Secondary development interface documentation

Document management

Version history

date	version	modify the record	author

Content

1 Introduction

This document will provide a comprehensive introduction to the application development interface instructions to help application developers to better secondary development.

2 System Module

2.1 API Interface

2.1.1 mf_driverlib_init

Interface Prototype:

int mf_driverlib_init (void)

Function Description:

System driver initialization, this function must be the main function to start calling the first function.

Interface Description:

	Parameter name	Effective value	Description

Input parameters	No		
Output parameters	No		
return value		0	Initialized successfully
		other	initialization failed

Note:

main function call the first function must be mf_driverlib_init, otherwise many device drivers can not be used.

2.1.2 osl_app_init

Interface Prototype:

int osl_app_init (int flag)

Function Description:

System component initialization function.

Interface Description:

	Parameter name	Effective value	Description
Input parameters	flag		Reserved

Output parameters	No		
return value		0	Initialized successfully
		other	initialization failed

2.1.3 mf_log_debug

Interface Prototype:

void mf_log_debug (const char * pcString, ...)

Function Description:

Output log, can not be called concurrently.

Interface Description:

	Parameter name	Effective value	Description
Input parameters	pcString		Log content
Output parameters	no		
return value	no		

2.1.4 mf_get_cpuid

Interface Prototype:

void mf_get_cpuid (unsigned char * id)

Function Description:

Get cpuid, 16 bytes in length

Interface Description:

	Parameter name	Effective value	Description
Input parameters	NO		
Output parameters	id		Returns CPU ID data, 16 bytes in length。
return value	NO		

2.1.5 mf_console_switch

Interface Prototype:

void mf_console_switch (int on)

Function Description:

Enable or disable log output.

Interface Description:

	Parameter name	Effective value	Description
Input parameters	on		1: Enable log output 0: Turn off log output

Output parameters	no		
	no		

2.2 programming case

2.2.1 Typical system initialization process

```
void main()
{

    mf_driverlib_init();   //initialize the associated hardware
    mf_console_switch(1); //enable log output
    osl_app_init(0);
    while(1) {
    mf_log_debug("hello,xpos\n"); //print the message
       mdelay(200); //delay 200 milliseconds
    }
}
```

2.2.2 Get the version number case

```
void version_probe(void)
{
    mf_log_debug("hardware version:0x%x\n", mf_hardware_ver());
    mf_log_debug("boot version:%s\n", mf_boot_ver());
}
```

3 Peripheral Modules

3.1 RFID Module

supports TYPE A and TYPE B contactless cards conforming to ISO14443-4. For detailed communication protocols and procedures please refer to ISO14443-3 specification.

supports Mifare one card.

3.1.1 API Interface

3.1.1.1 mf_rfid_tcl_open

Interface Prototype:

int mf_rfid_tcl_open (void)

Function Description:

Find and activate contactless cards.

Interface description:

	Parameter name	Effective value	Description
Input parameters	no		
Output parameters	no		
return value		0	Find a contactless card and activate it successfully

		-1	No contactless or activation card failed.

Note：

The interface to achieve the card, anti-conflict, the election card process.

3.1.1.2 mf_rfid_tcl_close

Interface Prototype:

int mf_rfid_tcl_close (void)

Function Description:

Turn off contactless cards.

Interface Description:

	Parameter name	Effective value	Description
Input parameters	no		
Output parameters	no		
return value		0	Close contactless card success
		-1	Close contactlesscard failed

3.1.1.3 mf_rfid_getuid

Interface Prototype:

int mf_rfid_getuid (unsigned char * uid)

Function Description:

Get the contactless card UID

Interface Description:

	Parameter name	Effective value	Description
Input parameters	no		
Output parameters	uid		Returns the UID of the non-card, up to 10 bytes.
return value		0	Get success
		-1	Failed to get

3.1.1.4 mf_rfid_tcl_exchange

Interface prototype:

int mf_rfid_tcl_exchange(unsigned char *tbuf, unsigned short tlen, unsigned char **rbuf, unsigned short *rlen)

Function Description:

Data interaction with contactless cards.

Interface Description:

	Parameter name	Effective value	Description
Input parameters	tbuf		Send data buffer
	tlen		Send data length
Output parameters	rbuf		Returns the pointer to the internal data receive buffer.
	rlen		Returns the length of valid data.
return value		0	success
		-1	failure

3.1.1.5 mf_rfid_mfcl_open

Interface Prototype:

int mf_rfid_mfcl_open (void)

Function Description:

Find and activate mifare one cards.

Interface Description:

	Parameter name	Effective value	Description

Input parameters	no		
Output parameters	no		
return value		0	Find a contactless card and activate it successfully
		-1	No contactless or activation card failed.

3.1.1.6 mf_rfid_mfcl_close

Interface Prototype:

int mf_rfid_mfcl_close (void)

Function Description:

Turn off contactless cards.

Interface Description:

	Parameter name	Effective value	Description
Input parameters	no		
Output parameters	no		

return value		0	Close contactless card success
		-1	Close contactless card failed

3.1.1.7 mf_rfid_mfcl_setkey

Interface Prototype:

int mf_rfid_mfcl_setkey(unsigned char *key)

Function Description:

Set the card authentication key

Interface Description:

	Parameter name	Effective value	Description
Input parameters	key		Key data
Output parameters	no		
return value		0	Set up successfully
		-1	Set up failed

3.1.1.8 mf_rfid_mfcl_auth

Interface Prototype:

int mf_rfid_mfcl_auth(int cmd, int sector)

Function Description:

The specified sector is authenticated according to the card authentication key

Interface Description:

	Parameter name	Effective value	Description
Input parameters	cmd	0x60	A set of key authentication
		0x61	B key authentication
	sector		Sector number
Output parameters	no		
return value		0	Certification is successful
		-1	Authentication failed

Note:

1、 into the cmd type macro as follows:

#define RFID_CMD_MIFARE_AUTH1A 0x60

#define RFID_CMD_MIFARE_AUTH1B 0x61

3.1.1.9 mf_rfid_mfcl_read

Interface Prototype:

int mf_rfid_mfcl_read(int block, unsigned char *buf, int *len)

Function Description:

Read the data to the card

Interface Description:

	Parameter name	Effective value	Description
Input parameters	block		Sector number
Output parameters	buf		Read sector data buffer
	len		Read the data length pointer
return value		0	Read data successfully
		other	Read data failed

3.1.1.10 mf_rfid_mfcl_write

Interface Prototype:

int mf_rfid_mfcl_write (int block, unsigned char *buf, int *len)

Function Description:

Write the data to the card

Interface Description:

	Parameter name	Effective value	Description

Input parameters	block		Sector number
Output parameters	buf		Write sector data buffer
	len		Write data length
return value		0	Write data success
		other	Write data failed

3.1.1.11 mf_rfid_mfcl_increment

Interface Prototype:

int mf_rfid_mfcl_increment(int block, int operand)

Function Description:

Add the action value to the specified data block

Interface Description:

	Parameter name	Effective value	Description
Input parameters	block		Data block number
	operand		Operation value
Output parameters			
return value		0	success
		other	failure

3.1.1.12 mf_rfid_mfcl_decrement

Interface Prototype:

int mf_rfid_mfcl_decrement (int block, int operand)

Function Description:

Specify the data block minus the operation value

接口说明:

	Parameter name	Effective value	Description
Input parameters	block		Data block number
	operand		Operation value
Output parameters			
return value		0	success
		other	failure

3.1.1.13 mf_rfid_mfcl_transfer

Interface Prototype:

int mf_rfid_mfcl_transfer (int block)

Function Description:

Copy the card memory data into the specified data block

Interface Description:

	Parameter name	Effective value	Description
Input parameters	block		Data block number
Output parameters			
return value		0	success
		other	failure

3.1.1.14 mf_rfid_mfcl_restore

Interface Prototype:

int mf_rfid_mfcl_restore (int block)

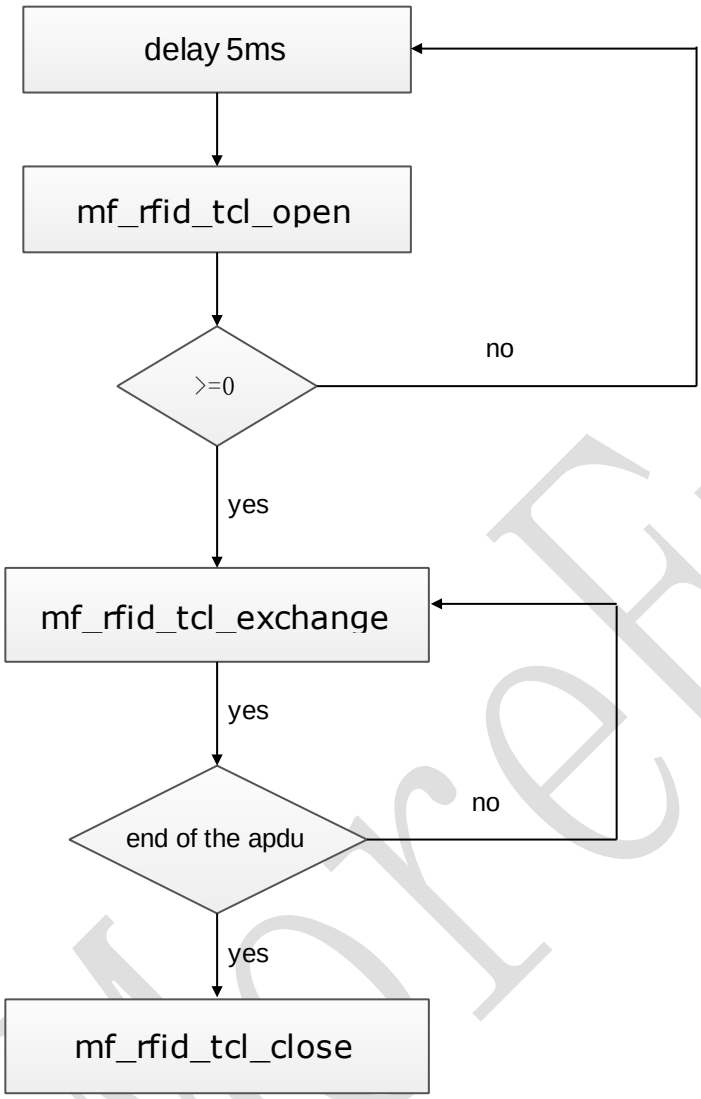
Function Description:

Copy the contents of the specified data block to the card memory

Interface Description:

	Parameter name	Effective value	Description
Input parameters	block		Data block number
Output parameters			
return value		0	success
		other	failure

3.1.2 **Call the process**



3.1.3 **Programming case**

3.1.3.1 Example of a CPU card

```
void rfid_test(void)
{
	int rc;
	unsigned char *rxbuf;
	unsigned short rxlen = 0;
	unsigned char uid[16];
	int uidlen = 0;
```

```
	unsigned char cmd1[19]={"\x00\xa4\x04\x00\x0e\x32\x50\x41\x59\x2e\x53\x59\x53\x2e\x44\x44\x46\x30\x31"};
	mf_log_debug("start rfid tcl test\n");
	while(1) {
		mdelay(200);
		rc = mf_rfid_tcl_open(); //find card
		mf_log_debug("mf_rfid_tcl_open %d\n",rc);
		if(rc >= 0) { //If the card is found
			uidlen = mf_rfid_getuid(uid); //Get the UID of the card
			mf_log_debug("mf_rfid_getuid %s\n", hexdump(uid, uidlen));
			rc = mf_rfid_tcl_exchange(cmd1, sizeof(cmd1), &rxbuf, &rxlen); //Perform APDU interaction
			mf_log_debug("mf_rfid_tcl_exchange rc %d, rxlen %d\n", rc, rxlen);
			if(rc == 0)mf_log_debug("PUT < %s\n", hexdump(rxbuf, rxlen));
			mf_rfid_tcl_close();  //Close the card
		}
	}
}
```

3.1.3.2 MIFARE ONE card example

```
void rfid_mfcl_test(void)
{
	int rc;
	unsigned char *rxbuf;
	unsigned short rxlen = 0;
	unsigned char uid[16];
	int uidlen = 0;

	mf_log_debug("start rfid mfcl test\n");

	unsigned char key[] = {0xff, 0xff, 0xff, 0xff, 0xff, 0xff};

	while(1) {
		rc = mf_rfid_mfcl_open();
		if(rc >= 0) {
			uidlen = mf_rfid_getuid(uid);
			mf_log_debug("mf_rfid_getuid %s\n", hexdump(uid, uidlen));
			mf_rfid_mfcl_setkey(key);

			do {

			if(mf_rfid_mfcl_auth(RFID_CMD_MIFARE_AUTH1A, 0) == 0) {
				mf_log_debug("auth ok....\n");
```

```
//break;

unsigned char buf[MIFARE_CL_PAGE_SIZE];
int len = 16;

rc = mf_rfid_mfcl_read(0, buf, &len);
if(rc != 0) {
      mf_log_debug("read error\n");
      break;
}

mf_log_debug("read block 0 dump %s\n", hexdump(buf, len));

memset(buf, 0x0, sizeof(buf));

buf[4] = 0xff;
buf[5] = 0xff;
buf[6] = 0xff;
buf[7] = 0xff;

buf[12] = ~0;
buf[13] = 0;
buf[14] = ~0;
buf[15] = 0;

rc = mf_rfid_mfcl_write(1, buf, 16);
if(rc != 0) {
      mf_log_debug("write block 1 error\n");
      break;
}

mf_log_debug("mf_rfid_mfcl_write ok\n");

len = 16;
rc = mf_rfid_mfcl_read(1,(unsigned char*) buf, (unsigned int*)&len);
if(rc != 0)
      break;
mf_log_debug("read page 1 dump %s\n", hexdump(buf, len));

//////////////////////////////

if(mf_rfid_mfcl_increment(1, 10) < 0)
{
```

```
        mf_log_debug("mfcl_increment :failed\r\n");
        break;
    }

    mf_log_debug(" mfcl_increment OK\n");

    if(mf_rfid_mfcl_transfer(1) < 0)
    {
        mf_log_debug("mfcl_transfer :failed\r\n");
        break;
    }

    mf_log_debug(" mfcl_transfer OK\n");
    len = 16;
    rc = mf_rfid_mfcl_read(1,(unsigned char*) buf, (unsigned int*)&len);
    if(rc != 0)
        break;
    mf_log_debug("read page 1 dump %s\n", hexdump(buf, len));

    if(mf_rfid_mfcl_restore(1) < 0)
    {
        mf_log_debug("mfcl_restore :failed\r\n");
        break;
    }

    mf_log_debug(" mfcl_restore OK\n");

    if(mf_rfid_mfcl_decrement(1,10) < 0)
    {
        mf_log_debug("mfcl_decrement :failed\r\n");
        break;
    }

    mf_log_debug(" mfcl_decrement OK\n");

    if(mf_rfid_mfcl_transfer(1) < 0)
    {
        mf_log_debug("mfcl_transfer :failed\r\n");
        break;
    }

    mf_log_debug(" mfcl_transfer OK\n");
```

```
                    if(mf_rfid_mfcl_restore(1) < 0)
                    {
                        mf_log_debug("mfcl_restore :failed\r\n");
                        break;
                    }

                    mf_log_debug(" mfcl_restore OK\n");



                    len = 16;
                    rc = mf_rfid_mfcl_read(1,(unsigned char*) buf,  (unsigned int*)&len);
                    if(rc != 0)
                        break;
                    mf_log_debug("read page 1 dump %s\n", hexdump(buf, len));

                    ////////////////////////////

                } else {
                    mf_log_debug("auth error....\n");
                }

                }while(0);

                mf_rfid_mfcl_close();
            } else {
                mdelay(10);
            }
        }
}
```

3.2 RTC module

3.2.1 **API Interface**

3.2.1.1 mf_rtc_set_time

Interface Prototype:

int mf_rtc_set_time(struct rtc_time *tm)

Function Description:

Set the date and time

Interface Description:

	Parameter name	Effective value	Description
Input parameters	tm		Time structure
Output parameters	no		
return value		0	Clear success
		other	Clear failure

Note:

1, the time structure tm Description:

```
struct rtc_time {
    int tm_sec;
    int tm_min;
    int tm_hour;
    int tm_mday;
    int tm_mon;
    int tm_year;
    int tm_wday;
    int tm_yday;
    int tm_isdst;
};
```

The structure includes year, month, day, hour, minute, second and

other parameters

3.2.1.2 mf_rtc_get_time

Interface Prototype:

int mf_rtc_get_time(struct rtc_time *tm)

Function Description:

Get date and time

Interface Description:

	Parameter name	Effective value	Description
Input parameters	no		
Output parameters	tm		Time structure
return value		0	Read the card information failed
		1	Read the magnetic card information is successful

3.2.2 **Programming case**

```
void rtc_probe(void)
{
    struct rtc_time time;
```

```
    struct rtc_time *tm = &time;

    tm->tm_year = 2014;
    tm->tm_mon = 6;
    tm->tm_mday = 4;
    tm->tm_hour = 20;
    tm->tm_min = 03;
    tm->tm_sec = 20;

    mf_rtc_set_time(tm);

    while(1) {
        mf_rtc_get_time(tm);
        mf_log_debug("read time %04d.%02d.%02d %02d/%02d/%02d\n",tm->tm_year, tm->tm_mon,
tm->tm_mday,tm->tm_hour, tm->tm_min, tm->tm_sec);
        mdelay(1000);
    }
}
```

3.3 Serial module

3.3.1 **API Interface**

3.3.1.1 mf_serial_open

Interface Prototype:

int mf_serial_open(int com, int baudrate, int wordlength, int stopbits, int parity)

Function Description:

Open the serial port and set the relevant parameters

Interface Description:

	Parameter name	Effective value	Description
Input parameters	com		Serial port number

	baudrate		Baud rate
	wordlength		Data bits
	stopbits		Stop bit
	parity		Parity bit
Output parameters	no		
return value		0	Open successfully
		Other	Open failed

Note:

1, serial port port number com Parameter Description:

```
enum
{
    MF_UART_COM1 = 1,
    MF_UART_COM2,
    MF_UART_COM3,
    MF_UART_COM4,

    MF_UART_COM20 = 20,
    MF_UART_COM21,
    MF_UART_COM22,
    MF_UART_COM23,
    MF_UART_COM24,
    MF_UART_COM25,

    MF_UART_COM30 = 30,
```

};

Involving the use of serial modules including contact IC card, wireless module, one-dimensional head, program upgrade, Bluetooth module

2、the baud rate baudrate parameter description:

Support baud rate: 9600,38400,115200

3、Data bit wordlength Parameter Description:

#define MF_UART_WordLength_8b (0x0000)

#define MF_UART_WordLength_9b (0x1000)

Data bits support 8 or 9 bit modes

4、stop bit stopbits parameter description:

#define MF_UART_StopBits_1 (0x0000)

#define MF_UART_StopBits_0_5 (0x1000)

#define MF_UART_StopBits_2 (0x2000)

#define MF_UART_StopBits_1_5 (0x3000)

Stop bit supports 1,0.5,2,1.5 bit mode

5、Parity bit parity parameter description:

#define MF_UART_Parity_No (0x0000)

#define MF_UART_Parity_Even (0x0400)

#define MF_UART_Parity_Odd (0x0600)

The parity bit supports odd, even, and no parity

3.3.1.2 mf_serial_close

Interface Prototype:

int mf_serial_close (int com)

Function Description:

Close the serial port

Interface Description:

	Parameter name	Effective value	Description
Input parameters	com		Serial port number
Output parameters	no		
return value		0	Closed successfully
		other	Close failed

3.3.1.3 mf_serial_write

Interface Prototype:

int mf_serial_write(int com, unsigned char *buf, int size)

Function Description:

Write the data to the serial send buffer. There will be less case, the need for application processing.

Interface Description:

	Parameter name	Effective value	Description

Input parameters	com		Serial port number
	buf		Send data content
	size		Send data size
Output parameters	no		
return value		<0	Fill in failed
		other	The actual size of the data being written

3.3.1.4 mf_serial_read

Interface Prototype:

int mf_serial_read(int com, unsigned char *buf, int size, int milliseconds)

Function Description:

Read from the serial buffer data, there will be less read the situation, the need for application processing.

Interface Description:

	Parameter name	Effective value	Description
Input parameters	com		Serial port number
	size		Send data size

	milliseconds		Receive timeout
Output parameters	buf		Unit: ms
return value		<0	Receive data content
		other	Reception failed

3.3.1.5 mf_serial_flush

Interface prototype:

int mf_serial_flush(int com)

Function Description:

Clear the buffer data received and sent by the serial port

Interface Description:

	Parameter name	Effective value	Description
Input parameters	com		Serial port number
Output parameters	no		
return value		0	Emptied successfully
		other	Empty failed

3.3.1.6 mf_serial_data_avail

Interface prototype:

int mf_serial_data_avail(int com, int *txlen, int *rxlen)

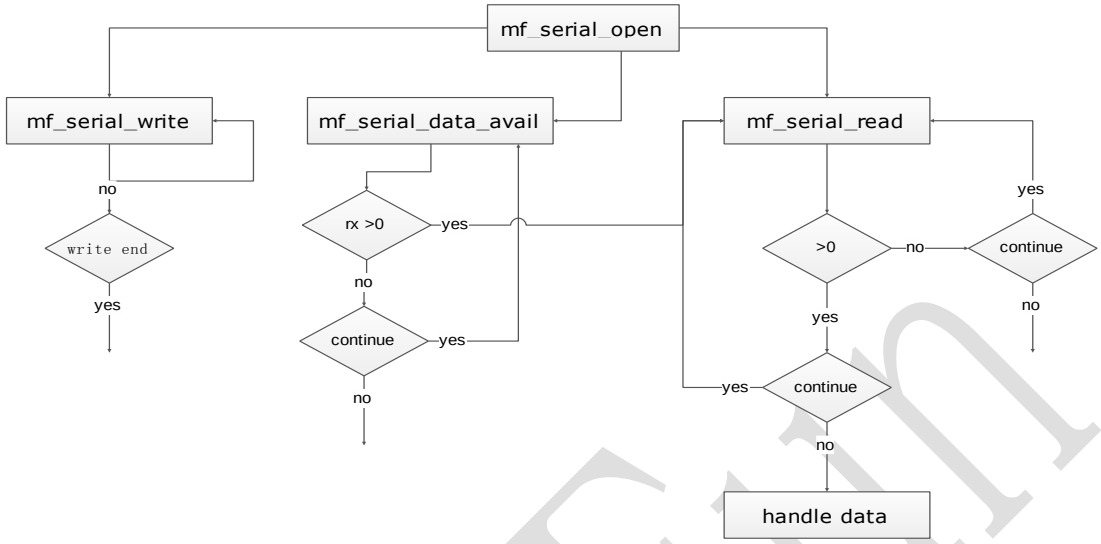
Function Description:

Get the serial receive and transmit buffer data length

Interface Description:

	Parameter name	Effective value	Description
Input parameters	com		Serial port number
Output parameters	txlen		Returns the available size of the send buffer
	rxlen		Returns the valid data length in the receive buffer
return value		0	Get success
		other	Failed to get

3.3.2 Call the process



3.3.3 Programming Examples

```
void uart_probe(void)
{
	unsigned char rbuf[512];
	int ret;
	int rxlen;
	int total = 0;
	int com = MF_UART_COM20;

	mf_serial_open(com, 115200, MF_UART_WordLength_8b, MF_UART_StopBits_1, MF_UART_Parity_No); //open uart
	mf_serial_flush(com);
	while(1) {
		mf_serial_data_avail(com, NULL, &rxlen); //Gets the length of the data in the receive buffer
		ret = mf_serial_read(com, rbuf, rxlen, 500); //Read data
		if(ret > 0) {
			mf_serial_write(com, rbuf, ret);
			if(rbuf[0] == 0x55)
				goto out;
		}
	}
out:
	mf_serial_close(com);

	return;
}
```

3.4 Magnetic stripe cards module

3.4.1 **API Interface**

3.4.1.1 mf_magtek_flush

Interface Prototype:

int mf_magtek_flush(void)

Function Description:

Clear disk data buffer

Interface Description:

	Parameter name	Effective value	Description
Input parameters	no		
Output parameters	no		
return value		0	Clear success
		other	Failed to clear

3.4.1.2 mf_magtek_read

Interface prototype:

int mf_magtek_read(struct magtek_track_info * info)

Function Description:

Read the magnetic card information

Interface Description:

	Parameter name	Effective value	Description
Input parameters	NO		
Output parameters	info		Track information
return value		0	Read the card information failed
		1	Read the magnetic card information is successful

Note:

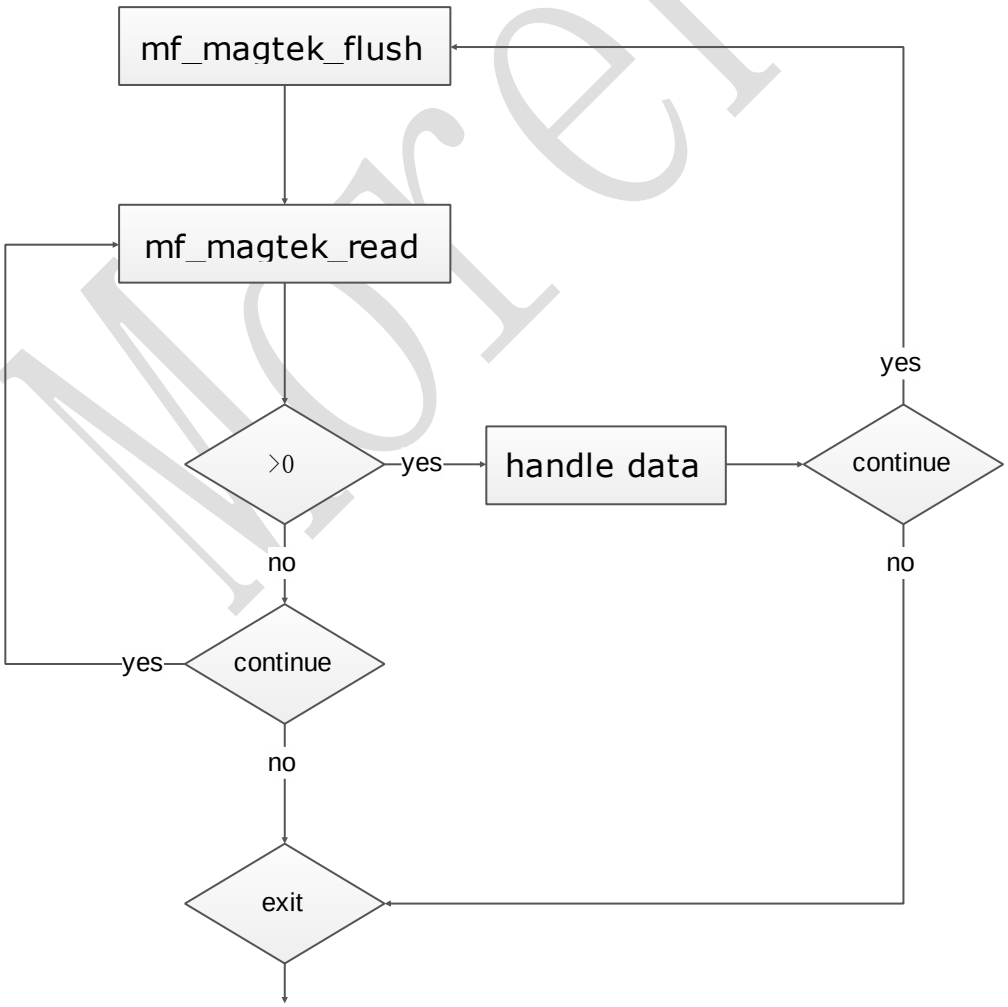
1、track information info structure is as follows:

```
#define MAGTEK_TRACK_7BIT_MAX_CHARS     (100)
#define MAGTEK_TRACK_5BIT_MAX_CHARS     (140)
#define MAGTEK_TRACK_BUFFER_BITS        (704)

struct magtek_track_a_chars {
    unsigned char chars[MAGTEK_TRACK_7BIT_MAX_CHARS];
};

struct magtek_track_b_chars {
    unsigned char chars[MAGTEK_TRACK_5BIT_MAX_CHARS];
};

struct magtek_track_c_chars {
    unsigned char chars[MAGTEK_TRACK_5BIT_MAX_CHARS];
```

```
};

struct magtek_track_info {
    struct magtek_track_a_chars a_chars;
    struct magtek_track_b_chars b_chars;
    struct magtek_track_c_chars c_chars;
};
```

Magnetic card includes 1/2/3 orbit, each track information is not the same, all end with 0.

3.4.2 **Call the process**



3.4.3 Programming Examples

```
void magtek_probe(void)
{
	struct magtek_track_info info;

	mf_magtek_flush(); //Clear the cache data

	while(1)
	{
		if(mf_magtek_read(&info) == 1){ //Get the read information
			mf_log_debug("MAGTEK A TRACK:\n%s\n", info.a_chars.chars);
			mf_log_debug("MAGTEK B TRACK:\n%s\n", info.b_chars.chars);
			mf_log_debug("MAGTEK C TRACK:\n%s\n", info.c_chars.chars);
		}

		mf_mdelay(30);
	}
}
```

3.5 Contact IC card module

Implement a complete ISO7816 protocol that conforms to the EMV specification and provides an interface to interface with the IC card for APDU.

3.5.1 API Interface

3.5.1.1 icc_open

Interface Prototype:

int icc_open(int socket)

Function Description:

Open the IC card reader

Interface Description:

	Parameter name	Effective	Description

		value	
Input parameters	socket	0	Card slot number
Output parameters	no		
return value		0	Open the card reader successfully
		other	Open the card reader failed

Note:

1, card slot port number description

enum

{

ICC_SOCKET1 = 0, //SMART CARD

ICC_SOCKET2, //TSAM

ICC_SOCKET3,

};

Contact IC card communication card slot port number is ICC_SOCKET1

3.5.1.2 icc_close

Interface prototype:

int icc_close(int socket)

Function Description:

Close the IC card reader

Interface Description:

	Parameter name	Effective value	Description
Input parameters	socket	0	Card slot number
Output parameters	no		
return value		0	Closed successfully
		-1	Close failed

3.5.1.3 icc_present

Interface Prototype：

int icc_present (int socket)

Function Description:

Card in position detection

Interface Description:

	Parameter name	Effective value	Description
Input parameters	socket		Card slot number
Output parameters	no		
return value		0	No card insertion detected

		1	A card insertion is detected

3.5.1.4 icc_powerup

Interface ：

int icc_powerup(int socket, unsigned char * atrstr, int buflen)

Function Description:

Activation Sequence

Interface Description:

	Parameter name	Effective value	Description
Input parameters	socket	0	Card slot number
	buflen		Reset the response buffer size
Output parameters	atrstr		Returns the ATR data
return value		<= 0	Activation IC card failed
		other	Activation of IC card is successful

3.5.1.5 icc_powerdown

Interface Prototype：

int icc_powerdown (int socket)

Function Description:

Deactivation Sequence

Interface Description:

	Parameter name	Effective value	Description
Input parameters	socket		Card slot number
Output parameters	no		
return value		0	Deactivation IC card failed
		other	Deactivation IC card is successful

3.5.1.6 icc_send_apdu

Interface Prototype:

int icc_send_apdu(int socket, unsigned char *buffer, unsigned short length, unsigned char *rbuffer)

Function Description:

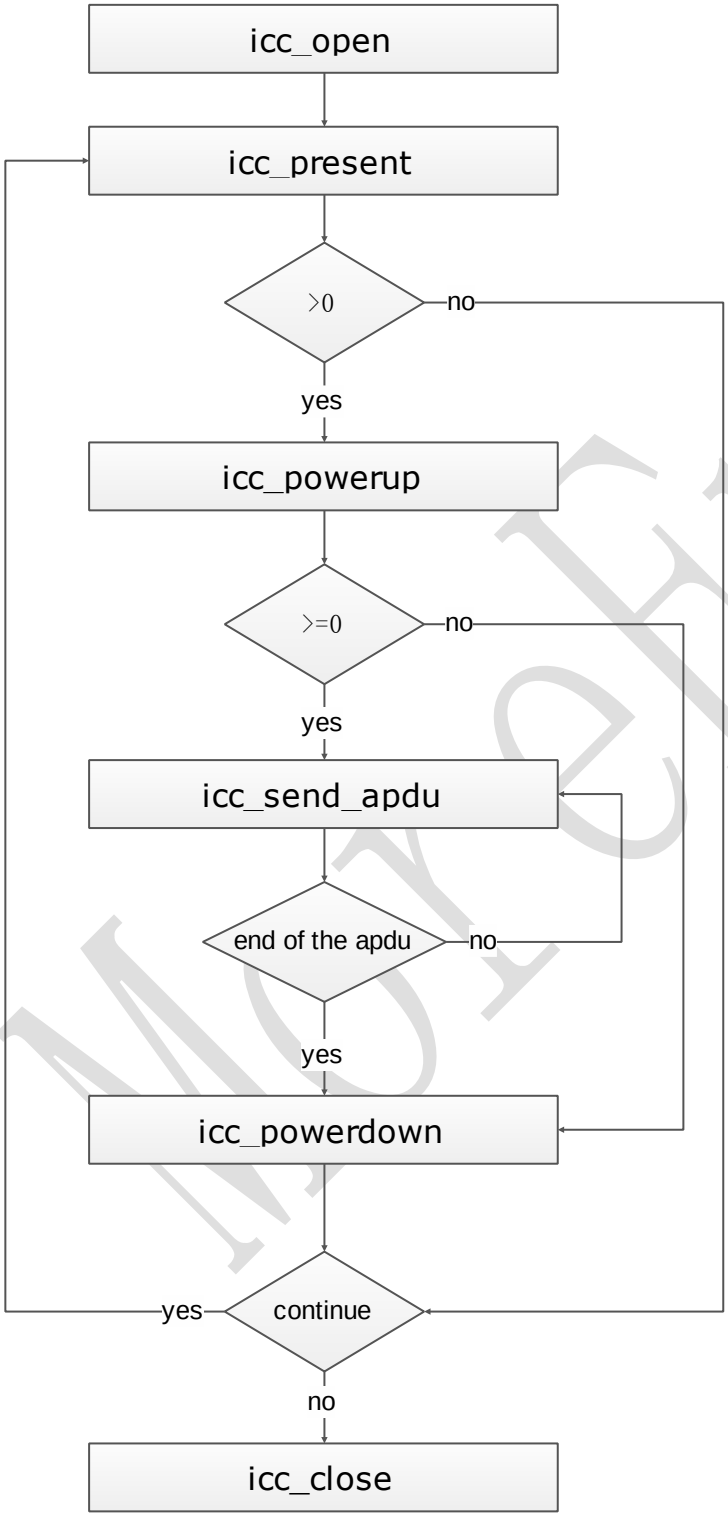
Card data interaction

Interface Description:

	Parameter name	Effective value	Description
Input parameters	socket		Card slot number

	buffer		Send data to the buffer
	length		The length of the data sent
Output parameters	rbuffer		Returns the response data for the card (this buffer size is at least 256 bytes)
return value		<=0	Interaction failed
		other	The length of the card response data.

3.5.2 **Call the process**



3.5.3 **Programming examples**

```
void smart_icc_test(void)
{
	unsigned char atr[64], rbuffer[64];
	int ret, atrlen;
	unsigned char buffer[128];
	unsigned char test_cmd[] = {0x00,0xA4,0x04,0x00,0x08,0xA0,0x00,0x00,0x03,0x33,0x01,0x01,0x01,0x00};

loop:
	atrlen = -1;
	ret = 0;
	icc_open(ICC_SOCKET1); //Open the card reader

	if(icc_present(ICC_SOCKET1) != 1) { //Determine whether the card is inserted
		mf_log_debug("no smartcard find....\n");
		icc_close(ICC_SOCKET1);
		mdelay(1000);
		goto loop;
	}

	mf_log_debug("find smard card......\n");
	mdelay(500);
	ret = icc_powerup(ICC_SOCKET1, atr, sizeof(atr)); //Perform a reset power up operation
	if(ret < 0) {
		mf_log_debug("@icc_powerup error %d\n", ret);
		mdelay(1000);
		goto out;
	}

	mf_log_debug("icc_powerup ok %d\n", ret);
	mf_log_debug("ATR:\n");
	mf_log_hexdump(ret, atr);
	atrlen = ret;

	int try = 1;
	while(try--) {
		buffer[0] = 0xa0;
		buffer[1] = 0xfa;
		buffer[2] = 0x0;
		buffer[3] = 0x0;
		buffer[4] = 0x0;
```

```
            ret = icc_send_apdu(ICC_SOCKET1, test_cmd, sizeof(test_cmd), rbuffer); //Send APDU command
            mf_log_debug("icc_send_apdu ret %d\n", ret);
            if(ret < 0) {
                  mf_log_debug("no resp....%d\n", ret);
            } else {
                  mf_log_debug("rep:\n");
                  mf_log_hexdump(ret, rbuffer);
            }
      }
out:
      icc_powerdown(ICC_SOCKET1); //Execute card power down
      icc_close(ICC_SOCKET1); //Close the card reader
      mdelay(500);

      goto loop;
}
```

3.6 1D code module

One-dimensional code decoding data through the serial port to read, the corresponding port number MF_UART_COM22, serial port parameters 9600, 8BIT DATA, 1 BIT STOP, NO PARITY.

3.6.1 **API Interface**

3.6.1.1 mf_barcode_trig

Interface Prototype:

int mf_barcode_trig(int on)

Function Description:

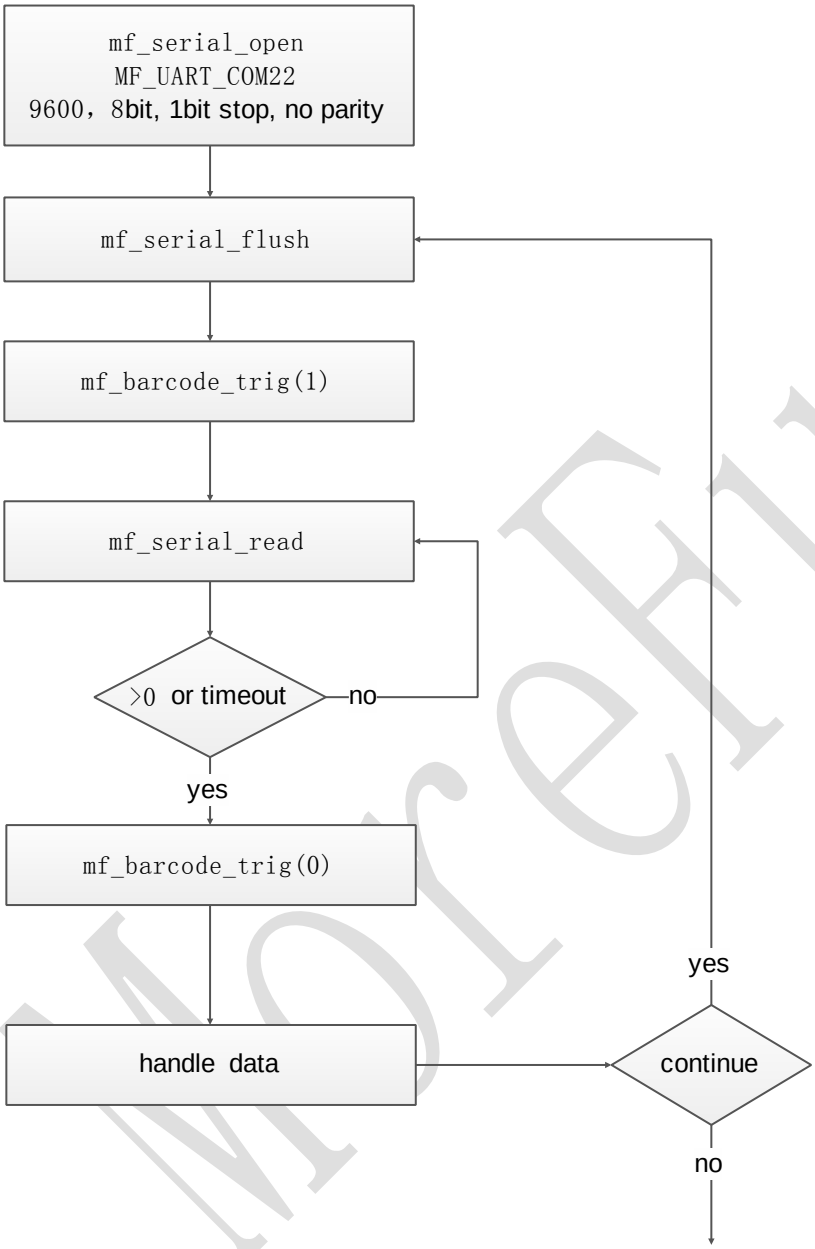
Trigger one-dimensional head scan.

Interface Description:

	Parameter name	Effective	Description

		value	
Input parameters	on		1: startscanning 0: Stop scanning
Output parameters	no		
return value		0	success
		other	failure

3.6.2 **Call the process**



3.6.3 **Programming examples**

```
void barcode_test()
{
	unsigned char rbuf[64];
	int ret;

	mf_log_debug("barcode_test.....\n");
```

```
    while(1) {
        //open uart。
        mf_serial_open(MF_UART_COM22, 9600, MF_UART_WordLength_8b, MF_UART_StopBits_1, MF_UART_Parity_No);
        mf_barcode_trig(1); //Trigger scan
        ret = mf_serial_read(MF_UART_COM22, rbuf, sizeof(rbuf), 1000 * 2); //Read data
        mf_barcode_trig(0); //Stop scanning
        mf_serial_close(MF_UART_COM22);
        printf("barcode read ret %d\n", ret);

        if(ret>0) {
            mf_log_hexdump(ret, rbuf);
        }

        mdelay(100);
    }

}
```

3.7 Buzzer and LED module

3.7.1 **API Interface**

3.7.1.1 mf_buzzer_control

Interface Prototype:

int mf_buzzer_control(int status)

Function Description:

Buzzer control

Interface Description:

	Parameter name	Effective value	Description

Input parameters	status		Buzzer status 0 - stop the buzzer ring Others - start the buzzer ring
Output parameters	no		
return value		0	Control success
		other	Control failed

3.7.1.2 mf_led_control

Interface Prototype：

int mf_led_control(int led , int status)

Function Description:

led lights control

Interface Description:

	Parameter name	Effective value	Description
Input parameters	status		Light status: 0 - light dark Other - light on
	led		Lamp number

Output parameters	no		
return value		0	Control success
		other	Control failed

Note:

1、 light number Description:

```
enum
{
    MF_LED1,
    MF_LED2,
    MF_LED3,
    MF_LED4,
    MF_LED5
};
```

There are five main lights, including four contactless lights and a status light

3.7.2 **Programming examples**

3.7.2.1 Lamp control example

```
void led_probe(void)
{
      int led = 0;
      while(1) {
                mf_led_control(led, 1);
                mdelay(200);
                mf_led_control(led, 0);
                led++;
                if(led == 4)led = 0;
      }
```

```
}
```

3.7.2.2 Example of buzzer control

```
void buzzer_probe(void)
{
      while(1) {
           mf_buzzer_control(1);
           mdelay(2000);
           mf_buzzer_control(0);
           mdelay(2000);
      }
}
```

4 NET Module

4.1 API Interface

4.1.1 **Connect the network**

4.1.1.1 net_func_link

Interface prototype：

```
intnet_func_link(char * title, char * apn ,inttimetry, inttimeover);
```

Function Description:

Connect Network

Interface Description:

	Parameter name	Effective value	Description
Input parameters	title		Show title
Input parameters	apn		Apn name, wireless module using wifi can be empty
Input parameters	timetry		number of retries
Input parameters	timeover		overtime time
Output parameters	no		
return value			0 successful

4.1.1.2 net_func_unlink

Interface prototype:

```
intnet_func_unlink()
```

Function Description:

Disconnect the network

Interface Description:

	Parameter name	Effective value	Description
Input parameters	no		
Output parameters	no		

return value			0 successful

4.1.2 SOCK communication

4.1.2.1 mf_sock_connect

Interface prototype:

```
intmf_sock_connect(ints,constchar * pIp, intnPort);
```

Function Description:

sock connection

Interface Description:

	Parameter name	Effective value	Description
Input parameters	s		Channel number
Input parameters	pIp		Server IP
Input parameters	nPort		Server port
Output parameters	no		
return value			0 successful

4.1.2.2 mf_sock_recv

Interface prototype:

```
intmf_sock_recv(ints, unsignedchar * buff, intlen, unsignedinttimeover)
```

Function Description:

Receive data

Interface Description:

	Parameter name	Effective value	Description
Input parameters	s		Channel number
Input parameters	Len		Buffer size
Input parameters	timeover		overtime time
Output parameters	buff		Receive buffer
Return value			0 successful

4.1.2.3 mf_sock_send

Interface prototype：

```
intmf_sock_send(ints, unsignedchar * buff , intsize);
```

Function Description:

send data

Interface description

	Parameter name	Effective value	Description
Input parameters	s		Channel number
Input parameters	Len		Send data size
Input parameters	buff		Send buffer
Return value			0 successful

4.1.2.4 mf_sock_close

Interface prototype：

```
intmf_sock_close(ints);
```

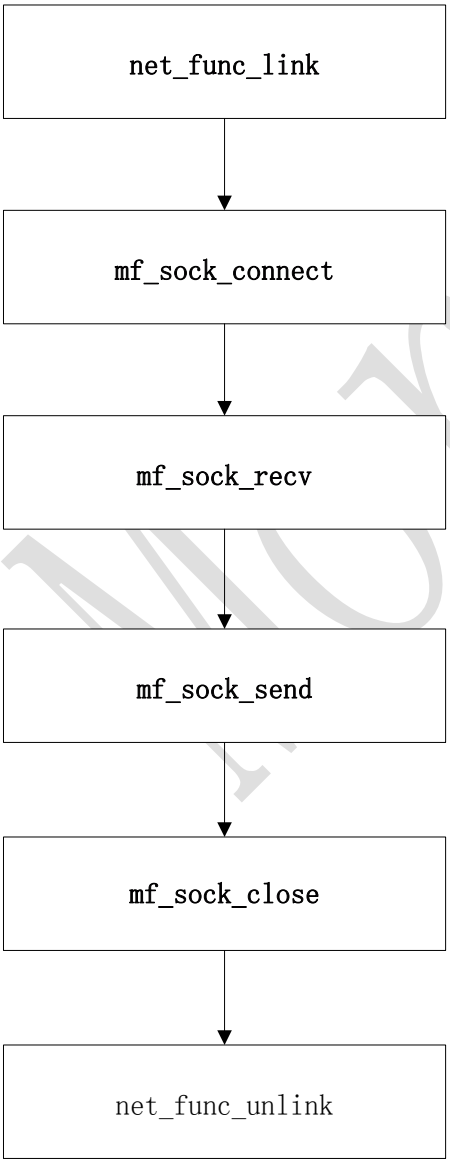
Function Description:

Close sock

Interface Description:

	Parameter name	Effective value	Description
Input parameters	s		Channel number
Output parameters	no		
Return value			0 successful

4.2 Call the process



4.3 Programming examples

Please refer to xpos-sys-demo.

XGUI function interface

Document management

Version history

date	version	Modify the record	author

5 XGUI MODULE

5.1 API Interface

5.1.1 Basic operation

5.1.1.1 xgui_BeginBatchPaint

Interface prototype：

```
voidxgui_BeginBatchPaint()
```

Function Description:

Batch refresh starts

Interface Description:

	Parameter name	Effective value	Description
Input parameters	no		
Output parameters	no		
Return value			

5.1.1.2 xgui_EndBatchPaint

Interface prototype：

```
voidxgui_EndBatchPaint()
```

Function Description:

Batch refresh ends

Interface Description:

	Parameter name	Effective value	Description
Input parameters	no		
Output parameters	no		
Return value			

5.1.1.3 xgui_SetColor

Interface prototype:

```
voidxgui_SetColor(intnColor);
```

Function Description:

Set the foreground color.

Interface Description:

	Parameter name	Effective value	Description
Input parameters	nColor	Int	Foreground color
Output parameters	no		
Return value	no		

5.1.1.4 xgui_GetColor

Interface prototype:

```
intxgui_GetColor(void)
```

Function Description:

Read foreground color.

Interface Description:

	Parameter name	Effective value	Description
Input parameters	no		
Output parameters	no		
Return value		Int	Color value

5.1.1.5 xgui_SetBgColor

Interface prototype:

```
voidxgui_SetBgColor(intnColor);
```

Can explain:

Set background color.

Interface Description:

	Parameter name	Effective value	Description
Input parameters	nColor	Int	Color value
Output parameters	no		
Return value			

5.1.1.6 xgui_GetBgColor

Interface Prototype:

```
intxgui_GetBgColor(void)
```

Function Description:

Read the background color

Interface Description:

	Parameter name	Effective value	Description
Input parameters	no		
Output parameters	no		
Return value		Int	Background color

5.1.1.7 xgui_Pixel

Interface prototype:

```
intxgui_Pixel(intnX, intnY)
```

Function Description:

Use foreground color points

Interface Description:

	Parameter name	Effective value	Description
Input parameters	nX		Horizontal offset
Input parameters	nY		Vertical offset
Output parameters	no		
Return value			Keep it

5.1.1.8 xgui_LineTo

Interface prototype:

```
voidxgui_LineTo(intnX, intnY)
```

Function Description:

Draw lines from the current position

Interface Description:

	Parameter name	Effective value	Description

Input parameters	nX		Horizontal offset
Input parameters	nY		Vertical offset
Output parameters	no		
Return value			

5.1.1.9 xgui_Bar_RC

Interface prototype:

```
voidxgui_Bar_RC(constRECT *pRect)
```

Function Description:

Fill a piece of area

Interface Description:

	Parameter name	Effective value	Description
Input parameters	pRect		area
Output parameters	no		
Return value			

5.1.1.10 xgui_SetBarFill

Interface prototype:

```
voidxgui_SetBarFill(intnColor)
```

Function Description:

Set the fill color

Interface Description:

	Parameter name	Effective value	Description
Input parameters	nColor	Int	Color value
Output parameters	no		

Return value			

5.1.1.11 xgui_GetBarFill

Interface prototype:

```
intxgui_GetBarFill(void)
```

Function Description:

Read the fill color

Interface Description:

	Parameter name	Effective value	Description
Input parameters	NO		
Output parameters	NO		
Return value			Color value

5.1.1.12 xgui_SetFont

Interface prototype:

```
voidxgui_SetFont(enumUFONTenuFont)
```

Function Description:

Set the text font

Interface Description:

	Parameter name	Effective value	Description
Input parameters	enuFont		enuFont font
Output parameters	NO		
Return value			

5.1.1.13 xgui_GetFont

Interface prototype:

```
enumUFONTxgui_GetFont(void)
```

Function Description:

Read the current font

Interface Description:

	Parameter name	Effective value	Description
Input parameters	no		
Output parameters	no		
Return value			enuFont font

5.1.1.14 xgui_SetTextColor

Interface prototype:

```
voidxgui_SetTextColor(intnColor)
```

Function Description:

Set the text color

Interface Description:

	Parameter name	Effective value	Description
Input parameters	nColor		Color value
Output parameters	NO		
Return value	NO		

5.1.1.15 xgui_GetTextColor

Interface prototype:

```
intxgui_GetTextColor(void)
```

Function Description:

Read the current text color

Interface Description:

	Parameter name	Effective value	Description
Input parameters	no		
Output parameters	no		
Return value			Color value

5.1.1.16 xgui_SetTextBgColor

Interface prototype:

```
voidxgui_SetTextBgColor(intnColor) ;
```

Function Description:

Set the text background color

Interface Description:

	Parameter name	Effective value	Description
Input parameters	nColor		Color value
Output parameters	no		
Return value	no		

5.1.1.17 xgui_GetTextBgColor

Interface prototype:

```
intxgui_GetTextBgColor(void)
```

Function Description:

Read the current text color

Interface Description:

	Parameter name	Effective value	Description
Input parameters	no		
Output parameters	no		
Return value			Color value

5.1.1.18 xgui_SetCurrent

Interface Prototype:

```
voidxgui_SetCurrent(constPOINT* pPoint)
```

Can explain:

Set the current drawing position

Interface Description:

	Parameter name	Effective value	Description
Input parameters	pPoint		Point position
Output parameters	no		
Return value			

5.1.1.19 xgui_GetCurrent

Interface prototype：

```
voidxgui_GetCurrent(POINT* pPoint)
```

Function Description:

Read the current drawing position

Interface Description:

	Parameter name	Effective value	Description

Input parameters	no		
Output parameters	no		
Return value			Point position

5.1.1.20 xgui_ClearDC

Interface prototype：

```
voidxgui_ClearDC(void)
```

Function Description:

Clear the current view

Interface Description:

	Parameter name	Effective value	Description
Input parameters	no		
Output parameters	no		
Return value			

5.1.1.21 xgui_CPixel

Interface prototype：

```
intxgui_CPixel(intnX, intnY, intnColor)
```

Function Description:

points

接口说明：

	Parameter name	Effective value	Description
Input parameters	nX		Horizontal offset
Input parameters	nY		Vertical offset

Input parameters	nColor		Colour
Output parameters			
Return value	no		

5.1.1.22 xgui_GetPixel

Interface prototype：

```
intxgui_GetPixel(intnX, intnY)
```

Function Description:

Read the color of the point

Interface Description:

	Parameter name	Effective value	Description
Input parameters	nX		Horizontal offset
Input parameters	nY		Vertical offset
Output parameters	no		
Return value			Colour

5.1.1.23 xgui_TextOut

Interface prototype：

```
intxgui_TextOut(intnX, intnY, constchar *strText)
```

Function Description:

Output string

Interface description:

	Parameter name	Effective value	Description
Input parameters	nX		Horizontal offset
Input parameters	nY		Vertical offset

Input parameters	strText		String
Output parameters	no		
Return value			

5.1.1.24 xgui_GetTextWidth

Interface Prototype:

```
intxgui_GetTextWidth(constchar *szText)
```

Function Description:

Read the string width

Interface Description:

	Parameter name	Effective value	Description
Input parameters	szText		String
Output parameters	no		
Return value			width

5.1.1.25 xgui_GetTextHeight

Interface prototype:

```
intxgui_GetTextHeight(constchar *szText)
```

Function Description:

Read the string height

Interface Description:

	Parameter name	Effective value	Description
Input parameters	szText		String
Output parameters	no		
Return value			height

5.1.1.26 xgui_CLine

Interface peototype：

```
voidxgui_CLine(intnX1, intnY1, intnX2, intnY2, intnColor)
```

Function Description:

Draw lines

Interface Description:

	Parameter name	Effective value	Description
Input parameters	nX1		Lateral offset
Input parameters	nY1		Longitudinal offset
Input parameters	nX2		Horizontal offset 2
Input parameters	nY2		Longitudinal offset 2
Input parameters	nColor		Colour
Output parameters	no		
Return value			

5.1.1.27 xgui_ClearRect

Interface prototype：

```
voidxgui_ClearRect(intnLeft, intnTop, intnRight, intnBottom, intnColor);
```

Function Description:

Clear a piece of area

Interface Description:

	Parameter name	Effective value	Description

Input parameters	nLeft		left
Input parameters	nTop		on
Input parameters	nRight		right
Input parameters	nBottom		under
Output parameters	NO		
Return value			

5.1.1.28 xgui_GetWidth

Interface prototype：

```
intxgui_GetWidth(void);
```

Function Description:

Read the screen width

Interface Description:

	Parameter name	Effective value	Description
Input parameters	no		
Output parameters	no		
Return value			Screen width

5.1.1.29 xgui_GetHeight

Interface prototype：

```
intxgui_GetHeight(void);
```

Function Description:

Read the screen height

Interface Description:

	Parameter name	Effective value	Description
Input parameters	no		
Output parameters	no		
Return value			Screen height

5.1.1.30 xgui_Page_OP_Paint

Interface prototype:

```
void xgui_Page_OP_Paint(char * szLeftOp, char * szRightOp);
```

Function Description:

2 buttons are displayed in the lower left and lower right corners of the screen

Interface Description:

	Parameter name	Effective value	Description
Input parameters	szLeftOp		What is shown on the left
Input parameters	szRightOp		What is shown on the right
Output parameters	no		
Return value			

5.1.2 Input method

5.1.2.1 xgui_ImeSetMode

Interface prototype:

```
intxgui_ImeSetMode(intnDefMode, intnAllowMode, intbPassword, intbSmsInput);
```

Function Description:

Set the input method

Interface Description:

	Parameter name	Effective value	Description
Input parameters	nDefMode		The default input method
	nAllowMode		
	bPassword		Whether the password is entered
	bSmsInput	0	Keep it
Output parameters	no		
Return value			

5.1.2.2 xgui_ImeStartInput

Interface prototype:

```
intxgui_ImeStartInput(char * pcInputBuffer, intnMaxLength, int * pPosition,
char * pcHelpString)
```

Function Description:

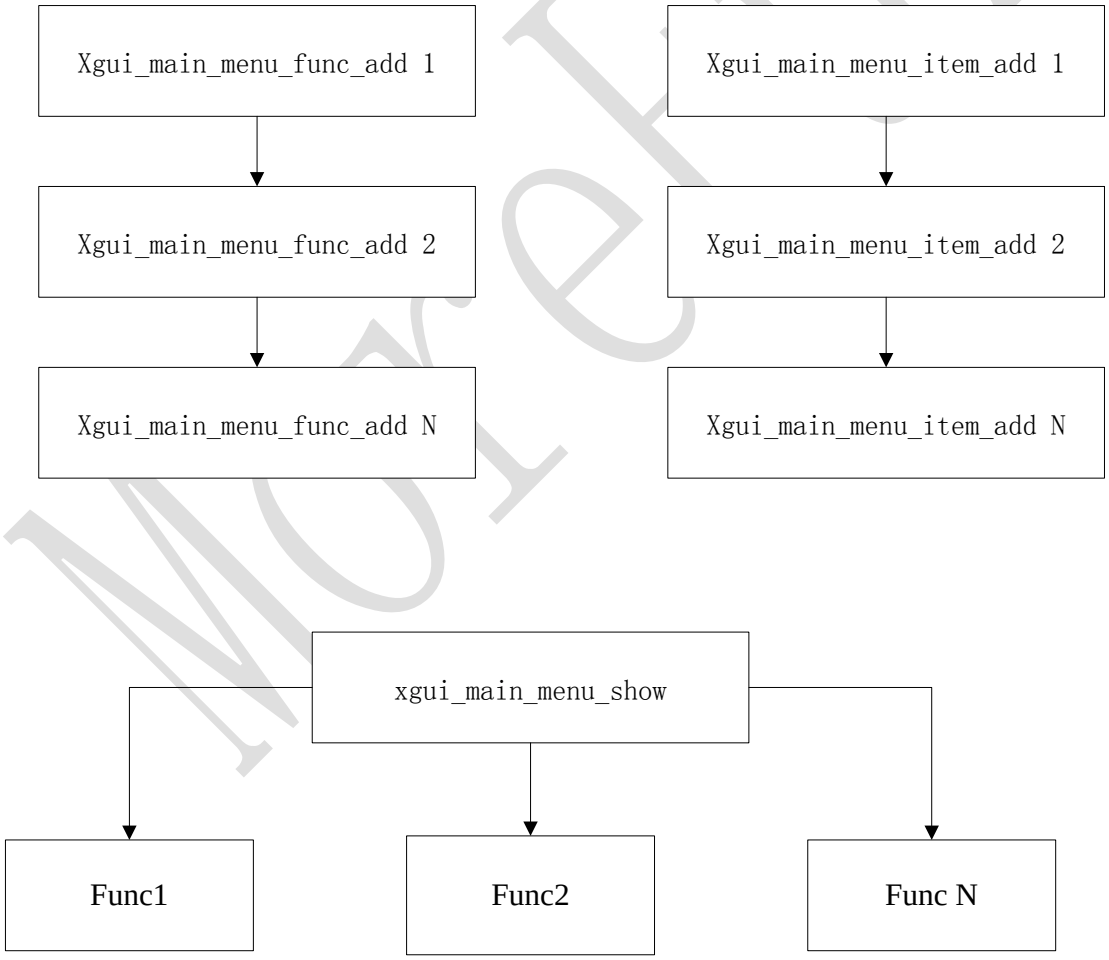
xxxx

Interface Description:

	Parameter name	Effective value	Description
Input parameters	pcInputBuffer		Input buffer

	nMaxLength		Maximum input length
	pPosition		Cursor position
	pcHelpString		Enter the title
Output parameters	no		
Return value			Enter the length

5.1.3 **Menu operation**



5.1.3.1 xgui_main_menu_func_add

Interface prototype：

```
intxgui_main_menu_func_add(void * pfunc)
```

Function Description:

Add a callback function

Interface Description:

	Parameter name	Effective value	Description
Input parameters	pfunc		Callback
Output parameters	no		
Return value			Keep it

5.1.3.2 xgui_main_menu_func_del

Interface prototype：

```
intxgui_main_menu_func_del(void * pfunc)
```

Function Description:

Remove the callback function

Interface Description:

	Parameter name	Effective value	Description
Input parameters	pfunc		Callback
Output parameters	no		
Return value			Keep it

5.1.3.3 xgui_main_menu_item_add

Interface prototype：

```
intxgui_main_menu_item_add(conststruct_main_menu_item_def * menu_item)
```

Function Description:

Add a menu item

Interface Description:

	Parameter name	Effective value	Description
Input parameters	menu_item		Menu structure
Output parameters	no		
Return value			0 successful Other failure

Note ： You can not add the same id of the menu item, otherwise it will return to failure.

5.1.3.4 xgui_main_menu_item_del

Interface prototype：

```
intxgui_main_menu_item_del(char *name ,char *id);
```

Function Description:

Delete the menu item

Interface Description:

	Parameter name	Effective value	Description
Input parameters	no		
Output parameters	no		
Return value			

5.1.3.5 xgui_main_menu_show

Interface prototype：

```
voidxgui_main_menu_show(char *id , inttimeover)
```

Function Description:

Display the menu

Interface Description:

	Parameter name	Effective value	Description
Input parameters	id		The displayed menu id
Input parameters	timeover		No operation exit time
Output parameters	no		
Return value			

5.1.4 message

5.1.4.1 xgui_PostMessage

Interface prototype：

```
unsignedintxgui_PostMessage(unsignedintnMsgID,unsignedintwParam,
unsignedintlParam);
```

Function Description:

Send a message

Interface Description:

	Parameter name	Effective value	Description
Input parameters	nMsgID		Message id
Input parameters	wParam		Parameter 1
Input parameters	lParam		Parameter 2
Output parameters	no		
Return value			

5.1.4.2 xgui_GetMessageWithTime

Interface prototype:

```
unsignedintxgui_GetMessageWithTime( PMESSAGEpMsg , inttimeover)
```

Function Description:

Take a message from the system message queue and wait for a timeout if there is no message

Interface Description:

	Parameter name	Effective value	Description
Input parameters	timeover		overtime time
Output parameters	pMsg		Message data
Return value			Read the results 0 successful

5.1.4.3 xgui_proc_default_msg

Interface prototype:

```
int xgui_proc_default_msg(PMESSAGE pMsg)
```

Function Description:

Let the system handle some default messages, such as shutdown, low battery voltage, etc.

Interface Description:

	Parameter name	Effective value	Description
Input parameters	pMsg		Message data
Output parameters			
Return value			0 unprocessed

5.1.5 dialog box

5.1.5.1 xgui_messagebox_show

Interface peototype：

```
intxgui_messagebox_show(char *title, char *msg , char* pszLeftOp, char* pszRightOp , inttimeover);
```

Function Description:

The dialog box is displayed

Interface Description:

	Parameter name	Effective value	Description
Input parameters	title		Dialog box title
Input parameters	msg		Dialog box contents
Input parameters	pszLeftOp		Left button
Input parameters	pszRightOp		Right button

Input parameters	timeover		overtime time
Output parameters	NO		
Return value			Operation result

5.1.6 image

5.1.6.1 xgui_load_bmp

Interface peototype:

```
char * xgui_load_bmp(char * filename , int *width , int *height);
```

Function Description:

Load monochrome bmp image into buffer

Interface Description:

	Parameter name	Effective value	Description
Input parameters	filename		Bmp file name
Output parameters	width		Bmp width
Output parameters	height		Bmp height
Return value			Buffer pointer, if 0 image read failed

5.1.6.2 xgui_out_bits

Interface peototype:

```
void xgui_out_bits(int x, int y, unsigned char *pbits, int width , int height,
int mode);
```

Function Description:

Load monochrome bmp image into buffer

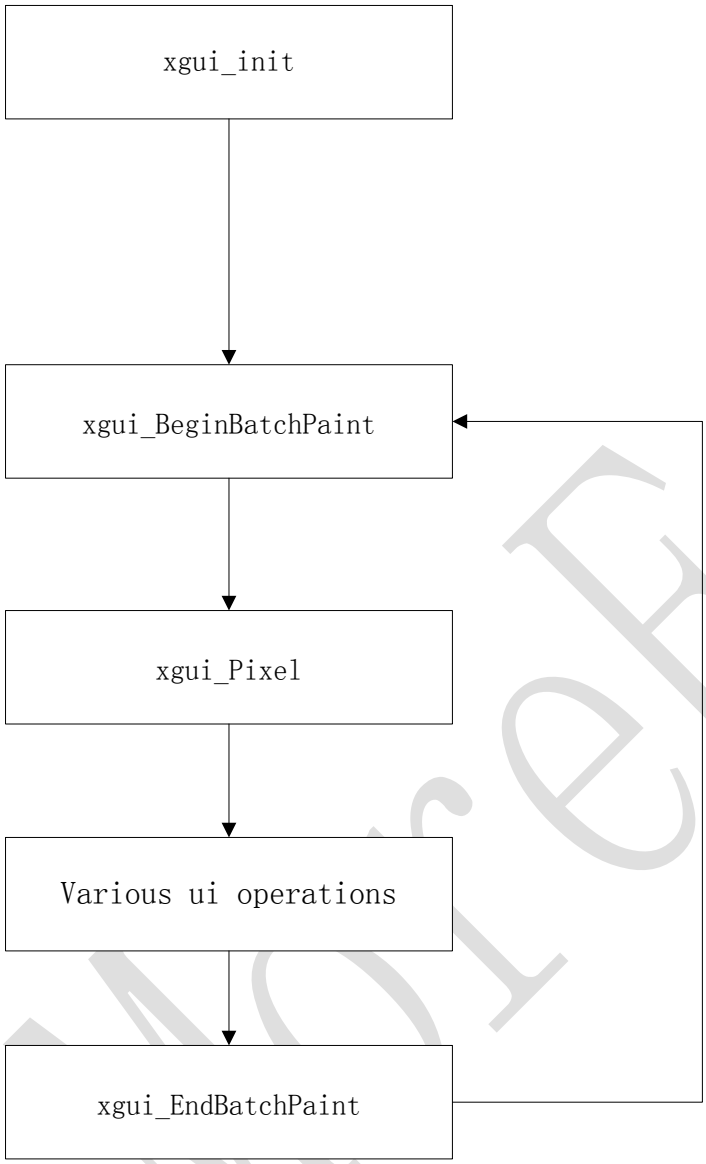
Interface Description:

	Parameter name	Effective value	Description
Input parameters	x		Display x coordinate
Input parameters	y		Display y coordinate
Input parameters	pbits		Bmp buffer
Input parameters	width		Bmp height
Input parameters	height		Bmp height
Input parameters	mode		Reserved set to 0
Return value			

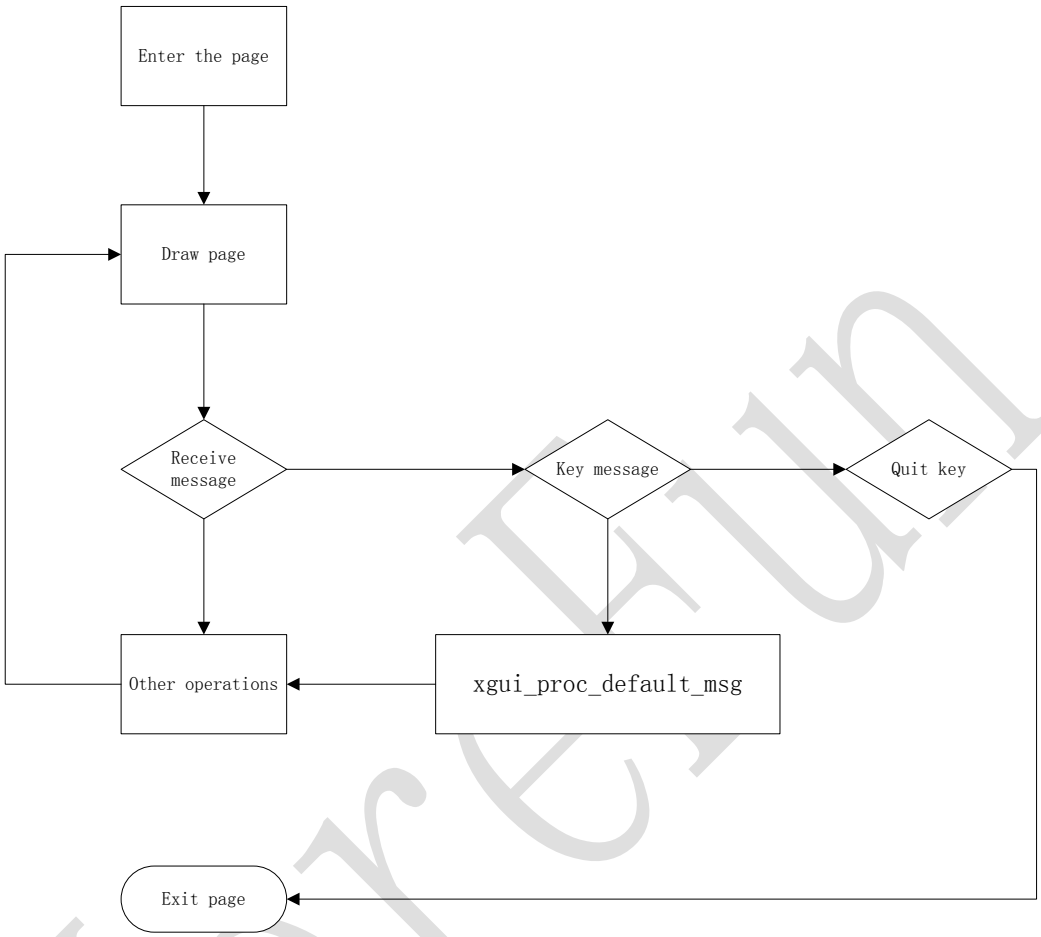
5.1.6.3 programming case

```
int logowidth;
int logoheight;
char * pbmp = xgui_load_bmp(TESTOIMG , &logowidth , &logoheight);
if (pbmp != 0){
        xgui_Clear_Win();
        xgui_out_bits(20, 10 ,pbmp , logowidth , logoheight , 0);
        free(pbmp);
}
```

5.2 Call the process



5.1 page the process



5.2 Programming examples

Please refer to xpos-sys-demo.

6 Print module

6.1 API Interface

6.1.1 **Formatting**

6.1.1.1 osl_print_cn_font_size

Interface prototype：

```
constchar *  osl_print_cn_font_size(intv);
```

Function Description:

Set the Chinese font

Interface Description:

	Parameter name	Effective value	Description
Input parameters	v		Font
Output parameters	no		
Return value			Format string

6.1.1.2 osl_print_en_font_size

Interface prototype：

```
constchar *  osl_print_en_font_size(intv);
```

Function Description:

Set the English font

Interface Description:

	Parameter name	Effective value	Description
Input parameters	v		Font
Output parameters	no		

Return value			Format string

6.1.1.3 osl_print_cn_font_zoom

Interface prototype:

```
constchar *  osl_print_cn_font_zoom(intw,inth);
```

Function Description:

Set Chinese to enlarge

Interface Description:

	Parameter name	Effective value	Description
Input parameters	W		Horizontal zoom
Input parameters	h		Vertical zoom
Output parameters	no		
Return value			Format string

6.1.1.4 osl_print_en_font_zoom

Interface prototype:

```
constchar *  osl_print_en_font_zoom(intw,inth);
```

Function Description:

Set English zoom

Interface Description:

	Parameter name	Effective value	Description
Input parameters	w		Horizontal zoom
Input parameters	h		Vertical zoom
Output parameters	no		

Return value			Format string

6.1.1.5 osl_print_line_space

Interface prototype：

```
constchar *  osl_print_line_space(intv);
```

Function Description:

Set line spacing

Interface Description:

	Parameter name	Effective value	Description
Input parameters	v		Line spacing
Output parameters	NO		
Return value			Format string

6.1.1.6 osl_print_col_space

Interface prototype：

```
constchar *  osl_print_col_space(intv);
```

Function Description:

Vertical paper

Interface Description:

	Parameter name	Effective value	Description
Input parameters	v		Rows
Output parameters	no		
Return value			Format string

6.1.1.7 osl_print_row_space

Interface prototype：

```
constchar *  osl_print_row_space(intv);
```

Function Description:

Horizontal paper

Interface Description:

	Parameter name	Effective value	Description
Input parameters	v		Rows
Output parameters	NO		
Return value			Format string

6.1.1.8 osl_print_align

Interface prototype：

```
constchar *  osl_print_align(intv);
```

Function Description:

Alignment

Interface Description:

	Parameter name	Effective value	Description
Input parameters	v	0-2	Alignment 0 left justified 1 in alignment 2 right justified
Output parameters	no		
Return value			Format string

6.1.1.9 osl_print_img

Interface prototype：

```
constchar *  osl_print_img(constchar *imgfile);
```

Function Description:

Print the picture

Interface Description:

	Parameter name	Effective value	Description
Input parameters	imgfile		Image path
Output parameters	no		
Return value			Format string

6.1.1.10 osl_print_heat_factor

Interface prototype：

```
constchar *  osl_print_heat_factor(intv)
```

Function Description:

heating time

Interface Description:

	Parameter name	Effective value	Description
Input parameters	v		heating time
Output parameters	no		
Return value			Format string

6.1.2 **Printout**

6.1.2.1 osl_print_add

Interface prototype：

```
voidosl_print_add(constchar * pbuff);
```

Function Description:

Add print content to the buffer

Interface Description:

	Parameter name	Effective value	Description
Input parameters	pbuff		Print content
Output parameters	no		
Return value			

6.1.2.2 osl_print_get

Interface prototype：

```
char * osl_print_get()
```

Function Description:

Get print content

Interface Description:

	Parameter name	Effective value	Description
Input parameters	v		Line spacing
Output parameters	no		
Return value			Print content

6.1.2.3 osl_print_write

Interface prototype：

```
intosl_print_write(char * data);
```

Function Description:

Print output

Interface Description:

	Parameter name	Effective value	Description
Input parameters	Data		Print content
Output parameters	no		
Return value			0 successful

6.1.2.4 osl_print_free

Interface prototype：

```
voidosl_print_free();
```

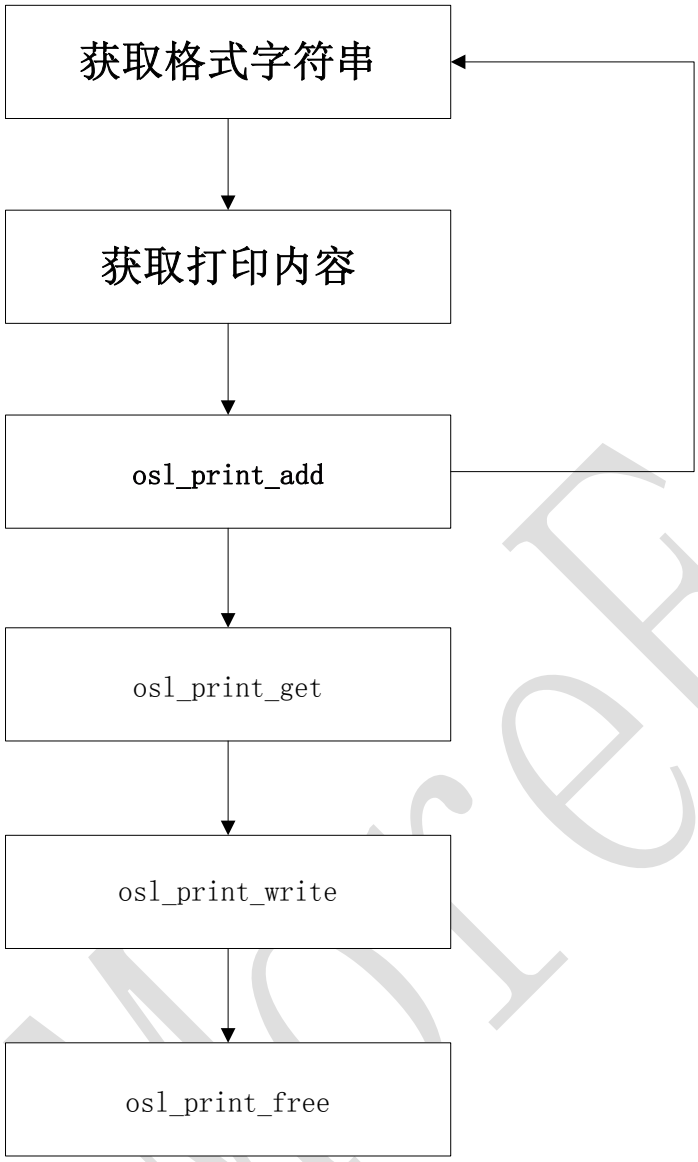
Function Description:

Print buffer is released

Interface Description:

	Parameter name	Effective value	Description
Input parameters	v		Line spacing
Output parameters	no		
Return value			Format string

6.2 Process



6.3 Programming examples

Please refer to xpos-sys-demo。

7 Timer Module

7.1 API Interface

7.1.1 osl_TimerInit

Interface prototype:

```
unsignedintosl_TimerInit(void)
```

Function Description:

Initialize timer management

Interface Description:

	Parameter name	Effective value	Description
Input parameters	no		
Output parameters	no		
Return value			

7.1.2 osl_TimerCreate

Interface prototype:

```
unsignedintosl_TimerCreate(TIMERFUNCTIONpWorkFunction, void* pParam,
unsignedintnPeriod, unsignedintnMode, unsignedint *pnErrorCode);
```

Function Description:

Create a timer

Interface Description:

	Parameter name	Effective value	Description

Input parameters	pWorkFunction		Processing function
Input parameters	pParam		parameter
Input parameters	nPeriod		Timing cycle
Input parameters	nMode		Keep it
Output parameters	pnErrorCode		error code
Return value			0 failedOther timer handles

7.1.3 osl_TimerEnable

Interface prototype:

```
unsignedintosl_TimerEnable (unsignedintnTimerNo)
```

Function Description:

Enable the specified timer

Interface Description:

	Parameter name	Effective value	Description
Input parameters	nTimerNo		定时器句柄
Output parameters	no		
Return value			0 success

7.1.4 osl_TimerDisable

Interface prototype：

```
unsignedintosl_TimerDisable (unsignedintnTimerNo)
```

Function Description:

Stop the specified timer

Interface Description:

	Parameter name	Effective value	Description
Input parameters	nTimerNo		Timer handle
Output parameters	no		
Return value			0 success

7.2 Programming examples

Please refer to xpos-sys-demo.

8 OS Layer Module

8.1 API Interface

8.1.1 osl_set_language

Interface prototype：

```
void osl_set_language(int val);
```

Function Description:

Set the common module display language

Interface Description:

	Parameter name	Effective value	Description
Input parameters	val		0 Chinese 1 English
Output parameters	no		
Return value			

8.1.2 osl_Sleep

Interface prototype:

```
void osl_Sleep(int ms);
```

Function Description:

Thread delay

Interface Description:

	Parameter name	Effective value	Description
Input parameters	ms		Thread delay milliseconds
Output parameters	no		
Return value			

8.1.3 osl_CheckTimeover

Interface prototype:

```
int osl_CheckTimeover(unsigned int tick1 , unsigned int timeover);
```

Function Description:

Check for timeout

Interface Description:

	Parameter name	Effective value	Description
Input parameters	Tick1		Start tick
Output parameters	timeover		overtime time
Return value			1 timeout

8.1.4 osl_GetTick

Interface prototype:

```
unsigned int osl_GetTick(void)
```

Function Description:

Get current tick

Interface Description:

	Parameter name	Effective value	Description
Input parameters	no		
Output parameters	No		
Return value			current tick

8.1.5 osl_getTerminalID

Interface prototype:

```
const char* osl_getTerminalID( char *outdata ,int nMaxLen )
```

Function Description:

Get Terminal ID

Interface Description:

	Parameter name	Effective value	Description
Input parameters	nMaxLen		Buff size
Output parameters	outdata		Terminal ID buff
Return value			Terminal ID

8.1.6 osl_scaner_open

Interface prototype:

```
int osl_scaner_open();
```

Function Description:

Open the scanning device

Interface Description:

	Parameter name	Effective value	Description
Input parameters			
Output parameters			

Return value			0 successful

8.1.7 osl_scaner_close

Interface prototype:

```
int osl_scaner_close();
```

Function Description:

Close the scanning device

Interface Description:

	Parameter name	Effective value	Description
Input parameters			
Output parameters			
Return value			0 successful

8.1.8 osl_scaner_get

Interface prototype:

```
int osl_scaner_get( char *outbuff,int maxlen ,int timeout);
```

Function Description:

Get the barcode, read it and exit immediately, can't read the wait timeout and return

Interface Description:

	Parameter name	Effective value	Description
Input parameters	maxlen		Barcode buffer maximum length

Input parameters	timeout		overtime time
Output parameters	outbuff		Barcode buffer
Return value			Barcode length, less than 0 failed

9 File system module

9.1 API Interface

9.1.1 mf_file_open

Interface prototype:

```
intmf_file_open(constchar * name , intflag , intmode);
```

Function Description:

open a file

Interface Description:

	Parameter name	Effective value	Description
Input parameters	name		file name
Input parameters	flag		Open the logo
Input parameters	mode		Open mode

Output parameters	no		
Return value			Less than 0 failed Other file handle

9.1.2 mf_file_lseek

Interface prototype:

```
intmf_file_lseek(intfp , intpos , intflag)
```

Function Description:

File location

Interface Description:

	Parameter name	Effective value	Description
Input parameters	fp		Handle
Input parameters	pos		Offset
Input parameters	flag		Logo
Output parameters	no		
Return value			Move position

9.1.3 mf_file_write

Interface prototype:

```
intmf_file_write(intfp ,constvoid * buff , unsignedintsize)
```

Function Description:

Write the file

Interface Description:

	Parameter name	Effective value	Description
Input parameters	fp		Handle
Input parameters	Buff		Write to the buffer
Input parameters	Size		Write length
Output parameters	no		
Return value			Successful write length

9.1.4 mf_file_read

Interface prototype:

```
intmf_file_read(intfp , void * buff , unsignedintsize)
```

Function Description:

Read the file

Interface Description:

	Parameter name	Effective value	Description
Input parameters	fp		Handle

Input parameters	Buff		Read buffer
Input parameters	flag		Read length
Output parameters	no		
Return value			Successfully read the length

9.1.5 mf_file_close

Interface prototype：

```
intmf_file_close(intfp )
```

Function Description:

Close the file

Interface Description:

	Parameter name	Effective value	Description
Input parameters	fp		Handle
Output parameters	no		
Return value			Keep it

9.1.6 mf_file_unlink

Interface prototype：

```
intmf_file_unlink(constchar * name)
```

Function Description:

Delete Files

Interface Description:

	Parameter name	Effective value	Description
Input parameters	name		file name
Output parameters	no		
Return value		0	0 successful

9.2 Programming examples

Please refer to xpos-sys-demo。

9.3 Programming examples

Please refer to xpos-sys-demo。

10 Communication module

10.1API Interface prototype

10.1.1 GPRS

10.1.1.1 atc_isreg

Interface prototype：

```
intatc_isreg();
```

Function Description:

To read the module whether to register the network

Interface Description:

	Parameter name	Effective value	Description
Input parameters			
Output parameters			
Return value			0 is not registered 1 is registered

10.1.1.2 atc_csq

Interface prototype:

```
intatc_csq()
```

Function Description:

Read the csq value

Interface Description:

	Parameter name	Effective value	Description
Input parameters			
Output parameters			
Return value			csq value

10.1.1.3 atc_imei

Interface prototype:

```
constchar * atc_imei()
```

Function Description:

Read imei

Interface Description:

	Parameter name	Effective value	Description
Input parameters			
Output parameters			
Return value			Imei

10.1.1.4 atc_imsi

Interface prototype:

```
constchar * atc_imsi()
```

Function Description:

Read imsi

Interface Description:

	Parameter name	Effective value	Description
Input parameters			
Output parameters			
Return value			Imsi

10.1.1.5 atc_signal

Interface prototype:

```
intatc_signal()
```

Function Description:

Read the signal

Interface Description:

	Parameter name	Effective value	Description
Input parameters			
Output parameters			
Return value			Signal value (1-4)

10.1.1.6 atc_model_ver

Interface prototype:

```
constchar * atc_model_ver()
```

Function Description:

Read the module version

Interface Description:

	Parameter name	Effective value	Description
Input parameters			
Output parameters			
Return value			Module version

10.1.2 WIFI

10.1.2.1 wifi_get_link_state

Interface prototype:

```
intwifi_get_link_state();
```

Function Description:

Read wifi connection status

Interface Description:

	Parameter name	Effective value	Description
Input parameters			
Output parameters			
Return value			1 is connected ap 2 is not connected

10.1.2.2 wifi_get_signal

Interface prototype:

```
intwifi_get_signal()
```

Function Description:

Read the number of signals

Interface Description:

	Parameter name	Effective value	Description
Input parameters			

Output parameters			
Return value			Wi-Fi signal (1-4)

10.1.2.3 wifi_model_ver

Interface prototype:

```
char * wifi_model_ver()
```

Function Description:

Read the wifi module version

Interface Description:

	Parameter name	Effective value	Description
Input parameters			
Output parameters			
Return value			Wifi module version

10.1.2.4 wifi_get_ssid

Interface prototype:

```
char * wifi_get_ssid()
```

Function Description:

Read wifi ssid

Interface Description:

	Parameter name	Effective value	Description
Input parameters			

Output parameters			
Return value			ssid

10.1.2.5 wifi_get_ap_mac

Interface prototype:

```
char * wifi_get_ap_mac()
```

Function Description:

Read ap mac

Interface Description:

	Parameter name	Effective value	Description
Input parameters			
Output parameters			
Return value			Ap mac

10.1.2.6 wifi_get_local_mac

Interface prototype:

```
char * wifi_get_local_mac()
```

Function Description:

Native mac

Interface Description:

	Parameter name	Effective value	Description
Input parameters			

Output parameters			
Return value			Mac

10.1.2.7 wifi_get_local_ip

Interface prototype:

```
char * wifi_get_local_ip()
```

Function Description:

Native ip

Interface Description:

	Parameter name	Effective value	Description
Input parameters			
Output parameters			
Return value			Ip

11 QR Module

11.1API Interface

11.1.1 mfGeneCodePic

Interface prototype:

```
int mfGeneCodePic(char * chData, int iLen, Param_QR_INFO *QRParam , char * bitmap);
```

Function Description:

Generate qr code

Interface Description:

	Parameter name	Effective value	Description
Input parameters	chData		Bar code input such as http://www.qr.com
Input parameters	QRParam		**nVersion;** version number: 1~40 **nLevel;** error correction level: 0 - low, 1- medium, 2- high, 3- highest, **moudleWidth;** module width (unit: pixel)
Output parameters	bitmap		QR code image data
Return value			

11.1.2 osl_TimerCreate

Interface prototype:

```
unsignedintosl_TimerCreate(TIMERFUNCTIONpWorkFunction, void* pParam,
unsignedintnPeriod, unsignedintnMode, unsignedint *pnErrorCode);
```

Function Description:

Create a timer

Interface Description:

	Parameter name	Effective value	Description
Input parameters	pWorkFunction		Processing function

Input parameters	pParam		parameter
Input parameters	nPeriod		Timing cycle
Input parameters	nMode		Keep it
Output parameters	pnErrorCode		error code
Return value			0 failedOther timer handles

12 Security Module

12.1API Interface

12.1.1 DUKPT

12.1.1.1 dukpt_init

Interface prototype：

```
void dukpt_init()
```

Function Description:

Dukpt initialization

Interface Description:

	Paramet	Effec	Description

	er name	tive value	
Input parameters			
Input parameters			
Output parameters			
Return value			

12.1.1.2 dukpt_load_init_key

Interface prototype:

```
int dukpt_load_init_key(unsigned char gid, unsigned char* init_ksn, unsigned char* init_key);
```

Function Description:

Set the initial key

Interface Description:

	Paramet er name	Effec tive value	Description
Input parameters	Gid	0	Key grouping
Input parameters	init_ksn		10 bytes of initial ksn
Input parameters	init_key		16 bytes of initial key

Output parameters			
Return value			0 successful

12.1.1.3 dukpt_get_key

Interface prototype:

```
int dukpt_get_key(unsigned char gid, unsigned char* key, unsigned char * ksn);
```

Function Description:

Export a key

Interface Description:

	Parameter name	Effective value	Description
Input parameters	Gid	0	Key grouping
Input parameters			
output parameters	key		16 bytes of key
Output parameters	ksn		10 bytes of key
Return value			0 successful

13 UCOS-II Module

UCOS interface as the standard interface, the description of the relevant interface see "uCOS-II-RefMan.pdf" document.

14 Standard C library interface support

14.1 Heap allocation

The total size of the heap depends on the configuration of the link script. Use the standard C library operating interface: malloc, free.

15 Power management module

System components implement a mechanism for battery management. As long as the initialization time to call the corresponding initialization interface can be. Refer to the "Typical System Initialization Flow" section.

15.1 API Interface

15.1.1 setbacklightflag

Interface prototype:

```
voidsetbacklightflag(intflag);
```

Function Description:

Sets whether or not to turn off the backlight automatically

Interface Description:

	Parameter name	Effective value	Description
Input parameters	flag	0-1	0 does not turn off automatically 1 is automatically turned off
Return value			0 success

16 key module

The key handling has been integrated into the XGUI. Please refer to xpos-sys-demo for specific use.