BK7251 RTOS SDK

API Reference

Better Life with Wireless
美好生活尽在无线

Version 1.0.0

Release Notes

Version	Date	Description
V1.0	2019.04	First Release
V1.1	2019.07	Modified about bootloader

Table Of Contents

1 ADC

1.1 ADC Introduction

BK7251 has multi-channel general purpose ADC and supports 10-16 bits output with internal decimation filter. Single, continuously and software read mode are supported. The range of voltage detection is 0 ~ 2.4V. The ADC channels are as follows:

channel	description
0	vbat voltage, the detected voltage value is 1/2 of vbat voltage
1	gpio4 voltage
2	gpio5 voltage
3	gpio23 voltage (multiplexing with JTAG pins)
4	gpio2 voltage
5	gpio3 voltage
6	gpio12 voltage
7	gpio13 voltage

Note: ADC channel 3 is multiplexed with JTAG, if you want to use this pin ,you must shut off the JTAG mode. The ADC channels correspond to GPIO mainly with the abo ve tables.

1.2 ADC Related API

ADC related APIs refer to \beken378\func\saradc_intf.h, APIs are as follows:

function	description
saradc_work_create()	creat adc work thread
adc_obj_init()	configure adc channel and callback
adc_obj_start()	start adc detection
adc_obj_stop()	stop adc detection

1.2.1 creat ADC work thread

```
void saradc_work_create(UINT32 scan_interval_ms);
```

parameters	description
scan_interval_ms	adc scan interval
return	null

1.2.2 configure ADC channel and callback

```
void adc_obj_init(ADC_OBJ* handle, adc_obj_callback cb, UINT32 channel, void *user_data);
```

parameters	description
handle	the structure of adc channel,include channel number and callback
adc_obj_callback_cb	callback function after reading voltage value, processing read results. callback has 2 parameters: 1.new_mv : voltage value 2.user_data: user data
channel	channel
user_data	user data
return	null

1.2.3 start ADC

```
int adc_obj_start(ADC_OBJ* handle);
```

parameters	description
handle	the structure of adc channel
return	0:success ; -1:fail

1.2.4 stop ADC

```
int adc_obj_stop(ADC_OBJ* handle);
```

parameters	description
handle	the structure of adc channel
return	0:success ; -1:fail

1.3 ADC Struct Reference

ADC_OBJ：adc object

user_data	user data
channel	adc channel
cb	callback
next	point to the next adc object

1.4 ADC Sample Code

ADC sample code refers to test\adc_test.c.After opening the macro CONFIG_

ADC_TEST, you can see the print information of battery power. After input cmd adc_channel_test <start> <channel>，and adjusting the voltage of the corresponding channel, you can see the voltage value also changing correspondingly. The voltage value detected by channel 0 needs to be multiplied by 2, which is the actual voltage value .

```
/*
*program list：  This is a sample coed of ADC. After opening the macro CONFIG_ADC_TEST and
* input cmd ,you can see voltage print information.
* command format：  start test：adc_channel_test start channel
*                             stop test：adc_channel_test stop
* program function：After inputing the command,you can see the voltage values measured by the
*corresponding ADC channel printed
*/

#include "include.h"
#include "arm_arch.h"
#include "error.h"
#include "include.h"
#include <rthw.h>
#include <rtthread.h>
#include <rtdevice.h>
#include <stdint.h>
#include <stdlib.h>
#include <finsh.h>
#include <rtdef.h>
#include "saradc_intf.h"
#include "sys_ctrl_pub.h"

#define CONFIG_ADC_TEST
#ifdef  CONFIG_ADC_TEST
static ADC_OBJ test_adc;

/****channel 1 - 7***/
static void adc_detect_callback(int new_mv, void *user_data)
{
        static int cnt = 0;
        test_adc.user_data = (void*)new_mv;

        if(cnt++ >= 100)
        {
                cnt = 0;
        rt_kprintf("adc channel%d voltage:%d,%x\r\n",test_adc.channel,new_mv,test_adc.user_data);
        }
}



void adc_channel_test(int argc,char *argv[])
{
        int channel;

        if (strcmp(argv[1], "start") == 0)
        {
                if(argc == 3)
                {
                        channel = atoi(argv[2]);
                        rt_kprintf("---adc channel:%d---\r\n",channel);
                        saradc_work_create(20);
                        adc_obj_init(&test_adc, adc_detect_callback, channel, &test_adc);
                        adc_obj_start(&test_adc);
                }
                else
                {
                        rt_kprintf("input param error\r\n");
                }
        }
        if(strcmp(argv[1], "stop") == 0)
        {
                adc_obj_stop(&test_adc);
        }
}

MSH_CMD_EXPORT(adc_channel_test,adc test);
#endif
```

2 PWM

2.1 PWM Introduction

BK7251 has six 32-bit PWM outputs. The PWM running clock can be either high speed clock or low power clock. Each PWM runs independently with its own duty cycle.

channel	description
0	gpio6
1	gpio7
2	gpio8
3	gpio9
4	gpio24
5	gpio26

Note: Pwm0 can not been used for pwm channel,because it has been used for system timer.

2.2 PWM Related API

Pwm related APIs refer to \beken378\func\user_driver\BkDriverPwm.h, APIs are as follows:

function	description
bk_pwm_initialize()	initialize PWM
bk_pwm_start()	start PWM
bk_pwm_stop()	stop PWM

2.2.1 pwm initialization

```
OSStatus  bk_pwm_initialize(bk_pwm_t pwm, uint32_t cycle, uint32_t duty_cycle);
```

parameters	description
pwm	pwm channel: 0 ~ 5
cycle	set pwm cycle
duty_cycle	set pwm duty cycle
return	0:success ; -1: fail

2.2.2 start pwm

```
OSStatus bk_pwm_start(bk_pwm_t pwm)；
```

parameters	description

pwm	pwm channel: 0 ~ 5
return	0:success ; -1: fail

2.2.3 stop pwm

```
OStatus bk_pwm_stop(bk_pwm_t pwm)；
```

parameters	**description**
pwm	pwm channel: 0 ~ 5
return	0:success ; -1: fail

2.3 PWM Enumeration

bk_pwm_t：

BK_PWM_0	pwm0
BK_PWM_1	pwm1
BK_PWM_2	pwm2
BK_PWM_3	pwm3
BK_PWM_4	pwm4
BK_PWM_5	pwm5

2.4 PWM Sample Code

Pwm sample code refers to test\pwm_test.c.After opening macro CONFIG_PWM_TEST and input command : pwm_test <channel> <duty_cycle> <cycle>, the pwm waveform can be detected on the corresponding pin.

```
/*
*program list：This is a sample coed of pwm. After opening the macro CONFIG_PWM_TEST,start test
  mode
* command format:   pwm_test   1   8000   16000
* program function：After inputing the command,you can detected pwm waveform on corresponding
* pin
*/
#include "rtos_pub.h"
#include "BkDriverPwm.h"
#include "pwm_pub.h"
#include "error.h"
#include <stdint.h>
#include <stdlib.h>
#include <finsh.h>
```

```
#define           CONFIG_PWM_TEST
#ifdef            CONFIG_PWM_TEST

static void pwm_test(int argc,char *argv[])
{
        UINT32 channel,duty_cycle,cycle;
        if(argc != 4)
                return;
        channel       = atoi(argv[1]);
        duty_cycle    = atoi(argv[2]);
        cycle         = atoi(argv[3]);
        if(cycle < duty_cycle)
        {
                rt_kprintf("pwm param error: end < duty\r\n");
                return;
        }
        rt_kprintf("---pwm %d test--- \r\n",channel);
        bk_pwm_initialize(channel, cycle, duty_cycle);      /*pwm initialization，set dutycycle*/
        bk_pwm_start(channel);                              /*start pwm */

        rt_thread_delay(100);

        bk_pwm_stop(channel);                               /*stop pwm */
        rt_kprintf("---pwm test stop---\r\n");
}

MSH_CMD_EXPORT(pwm_test,pwm test);
#endif
```

3 Audio

3.1 Audio Introduction

BK7251 supports audio playback and recording function. It collects audio data by microphone and outputs sound via dac.

3.2 Audio Related API

Audio related APIs refer to \driver\audio_device.h, APIs are as follows：

function	description
audio_device_init()	find sound and mic device
audio_device_mic_open()	open mic
audio_device_mic_set_channel()	set adc channel
audio_device_mic_set_rate ()	set adc sampling rate
audio_device_mic_read()	mic acquisition of sound data
audio_device_mic_close()	close mic device
audio_device_open()	open dac
audio_device_set_rate()	set dac sampling rate
audio_device_set_volume()	set dac volume: 0 ~ 16
audio_device_write()	audio play audio data
audio_device_close()	close dac

3.2.1 audio device initialization

Find“sound” and “mic” device .

```
int audio_device_init(void)
```

parameters	description
void	null
return	RT_EOK:success ; others: fail

3.2.2 open mic

Open mic device , set mode: read only.

```
void audio_device_mic_open(void)；
```

parameters	description
void	null
return	null

3.2.3 set mic data acquisition channel

```
void audio_device_mic_set_channel(int channel)；
```

parameters	description
channel	adc channel
return	null

3.2.4 set mic sampling rate

The sampling rate has 48k/44.1k/16k/8k.

```
void audio_device_mic_set_rate(int sample_rate);
```

parameters	description
sample_rate	adc channel
return	null

3.2.5 acquisition of sound data from mic

```
int audio_device_mic_read(void *buffer, int size)；
```

parameters	description
buffer	mic read buffer
size	the size of buffer
return	length: return the length of read data

3.2.6 close mic

```
void audio_device_mic_close(void);
```

parameters	description
void	null
return	null

3.2.7 open audio device

```
void audio_device_open(void)；
```

parameters	description
void	null

return	null

3.2.8 set dac sampling rate

The sampling rate has 48k/44.1k/16k/8k.

```
void audio_device_set_rate(int sample_rate);
```

parameters	**description**
sample_rate	sampling rate
return	null

3.2.9 set dac volume

```
void audio_device_set_volume(int volume);
```

parameters	**description**
volume	volume
return	null

3.2.10 write data to dac

```
void audio_device_write(void *buffer, int size);
```

parameters	**description**
buffer	dac buffer
size	the size of dac buffer
return	null

3.2.11 close dac

```
void audio_device_close(void);
```

parameters	**description**
void	null
return	null

3.3 Audio Struct Reference

audio_device:

struct rt_device *snd	sound handle
struct rt_device *mic	mic handle

struct rt_mempool mp	memory pool
state	audio state
void (*evt_handler)(void *parameter, int state)	interrupt process
parameter	audio parameter

3.4 Audio Macros

#define	**TEST_BUFF_LEN**	60*1024	/* the size of buffer */
#define	**READ_SIZE**	1024	/*the size of read buffer***/**

3.5 Audio Sample Code

Audio sample code refers to test\mic_record.c, After inputing command: record_and_play, you can hear the sound recorded from mic.Openning macro definitions : MICPHONE_TEST and closing macro definitions : CONFIG_SOUND_MIXER to start test.

```
/*
* program list：This is sample for recording and audio playback.
* command format:   record_and_play.
* program function：The device palys audio after recording data from mic.
*/
#include <rtthread.h>
#include <rtdevice.h>
#include <finsh.h>

#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include "board.h"
#include "audio_device.h"

#define MICPHONE_TEST
#ifdef MICPHONE_TEST

#define TEST_BUFF_LEN 60*1024
#define READ_SIZE 1024

static uint8_t *test_buf;

void record_and_play(int argc,char *argv[])
{
        int mic_read_len = 0;
        int actual_len,i;
        int dac_wr_len=0;
        uint16_t *buffer = NULL;

        int vad_on;

#if CONFIG_SOUND_MIXER
       mixer_pause();
#endif

       vad_on = atoi(argv[1]);

        test_buf = sdram_malloc(TEST_BUFF_LEN);
        if(test_buf == NULL)
        {
               rt_kprintf("===not enough memory===\r\n");
               return;
        }

        audio_device_init();                                        /*sound mic device init*/

        audio_device_mic_open();                                    /*open mic*/
        audio_device_mic_set_channel(1);                            /*set adc channel*/
        audio_device_mic_set_rate(16000);                           /*set adc sampling rate*/

        if (vad_on)
        {
              rt_kprintf("Vad is ON !!!!!!!!\r\n");                 /*enter into vad detection*/
               wb_vad_enter();
        }

        while(1)
        {
              if (vad_on)
                     rt_thread_delay(5);
               else
                      rt_thread_delay(20);

               int chunk_size = wb_vad_get_frame_len();//320
               char *val = NULL;
               if(mic_read_len > TEST_BUFF_LEN - READ_SIZE)
                      break;

               if (!vad_on)
               {
                     actual_len = audio_device_mic_read(test_buf+mic_read_len,READ_SIZE);
               }
               else
               {
                   /*mic collect data*/
                   actual_len = audio_device_mic_read(test_buf+mic_read_len,chunk_size);
                   if(wb_vad_entry(test_buf+mic_read_len, actual_len))
                     {
                           rt_kprintf("Vad Detected !!!!!!!!\r\n");                    /*detected voice*/
                            break;
                     }
               }

               mic_read_len += actual_len;
        }

        if (vad_on)
        {
               wb_vad_deinit();                                                        /* close vad detection */
        }

        rt_kprintf("mic_read_len is %d\r\n", mic_read_len);
        audio_device_mic_close();                                                       /*close mic*/

        audio_device_open();                                                            /*open dac*/
        audio_device_set_rate(8000);                                                   /*set dac sampling rate */

        while(1)
        {
               buffer = (uint16_t *)audio_device_get_buffer(RT_NULL);
               if(dac_wr_len >= mic_read_len)
               {
                      audio_device_put_buffer(buffer);
                      break;
                }

                memcpy(buffer,test_buf+dac_wr_len,READ_SIZE);
                dac_wr_len += READ_SIZE;

              audio_device_write((uint8_t *)buffer, READ_SIZE);               /*dac play audio*/
        }
        audio_device_close();                                                  /*close dac*/

        if(test_buf)
                sdram_free(test_buf);                                          /*free ram*/

#if CONFIG_SOUND_MIXER
       mixer_replay();
#endif
}
MSH_CMD_EXPORT(record_and_play, record play);
#endif
```

4 Airkiss

4.1 Airkiss Introduction

Airkiss is a WiFi device fast access configuration technology provided by the hardware platform of wechat. To configure the device access network by using the way of wechat client, it is necessary for the device to support airkiss technolog.

4.2 Airkiss Related API

Airkiss related APIs refer to \samples\airkiss\airkiss.h ,APIs are as follows：

function	description
airkiss()	start airkiss
airkiss_get_status()	get aikiss status
airkiss_get_result()	after airkiss_recv()return AIRKISS_STATUS_COMPLETE, get airkiss result.

4.2.1 start airkiss

```
int airkiss(void);
```

parameters	description
void	null
return	0:success ; -1:fail

4.2.2 get airkiss status

```
uint32_t airkiss_get_status(void);
```

parameters	description
void	null
return	airkiss status

4.2.3 get airkiss result

```
int airkiss_get_result(airkiss_context_t *context, airkiss_result_t *result);
```

parameters	description
airkiss_context_t* context,	memory allocated for the airkiss library
airkiss_result_t *result	result of airkiss decoding
return	0:success ; others:fail

4.3 Airkiss Struct Reference

airkiss_result_t：airkiss result

char *pwd	wifi password
char *ssid	wifi ssid
unsigned char pwd_length	length of wifi password
unsigned char ssid_length	length of wifi password ssid
unsigned char random	random value :send this data to 10000 through UDP when connect wifi
unsigned char reserved	reserved

4.4 Airkiss Enumeration

airkiss_status_t：

```
typedef enum
{
    /* The decoding is normal and no special processing is required. Continue calling airkiss_recv()
      until the decoding is successful */
    AIRKISS_STATUS_CONTINUE = 0,

    /* Wifi channel has been locked, the upper layer should stop switching channel immediately */
    AIRKISS_STATUS_CHANNEL_LOCKED = 1,

    /* decode success，call airkiss_get_result to get result */
    AIRKISS_STATUS_COMPLETE = 2

} airkiss_status_t;
```

4.5 Airkiss Sample Code

Airkiss sample code refers to \samples\airkiss\airkiss.c. The code creat airkiss thread and lock channel after connecting net in wechat platform. Input command: start_airkiss. The log information show whether the distribution network is successful after connected net in wechat platform.

```
/*
*program list：This is sample for airkiss airkiss
* command format:   start_airkiss
* program function：connectingdevices to networks via wechat platform
*/


#include <rtthread.h>
```

```
#include <rtdevice.h>
#include <rthw.h>
#include <wlan_dev.h>
#include <wlan_mgnt.h>
#include "airkiss.h"
#include "bk_rtos_pub.h"
#include <stdio.h>
#include <sys/socket.h>
#include "error.h"

int start_airkiss(int argc, char *argv[])
{
	airkiss_result_t *result;
	if(1 == airkiss())
		rt_kprintf("airkiss start\r\n");
	else
		rt_kprintf("airkiss fail\r\n");

	while(airkiss_get_status() != AIRKISS_STATUS_COMPLETE)
	{
		rt_thread_delay(rt_tick_from_millisecond(100));
	}

	result = airkiss_result_get();

	rt_kprintf("---ssid:%s , key:%s---\r\n", result->ssid,result->pwd);
}
#ifdef FINSH_USING_MSH
#include "finsh.h"

MSH_CMD_EXPORT(start_airkiss, start_ariksss);
#endif
```

5 Voice Cofigure Network

5.1 Voice Cofigure Network Introduction

Sound files must be in 6 bit, 48 kHz, 1 channel wav/pcm forma. BK7251 connects to the network by identifying files in this format.

5.2 Voice Cofigure Related API

voice cofigure network related APIs refer to samples\voice_config\include\voice_config.h, APIs are as follows：

function	description
voice_config_work()	open device
voice_config_stop()	user stop cofigure network
voice_config_version()	get version

5.2.1 start voice configure network

```
int voice_config_work(void *device,
                      uint32_t sample_rate,
                      uint32_t timeout,
                      struct voice_config_result *result)
```

parameters	description
device	record device
sample_rate	sampling rate(16000)
timeout	time out
result	voice recognition result
return	0:success ; others:fail

5.2.2 stop voice configure network

```
void voice_config_stop(void)
```

parameters	description
void	null
return	null

5.2.3 get version

```
const char *voice_config_version(void)
```

parameters	description
void	null
return	version

5.3 Voice Cofigure Network Struct Reference

voice_config_result：	
uint32_t ssid_len	ssid length
uint32_t passwd_len	password length
uint32_t custom_len	customer define length
char ssid[32+1]	ssid
char passwd[63+1]	password
char custom[16+1]	customer define data

5.4 Voice Cofigure Network Macros

#define	**SAMPLE_USING_VOICE_CONFIG**	open voice configure network mode

5.5 Voice Cofigure Network Sample Code

Voice configure network sample code refers to \test\samples\voice_config\ \voice_config.c. Openning macro definitions : VOICE_CONFIG_TEST to start test.

```
/*
*program list：This is a sample for voice configure network. Inputing command to let board get
*voice file which must be created from a tool. If succeeds, it prints the successful message.
* command format:   voice_config
* program function: playing voice file on pc, the demo board connect to the Internet by recognizing the
  sound played by pc.
  */
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdint.h>

#include <rtthread.h>
#include <rtdevice.h>
#include <rthw.h>
#include "bk_rtos_pub.h"
#include "error.h"
#include "voice_config.h"
```

```
/************************ voice config start ***************************/
static unsigned char voice_config_ssid[32 + 1] = {0};
static unsigned char voice_config_password[64 + 1] = {0};
int voice_config(int argc, char *argv[])
{
    if (tid)
    {
        rt_kprintf("voice config already init.\n");
        return -1;
    }

    tid = rt_thread_create("voice_config",
                           cmd_voice_config_thread,
                           RT_NULL,
                           1024 * 6,
                           20,
                           10);

    if (tid != RT_NULL)
    {
        rt_thread_startup(tid);
    }

    return 0;
}
static rt_thread_t tid = RT_NULL;
static void cmd_voice_config_thread(void *parameter)
{
    rt_device_t device = 0;
    struct voice_config_result result={0};
    int res;
    DEBUG_PRINTF("voice config version: %s\r\n", voice_config_version());     //get voice config
version

    /* open audio device and set tx done call back */
    device = rt_device_find("mic");
    if (device == RT_NULL)
    {
        DEBUG_PRINTF("audio device not found!\r\n");
        goto _err;
    }
    codec_device_lock();
    res = rt_device_open(device, RT_DEVICE_OFLAG_RDWR);

    /* set samplerate */
    if (RT_EOK == res)
    {
        int SamplesPerSec = SAMPLE_RATE;
        if (rt_device_control(device, CODEC_CMD_SAMPLERATE, &SamplesPerSec)
                != RT_EOK)
        {
            rt_kprintf("[record] audio device doesn't support this sample rate: %d\r\n",
                       SamplesPerSec);
            goto _err;
        }
    }
     else
     {
         goto _err;
     }
    rt_device_write(device, 0, 0, 100);                                     // start to record
    res = voice_config_work(device, SAMPLE_RATE, 1000 * 60 * 1, &result);     //start voice
configure
    if(res == 0)
    {
        rt_kprintf("ssid len=%d, [%s]\n", result.ssid_len, result.ssid);
        rt_kprintf("passwd L=%d, [%s]\n", result.passwd_len, result.passwd);
        rt_kprintf("custom L=%d, [%s]\n", result.custom_len, result.custom);

          station_connect(result.ssid,result.passwd);                     //connect station
    }
    else
    {
        rt_kprintf("voice_config res:%d\n", res);
    }

_err:
    if (device)
    {
        rt_device_close(device);                                        //close device
        codec_device_unlock();
    }
    tid = RT_NULL;

    return;
}

#ifdef FINSH_USING_MSH
#include "finsh.h"
MSH_CMD_EXPORT(voice_config, start voice config);
MSH_CMD_EXPORT(voice_config_stop, stop voice config);
```

6 Button

6.1 Button Introduction

Button function incudes short/double/long press.

6.2 Button Realated API

Button related APIs refer to samples\key\multi_button.h , APIs are as follows:

function	description
button_init()	button init
button_attach()	cofigure callback
button_start()	start button ,add handle to worklist
button_stops()	stop button

6.2.1 button init

```
void button_init(BUTTON_S* handle, uint8_t(*pin_level)(), uint8_t active_level,void
*user_data);
```

parameters	description
BUTTON_S* handle	button handle
uint8_t(*pin_level)	eead HAL gpio
uint8_t active_level	gpio level
void *user_data	user data
return	null

6.2.2 configure callback fuction

```
void button_attach(BUTTON_S* handle, PRESS_EVT event, btn_callback cb);
```

parameters	description
BUTTON_S* handle	button handle
PRESS_EVT event	press event
btn_callback cb	callback
return	null

6.2.3 start button

```
int button_start(BUTTON_S* handle);
```

parameters	description

BUTTON_S* handle	button handle
return	0:success ; others:fail

6.2.4 stop button

```
void button_stop(BUTTON_S* handle);
```

parameters	description
BUTTON_S* handle	button handle
return	null

6.3 Button Enumeration

Press event:

```
typedef enum {
        PRESS_DOWN = 0,
        PRESS_UP,
        PRESS_REPEAT,
        SINGLE_CLICK,
        DOUBLE_CLICK,
        LONG_RRESS_START,
        LONG_PRESS_HOLD,
        NUMBER_OF_EVENT,
        NONE_PRESS
}PRESS_EVT;
```

6.4 Button Sample Code

Button sample code refers to \samples\key\button_test.c. Openning macro definitions : BUTTON_TEST to start test.

```
/*
* program list :This is a sample for button test
* command format: button_test   gpio，input gpio number according to the schematic diagram
* program function:   long/short/double press the button, the status would be printed
  */


#include "include.h"
#include "typedef.h"
#include "arm_arch.h"
#include "gpio_pub.h"
#include "gpio_pub.h"
```

```
#include "uart_pub.h"
#include "multi_button.h"
#include "bk_rtos_pub.h"
#include "error.h"
#include "sys_ctrl_pub.h"

#define BUTTON_TEST
#ifdef   BUTTON_TEST

#define TEST_BUTTON 4
static beken_timer_t g_key_timer;

static void button_short_press(void *param)
{
        rt_kprintf("button_short_press\r\n");
}

static void button_double_press(void *param)
{
        rt_kprintf("button_double_press\r\n");
}
static void button_long_press_hold(void *param)
{
        rt_kprintf("button_long_press_hold\r\n");
}
static uint8_t key_get_gpio_level(BUTTON_S*handle)
{
        return bk_gpio_input((uint32_t)handle->user_data);
}
BUTTON_S gpio_button_test[2];

void button_test(int argc,char *argv[])
{
        OSStatus result;

        int gpio ;
        if(argc != 2)
        {
                rt_kprintf("---! ! !param error---\r\n");
        }
        else
        {
                gpio = atoi(argv[1]);

                rt_kprintf("---gpio%d as button : test start---n",gpio);

                if((gpio >=40)||(gpio >= 40))
                {
                        rt_kprintf("---! ! !gpio error---\r\n");
                        return;
                }

                /*gpio key config:input && pull up*/
                gpio_config(gpio,GMODE_INPUT_PULLUP);

                button_init(&gpio_button_test[0], key_get_gpio_level, 0,(void*)gpio);         /*initial button*/

               /*configure callback/
               button_attach(&gpio_button_test[0], SINGLE_CLICK,button_short_press);
               button_attach(&gpio_button_test[0], DOUBLE_CLICK,button_double_press);
                button_attach(&gpio_button_test[0], LONG_PRESS_HOLD,button_long_press_hold);

                button_start(&gpio_button_test[0]);                                       /*start button detect*/
                result = bk_rtos_init_timer(&g_key_timer,                                /*initial button timer */
                                        TICKS_INTERVAL,
                                        button_ticks,
                                        (void *)0);
        ASSERT(kNoErr == result);

        result = bk_rtos_start_timer(&g_key_timer);                                          /*open timer*/
        ASSERT(kNoErr == result);
        }
}

MSH_CMD_EXPORT(button_test,button test);

// eof
#endif
```

7 I2C

7.1 I2C Introduction

BK7251 supports two sets of I2C with normal 400 kHz clock speed, with 7 bit addressing. However, there is a driver file with GPIO analog I2C in RTOS, and it has been associated with the standard set of device operation functions of RTOS. SCL and SDA correspond to gpio2 and gpio3 respectively.

7.2 I2C Realated API

I2C related APIs refer to \rt-thread\components\drivers\include\drivers\i2c.h, APIs are as follows：

function	description
rt_i2c_bus_device_find()	find i2c bus
rt_i2c_transfer()	read/write data

7.2.1 find i2c bus handle

```
struct   rt_i2c_bus_device *rt_i2c_bus_device_find (const char *bus_name)；
```

parameters	description
const char *bus_name	bus name
return	bus handle

7.2.2 read/write data to slave device

```
rt_size_t rt_i2c_transfer(struct rt_i2c_bus_device *bus,
                          struct rt_i2c_msg              msgs[],
                          rt_uint32_t                    num)；
```

parameters	description
struct rt_i2c_bus_device *bus	i2c bus handle
struct rt_i2c_msg msgs[ ]	i2c message
rt_uint32_t num	number of trans
return	RT_EOK:success ; others:fail

7.3 I2C Struct Reference

rt_i2c_bus_device：

rt_device parent	parent
rt_i2c_bus_device_ops* ops	i2c opration

rt_uint16_t addr	slave address
rt_uint16_t flags	read/write flags
rt_mutex lock	lock
rt_uint32_t timeout	timeout
rt_uint32_t retries	retry count
void* priv	private data

rt_i2c_msg：

rt_uint16_t addr	address
rt_uint16_t flags	read/write flags
rt_uint16_t len	buffer length
rt_uint8_t *buf	transfer buffer

7.4 I2C Macros

#define	**RT_USING_I2C**	use i2c device
#define	**RT_USING_I2C_BITOPS**	use gpio analog i2c bus
#define	**BEKEN_USING_IIC**	use i2c driver

7.5 I2C Sample Code

I2C sample code refers to \test\i2c_rtt_test.c. Openning macro definitions : I2C_RTT_TEST to start test.

Note: There are no i2c slave devices in the chip, so an EEPROM needs to be plugged in to test its accuracy.

```
/*
* program list: This is a sample for i2c driver. Slave is EEPROM ,it’s address :0x55.
* command format: i2c_test_rtt
* program function: bk7251 control reading and writing of EEPROM by i2c driver.
  */
#include <rtthread.h>
#include <rtdevice.h>
#include "finsh.h"
#include <rthw.h>
#include <string.h>
#include <time.h>
#include <drv_iic.h>
#define eeprom_addr     0x50;   /* 1010A2A1A0 -R/W */
/************************I2C sample*********************/
static int i2c_test_rtt(int argc, char *argv)
```

```
{
    const char *i2c_bus_device_name = "i2c";
    struct rt_i2c_bus_device *i2c_device;
    struct rt_i2c_msg msgs[2];

    rt_uint8_t buffer1[2];
    rt_uint8_t buffer2[3];
    rt_size_t i, ret;
    rt_uint8_t ret1;

    ret1 = iic_bus_attach( );                               /*gpio init and add bus*/
    rt_kprintf("iic_bus_attach   ret:%d\n", ret1);
    i2c_device = rt_i2c_bus_device_find(i2c_bus_device_name);
    if (i2c_device == RT_NULL)
    {
        rt_kprintf("i2c bus device %s not found!\n", i2c_bus_device_name);
        return -RT_ENOSYS;
    }
    else
    {
        rt_kprintf("find i2c success\n");
    }
    /**********************step 1: read out.***********************/
    buffer1[0] = 0;
    msgs[0].addr = eeprom_addr;
    msgs[0].flags = RT_I2C_WR;                              /* write to slave */
    msgs[0].buf = buffer1;                                  /* eeprom offset. */
    msgs[0].len = 1;

    msgs[1].addr = eeprom_addr;
    msgs[1].flags = RT_I2C_RD;                               /* read from slave */
    msgs[1].buf = buffer2;
    msgs[1].len = sizeof(buffer2);

     if ( rt_i2c_transfer(i2c_device, msgs, 2) !=2 )        /* write or read data */
     {
         rt_kprintf("--read eeprom fail--\r\n");
     }
     else
     {
```

```
                rt_kprintf("--read eeprom sucess--\r\n");
         }
        for(i=0; i<sizeof(buffer2); i++)
        {
              rt_kprintf("%02X ", buffer2[i]);
        }

         rt_thread_delay(rt_tick_from_millisecond(50));
        rt_kprintf("\r\n---read test over---\r\n");
/**********************step 2: write back.***********************/

         for(i=0; i<sizeof(buffer2); i++)
         {
                buffer2[i] = buffer2[i]+5 ;
         }
         msgs[0].addr = eeprom_addr;
        msgs[0].flags = RT_I2C_WR;                    /* write to slave */
        msgs[0].buf = buffer1;                        /* eeprom offset. */
        msgs[0].len = 1;

        msgs[1].addr = eeprom_addr;
        msgs[1].flags = RT_I2C_WR;
        msgs[1].len = sizeof(buffer2);
         if ( rt_i2c_transfer(i2c_device, msgs, 2) !=2 )
         {
                rt_kprintf("---write eeprom fail---\r\n");
         }
         else

         {
                rt_kprintf("---write eeprom sucess---\r\n");
         }

         rt_thread_delay(rt_tick_from_millisecond(50));

        for(i=0; i<msgs[1].len; i++)
        {
              rt_kprintf("%02X ", buffer2[i]);
        }
        rt_kprintf("\r\n ---write test over---\r\n");
```

```
/**********************step 3: read out.************************/
       buffer1[0] = 0;
        msgs[0].addr = eeprom_addr;
        msgs[0].flags = RT_I2C_WR;                          /* write to slave */
        msgs[0].buf = buffer1;                              /* eeprom offset. */
        msgs[0].len = 1;

        msgs[1].addr = eeprom_addr;
        msgs[1].flags = RT_I2C_RD;                          /* Read from slave */
        msgs[1].buf = buffer2;
        msgs[1].len = sizeof(buffer2);

        if ( rt_i2c_transfer(i2c_device, msgs, 2) !=2 )
        {
                rt_kprintf("---re-read eeprom fail---\r\n");
        }
        else
        {
                rt_kprintf("---re-read eeprom sucess---\r\n");
        }

        rt_thread_delay(rt_tick_from_millisecond(50));

        for(i=0; i<msgs[1].len; i++)
        {
                rt_kprintf("%02X ", buffer2[i]);
        }
        rt_kprintf("\r\n ---re-read test over---\r\n");
       return 0;
}
#ifdef RT_USING_FINSH
#include <finsh.h>
FINSH_FUNCTION_EXPORT_ALIAS(i2c_test_rtt, __cmd_i2c_test_rtt, i2c test rtt cm);
#endif
```

8 I2S

8.1 I2S Introduction

The I2S interface supports both master and slave mode, with sample rate from 8 kHz to 96 kHz. The I2S interface contains four signal lines: I2S_CLK(gpio2)、I2S_SYNC(gpio3)、 I2S_DIN(gpio4) and I2S_DOUT(gpio5). I2S mode can be divided into I2S, Left justifie d, Right justified and 2B+D .I2S_CLK: BCLK ,l2S_SYNC:sampling rate, I2S_DIN:input,I2S_DOUT:output.

8.2 I2S Realated API

I2S related APIs refer to \rt-thread\components\drivers\include\drivers\i2s.h, APIs are as follows:

function	description
i2s_configure()	i2s configure
i2s_transfer()	master/slave read/write data

8.2.1 i2s parameters configure

```
i2s_configure(UINT32 fifo_level, UINT32 sample_rate, UINT32 bits_per_sample, UINT32 mode)；
```

parameters	description
fifo_level	configure fifo level
sample_rate	set sampling rate
bits_per_sample	bit number of 1 channel
mode	mdoe
return	I2S_SUCCESS:success;others:fail

8.2.2 i2s master/slave receive/send data

```
UINT32 i2s_transfer(UINT32 *i2s_send_buf , UINT32 *i2s_recv_buf, UINT32 count, UINT32 param )；
```

parameters	description
i2s_send_buf	send buffer
i2s_recv_buf	receive buffer
count	the count of send buffer
param	1:master ; 0:lave
return	0:success; others:fail

8.3 I2S Struct Reference

i2s_trans_t：

p_tx_buf	send buffer
***p_rx_buf;**	receive buffer
trans_done	trans data over flag
tx_remain_data_cnt;	tx remain data
rx_remain_data_cnt	rx remain data

i2s_message：

send_buf	send buffer
send_len	send buffer length
recv_buf	receive buffer
recv_len	receive buffer length

8.4 I2S Macros

#define RT_USING_I2S	use I2S module
#define BEKEN_USING_IIS	use I2S driver

I2S work mode macros:

#define I2S_MODE	**(0 << 0)**	I2S mode
#define I2S_LEFT_JUSTIFIED	**(1 << 0)**	left justified
#define I2S_RIGHT_JUSTIFIED	**(2 << 0)**	right justified
#define I2S_RESERVE	**(3 << 0)**	reserve
#define I2S_SHORT_FRAME_SYNC	**(4 << 0)**	short frame sync
#define I2S_LONG_FRAME_SYNC	**(5<< 0)**	long frame sync
#define I2S_NORMAL_2B_D	**(6 << 0)**	normal 2B+D
#define I2S_DELAY_2B_D	**(7 << 0)**	delay 2B+D
#define I2S_LRCK_NO_TURN	**(0 << 3)**	lrck not turn
#define I2S_SCK_NO_TURN	**(0 << 4)**	sck not turn
#define I2S_MSB_FIRST	**(0 << 5)**	MSB first send
#define I2S_SYNC_LENGTH_BIT	**(8)**	sync length(only work in long frame sync)
#define I2S_PCM_DATA_LENGTH_BIT	**(12)**	D length(only work in 2B+D mode)

8.5 I2S Sample Code

I2S sample code refers to \test\i2s_test.c. Openning macro definitions : I2S_TEST to start test.

```
/*
* program list:This is a sample for i2s.it need 2 demo board ,one as mater ,the other as slave.
* command format: i2s_test   master/slave   rate   bit_length
* program function: The sample code achieves the data receiving and sending by master and slave
   devices respectively, and tests whether the data can be received or sent normally and whether the
   frequency and bit width can meet the requirements.
*/
#include "include.h"
#include "arm_arch.h"
#include <rtthread.h>
#include <rthw.h>
#include <rtdevice.h>
#include <stdio.h>
#include <string.h>
#include <stdint.h>
#include <stdlib.h>

#include "typedef.h"
#include "icu_pub.h"
#include "i2s.h"
#include "i2s_pub.h"

#include "sys_ctrl_pub.h"

#include "drv_model_pub.h"
#include "mem_pub.h"

#include "sys_config.h"
#include "error.h"
#include "rtos_pub.h"

#define I2S_DATA_LEN            0x100

extern UINT32 i2s_configure(UINT32 fifo_level, UINT32 sample_rate, UINT32 bits_per_sample,
UINT32 mode);

volatile i2s_trans_t i2s_trans;
i2s_level_t   i2s_fifo_level;
```

```
 int i2s_test(int argc, char** argv)
{
        struct rt_device *i2s_device;
        struct i2s_message msg;
        uint32 i,rate,bit_length ;
        uint32 i2s_mode = 0;
        if(argc != 4)
        {
                rt_kprintf("---cmd error--\r\n");
                return RT_ERROR;
        }
        rate        = atoi(argv[2]);
        bit_length = atoi(argv[3]);

        msg.recv_len = I2S_DATA_LEN;
        msg.send_len = I2S_DATA_LEN;

        msg.recv_buf = rt_malloc(I2S_DATA_LEN * sizeof(msg.recv_buf[0]));
        if(msg.recv_buf == RT_NULL)
        {
                rt_kprintf("msg.recv_buf malloc failed\r\n");
        }
        //rt_kprintf("msg.recv_buf=%x\r\n",msg.recv_buf);

        msg.send_buf = rt_malloc(I2S_DATA_LEN * sizeof(msg.send_buf[0]));
        if(msg.send_buf == RT_NULL)
        {
                rt_kprintf("msg.send_buf malloc failed\r\n");
        }
        //rt_kprintf("msg.send_buf=%x\r\n",msg.send_buf);

        /* find device*/
        i2s_device = rt_device_find("i2s");
        if(i2s_device == RT_NULL)
        {
                rt_kprintf("---i2s device find failed---\r\n ");
                return 0 ;
        }
```

```
/* init device*/
if(rt_device_init( i2s_device) != RT_EOK)
{
        rt_kprintf(" --i2s device init failed---\r\n ");
        return 0;
}

/* open audio , set fifo level set sample rate/datawidth */
 i2s_mode = i2s_mode| I2S_MODE| I2S_LRCK_NO_TURN| I2S_SCK_NO_TURN|
 I2S_MSB_FIRST| (0<<I2S_SYNC_LENGTH_BIT)| (0<<I2S_PCM_DATA_LENGTH_BIT);

/* write and recieve */
if(strcmp(argv[1], "master") == 0)
{
        rt_kprintf("---i2s_master_test_start---\r\n");

        if(msg.send_buf == NULL)

        {
                rt_kprintf("---msg.send_buf error --\r\n ");
                return 0;
        }

        for(i=0; i<I2S_DATA_LEN; i++)
        {
                msg.send_buf[i]= ((i+1)<<24) | ((i+1)<<16)    | ((i+1)<<8) | ((i+1)<<0);
        }
       i2s_configure(FIFO_LEVEL_32, rate, bit_length, i2s_mode);
        i2s_transfer(msg.send_buf, msg.recv_buf, I2S_DATA_LEN, MASTER);

        for(i=0; i<I2S_DATA_LEN; i++)
       {
                rt_kprintf("msg.send_buf[%d]=0x%x ---msg.recv_buf[%d]=0x%x \r\n", i,
           msg.send_buf[i],i, msg.recv_buf[i]);
        }
        rt_kprintf("---i2s_master_test_over---\r\n");

}
else if(strcmp(argv[1], "slave") == 0)                                    //slave
{
        rt_kprintf("---i2s_slave_test_start---\r\n");

        if(msg.send_buf == NULL)

        {
            rt_kprintf("---msg.send_buf error --\r\n ");
            return 0;
        }

        for(i=0; i<I2S_DATA_LEN; i++)
        {
            msg.send_buf[i]= ((i+1)<<24) | ((i+1)<<16)    | ((i+1)<<8) | ((i+1)<<0) |0x80808080;
        }

         i2s_configure(FIFO_LEVEL_32, rate, bit_length, i2s_mode);
        i2s_transfer(msg.send_buf, msg.recv_buf, I2S_DATA_LEN, SLAVE);

       for(i=0; i<I2S_DATA_LEN; i++)
       {
          rt_kprintf("msg.send_buf[%d]=0x%x , msg.recv_buf[%d]=0x%x \r\n", i, msg.send_buf[i],i,
          msg.recv_buf[i]);
        }

        rt_kprintf("---i2s_slave_test_over---\r\n");
    }
    else
    {
        rt_kprintf("---no test command--\r\n");
    }



    i2s_trans.p_rx_buf = RT_NULL;
    i2s_trans.p_tx_buf = RT_NULL;

    if(msg.send_buf != RT_NULL)
    {
        os_free(msg.send_buf);
        msg.send_buf= RT_NULL;
    }
```

```
        if(msg.recv_buf != RT_NULL)
        {
                os_free(msg.recv_buf);
                msg.recv_buf= RT_NULL;
        }

        rt_kprintf("---i2s_test_over---\r\n");
        return 0;
}

MSH_CMD_EXPORT(i2s_test, i2s_test);
```

9 General SPI

9.1 General SPI Introduction

BK7251 supports spi mode and both slave and master,with follwing features:

a) the length of data exchange can be matched, usually in byte units, MSB first, LSB later;

b) maximum clock speed on master mode:30MHz;

c) maximum clock speed on slave mode:10MHz;

d) CPOL and CPHA are configurable.

e) support four-wire full duplex mode（MOSI、MISO、CSN、CLK）and three-wire half-duplex（DATA、CS、CLK）.

9.2 General SPI Related API

General spi driver has been associated with the standard set of device operation functions of RTOS.So call RTOS standard device operation interface directly.General spi APIs refer to \rt-thread\components\drivers\include\drivers\spi.h. APIs are as follows:

function	description
rt_spi_configure()	spi configure
rt_spi_send()	spi send data (slave may be hang up)
rt_spi_recv()	spi receive data(slave may be hang up)

9.2.1 spi configure

```
rt_err_t rt_spi_configure(struct rt_spi_device *device,     struct rt_spi_configuration *cfg);
```

parameters	description
device	spi device
cfg	spi configure sturct
return	RT_EOK:success ; others:fail

9.2.2 spi send data

```
rt_inline rt_size_t rt_spi_send(struct rt_spi_device *device,
                                const void       *send_buf,
                                rt_size_t         length);
```

parameters	description
device	spi device pointer
send_buf	send buffer

length	send buffer length
return	send byte

As master, all data is sent back immediately. As slave mode, it may hang until the spi that communicates with it initiates the spi sequence, and all data is sent.

9.2.3 spi receive data

```
rt_inline rt_size_t rt_spi_recv(struct rt_spi_device *device,
                                void           *recv_buf,
                                rt_size_t       length);
```

parameters	description
device	spi device pointer
recv_buf	receive buffer
length	receive buffer length
return	receive byte

As master, all data is sent back immediately. As slave mode, it may hang until the spi that communicates with it initiates the spi sequence, and will be returned when non-zero length received.

9.3 General SPI Struct Reference

rt_spi_device:

parent	spi device object
bus	spi bus handle
config	spi configure struct

rt_spi_configuration:

mode	spi work mode
data_width	send/receive data_width
reserved	reserve
max_hz	spi frequency,only master work

9.4 General SPI Macros

#define RT_USING_SPI	open spi mode
#define CFG_USE_SPI_MASTER	open master
#define CFG_USE_SPI_SLAVE	start slave

spi work mode:

#define RT_SPI_CPHA (1<<0)	second edge sampling data
#define RT_SPI_CPOL (1<<1)	sck at a high level in idle time

#define RT_SPI_LSB (0<<2)	0-LSB
#define RT_SPI_MSB (1<<2)	1-MSB
#define RT_SPI_MASTER (0<<3)	master
#define RT_SPI_SLAVE (1<<3)	slave
#define RT_SPI_MODE_0 (0 | 0)	CPOL = 0, CPHA = 0
#define RT_SPI_MODE_1 (0 | RT_SPI_CPHA)	CPOL = 0, CPHA = 1
#define RT_SPI_MODE_2 (RT_SPI_CPOL | 0)	CPOL = 1, CPHA = 0
#define RT_SPI_MODE_4 (RT_SPI_CPOL | RT_SPI_CPHA)	CPOL = 1, CPHA = 1
#define RT_SPI_MODE_MASK (RT_SPI_CPHA | RT_SPI_CPOL | RT_SPI_MSB | RT_SPI_SLAVE)	all bit is1

9.5 General SPI Sample Code

General spi sample code refers to \test\general_spi_test.c. Sample code describes how to use spi related APIs. Openning macro definitions :GENERAL_SPI_TEST to start test.

```
/*
* program      list : This is a sample code about using general spi driver .Make sure that
  rt_hw_spi_device_init() has been called when system initialization.
* command format: gspi_test   master/slave   tx/rx   rate   len
* program function: configure spi interface master/slave and transmission speed, finish receive/send
*/
#include <rtthread.h>
#include <rthw.h>
#include <rtdevice.h>
#include <stdio.h>
#include <string.h>
#include "sys_config.h"

#define SPI_BAUDRATE           (10 * 1000 * 1000)
#define SPI_TX_BUF_LEN         (32)
#define SPI_RX_BUF_LEN         (32)

/* must open CFG_USE_SPI_MASTER and CFG_USE_SPI_SLAVE macros,refer to sys_config.h */
#if ((CFG_USE_SPI_MASTER) &&(CFG_USE_SPI_SLAVE))
int gspi_test(int argc, char** argv)
{
        struct rt_spi_device *spi_device;
       struct rt_spi_configuration cfg;
```

```
/*find device*/
spi_device = (struct rt_spi_device *)rt_device_find("gspi");
if (spi_device == RT_NULL)
{
    rt_kprintf("spi device %s not found!\r\n", "gspi");
    return -RT_ENOSYS;
}
cfg.data_width = 8;
if(strcmp(argv[1], "master") == 0)
{
    /*set master mode MSB、CPOL = 0, CPHA = 0*/
    cfg.mode = RT_SPI_MODE_0 | RT_SPI_MSB | RT_SPI_MASTER;
}
else if(strcmp(argv[1], "slave") == 0)
{
    /*set slave modeMSB、CPOL = 0, CPHA = 0*/
    cfg.mode = RT_SPI_MODE_0 | RT_SPI_MSB | RT_SPI_SLAVE;
}
else
{
    rt_kprintf("gspi_test master/slave    tx/rx   rate   len\r\n");
    return -RT_ENOSYS;
}
/* SPI Interface with Clock Speeds Up to 30 MHz */
if(argc == 5)
    cfg.max_hz = atoi(argv[3]);
else
    cfg.max_hz = SPI_BAUDRATE;
rt_kprintf("cfg:%d, 0x%02x, %d\r\n", cfg.data_width, cfg.mode, cfg.max_hz);
/*configure device*/
rt_spi_configure(spi_device, &cfg);
if(strcmp(argv[2], "tx") == 0)
{
    rt_uint8_t *buf;
    int tx_len;
    if(argc < 4)
        tx_len = SPI_TX_BUF_LEN;
    else
        tx_len = atoi(argv[4]);
    rt_kprintf("spi init tx_len:%d\n", tx_len);
```

```
        buf = rt_malloc(tx_len * sizeof(rt_uint8_t));
        if(buf)
        {
            rt_memset(buf, 0, tx_len);
            for(int i=0; i<tx_len; i++)
            {
                buf[i] = i & 0xff;
            }
            /*send data ,slave may be hang up*/
            rt_spi_send(spi_device, buf, tx_len);
            for(int i=0; i<tx_len; i++)
            {
                rt_kprintf("%02x,", buf[i]);
                if((i+1)%32 == 0)
                    rt_kprintf("\r\n");
            }
            rt_kprintf("\r\n");
            rt_free(buf);
        }
    }
    else if(strcmp(argv[2], "rx") == 0)
    {
        rt_uint8_t *buf;
        int rx_len;
        if(argc < 4)
            rx_len = SPI_RX_BUF_LEN;
        else
            rx_len = atoi(argv[4]);
        rt_kprintf("spi init rx_len:%d\n", rx_len);
        buf = rt_malloc(rx_len * sizeof(rt_uint8_t));
        if(buf)
        {
            rt_memset(buf, 0, rx_len);
            /* receive data ,slave may be hang up */
            rx_len = rt_spi_recv(spi_device, buf, rx_len);
            rt_kprintf("rx ret:%d\r\n", rx_len);
            for(int i=0; i<rx_len; i++)
            {
                rt_kprintf("%02x,", buf[i]);
                if((i+1)%32 == 0)
```

```
                    rt_kprintf("\r\n");
                }
                rt_kprintf("\r\n");
                rt_free(buf);
            }
    }
    else
    {
        rt_kprintf("gspi_test master/slave     tx/rx    rate    len\r\n");
    }
}
MSH_CMD_EXPORT(gspi_test, gspi_test);
```

10 General SPI Flash

10.1 General SPI Flash Introduction

SPI Flash is standard Flash plug-in, with follwing features:

a) support four-wire full duplex mode;
b) maximum clock speed on master mode:30MHz.

10.2 General SPI Flash Related API

General spi flash driver has been associated with the standard set of device operation functions of RTOS.So call RTOS standard device operation interface directly.

function	description
rt_device_control()	spi flash operation such as: erase,remove/add write protection

10.2.1 control device

```
rt_err_t rt_device_control(rt_device_t dev, int cmd, void *arg);
```

parameters	description
dev	spi flash pointer
cmd	command of device
arg	device’s args
return	RT_EOK:success ; others:fail

10.3 General SPI Flash Macros

Flash work command:

#define	**BK_SPI_FLASH_ERASE_CMD**	erase command
#define	**BK_SPI_FLASH_PROTECT_CMD**	falsh write protection
#define	**BK_SPI_FLASH_UNPROTECT_CMD**	flash write unprotection

10.4 General SPI Flash Sample Code

General spi flash sample code refers to \test\general_spi_flash_test.c. Sample code describes how to use spi flash related APIs. Openning macro definitions : SPI_FLASH_TEST to start test.

```
/*
* program       list: This is a sample code about using general spi flash driver .Make sure that
  rt_spi_flash_hw_init() has been called when system initialization.
```

```
* command format: gspi_flash_test
* program function: test spi flash read/write data
*/
#include <rtthread.h>
#include <rthw.h>
#include <rtdevice.h>
#include <stdio.h>
#include <string.h>
#include "sys_config.h"
#ifdef BEKEN_USING_SPI_FLASH

/*spi flash must be related with 3 macros: BEKEN_USING_SPI_FLASH、
CFG_USE_SPI_MASTER 、CFG_USE_SPI_MST_FLASH */
#if ((CFG_USE_SPI_MASTER == 0) || (CFG_USE_SPI_MST_FLASH == 0))
#error "test gspi psram need 'CFG_USE_SPI_MASTER' and 'CFG_USE_SPI_MST_FLASH'"
#endif

#include "drv_spi_flash.h"
#define FTEST_BUF_SIZE          1024
#define FTEST_BASE              0x40
#define FTEST_ADDR              0x100000
void gspi_flash_test(int argc, char** argv)
{
        struct rt_device *flash;
        /*find device*/
        flash = rt_device_find("spi_flash");
        if (flash == NULL)
        {
                rt_kprintf("psram not found \n");
                return;
        }
        /*device init */
        if (rt_device_init(flash) != RT_EOK)
        {
                return;
        }
        /*open device*/
        if (rt_device_open(flash, 0) != RT_EOK)
        {
                return;
        }
        uint8_t buffer[FTEST_BUF_SIZE], *ptr;
        int i;
        rt_kprintf("[SPIFLASH]: SPIFLASH test begin\n");
        rt_memset(buffer, 0, FTEST_BUF_SIZE);
        /*read once */
        rt_device_read(flash, FTEST_ADDR, buffer, FTEST_BUF_SIZE);
        /*print read data */
        ptr = buffer;
        rt_kprintf("flash data:%x\r\n", FTEST_ADDR);
        for(i=0; i<FTEST_BUF_SIZE; i++)
        {
                rt_kprintf("0x%02x,", ptr[i]);
                if((i+1)%16 == 0)
                    rt_kprintf("\r\n");
        }
        rt_kprintf("\r\n");

        /*initial the write dara, the data come from FTEST_BASE */
        ptr = (uint8_t *)FTEST_BASE;
        rt_kprintf("base data:%08x\r\n", ptr);
        for(i=0; i<FTEST_BUF_SIZE; i++)
        {
                rt_kprintf("0x%02x,", ptr[i]);
                buffer[i] = ptr[i];
                if((i+1)%16 == 0)
                    rt_kprintf("\r\n");
        }
        rt_kprintf("\r\n");
        /*unprotection before write*/
        rt_device_control(flash, BK_SPI_FLASH_UNPROTECT_CMD, NULL);
        /*write data */
        rt_device_write(flash, FTEST_ADDR, buffer, FTEST_BUF_SIZE);
        rt_kprintf("write fin\r\n");
        /*clear buffer */
        rt_memset(buffer, 0, FTEST_BUF_SIZE);
        /*read return */
        rt_device_read(flash, FTEST_ADDR, buffer, FTEST_BUF_SIZE);
        rt_kprintf("read fin\r\n");
        ptr = buffer;
```

```
    rt_kprintf("flash data:%x\r\n", FTEST_ADDR);
    for(i=0; i<FTEST_BUF_SIZE; i++)
    {
        rt_kprintf("0x%02x,", ptr[i]);
        if((i+1)%16 == 0)
            rt_kprintf("\r\n");
    }
    rt_kprintf("\r\n");
    /*erase flash */
    rt_kprintf("earase\r\n");
    BK_SPIFLASH_ERASE_ST erase_st;
    erase_st.addr = FTEST_ADDR;
    erase_st.size = 4 * 1024;
    rt_device_control(flash, BK_SPI_FLASH_ERASE_CMD, &erase_st);
    rt_kprintf("[SPIFLASH]: SPIFLASH test end\n");
    /*protection */
    rt_device_control(flash, BK_SPI_FLASH_PROTECT_CMD, NULL);
    /*close device*/
    rt_device_close(flash);
}
MSH_CMD_EXPORT(gspi_flash_test, gspi_flash_test);
#endif // BEKEN_USING_SPI_FLASH
```

11 General SPI PSRAM

11.1 General SPI PSRAM Introduction

SPI PSRAM is standard ram plug-in, with follwing features:

a) support four-wire full duplex mode;

b) maximum clock speed on master mode:30MHz.

11.2 General SPI PSRAM Related API

General spi psram driver has been associated with the standard set of device operation functions of RTOS.So call RTOS standard device operation interface directly.

11.3 General SPI PSRAM Struct Reference

PSRAM: rt_device struct

11.4 General SPI PSRAM Macros

#define	**BEKEN_USING_SPI_PSRAM**	open spi psram moudle

11.5 General SPI PSRAM Sample Code

General spi psram sample code refers to \test\general_spi_psram_test.c. Sample code describe how to use spi psram related APIs. Openning macro definitions : SPI_PSRAM_TEST to start test.

```
/*
* program       list : This is a sample code about using general spi psram driver .Make sure that
   rt_spi_psram_hw_init() has been called when system initialization.
* command format: spi_psram_test
* program function:test spi psram read/write data
*/
#include <rtthread.h>
#include <rthw.h>
#include <rtdevice.h>
#include <stdio.h>
#include <string.h>
#include "sys_config.h"

#ifdef BEKEN_USING_SPI_PSRAM
/* spi psram must be related with 3 macros:BEKEN_USING_SPI_PSRAM、
CFG_USE_SPI_MASTER 、CFG_USE_SPI_MST_PSRAM */
#if ((CFG_USE_SPI_MASTER == 0) || (CFG_USE_SPI_MST_PSRAM == 0))
```

```
#error "test gspi psram need 'CFG_USE_SPI_MASTER' and 'CFG_USE_SPI_MST_PSRAM'"
#endif

void spi_psram_test(int argc, char** argv)
{
      struct rt_device *psram;
      /*find device*/
      psram = rt_device_find("spi_psram");
      if (psram == NULL)
      {
            rt_kprintf("psram not found \n");
            return;
      }
      /*initial device*/
      if (rt_device_init(psram) != RT_EOK)
      {
            return;
      }
      /*open device*/
      if (rt_device_open(psram, 0) != RT_EOK)
      {
            return;
      }
       uint8_t buffer[4096];
       int i;
       rt_kprintf("[PSRAM]: SPRAM test begin\n");
       for(i = 0; i < sizeof(buffer); i++)
       {
              buffer[i] = (uint8_t)i;
       }
      /*write device*/
      rt_device_write(psram, 0, buffer, sizeof(buffer));
      /*clear buffer*/
       rt_memset(buffer, 0, sizeof(buffer));
      /*read device*/
      rt_device_read(psram, 0, buffer, sizeof(buffer));
      /*compare read data with write data */
       for(i = 0; i < sizeof(buffer); i++)
       {
              if(buffer[i] != (uint8_t)i)
```

```
                {
                        rt_kprintf("[%02d]: %02x - %02x\n", i, (uint8_t)i, buffer[i]);
                }
        }
        rt_kprintf("[PSRAM]: SPRAM test end\n");
       /*close device*/
       rt_device_close(psram);
}
MSH_CMD_EXPORT(spi_psram_test, spi_psram_test);
#endif    // BEKEN_USING_SPI_PSRAM
```

12 Highspeed SPI Slave

12.1 Highspeed SPI Slave Introduction

Highspeed spi slave is designed to solve the problem that general SPI can not withstand high speed, with follwing features:

a) support four-wire full-duplex and three-wire half-duplex modules;
b) support MSB/LSB configuration;
c) support DMA;
d) maximum clock speed:50MHz.

Note: For convenience and simplicity, the configuration of spi_hs is fixed as follows: four-wire mode, MSB, sending and receiving using DMA.

12.2 Highspeed SPI Slave Related API

Highspeed spi driver has been associated with the standard set of device operation functions of RTOS.So call RTOS standard device operation interface directly.

12.3 Highspeed SPI Slave Struct Reference

spi_hs: rt_device struct

12.4 Highspeed SPI Slave Macros

#define	**BEKEN_USING_SPI_HSLAVE**	open spi slave moudle
#define	**SPI_TX_BUF_LEN**	length of send data length
#define	**SPI_RX_BUF_LEN**	length of receive data length

12.5 Highspeed SPI Slave Sample Code

Highspeed spi slave sample code refers to \test\highspeed_spi_slave_test.c. Sample code describe how to use spi psram related APIs. Openning macro definitions : SPI_HSLAVE_TEST to start test.

```
/*
* program      list: This is a sample code about using highspeed spi slave driver .Make sure that
   rt_spi_hslave_hw_init() has been called when system initialization.
* command format: spi_hs_test   tx/rx   len
* program function: test highspeed spi slave read/write data
*/
#include <rtthread.h>
#include <rthw.h>
#include <rtdevice.h>
#include <stdio.h>
```

```
#include <string.h>
#include "sys_config.h"
#define SPI_TX_BUF_LEN         (512)
#define SPI_RX_BUF_LEN         (512)
#ifdef BEKEN_USING_SPI_HSLAVE

/* highspeed spi slave must be related with 2 macros:BEKEN_USING_SPI_HSLAVE、
CFG_USE_HSLAVE_SPI */
#if (CFG_USE_HSLAVE_SPI == 0)
#error "spi_hs_test need 'CFG_USE_HSLAVE_SPI' and 'CFG_USE_SPI_MST_PSRAM'"
#endif
int spi_hs_test(int argc, char** argv)
{
     struct rt_device *spi_hs;
    /*find device*/
    spi_hs = (struct rt_device *)rt_device_find("spi_hs");
    if (spi_hs == RT_NULL)
    {
        rt_kprintf("spi device %s not found!\r\n", "spi_hs");
        return -RT_ENOSYS;
    }
    /*open device*/
    if (rt_device_open(spi_hs, 0) != RT_EOK)
    {
        return 0;
    }
    if(strcmp(argv[1], "tx") == 0)
    {
        rt_uint8_t *buf;
        int tx_len;
        if(argc < 3)
            tx_len = SPI_TX_BUF_LEN;
        else
            tx_len = atoi(argv[2]);
        rt_kprintf("spi hs tx_len:%d\n", tx_len);
        buf = rt_malloc(tx_len * sizeof(rt_uint8_t));
        if(buf)
        {
            rt_memset(buf, 0, tx_len);
            for(int i=0; i<tx_len; i++)
```

```
                {
                        buf[i] = i & 0xff;
                }
                /*write data*/
                rt_device_write(spi_hs, 0, (const void *)buf, tx_len);
                for(int i=0; i<tx_len; i++)
                {
                        rt_kprintf("%02x,", buf[i]);
                        if((i+1)%32 == 0)
                                rt_kprintf("\r\n");
                }
                rt_kprintf("\r\n");
                rt_free(buf);
        }
}
else if(strcmp(argv[1], "rx") == 0)
{
        rt_uint8_t *buf;
        int rx_len;
        if(argc < 3)
                rx_len = SPI_RX_BUF_LEN;
        else
                rx_len = atoi(argv[2]);
        rt_kprintf("spi hs rx_len:%d\n", rx_len);
        buf = rt_malloc(rx_len * sizeof(rt_uint8_t));
        if(buf)
        {
                rt_memset(buf, 0, rx_len);
                /*receive data*/
                rx_len = rt_device_read(spi_hs, 0, buf, rx_len);
                rt_kprintf("rx ret:%d\r\n", rx_len);
                for(int i=0; i<rx_len; i++)
                {
                        rt_kprintf("%02x,", buf[i]);
                        if((i+1)%32 == 0)
                                rt_kprintf("\r\n");
                }
                rt_kprintf("\r\n");
                rt_free(buf);
        }
```

```
    }
    else
    {
        rt_kprintf("spi_hs_test tx/rx len\r\n");
    }
    /*close device*/
    rt_device_close(spi_hs);
}
MSH_CMD_EXPORT(spi_hs_test, spi_hs_test);
#endif // BEKEN_USING_SPI_HSLAVE
```

13 GPIO

13.1 GPIO Introduction

BK7251 has total 40 GPIOs and anyone could be set an interrupt source to interrupt system at active mode or wake up system from sleep mode.

13.2 GPIO Related API

Gpio related APIs refer to \rt-thread\components\drivers\include\drivers\pin.h, APIs are as follows;

function	description
rt_pin_mode()	set pin mode
rt_pin_write()	set pin output level
rt_pin_read()	read pin level
rt_pin_attach_irq()	attach pin callback
rt_pin_irq_enable()	enable pin interrupt
rt_pin_detach_irq()	detach pin interrupt callback

13.2.1 set pin mode

```
void rt_pin_mode(rt_base_t pin, rt_base_t mode);
```

parameters	description
pin	pin number
mode	pin woek mode
return	null

13.2.2 set pin output level

```
void rt_pin_write(rt_base_t pin, rt_base_t value);
```

parameters	description
pin	pin number
value	pin level logic value,has 2 macros: PIN_LOW/PIN_HIGH
return	null

13.2.3 read pin leve

```
int rt_pin_read(rt_base_t pin);
```

parameters	description
pin	pin number
return	PIN_LOW/PIN_HIGH

13.2.4 attach pin interrupt callback fuction

When pin interruption occurs, the callback function is executed after the callback attached.

```
rt_err_t rt_pin_attach_irq(rt_int32_t pin, rt_uint32_t mode,void (*hdr)(void *args), void *args);
```

parameters	description
pin	pin number
mode	interrupt trigger mode
hdr	interrupt callback
args	interrupt callback arg,not use:RT_NULL
return	RT_EOK:success;others:fail

13.2.5 enable pin interrupt

```
rt_err_t rt_pin_irq_enable(rt_base_t pin, rt_uint32_t enabled);
```

parameters	description
pin	pin number
enabled	2 state:PIN_IRQ_ENABLE:open PIN_IRQ_DISABLE:close
return	RT_EOK:success;others:fail

13.2.6 detach pin interrupt callback

```
rt_err_t rt_pin_detach_irq(rt_int32_t pin);
```

parameters	description
pin	pin number
return	RT_EOK:success;others:fail

13.3 GPIO Macros

Each gpio operating modes support four types.

#define	**PIN_MODE_OUTPUT**	**0x00**	output
#define	**PIN_MODE_INPUT**	**0x01**	input
#define	**PIN_MODE_INPUT_PULLUP**	**0x02**	pull up input

#define	**PIN_MODE_INPUT_PULLDOWN**	**0x03**	pull down input
#define	**PIN_IRQ_MODE_RISING**	**0x00**	rising edge trigger
#define	**PIN_IRQ_MODE_FALLING**	**0x01**	falling edge trigger
#define	**PIN_IRQ_MODE_RISING_FALLING**	**0x02**	both the rising and falling edge trigger
#define	**PIN_IRQ_MODE_HIGH_LEVEL**	**0x03**	high level trigger
#define	**PIN_IRQ_MODE_LOW_LEVEL**	**0x04**	low level trigger

13.4 GPIO Sample Code

GPIO sample code refers to \test\gpio.c, Sample code describe how to use spi psram related APIs. The macros LED_PIN_NUM、LED1_PIN_NUM、LED2_PIN_NUM、KEY0_PIN_NUM、KEY1_PIN_NUM change according to actual harware. Openning macro definitions : GPIO_DEMO to start test.

```
/*
* program      list: This is a sample code about using gpio as pin.
* command format: pin_led_sample
* program function: control led through button ,and level state control the corresponding pins.
*/
#include <rtthread.h>
#include <rtdevice.h>

#define LED_PIN_NUM 30
#define LED1_PIN_NUM 13
#define LED2_PIN_NUM 27
#define KEY0_PIN_NUM 2
#define KEY1_PIN_NUM 3

void led_on(void *args) {
    rt_kprintf("turn on led!\n");
    rt_pin_write(LED_PIN_NUM, PIN_HIGH);
}
void led_off(void *args) {
    rt_kprintf("turn off led!\n");
    rt_pin_write(LED_PIN_NUM, PIN_LOW);
}
static void pin_led_sample(void) {
    /* led pin set as output mode*/
    rt_pin_mode(LED_PIN_NUM, PIN_MODE_OUTPUT);
    /* default low level*/
```

```
        rt_pin_write(LED_PIN_NUM, PIN_LOW);
        /* key0 set pull up input mode*/
        rt_pin_mode(KEY0_PIN_NUM , PIN_MODE_INPUT_PULLUP);
        /* attach int,falling trigger mode, callback function is led_on */
        rt_pin_attach_irq(KEY0_PIN_NUM , PIN_IRQ_MODE_FALLING , led_on, RT_NULL);
        /* enable int*/
        rt_pin_irq_enable(KEY0_PIN_NUM , PIN_IRQ_ENABLE);
        /* key1 set pull up input mode */
        rt_pin_mode(KEY1_PIN_NUM , PIN_MODE_INPUT_PULLUP);
        /* attach int,falling trigger mode, callback function is led_off */
        rt_pin_attach_irq(KEY1_PIN_NUM , PIN_IRQ_MODE_FALLING , led_off, RT_NULL);
        /* enable int */
      rt_pin_irq_enable(KEY1_PIN_NUM , PIN_IRQ_ENABLE);
         /* led set output mode*/
         rt_pin_mode(LED1_PIN_NUM, PIN_MODE_OUTPUT);
         /* default low level*/
         rt_pin_write(LED1_PIN_NUM, PIN_LOW);
         /* l set output mode */
         rt_pin_mode(LED2_PIN_NUM, PIN_MODE_OUTPUT);
         /* default low level */
         rt_pin_write(LED2_PIN_NUM, PIN_HIGH);
}

/* command function*/
MSH_CMD_EXPORT(pin_led_sample , pin led sample);
```

14 UART

14.1 UART Introduction

BK7251 has two sets of UART. The maximum baud rate can be up to 6 Mbps. It supports 5, 6, 7 and 8 bits data mode, and supports even, odd or none parity check. The stop bit can be either 1 or 2 bits. The UART1 supports hardware and software flow control with RTS and CTS signal.

14.2 UART Related API

UART driver has been associated with the standard set of device operation functions of RTOS.So call RTOS standard device operation interface directly.

function	description
rt_device_control()	control device

14.2.1 control device

```
rt_err_t rt_device_control(rt_device_t dev, rt_uint8_t cmd, void* arg);
```

parameters	description
dev	device handle
cmd	commande control,refer to macro
arg	control parameter:type: struct serial_configure
return	RT_EOK:success ; others:fail

14.3 UART Struct Reference

serial:rt_device struct

serial_configure：

rt_uint32_t baud_rate	baud rate,defult:115200
rt_uint32_t data_bits	data bit,default:8bit
rt_uint32_t stop_bits	stop bit,default:1
rt_uint32_t parity	parity bit:no parity
rt_uint32_t bit_order	litter endian
rt_uint32_t invert	mode invert:not invert
rt_uint32_t bufsz	receive buffer size
rt_uint32_t reserved	reserve

14.4 UART Struct Macros

BK7251SDK support default macros:

#define	**BAUD_RATE_115200**	**115200**	baud rate:115200
#define	**DATA_BITS_8**	**8**	data bit:8bit
#define	**STOP_BITS_1**	**1**	stop bit:1
#define	**PARITY_NONE**	**0**	no parity
#define	**BIT_ORDER_LSB**	**0**	litter endian
#define	**NRZ_NORMAL**	**0**	mode invert:not invert
#define	**RT_SERIAL_RB_BUFSZ**	**64**	receive buffer size:64

configure device macros：

#define	**RT_DEVICE_CTRL_CONFIG 0x03**	configure corresponding device

baund rate set:

#define	**BAUD_RATE_2400**	2400
#define	**BAUD_RATE_4800**	4800
#define	**BAUD_RATE_9600**	9600
#define	**BAUD_RATE_19200**	19200
#define	**BAUD_RATE_38400**	38400
#define	**BAUD_RATE_57600**	57600
#define	**BAUD_RATE_115200**	115200
#define	**BAUD_RATE_203400**	203400
#define	**BAUD_RATE_460800**	460800
#define	**BAUD_RATE_921600**	921600
#define	**BAUD_RATE_2000000**	2000000
#define	**BAUD_RATE_3000000**	3000000

data bit set:

#define	**DATA_BITS_5**	5
#define	**DATA_BITS_6**	6
#define	**DATA_BITS_7**	7
#define	**DATA_BITS_8**	8
#define	**DATA_BITS_9**	9

stop bit set:

#define	**STOP_BITS_1**	0
#define	**STOP_BITS_2**	1
#define	**STOP_BITS_3**	2
#define	**STOP_BITS_4**	3

parity bit set:

#define	**PARITY_NONE**	0
#define	**PARITY_ODD**	1
#define	**PARITY_EVEN**	2

big/little endian set

#define	**BIT_ORDER_LSB**	0 little-endian
#define	**BIT_ORDER_MSB**	1 big-endian

mode select

#define	**NRZ_NORMAL**	0 normal mode
#define	**NRZ_INVERTED**	1 inverted mode

14.5 UART Sample Code

UART sample code refers to \test\uart_demo.c. Sample code describe how to use UART related APIs. Openning macro definitions : UART_DEMO to start test.

```
/*
* program     list: This is a sample code about using uart driver.
* command format: uart_sample uart2
* program function:input character string "hello RT-Thread!" ,then dislocation output.
*/
#include <rtthread.h>

#define SAMPLE_UART_NAME "uart2"
static struct rt_semaphore rx_sem;
static rt_device_t serial;
/* receive data callback function*/
static rt_err_t uart_input(rt_device_t dev, rt_size_t size) {
    /*interrupt occurs when serial port receives data, then send and receive signal amount by this call
  back*/
     rt_sem_release(&rx_sem);
    return RT_EOK;
}
static void serial_thread_entry(void *parameter) {
    char ch;
    while (1) {
        /* read data from uart*/
        while (rt_device_read(serial, -1, &ch, 1) != 1) {
            /* wait for semphore*/
            rt_sem_take(&rx_sem, RT_WAITING_FOREVER);
        }
        /* the read data is output by serial port misalignment */
        ch = ch + 1;
        rt_device_write(serial, 0, &ch, 1);
    }
}
static int uart_sample(int argc, char *argv[]) {
    rt_err_t ret = RT_EOK;
    char uart_name[RT_NAME_MAX];
    char str[] = "hello RT-Thread!\r\n";
    if (argc == 2) {
        rt_strncpy(uart_name, argv[1], RT_NAME_MAX);
    }
    else   {
        rt_strncpy(uart_name, SAMPLE_UART_NAME , RT_NAME_MAX);
    }
    /* find uart device*/
    serial = rt_device_find(uart_name);
    if (!serial) {
        rt_kprintf("find %s failed!\n", uart_name);
        return RT_ERROR;
    }
    /* init semphore*/
    rt_sem_init(&rx_sem, "rx_sem", 0, RT_IPC_FLAG_FIFO);
    /* opening serial port device in interrupt receiving and polling sending mode */
    rt_device_open(serial, RT_DEVICE_FLAG_INT_RX);
    /*setting receive callback function */
    rt_device_set_rx_indicate(serial, uart_input);
    /* send string*/
    rt_device_write(serial, 0, str, (sizeof(str) - 1));
    /* creat serial thread*/
    rt_thread_t thread = rt_thread_create("serial", serial_thread_entry , RT_NULL,1024, 25, 10);
    /* start thread*/
    if (thread != RT_NULL) {
        rt_thread_startup(thread);
    }
    else {
        ret = RT_ERROR;
    }
    return ret;
}
MSH_CMD_EXPORT(uart_sample, uart device sample);
```

15 List Player

15.1 List Player Introduction

List Player provides list creation and playback capabilities, and supports switching multiple lists.

15.2 List Player Related API

List player related APIs refer to \rt-thread\components\list_player.h, APIs are as follows：

function	description
list_player_init()	player init
list_player_current_item()	get current song handle
list_player_current_index()	get current song index
list_player_current_state()	get current player state
list_player_current_position()	get current song position
list_player_current_items()	get current items
list_player_is_exist()	query for players and lists
list_player_play()	play the specified lists
list_player_switch()	switch play lists
list_player_play_index()	play the specified index track in the specified list
list_player_play_item()	play the specified handle track in the specified list
list_player_stop()	stop
list_player_pause()	pause
list_player_resume()	resume
list_player_prev()	play last song
list_player_next()	play next song
list_player_detach_items()	delete playlist
list_player_set_mode()	set play mode
list_player_items_create()	creat playlist
list_player_items_delete()	delete playlist
list_player_items_empty()	clear the playlist
list_player_set_table_handler()	callback function at playlist completion
list_player_items_get_num()	number of songs in playlist
list_player_items_get_index()	index of current playlist in playlist
list_player_items_get_item()	current playlist in playlist
list_player_item_add()	add tracks to the specified playlist
list_player_item_del()	delete tracks from the specified playlist
list_player_item_del_by_index()	delete the specified index track from the specified playlist

list_player_item_get()	get the specified index track from the specified playlist
list_player_index_get()	get the specified track index from the specified playlist

15.2.1 player init

int list_player_init(void)

parameters	description
void	null
return	0:success ; -1:fail

15.2.2 get current song handle

music_item_t list_player_current_item(void);

parameters	description
void	null
return	!=0:the handle of the music being played; 0:not initialized/not associated music list/not play

15.2.3 get current song index

int list_player_current_index(void);

parameters	description
void	null
return	>=0:the index of music being played; -1:no music find

15.2.4 get current player state

int list_player_current_state(void);

parameters	description
void	null
return	player status; -1:not playing

15.2.5 get current song position

int list_player_current_position(void);

parameters	description
void	null
return	>=0:the position;　-1:table is a null

15.2.6 get current items

```
music_list_t   list_player_current_item(void);
```

parameters	description
void	null
return	!=0: handle of the music being played; 0:not initialized / not associated music list

15.2.7 query for players and lists

```
int list_player_is_exist (void);
```

parameters	description
void	null
return	0:there is no;　1:there are

15.2.8 play the specified lists

```
int list_player_play (music_list_t table);
```

parameters	description
table	specify playlist
return	0: playback successful;　-6:not initialized: -10:table is a null pointer;　-1:playback failed

15.2.9 switch play lists

Switch from the current list to the specified list.

```
int list_player_switch(music_list_t table, int index, int position, int state);
```

parameters	description
table	specify playlist
index	play index
position	play position
state	play state

return	-1:no player;　-2:no playlist;　0:success

15.2.10 play the specified index track in the specified list

```
int list_player_play_index(int index);
```

parameters	description
Index	specify index
return	0:success;　1:fail

15.2.11 play the specified track

```
int list_player_play_item(music_item_t item);
```

parameters	description
Item	specify item
return	0:play successfully;　1:fail

15.2.12 stop play

```
void list_player_stop(void);
```

parameters	description
void	null
return	null

15.2.13 pause play

```
void list_player_pause(void);
```

parameters	description
void	null
return	null

15.2.14 resume play

```
void list_player_resume(void);
```

parameters	description
void	null
return	null

15.2.15 play last song

```
void list_player_prev(void);
```

parameters	description
void	null
return	-1:no player; -2:no playlist; 0:success

15.2.16 play next song

```
void list_player_next(void);
```

parameters	description
void	null
return	null

15.2.17 delete playlist

delete playlist but not free list:

```
music_list_t list_player_detach_items (void);
```

parameters	description
void	null
返回	the delete list

15.2.18 clear the playlist

```
int list_player_empty(void);
```

parameters	description
void	null
return	-1: list player not initialized; 0:success

15.2.19 set play mode

```
int list_player_set_mode(int mode);
```

parameters	description
mode	play mode
return	-1:list player not initialized; 0:success

15.2.20 creat playlist

```
music_list_t list_player_items_create(void);
```

parameters	description
void	null
return	!=0:creat successful; 0:no memory

15.2.21 delete playlist

```
void list_player_items_delete(music_list_t table);
```

parameters	description
table	the handle of music list
return	null

15.2.22 register callback function at playlist completion

```
void list_player_set_table_handler(music_list_t table, list_event_handler handler, void *arg);
```

parameters	description
table	the handle of music list
handler	callback function
arg	callback function parameter
return	null

15.2.23 get number of songs in playlist

```
int list_player_items_get_num(music_list_t table);
```

parameters	description
table	specify playlist
return	-1:table is null; others:the number of musics

15.2.24 get index of current playlist in playlist

```
int list_player_items_get_index(music_list_t table);
```

parameters	description
table	the handle of music list
return	>=0:the number of music; -1:table is null

15.2.25 get current playlist in playlis

```
music_item_t list_player_items_get_item(music_list_t table);
```

parameters	description
table	the handle of music list
return	!=0:last music item; 0:table is null

15.2.26 add tracks to the specified playlist

```
int list_player_item_add(music_list_t table, music_item_t item, int index);
```

parameters	description
table	the handle of music list
item	the handle of music that need to be added
index	0:add successful; others:fail

15.2.27 delete tracks from the specified playlist

```
int list_player_item_del(music_list_t table, struct music_item *item);
```

parameters	description
table	the handle of music list
item	the handle of music that need to be deleted
index	0:delete successful; others:fail

15.2.28 delete the specified index track from the specified playlist

```
int list_player_item_del_by_index(music_list_t table, int index);
```

parameters	description
table	the handle of music list
index	the index of music that need to be deleted
return	0:delete successful; others:fail

15.2.29 get the specified index track from the specified playlist

```
music_item_t list_player_item_get(music_list_t table, int index);
```

parameters	description
table	the handle of music list
index	the index of music

return	!=0:get item successful; 0:no item

15.2.30 get the specified track index from the specified playlist

```
int list_player_index_get(music_list_t table, music_item_t item);
```

parameters	description
table	the handle of music list
item	the handle of music lisy
return	>=0:item index; others:fail

15.3 List Player Macros

set player mode:

#define LISTER_NONE_MODE (0x00)	no mode
#define LISTER_LIST_ONCE_MODE (0x01)	list playback
#define LISTER_SONG_ONCE_MODE (0x02)	single play
#define LISTER_LIST_REPEAT_MODE (0x03)	repeated playback of lists
#define LISTER_SONG_REPEAT_MODE (0x04)	single tune circulation

15.4 List Player Sample Code

List player sample code refers to \samples\Player\list_player_demo.c. Sample code describe how to use List player related APIs. Openning macro definitions : LIST_PLAY_TEST to start test.

```
/* program        list:   sample Local Player for file playing
  * command format:   list_player   http://192.168.44.23/Kiss_The_Rain.mp3
  * program function: The sample code implements generating playlists and playing songs.
  */

#include <rtthread.h>
#include "player.h"
#include "list_player.h"
#include "player_app.h"
#include <finsh.h>

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

/* structure for storing list information */
typedef struct play_list_struct{
        music_list_t which_playlist;
        int play_list_status;
        int play_list_position;
        int play_list_num;
        music_item_t play_list_content;
        char backup_url[128];
}play_list_struct;

/* save the current playlist information */
play_list_struct saved_list;
static void save_current_playlist_status(void)
{
        int state = list_player_current_state();
        int num = list_player_current_index();
        int postion    =list_player_current_position();
        music_list_t tmp_list =list_player_current_items();
        int list_num = list_player_items_get_num(tmp_list);
        music_item_t tmp = list_player_current_item();

        saved_list.which_playlist = tmp_list;
        saved_list.play_list_status = state;
        saved_list.play_list_num = num;
        saved_list.play_list_position = postion;
        saved_list.play_list_content = tmp;
}

/* restore the saved playlist and play it */
static void bell_list_handle(void)
{
     list_player_switch(saved_list.which_playlist,
                saved_list.play_list_num,
                saved_list.play_list_position,
                saved_list.play_list_status);
}

/* generate playlists, add songs and play them */
music_list_t song_list = NULL;
int list_player(int argc, char** argv)
{
        struct music_item items = {0};
        items.name = ("Stream");

        items.URL = argv[1];

         /* generate playlists */
        if (!song_list)
        {
              song_list =list_player_items_create();
        }

         /*setting player mode */
        list_player_mode_set(LISTER_LIST_ONCE_MODE);
         /* adding songs */
        list_player_item_add(song_list, &items,-1);
         /* play songs in playlist */
        list_player_play(song_list);

       rt_kprintf("list player test\r\n");
}

/*generate a list of prompt tones, interrupt the current song playing, play the prompt tone, and return to
  the song playing after the end of the prompt tone */
music_list_t bell_list = NULL;
int bell_player(int argc, char** argv)
{
       struct music_item items = {0};
        items.name = ("Bell");

        items.URL = argv[1];

       /* save the current playlist */
       save_current_playlist_status();

        /* generate a list of prompt sounds */
        if (!bell_list)
        {
               bell_list =list_player_items_create();
        }

       /* setting player mode */
```

```
        list_player_mode_set(LISTER_LIST_ONCE_MODE);
        /* add songs */
         list_player_item_add(bell_list, &items,-1);
        /* switch player to new list and play */
         list_player_switch(bell_list,0,0,PLAYER_STAT_PLAYING);
        /* pre-restore playback */
        list_player_set_table_handler(bell_list,bell_list_handle,NULL);
}

MSH_CMD_EXPORT(list_player, list_player test);
MSH_CMD_EXPORT(bell_player, bell_player test);
```

16 Network Interface

16.1 Network Interface Introduction

BK7251 network interface provided for the upper application is used for:1.start station mode,connecting the specified network.2.close station 3.start AP mode.4.close AP.5.start monitor mode.6.close monitor mode.7.get ip/link status.8,set channel.9.star t scan,and get scan result.

16.2 Network Interface Related API

Network interface related APIs refer to beken378\func\include\wlan_ui_pub.h. APIs are as follows：

function	description
bk_wlan_start()	start station and AP
bk_wlan_start_sta_adv()	start station fast connection
bk_wlan_stop()	close station and AP
bk_wlan_start_scan()	start scan
bk_wlan_scan_ap_reg_cb()	callback function after registration scan
bk_wlan_start_assign_scan()	scan specific networks
bk_wlan_start_monitor()	start monitor
bk_wlan_stop_monitor()	close monitor
bk_wlan_register_monitor_cb()	register monitor callback function
bk_wlan_get_ip_status()	get ip status
bk_wlan_get_link_status()	get link status
bk_wlan_get_channel()	get current channel
bk_wlan_set_channel()	set channel

16.2.1 start wlan

Start the network after the upper layer get SSID and password.

```
OSStatus bk_wlan_start(network_InitTypeDef_st *inNetworkInitPara);
```

parameters	description
inNetworkInitPara	specifies wlan parameters
return	kNoErr :success; others:fail

16.2.2 start station fast connection

```
OSStatus bk_wlan_start_sta_adv(network_InitTypeDef_adv_st *inNetworkInitParaAdv);
```

parameters	description

inNetworkInitParaAdv	specifies the precise wlan parameters
return	kNoErr :success; others:fail

16.2.3 close station and AP

```
int bk_wlan_stop(char mode);
```

parameters	**description**
mode	the mode need to be closed
return	kNoErr :success; others:fail

16.2.4 start scan

```
void bk_wlan_start_scan(void);
```

parameters	**description**
void	null
return	null

16.2.5 callback function after registration scan

```
void bk_wlan_scan_ap_reg_cb(FUNC_2PARAM_PTR ind_cb);
```

parameters	**description**
ind_cb	callback function after scan. function define: typedef void (*FUNC_2PARAM_PTR)(void *arg, uint8_t vif_idx);
return	null

16.2.6 scan specific networks

```
void bk_wlan_start_assign_scan(UINT8 **ssid_ary, UINT8 ssid_num);
```

parameters	**description**
ssid_ary	SSID of the specified network
ssid_num	number of the specified network
return	null

16.2.7 start monitor

```
int bk_wlan_start_monitor(void);
```

parameters	description
void	null
return	kNoErr :success; others:fail

16.2.8 close monitor

```
int bk_wlan_stop_monitor(void);
```

parameters	description
void	null
return	kNoErr :success; others:fail

16.2.9 register monitor callback function

```
void bk_wlan_register_monitor_cb(monitor_data_cb_t fn);
```

parameters	description
fn	registered callback function.function define: typedef void (*monitor_data_cb_t)(uint8_t *data, int len, hal_wifi_link_info_t *info);
return	null

16.2.10 get ip status

```
OSStatus bk_wlan_get_ip_status(IPStatusTypedef *outNetpara, WiFi_Interface inInterface);
```

parameters	description
outNetpara	architecture to save the acquired network state
inInterface	mode that require access to network state
return	kNoErr :success; others:fail

16.2.11 get link status

```
OSStatus bk_wlan_get_link_status(LinkStatusTypeDef *outStatus);
```

parameters	description
outStatus	save the acquired connection status
return	kNoErr :success; others:fail

16.2.12 get current channel

```
int bk_wlan_get_channel(void);
```

parameters	description
void	null
return	channel

16.2.13 set channel

```
int bk_wlan_set_channel(int channel);
```

parameters	description
channel	input channel number
return	kNoErr :success; others:fail

16.3 Network Interface Struct Reference

network_InitTypeDef_st：

wifi_mode	DHCP mode
wifi_ssid	SSID of the wlan needs to be connected
wifi_key	security key of the wlan needs to be connected
local_ip_addr	static IP configuration, local IP address
net_mask	static IP configuration, netmask
gateway_ip_addr	static IP configuration, router IP address
dns_server_ip_addr	static IP configuration, DNS server IP address
dhcp_mode	DHCP mode

network_InitTypeDef_adv_st:

ap_info	network information that need to be fast connected
key	security key or PMK of the wlan
key_len	the length of network key
local_ip_addr	static IP configuration, Local IP address
net_mask	static IP configuration, Netmask
gateway_ip_addr	static IP configuration, Router IP address
dns_server_ip_addr	static IP configuration, DNS server IP address
dhcp_mode	DHCP mode

apinfo_adv_t:

ssid	SSID that need to be fast connected
bssid	BSSID that need to be fast connected
channel	network channel that need to be fast connected

security	network encryption that need to be fast connected

IPStatusTypedef:

dhcp	DHCP mode
ip	local IP address on the target wlan interface
gate	router IP address on the target wlan interface
mask	netmask on the target wlan interface
dns	DNS server IP address
mac	MAC address
broadcastip	obtained broadcastip IP

LinkStatusTypeDef:

conn_state	the link to wlan is established or not
wifi_strength	Signal strength of the current connected AP
ssid	SSID of the current connected wlan
bssid	BSSID of the current connected wlan
channel	channel of the current connected wlan
security	the encryption mode of current network

16.4 Network Interface Enumeration

network mode:

```
typedef enum
{
    SOFT_AP,                                  /*AP mode*/
    STATION                                   /*STATION mode*/
} wlanInterfaceTypedef;
```

connection status description:

```
typedef enum {
     MSG_IDLE = 0,                    /*not connection*/
     MSG_CONNECTING,                  /*is connecting*/
     MSG_PASSWD_WRONG,                /*password wrong*/
     MSG_NO_AP_FOUND,                 /*nof find ap*/
     MSG_CONN_FAIL,                   /*connect fail*/
     MSG_CONN_SUCCESS,                /*connect success*/
     MSG_GOT_IP,                      /*get IP*/
}msg_sta_states;
```

encryption mode

```
enum wlan_sec_type_e
{
    SECURITY_TYPE_NONE,              /* Open system */
    SECURITY_TYPE_WEP,               /* Wired Equivalent Privacy. WEP security. */
    SECURITY_TYPE_WPA_TKIP,          /* WPA /w TKIP */
    SECURITY_TYPE_WPA_AES,           /* WPA /w AES */
    SECURITY_TYPE_WPA2_TKIP,         /* WPA2 /w TKIP */
    SECURITY_TYPE_WPA2_AES,          /* WPA2 /w AES */
    SECURITY_TYPE_WPA2_MIXED,        /* WPA2 /w AES or TKIP */
    SECURITY_TYPE_AUTO,              /* It is used when calling */
};
```

16.5 Network Interface Macros

DHCP mode：

#define DHCP_DISABLE (0)	/*DHCP disable */
#define DHCP_CLIENT (1)	/*DHCP client mode*/
#define DHCP_SERVER (2)	/* DHCP server mode*/

16.6 Network Interface Sample Code

start a station connection

```
void demo_sta_app_init(char *oob_ssid,char *connect_key)
{
     /* define a structure for passing in parameters */
    network_InitTypeDef_st wNetConfig;
    int len;
    /*clear structure*/
    os_memset(&wNetConfig, 0x0, sizeof(network_InitTypeDef_st));

    /* check the length of SSID, not more than 32 bytes */
    len = os_strlen(oob_ssid);
    if(SSID_MAX_LEN < len)
    {
        bk_printf("ssid name more than 32 Bytes\r\n");
        return;
    }

    /* pass SSID and password into the structure */
     os_strcpy((char *)wNetConfig.wifi_ssid, oob_ssid);
     os_strcpy((char *)wNetConfig.wifi_key, connect_key);

    /* current client mode */
     wNetConfig.wifi_mode = STATION;
    /* using DHCP CLIENT to obtain IP address dynamically from router */
     wNetConfig.dhcp_mode = DHCP_CLIENT;
     wNetConfig.wifi_retry_interval = 100;

     bk_printf("ssid:%s key:%s\r\n", wNetConfig.wifi_ssid, wNetConfig.wifi_key);
    /*start wifi connection*/
     bk_wlan_start(&wNetConfig);
}
```

start AP mode to provide other client connections：

```
void demo_softap_app_init(char *ap_ssid,char *ap_key)
{
   /* define a structure for passing in parameters */
    network_InitTypeDef_adv_st     wNetConfigAdv;
     int len;
          /*clear structure*/
    os_memset( &wNetConfigAdv, 0x0, sizeof(network_InitTypeDef_adv_st) );
    len = os_strlen(ap_ssid);
     if(SSID_MAX_LEN < len)
     {
          bk_printf("ssid name more than 32 Bytes\r\n");
          return;
     }
    /*pass ap ssid and ap key*/
   os_strcpy((char *)wNetConfig.wifi_ssid, ap_ssid);
   os_strcpy((char *)wNetConfig.wifi_key, ap_key);

   /* current client mode */
   wNetConfig.wifi_mode = SOFT_AP;
   /* DHCP SERVER mode，needs assign a static address as local address */
   wNetConfig.dhcp_mode = DHCP_SERVER;
   wNetConfig.wifi_retry_interval = 100;
   os_strcpy((char *)wNetConfig.local_ip_addr, WLAN_DEFAULT_IP);
   os_strcpy((char *)wNetConfig.net_mask, WLAN_DEFAULT_MASK);
   os_strcpy((char *)wNetConfig.dns_server_ip_addr, WLAN_DEFAULT_IP);
```

```
        bk_printf("ssid:%s    key:%s\r\n", wNetConfig.wifi_ssid, wNetConfig.wifi_key);
        /*start ap*/
         bk_wlan_start(&wNetConfig);}
```

start station fast connection：

```
void demo_sta_adv_app_init(char *oob_ssid,char *connect_key)
{
        /* define a structure for passing in parameters */
         network_InitTypeDef_adv_st     wNetConfigAdv;
        /* clear structure */
         os_memset( &wNetConfigAdv, 0x0, sizeof(network_InitTypeDef_adv_st) );
         /* pass SSID */
         os_strcpy((char*)wNetConfigAdv.ap_info.ssid, oob_ssid);
        /* pass bssid */
         hwaddr_aton("12:34:56:00:00:01", wNetConfigAdv.ap_info.bssid);
        /* encryption to connect to the network. Specific parameters refer to the description of the
        structure.*/
         wNetConfigAdv.ap_info.security = SECURITY_TYPE_WPA2_MIXED;
        /* channels of the network to be connected */
         wNetConfigAdv.ap_info.channel = 11;
         /* network password to connect and password length */
         os_strcpy((char*)wNetConfigAdv.key, connect_key);
         wNetConfigAdv.key_len = os_strlen(connect_key);
        /* getting network information such as IP address by DHCP */
         wNetConfigAdv.dhcp_mode = DHCP_CLIENT;
         wNetConfigAdv.wifi_retry_interval = 100;
        /* start fast connection */
         bk_wlan_start_sta_adv(&wNetConfigAdv);
}
```

start scan，and analyze the results of scan：

```
/*callback function*/
static void scan_cb(void *ctxt, uint8_t param)
{
        /* pointer to scan result */
        struct scanu_rst_upload *scan_rst;
        /* structures that preserve analytical results */
        ScanResult apList;
        int i;
```

```
    apList.ApList = NULL;
    /*start scan*/
    scan_rst = sr_get_scan_results();

    if( scan_rst == NULL )
    {
        apList.ApNum = 0;
        return;
    }
    else
    {
        apList.ApNum = scan_rst->scanu_num;
    }
    if( apList.ApNum > 0 )
    {
        /* apply the corresponding memory to save the result of scan */
        apList.ApList = (void *)os_malloc(sizeof(*apList.ApList) * apList.ApNum);
        for( i = 0; i < scan_rst->scanu_num; i++ )
        {
            /*save the ssid and rssi that scaned*/
            os_memcpy(apList.ApList[i].ssid, scan_rst->res[i]->ssid, 32);
            apList.ApList[i].ApPower = scan_rst->res[i]->level;
        }
    }
    if( apList.ApList == NULL )
    {
        apList.ApNum = 0;
    }
     /*print the result of scan*/
     bk_printf("Got ap count: %d\r\n", apList.ApNum);
     for( i = 0; i < apList.ApNum; i++ )
     {
          if(os_strlen(apList.ApList[i].ssid) >= SSID_MAX_LEN)
          {
               char temp_ssid[33];
               os_memset(temp_ssid, 0, 33);
               os_memcpy(temp_ssid, apList.ApList[i].ssid, 32);
               bk_printf("    %s, RSSI=%d\r\n", temp_ssid, apList.ApList[i].ApPower);
          }
          else
```

```
                {
                        bk_printf("       %s, RSSI=%d\r\n", apList.ApList[i].ssid, apList.ApList[i].ApPower);
                }
        }
        bk_printf("Get ap end.......\r\n\r\n");

       if( apList.ApList != NULL )
       {
              os_free(apList.ApList);
              apList.ApList = NULL;
       }

#if CFG_ROLE_LAUNCH
        rl_pre_sta_set_status(RL_STATUS_STA_LAUNCHED);
#endif
       sr_release_scan_results(scan_rst);
}

void demo_scan_app_init(void)
{
       /*register scan callback function*/
        mhdr_scanu_reg_cb(scan_cb, 0);
       /*start scan*/
        bk_wlan_start_scan();
}
```

When the connection is successful, get the network status after the connection

```
void demo_ip_app_init(void)
{
       /* define a structure for saving network state */
       IPStatusTypedef ipStatus;
       /* clear structure */
        os_memset(&ipStatus, 0x0, sizeof(IPStatusTypedef));
       /* get the network state and save it in the structure */
        bk_wlan_get_ip_status(&ipStatus, STATION);
       /* print network status    */
        bk_printf("dhcp=%d ip=%s gate=%s mask=%s mac=" MACSTR "\r\n",
                              ipStatus.dhcp, ipStatus.ip, ipStatus.gate,
                              ipStatus.mask, MAC2STR((unsigned char*)ipStatus.mac));
}
```

When the connection is successful, get the connection status:

```
void demo_state_app_init(void)
{
    /* define structure to save connection state */
     LinkStatusTypeDef linkStatus;
    network_InitTypeDef_ap_st ap_info;
    char ssid[33] = {0};
    #if CFG_IEEE80211N
        bk_printf("sta: %d, softap: %d, b/g/n\r\n",sta_ip_is_start(),uap_ip_is_start());
    #else
        bk_printf("sta: %d, softap: %d, b/g\r\n",sta_ip_is_start(),uap_ip_is_start());
    #endif



    if( sta_ip_is_start() )
    {
     os_memset(&linkStatus, 0x0, sizeof(LinkStatusTypeDef));

     bk_wlan_get_link_status(&linkStatus);

    os_memcpy(ssid, linkStatus.ssid, 32);

     bk_printf("sta:rssi=%d,ssid=%s,bssid=" MACSTR ",channel=%d,cipher_type:",
          linkStatus.wifi_strength, ssid, MAC2STR(linkStatus.bssid), linkStatus.channel);
        switch(bk_sta_cipher_type())
        {
            case SECURITY_TYPE_NONE:
                bk_printf("OPEN\r\n");
                break;
             case SECURITY_TYPE_WEP :
                bk_printf("WEP\r\n");
                break;
             case SECURITY_TYPE_WPA_TKIP:
                bk_printf("TKIP\r\n");
                break;
             case SECURITY_TYPE_WPA2_AES:
                bk_printf("CCMP\r\n");
                break;
             case SECURITY_TYPE_WPA2_MIXED:
```

```
                        bk_printf("MIXED\r\n");
                        break;
                  case SECURITY_TYPE_AUTO:
                        bk_printf("AUTO\r\n");
                        break;
                  default:
                        bk_printf("Error\r\n");
                        break;
            }
      }

      if( uap_ip_is_start() )
      {
       os_memset(&ap_info, 0x0, sizeof(network_InitTypeDef_ap_st));
       bk_wlan_ap_para_info_get(&ap_info);
      os_memcpy(ssid, ap_info.wifi_ssid, 32);
      bk_printf("softap:ssid=%s,channel=%d,dhcp=%d,cipher_type:",
      ssid, ap_info.channel,ap_info.dhcp_mode);
      switch(ap_info.security)
            {
                 case SECURITY_TYPE_NONE:
                        bk_printf("OPEN\r\n");
                        break;
                  case SECURITY_TYPE_WEP :
                        bk_printf("WEP\r\n");
                        break;
                  case SECURITY_TYPE_WPA_TKIP:
                        bk_printf("TKIP\r\n");
                        break;
                  case SECURITY_TYPE_WPA2_AES:
                        bk_printf("CCMP\r\n");
                        break;
                  case SECURITY_TYPE_WPA2_MIXED:
                        bk_printf("MIXED\r\n");
                        break;
                  case SECURITY_TYPE_AUTO:
                        bk_printf("AUTO\r\n");
                        break;
                  default:
                        bk_printf("Error\r\n");
```

```
                    break;
            }
            bk_printf("ip=%s,gate=%s,mask=%s,dns=%s\r\n",
                    ap_info.local_ip_addr, ap_info.gateway_ip_addr, ap_info.net_mask,
ap_info.dns_server_ip_addr);
    }
}



/* monitor callback function*/
void bk_demo_monitor_cb(uint8_t *data, int len, hal_wifi_link_info_t *info)
{
    os_printf("len:%d\r\n", len);

    //Only for reference
    /*
    User can get ssid and key by prase monitor data,
    refer to the following code, which is the way airkiss
    use monitor get wifi info from data
    */
#if 0
    int airkiss_recv_ret;
    airkiss_recv_ret = airkiss_recv(ak_contex, data, len);
#endif

}
/* program    list: This is a sample code about how to use network interface
* command format: wifi_demo   sta oob_ssid   connect_key
* program function: input corresponding command let the chip connect net.
*/
int wifi_demo(int argc, char **argv)
{
    char *oob_ssid = NULL;
    char *connect_key;

    if (strcmp(argv[1], "sta") == 0)
    {
        os_printf("sta_Command\r\n");
        if (argc == 3)
        {
                oob_ssid = argv[2];
                connect_key = "1";
            }
            else if (argc == 4)
            {
                oob_ssid = argv[2];
                connect_key = argv[3];
            }
            else
            {
                os_printf("parameter invalid\r\n");
                return -1;
            }
            if(oob_ssid)
            {
                demo_sta_app_init(oob_ssid, connect_key);
            }
            return 0;
    }

     if(strcmp(argv[1], "adv") == 0)
     {
          os_printf("sta_adv_Command\r\n");
          if (argc == 3)
          {
               oob_ssid = argv[1];
               connect_key = "1";
          }
          else if (argc == 4)
          {
               oob_ssid = argv[1];
               connect_key = argv[2];
          }
          else
          {
               os_printf("parameter invalid\r\n");
               return -1;
          }
          if(oob_ssid)
          {
                demo_sta_adv_app_init(oob_ssid, connect_key);
            }
            return 0;
    }
    if(strcmp(argv[1], "softap") == 0)
    {

            os_printf("SOFTAP_COMMAND\r\n\r\n");
            if (argc == 3)
            {
                    oob_ssid = argv[1];
                    connect_key = "1";
            }
            else if (argc == 4)
            {
                    oob_ssid = argv[1];
                    connect_key = argv[2];
            }
            else
            {
                os_printf("parameter invalid\r\n");
                return -1;
            }

            if(oob_ssid)
            {
                    demo_softap_app_init(oob_ssid, connect_key);
            }
            return 0;
    }
    if(strcmp(argv[1], "monitor") == 0)
    {
            if(argc != 3)
            {
                    os_printf("parameter invalid\r\n");
            }
            if(strcmp(argv[2], "start") == 0)
            {
                    bk_wlan_register_monitor_cb(bk_demo_monitor_cb);
                    bk_wlan_start_monitor();
```

```
        }
        else if(strcmp(argv[2], "stop") == 0)
        {
            bk_wlan_stop_monitor();
        }
        else
        {
            os_printf("parameter invalid\r\n");
        }
    }
    return 0;
}
MSH_CMD_EXPORT(wifi_demo, wifi_demo command);
```

17 RTOS Interface

17.1 RTOS Interface Introduction

RTOS interface supports the operation about thread/timer/semaphore/mutex/quenene.

17.2 RTOS Related APIs

RTOS related APIs refer to beken378\rttos\include\bk_rtos_pub.h. APIs are as follows：

function	description
bk_rtos_create_thread()	create and starts a new thread
bk_rtos_delete_thread()	delete a terminated thread
bk_rtos_thread_join()	sleep until another thread has terminated
bk_rtos_thread_sleep()	suspend current thread for a specific time
bk_rtos_init_semaphore()	initialize semaphore and set count to 0
bk_rtos_set_semaphore()	set a semaphore
bk_rtos_get_semaphore()	get a semaphore
bk_rtos_deinit_semaphore()	deinit a semaphore
bk_rtos_init_mutex()	initialize a mutex
bk_rtos_lock_mutex()	obtain the lock on a mutex
bk_rtos_unlock_mutex()	release the lock on a mutex
bk_rtos_deinit_mutex()	de-initialize a mutex
bk_rtos_init_queue()	initialize a FIFO queue
bk_rtos_push_to_queue()	push an object onto a queue
bk_rtos_pop_from_queue()	pop an object off a queue
bk_rtos_deinit_queue()	de-initialize a queue
bk_rtos_is_queue_empty()	check if a queue is empty
bk_rtos_is_queue_full()	check if a queue is full
bk_rtos_init_timer()	initialize a RTOS timer
bk_rtos_start_timer()	start a RTOS timer running
bk_rtos_stop_timer()	stop a running RTOS timer
bk_rtos_reload_timer()	reload a RTOS timer that has expired
bk_rtos_deinit_timer()	de-initialize a RTOS timer
bk_rtos_is_timer_running()	get whether the timer is running

17.2.1 creat a thread

```
OSStatus bk_rtos_create_thread( beken_thread_t* thread,
                                uint8_t priority,
                                const char* name,
                                beken_thread_function_t function,
                                uint32_t stack_size,
                                beken_thread_arg_t arg );
```

function	description
thread	Pointer to variable that will receive the thread handle
priority	priority number
name	thread name
function	the main thread function
stack_size	stack size for this thread
arg	argument which will be passed to thread function
return	kNoErr :success; others:fail

17.2.2 delete a terminated thread

```
OSStatus bk_rtos_delete_thread( beken_thread_t* thread );
```

parameters	description
thread	the handle of the thread to delete
return	kNoErr :success; others:fail

17.2.3 sleep until another thread has terminated

```
OSStatus bk_rtos_thread_join(beken_thread_t* thread);
```

parameters	description
thread	the handle of the other thread which will terminate
return	kNoErr :success; others:fail

17.2.4 suspend current thread for a specific time

```
void bk_rtos_thread_sleep(uint32_t seconds);
```

parameters	description
seconds	seconds : a time interval (unit: seconds)
return	null

17.2.5 initialize semaphore and set count to 0

```
OSStatus bk_rtos_init_semaphore( beken_semaphore_t* semaphore, int maxCount );
```

parameters	description
semaphore	a pointer to the semaphore handle to be initialised
maxCount	the max count number of this semaphore
return	kNoErr :success;　others:fail

17.2.6 set a semaphore

```
int bk_rtos_set_semaphore( beken_semaphore_t* semaphore );
```

parameters	description
semaphore	a pointer to the semaphore handle to be set
return	kNoErr :success;　others:fail

17.2.7 get a semaphore

```
OSStatus bk_rtos_get_semaphore( beken_semaphore_t* semaphore, uint32_t timeout_ms );
```

parameters	description
semaphore	a pointer to the semaphore handle to be set
timeout_ms	the number of milliseconds to wait before returning
return	kNoErr :success;　others:fail

17.2.8 deinit a semaphore

```
OSStatus bk_rtos_deinit_semaphore( beken_semaphore_t* semaphore );
```

parameters	description
semaphore	a pointer to the semaphore handle to be set
return	kNoErr :success;　others:fail

17.2.9 initialize a mutex

```
OSStatus bk_rtos_init_mutex( beken_mutex_t* mutex );
```

parameters	description
mutex	a pointer to the mutex handle to be initialised
return	kNoErr :success;　others:fail

17.2.10 obtain the lock on a mutex

```
OSStatus bk_rtos_lock_mutex( beken_mutex_t* mutex );
```

parameters	description
mutex	a pointer to the mutex handle to be locked
return	kNoErr :success; others:fail

17.2.11 release the lock on a mutex

```
OSStatus bk_rtos_unlock_mutex( beken_mutex_t* mutex );
```

parameters	description
mutex	a pointer to the mutex handle to be unlocked
return	kNoErr :success; others:fail

17.2.12 de-initialize a mutex

```
OSStatus bk_rtos_deinit_mutex( beken_mutex_t* mutex );
```

parameters	description
mutex	a pointer to the mutex handle
return	kNoErr :success; others:fail

17.2.13 initialize a FIFO queue

```
OSStatus bk_rtos_init_queue( beken_queue_t* queue,
                             const char* name,
                             uint32_t message_size,
                             uint32_t number_of_messages );
```

parameters	description
queue	a pointer to the queue handle to be initialised
name	name for the queue
message_size	size in bytes of objects that will be held in the queue
number_of_messages	depth of the queue
return	kNoErr :success; others:fail

17.2.14 push an object onto a queue

```
OSStatus bk_rtos_push_to_queue( beken_queue_t* queue,
                                void* message,
                                uint32_t timeout_ms );
```

parameters	description
queue	a pointer to the queue handle
message	the object to be added to the queue
timeout_ms	the number of milliseconds to wait before returning
return	kNoErr :success; others:fail

17.2.15 pop an object off a queue

```
OSStatus bk_rtos_pop_from_queue( beken_queue_t* queue,
                                 void* message,
                                 uint32_t timeout_ms );
```

parameters	description
queue	a pointer to the queue handle
message	pointer to a buffer that will receive the object being popped off queue
timeout_ms	the number of milliseconds to wait before returning
return	kNoErr :success; others:fail

17.2.16 de-initialize a queue

```
OSStatus bk_rtos_deinit_queue( beken_queue_t* queue );
```

parameters	description
queue	a pointer to the queue handle
return	kNoErr :success; others:fail

17.2.17 check if a queue is empty

```
BOOL bk_rtos_is_queue_empty( beken_queue_t* queue );
```

parameters	description
queue	a pointer to the queue handle
return	ture: empty; false:not empty

17.2.18 check if a queue is full

```
BOOL bk_rtos_is_queue_full( beken_queue_t* queue );
```

parameters	description

queue	a pointer to the queue handle
return	ture: full; false:not full

17.2.19 initialize a RTOS timer

```
OSStatus bk_rtos_init_timer( beken_timer_t *timer,
                             uint32_t time_ms,
                             timer_handler_t function,
                             void* arg );
```

parameters	description
timer	a pointer to the timer handle to be initialised
time_ms	timer period in milliseconds
function	the callback handler function that is called each time the timer expires
arg	an argument that will be passed to the callback function
return	kNoErr :success; others:fail

17.2.20 start a RTOS timer running

```
OSStatus bk_rtos_start_timer( beken_timer_t* timer );
```

parameters	description
timer	a pointer to the timer handle to start
return	kNoErr :success; others:fail

17.2.21 stop a RTOS timer running

```
OSStatus bk_rtos_stop_timer( beken_timer_t* timer );
```

parameters	description
timer	a pointer to the timer handle to stop
return	kNoErr :success; others:fail

17.2.22 reload a RTOS timer that has expired

```
OSStatus bk_rtos_reload_timer( beken_timer_t* timer );
```

parameters	description
timer	a pointer to the timer handle to reload

return	kNoErr :success; others:fail

17.2.23 de-initialize a RTOS timer

```
OSStatus bk_rtos_deinit_timer( beken_timer_t* timer );
```

parameters	description
timer	a pointer to the RTOS timer handle
return	kNoErr :success; others:fail

17.2.24 get whether the timer is running

```
BOOL bk_rtos_is_timer_running( beken_timer_t* timer );
```

parameters	description
timer	a pointer to the RTOS timer handle
return	1 :runing; 0:not runing

17.3 RTOS Struct Reference

beken_timer_t:

void * handle	a pointer to the handle of timer
timer_handler_t function	callback function of timer
void * arg	the argument of timer

beken_worker_thread_t:

beken_thread_t thread	a pointer to thread
beken_queue_t event_queue	a popinter to event queue of thread

beken_timed_event_t:

event_handler_t function	the handle of event function
void * arg	the argument of event function
beken_timer_t timer	timer struct
beken_worker_thread_t* thread	work thread

beken2_timer_t:

void * handle	a pointer to the handle of timer
timer_2handler_t function	a callback fuction of timer event
void * left_arg	the first argument of callback fuction
void * right_arg	the second argument of callback fuction
uint32_t beken_magic	magic number

17.4 RTOS Enumeration

waiting event description:

```
typedef enum
{
      WAIT_FOR_ANY_EVENT,      /*any event can wake up*/
      WAIT_FOR_ALL_EVENTS,    /* all event can wake up */
} beken_event_flags_wait_option_t;
```

17.5 RTOS Macros

return value:

#define RTOS_SUCCESS	**(1)**	/*success*/
#define RTOS_FAILURE	**(0)**	/*fail*/

RTOS priority configuration：

#define BEKEN_DEFAULT_WORKER_PRIORITY	**(6)**	/*default priority :6*/
#define BEKEN_APPLICATION_PRIORITY	**(7)**	/*application priority7*/

RTOS time configuration

#define kNanosecondsPerSecond	**1000000000UUL**
#define kMicrosecondsPerSecond	**1000000UL**
#define kMillisecondsPerSecond	**1000**
#define NANOSECONDS	**1000000UL**
#define MICROSECONDS	**1000**
#define MILLISECONDS	**(1)**
#define SECONDS	**(1000)**
#define MINUTES	**(60 * SECONDS)**
#define HOURS	**(60 * MINUTES)**
#define DAYS	**(24 * HOURS)**

RTOS time configuration

#define BEKEN_NEVER_TIMEOUT	**(0xFFFFFFFF)**
#define BEKEN_WAIT_FOREVER	**(0xFFFFFFFF)**
#define BEKEN_NO_WAIT	**(0)**

17.6 RTOS Sample Code

RTOS sample code refers to \bk7251_sdk\beken378\demo\os_demo.c. Sample code describe how to use RTOS related APIs.

```
#include <rtthread.h>
#include "include.h"
#include "bk_rtos_pub.h"
#include "uart_pub.h"
#include "Error.h"
#include "portmacro.h"

#define      OS_THREAD_DEMO            1
#define      OS_MUTEX_DEMO             1
#define      OS_SEM_DEMO               1
#define      OS_QUEUE_DEMO             1
#define      OS_TIMER_DEMO             1

/*thread 0 prints a log and exits */
static void thread_0( beken_thread_arg_t arg )
{
    (void)( arg );

    os_printf( "This is thread 0\r\n");
    bk_rtos_delay_milliseconds((TickType_t)1000 );

    /* Make with terminate state and IDLE thread will clean resources */
    bk_rtos_delete_thread(NULL);
}
/* thread 1 creates a sub-thread,and the entry function is thread_0, and wait until the subthread exits*/
static void thread_1( beken_thread_arg_t arg )
{
    (void)( arg );
    OSStatus err = kNoErr;
    beken_thread_t t_handler = NULL;

    while ( 1 )
    {
        /* Create a new thread, and this thread will delete its self and clean its resource */
        err = bk_rtos_create_thread( &t_handler,
                                     BEKEN_APPLICATION_PRIORITY,
                                     "Thread 0",
                                     thread_0,
                                    0x400,
                                    0);
        if(err != kNoErr)
        {
            os_printf("ERROR: Unable to start the thread 1.\r\n" );
        }
        /* wait thread 0 delete it's self */
        bk_rtos_thread_join( &t_handler );
    }
}

/* thread 2 entry function, print a log.*/
static void thread_2( beken_thread_arg_t arg )
{
    (void)( arg );

    while ( 1 )
    {
        os_printf( "This is thread 2\r\n" );
        bk_rtos_delay_milliseconds((TickType_t)600);
    }
}

/*this sample code creat 2 threads*/
static int thread_demo_start( void )
{
    OSStatus err = kNoErr;
    /*define 2 handles of threads*/
    beken_thread_t t_handler1 = NULL, t_handler2 = NULL;

    os_printf("\r\n\r\noperating system thread demo............\r\n" );
    /*creat the first thread ,entry function is thread_1,no argument*/
    err = bk_rtos_create_thread( &t_handler1, BEKEN_APPLICATION_PRIORITY,
                                "Thread 1",
                                thread_1,
                                0x400,
                                0);
    if(err != kNoErr)
    {
        os_printf("ERROR: Unable to start the thread 1.\r\n" );
```

```
            goto exit;
        }
        /* creat the second thread ,entry function is thread_2,no argument */
        err = bk_rtos_create_thread( &t_handler2, BEKEN_APPLICATION_PRIORITY,
                                      "Thread 2",
                                      thread_2,
                                      0x400,
                                      0);
        if(err != kNoErr)
        {
            os_printf("ERROR: Unable to start the thread 2.\r\n" );
            goto exit;
        }

exit:
        if ( err != kNoErr )
        {
            os_printf( "Thread exit with err: %d", err );

            if(t_handler1 != NULL)
            {
                bk_rtos_delete_thread(t_handler1);
            }

            if(t_handler2 != NULL)
            {
                bk_rtos_delete_thread(t_handler2);
            }
        }

        return err;
}
```

the semaphore example: Start two threads, one for setting semaphores and one for acquiring semaphores.

```
static beken_semaphore_t os_sem = NULL;

static void set_semaphore_thread( beken_thread_arg_t arg )
{
        while ( 1 )
```

```
    {
        os_printf( "release semaphore!\r\n" );
        bk_rtos_set_semaphore( &os_sem );
        bk_rtos_delay_milliseconds( 500 );
    }

exit:
    if(os_sem)
    {
        bk_rtos_deinit_semaphore(&os_sem);
    }
    bk_rtos_delete_thread( NULL );
}
static void get_semaphore_thread( beken_thread_arg_t arg )
{
    OSStatus err;

    while(1)
    {
        err = bk_rtos_get_semaphore(&os_sem, BEKEN_NEVER_TIMEOUT);
        if(err == kNoErr)
        {
            os_printf("Get_Sem Succend!\r\n");
        }
        else
        {
            os_printf("Get_Sem Err:%d\r\n", err);
            goto exit;
        }
    }

exit:
    if(os_sem)
    {
        bk_rtos_deinit_semaphore(&os_sem);
    }
    bk_rtos_delete_thread( NULL );
}
staic int sem_demo_start ( void )
{
    OSStatus err = kNoErr;
    beken_thread_t t_handler1 = NULL, t_handler2 = NULL;
    os_printf( "test binary semaphore\r\n" );
    /*init semaphore os_sem*/
    err = bk_rtos_init_semaphore( &os_sem, 1 ); //0/1 binary semaphore || 0/N semaphore
    /*check initialization is successful*/
    if(err != kNoErr)
    {
        goto exit;
    }
    /*creat a thread to get semaphore */
    err = bk_rtos_create_thread(&t_handler1,
                                BEKEN_APPLICATION_PRIORITY,
                                "get_sem",
                                get_semaphore_thread,
                                0x500,
                                0 );
    if(err != kNoErr)
    {
        goto exit;
    }
    /* creat a thread to set semaphore */
    err = bk_rtos_create_thread(&t_handler2,
                                BEKEN_APPLICATION_PRIORITY,
                                "set_sem",
                                set_semaphore_thread,
                                0x500,
                                0 );
    if(err != kNoErr)
    {
        goto exit;
    }

    return err;
exit:
    if ( err != kNoErr )
    {
        os_printf( "Thread exit with err: %d\r\n", err );
    }
    return err;
```

```
}
```

the mutex example, 2 threads print different strings using the same entry function at the same time:

```
static beken_mutex_t os_mutex = NULL; /*define a mutex*/
/*print the string */
static OSStatus mutex_printf_msg(char *s)
{
    OSStatus err = kNoErr;
    if(os_mutex == NULL)
    {
        return -1;
    }
    /*application mutex before print*/
    err = bk_rtos_lock_mutex(&os_mutex);
    if(err != kNoErr)
    {
        return err;
    }
    os_printf( "%s\r\n", s);
    /*release mutex after print*/
    err = bk_rtos_unlock_mutex(&os_mutex);
    if(err != kNoErr)
    {
        return err;
    }
    return err;
}

static void os_mutex_sender_thread( beken_thread_arg_t arg )
{
    OSStatus err = kNoErr;
    char *taskname = (char *)arg;
    char strprt[100];
    int rd;
    while ( 1 )
    {
        rd = rand() & 0x1FF;
        sprintf(strprt, "%s , Rand:%d", taskname, rd);
        err = mutex_printf_msg(strprt);
```

```
            if(err != kNoErr)
            {
                os_printf( "%s printf_msg error!\r\n", taskname);
                goto exit;
            }
            bk_rtos_delay_milliseconds( rd );
        }

exit:
        if ( err != kNoErr )
        {
            os_printf( "Sender exit with err: %d\r\n", err );
        }
        if(os_mutex != NULL)
        {
            bk_rtos_deinit_mutex(&os_mutex);
        }
        bk_rtos_delete_thread( NULL );
}
static int mutex_demo_start( void )
{
        OSStatus err = kNoErr;
        beken_thread_t t_handler1 = NULL, t_handler2 = NULL;
        err = bk_rtos_init_mutex( &os_mutex );
        if(err != kNoErr)
        {
            os_printf( "rtos_init_mutex err: %d\r\n", err );
            goto exit;
        }
        /*thread1 ,send "my name is thread1"。*/
        err = bk_rtos_create_thread(&t_handler1,
                                    BEKEN_APPLICATION_PRIORITY,
                                    "sender1",
                                    os_mutex_sender_thread,
                                    0x400,
                                    "my name is thread1");
        if(err != kNoErr)
        {
            goto exit;
        }
```

```
    /* thread2 ,send "I'm is task!"*/
    err = bk_rtos_create_thread(&t_handler2,
                                BEKEN_APPLICATION_PRIORITY,
                                "sender2",
                                os_mutex_sender_thread,
                                0x400,
                                "I'm is task!" );
    if(err != kNoErr)
    {
        goto exit;
    }
exit:
    if ( err != kNoErr )
    {
        os_printf( "Thread exit with err: %d\r\n", err );
    }
    return err;
}
```

the queue example,remove and print the data pushed into the queue:

```
typedef struct _msg
{
    int value;
} msg_t;                          /* define the type of data object put into the queue */
static beken_queue_t os_queue = NULL;

static void receiver_thread( beken_thread_arg_t arg )
{
    OSStatus err;
    msg_t received = { 0 };

    while ( 1 )
    {
        /* wait until there is data in the queue and take it out */
        err = bk_rtos_pop_from_queue( &os_queue, &received, BEKEN_NEVER_TIMEOUT);
        /* check the return value to verify that it is correctly taken out*/
        if(err == kNoErr)
        {
            os_printf( "Received data from queue:value = %d\r\n", received.value );
        }
        else
        {
            os_printf("Received data from queue failed:Err = %d\r\n", err);
            goto exit;
        }
    }

exit:
    if ( err != kNoErr )
        os_printf( "Receiver exit with err: %d\r\n", err );

    bk_rtos_delete_thread( NULL );
}
/* pushes data objects into a queue */
static void sender_thread( beken_thread_arg_t arg )
{
    OSStatus err = kNoErr;

    msg_t my_message = { 0 };

    while ( 1 )
    {
        my_message.value++;
        /*push data objects into queue*/
        err = bk_rtos_push_to_queue(&os_queue, &my_message, BEKEN_NEVER_TIMEOUT);
        /*check return value*/
        if(err == kNoErr)
        {
            os_printf( "send data to queue\r\n" );
        }
        else
        {
            os_printf("send data to queue failed:Err = %d\r\n", err);
        }
        bk_rtos_delay_milliseconds( 100 );
    }

exit:
    if ( err != kNoErr )
    {
        os_printf( "Sender exit with err: %d\r\n", err );
    }

    bk_rtos_delete_thread( NULL );
}
/*rtos queue demo*/
static int queue_demo_start ( void )
{
    OSStatus err = kNoErr;
    /*init queue os_queue。*/
    beken_thread_t t_handler1 = NULL, t_handler2 = NULL;
    err = bk_rtos_init_queue( &os_queue, "queue", sizeof(msg_t), 3 );
    /* check if initialization is successful */
    if(err != kNoErr)
    {
        goto exit;
    }
    /*creat a thread to push data objects into queues*/
    err = bk_rtos_create_thread(&t_handler1,
                                BEKEN_APPLICATION_PRIORITY,
                                "sender",
                                sender_thread,
                                0x500,
                                0 );
    if(err != kNoErr)
    {
        goto exit;
    }
    /* creat a thread to remove data objects from the queue*/
    err = bk_rtos_create_thread(&t_handler2,
                                BEKEN_APPLICATION_PRIORITY,
                                "receiver",
                                receiver_thread,
                                0x500,
                                0 );
    if(err != kNoErr)
    {
        goto exit;
    }
```

```
exit:
        if ( err != kNoErr )
        {
                os_printf( "Thread exit with err: %d\r\n", err );
        }
        return err;
}
```

the timer example:

```
beken_timer_t timer_handle, timer_handle2;

static void destroy_timer( void )
{
        /* stop timer (timer_handle) */
        bk_rtos_stop_timer( &timer_handle );
        /*deinit timer(timer_handle) */
        bk_rtos_deinit_timer( &timer_handle );
        /* stop timer (timer_handle2) */
        bk_rtos_stop_timer( &timer_handle2 );
        /* deinit timer(timer_handle2) */
        bk_rtos_deinit_timer( &timer_handle2 );
}
static void timer_alarm( void *arg )
{
        os_printf("I'm timer_handle1\r\n");
}
static void timer2_alarm( void *arg )
{
        os_printf("I'm timer_handle2,destroy timer!\r\n");

        destroy_timer();
}
static int timer_demo_start ( void )
{
        OSStatus err = kNoErr;

        os_printf("timer demo\r\n");

        /* init timer_handle,timeout :500ms,callback function :timer_alarm。*/
        err = bk_rtos_init_timer(&timer_handle, 500, timer_alarm, 0);    ///500mS
        if(kNoErr != err)
                goto exit;

          err = bk_rtos_init_timer(&timer_handle2, 2600, timer2_alarm, 0);    ///2.6S
        if(kNoErr != err)
                goto exit;
        /* start timer (timer_handle) */
        err = bk_rtos_start_timer(&timer_handle);
        if(kNoErr != err)
                goto exit;
        /* start timer (timer_handle2) */
        err = bk_rtos_start_timer(&timer_handle2);
        if(kNoErr != err)
                goto exit;
        return err;

exit:
        if( err != kNoErr )
                os_printf( "os timer exit with err: %d", err );
        return err;
}
/* program          list:    this demo creat thread, semaphore, mutex, queue and timer
  *program    format:    os_demo    thread/mutex/queue/semaphore/timer
*/
static int os_demo(int argc, char **argv)
{

        if(strcmp(argv[1], "thread") == 0)
        {
#if OS_THREAD_DEMO
                  thread_demo_start();
#endif
        }
         else if(strcmp(argv[1], "mutex") == 0)
         {
#if OS_MUTEX_DEMO
                  mutex_demo_start();
#endif
         }
         else if(strcmp(argv[1], "queue") == 0)
```

```
    {
#if OS_QUEUE_DEMO
        queue_demo_start();
#endif
    }
    else if(strcmp(argv[1], "semaphore") == 0)
    {
#if OS_SEM_DEMO
        sem_demo_start();
#endif
    }
    else if(strcmp(argv[1], "timer") == 0)
    {
#if OS_TIMER_DEMO
        timer_demo_start();
#endif
    }
    else
    {
        os_printf("os demo %s dosn't support.\n", argv[1]);
    }
}
MSH_CMD_EXPORT(os_demo, os_demo command);
```

18 OTA

18.1 OTA Introduction

BK7251 supports upgrading firmware remotely from network. It downloads OTA (Over-the-Air Technology)firmware from server using HTTP protocol and burn it to download partition. Bootloader copies the firmware of the OTA partition to the app run partition and loads the new app partition firmware after the device restarts. The OTA firmware supports compression and encryption, and the OTA firmware is manufactured using rt_ota_pac.

18.2 OTA Related API

OTA related APIs refer to\rt-thread\samples\ota\http\http_client_ota.c. APIs are as follows:

function	description
fal_init()	fal init
http_ota_fw_download()	download firmware

18.2.1 fal initialization

It Initialize all devices and partitions and must be called before http_ota_fw_ download.

```
int fal_init(void);
```

parameters	description
void	null
return	total partition number

18.2.2 remote download firmware

Download the OTA firmware from the server and burn it into the download partition.

```
int http_ota_fw_download(const char *url);
```

parameters	description
url	the file address in http server
return	0

18.3 OTA Sample Code

OTA sample code refers to samples\ota\http\http_client_ota.c.

```
/*
 * program      list: This is a sample code about how to use OTA
 * command format: http_ota url
 * program function: Download remote firmware to download partition via OTA
 */

void http_ota(uint8_t argc, char **argv)
{
    int parts_num;
    parts_num = fal_init();     //fal init

    if (parts_num <= 0)
    {
        log_e("Initialize failed! Don't found the partition table.");
        return;
    }
    if (argc < 2)
    {
        rt_kprintf("using url: " HTTP_OTA_URL "\n");
        http_ota_fw_download(HTTP_OTA_URL);     //download firmware
    }
    else
    {
        http_ota_fw_download(argv[1]);
    }
}
/**
 * msh />http_ota [url]
*/
MSH_CMD_EXPORT(http_ota, OTA by http client: http_ota [url]);
```

19 Low Power Consumption

19.1 Low Power Consumption Introduction

BK7251 low power consumption includes MCU sleep, RF sleep and deep sleep. The waking up way from deep sleep includes RTC and GPIO.

19.2 Low Power Consumption Related API

Low power consumption related APIs refer to \beken378\func\include\wlan_ui_pub.h and manual_ps_pub.h. APIs are as follows:

function	description
bk_wlan_enter_powersave()	enter MCU/RF sleep mode
bk_enter_deep_sleep_mode()	deep_sleep mode

19.2.1 enter MCU/RF sleep mode

```
int bk_wlan_enter_powersave(struct rt_wlan_device *device, int level)；
```

parameters	description
struct rt_wlan_device *device	the handle of wlan device
level	mode : 0- mcu, rf do not sleep; 1- mcu sleep,rf not; 2- mcu do not sleep,rf sleep 3- mcu and rf sleep
return	0 :success; others:fail

19.2.2 deep_sleep mode

```
void bk_enter_deep_sleep_mode(PS_DEEP_CTRL_PARAM *deep_param)；
```

parameters	description
PS_DEEP_CTRL_PARAM *deep_param	parameters of deep_sleep
return	null

19.3 Low Power Consumption Struct Reference

PS_DEEP_CTRL_PARAM：

PS_DEEP_WAKEUP_WAY deep_wkway	the way of waking up from deep sleep
UINT32 gpio_index_map	The gpio bitmap which set 1 enable wakeup deep sleep.gpio_index_map is hex and every bits is map to gpio0-gpio31.

UINT32 gpio_edge_map	The gpio edge bitmap for wakeup gpios, gpio_edge_map is hex and every bits is map to gpio0-gpio31.(gpio1 as uart rx,must be 1)
UINT32 gpio_last_index_map	The gpio bitmap which set 1 enable wakeup deep sleep. low 8 bits of gpio_last_index_map is map to gpio32-gpio39.
UINT32 gpio_last_edge_map	The gpio edge bitmap for wakeup gpios, low 8 bits of gpio_last_edge_map is map to gpio32-gpio39.
UINT32 sleep_time	sleep time in waking up from rtc mode

19.4 Low Power Consumption Enumeration

It supports 3 ways of waking up from deep sleep.

```
typedef enum {
        PS_DEEP_WAKEUP_GPIO = 0,         //GPIO wake up
        PS_DEEP_WAKEUP_RTC = 1,          //RTC timer wake up mode
        PS_DEEP_WAKEUP_GPIO_RTC = 2, //GPIO and RTC all wake up mode
} PS_DEEP_WAKEUP_WAY;
```

19.5 Low Power Consumption Macros

#define	**CFG_USE_MCU_PS**	must be opened in low power consumption
#define	**CFG_USE_STA_PS**	using RF low power consumption mode

19.6 Low Power Consumption Sample Code

Mcu sleep and rf sleep refer to \test\test_pm.c ,deep sleep mode refer to \test deep_sleep.c. Sample code describe how to use low power consumption related APIs. Openning macro definitions : TEST_PM and TEST_DEEP_SLEEP to start test.

```
/*
* program        list: this is a sample code about low power consumption
* command format: input wifi ap and wifi w0 join wifiname password to connect net.
                           input pm_level    level to enter into low power consumption mode
                           deep sleep: input command: sleep_mode    1c    0    1c    0    10
* program function: realizing low power consumption and deep_sleep wake-up Function
*/
#include "error.h"
#include "include.h"
#include "arm_arch.h"
```

```
#include "gpio_pub.h"
#include "uart_pub.h"
#include "music_msg_pub.h"
#include "manual_ps_pub.h"

#include "co_list.h"
#include "saradc_pub.h"
#include "temp_detect_pub.h"
#include "sys_rtos.h"
#include "rtos_pub.h"
#include "saradc_intf.h"
#include "pwm_pub.h"
#include "pwm.h"
#include <stdint.h>
#include <stdlib.h>
#include <finsh.h>

/* mcu sleep and rf sleep mode*/
static int pm_level(int argc, char **argv)
{
    uint32_t level;
        if(argc != 2)
        {
            rt_kprintf("input argc is err!\n");
            return -1;
        }
        level = atoi(argv[1]);
        if(level > 3)
        {
           rt_kprintf("nonsupport level %d\n", level);
            return -1;
        }

        {
            struct rt_wlan_device *sta_device = (struct rt_wlan_device
*)rt_device_find(WIFI_DEVICE_STA_NAME);
            if (NULL != sta_device)
            {
                bk_wlan_enter_powersave(sta_device, level);
            }
        }
    return 0;
}

static void enter_deep_sleep_test(int argc,char **argv[])
{
    rt_thread_sleep(200);
    PS_DEEP_CTRL_PARAM deep_sleep_param;
    deep_sleep_param.deep_wkway                          = 0;
    deep_sleep_param.gpio_index_map             = atoi(argv[1]);
    deep_sleep_param.gpio_edge_map              = atoi(argv[2]);
    deep_sleep_param.gpio_last_index_map        = atoi(argv[3]);
    deep_sleep_param.gpio_last_edge_map         = atoi(argv[4]);
    deep_sleep_param.sleep_time                  = atoi(argv[5]);
    deep_sleep_param.deep_wkway                 = atoi(argv[6]);
    if(argc == 7)
    {
        rt_kprintf("enter enter_deep_sleep: 0x%0X 0x%0X 0x%0X 0x%0X %d %d\r\n",
                        deep_sleep_param.gpio_index_map,
                        deep_sleep_param.gpio_edge_map,
                        deep_sleep_param.gpio_last_index_map,
                        deep_sleep_param.gpio_last_edge_map,
                        deep_sleep_param.sleep_time,
                        deep_sleep_param.deep_wkway);
        bk_enter_deep_sleep_mode(&deep_sleep_param);
    }
    else
    {
        rt_kprintf(" argc error \r\n");
    }
}

FINSH_FUNCTION_EXPORT_ALIAS(enter_deep_sleep_test, __cmd_sleep_mode, test sleep
mode);
```

20 Bootloader

20.1 Bootloader Introduction

Bootloader is divided into two levels, one is L_boot and the other is UP_boot. First-level boot provides UART download function, second-level boot implements OTA function. Before using bootloader, you need to determine the partition according to the project and save the partition information to the original bootloader. bin

20.2 Settings of partition tables

20.2.1 Bootloader partition

Flash-name is beken_onchip_crc, offset = 0x10000. The actual physical address of the partition in flash is also 0, and the actual size of the partition: len = (60K*34)/32 = 65280 Byte;

20.2.2 App partition

The partition is application code. Flash-name is beken_onchip_crc, so logic address is offset = 0x10000. The actual physical address of the partition in FLASH is: (0x10000 * 34)/32 = 0x0011000, and the actual size of the partition: len = (1152K*34)/32 = 1224 K;

20.2.3 Download partition

The partition is an OTA download data store. Flash-name is beken_onchip_crc, the actual physical address of the partition in flash is offset = 0x143000, and the actual size of the partition is 748K.

In addition to the above three partitions, other partitions can be added as required. In addition, the starting address and length of bootloader partition can not be changed, and the starting address of APP partition can not be changed, but the length can be changed. The starting address and length of other partitions can be modified according to the actual situation of the scheme.

Take the 2M partition table information provided in BK7251 SDK as an example (partition_audio_2M.json).Its format is explained as follows:

field	description
name	partition name, the basis for finding partitions in firmware, cannot be repeated
flash_name	The name of the medium is usually FLASH. Usually are beken_onchip_crc and beken_on chip.beken_onchip_crc : its offset and len fields are represented by logical addresses beken_onchip: by physical address

offset	partition start address, hexadecimal representation
len	partition length, decimal representation

20.3 L_boot

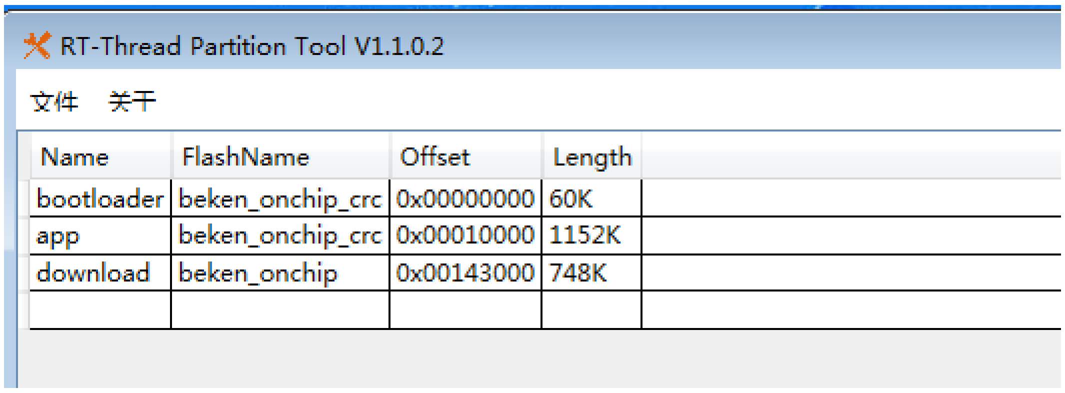
The first-level boot file is located in packages boot l_boot. bin and contains the UART download function. First-level boot should be burned to flash 0 address and run to second-level boot: CPU address 0x1F00.

20.4 UP_boot

UP_boot must start at address 0x1F00. UP_boot supports the OTA upgrade function of rttos. The OTA upgrade function decrypts the RBL file of download partition and decompresses it into OS execution partition. After the second boot runs, jump to OS partition: CPU address 0x10000. UP_boot is located in packages \boot up_boot.bin.

20.5 Get bootloader.bin

Open rt_partition_tool, loading the original bootloader.bin,then importing the partition table partition_audio_2M_sd.json. Finally, save the partition table to bootloader. bin. After the operation completed, the bootloader. bin can generate the final bin file with the application code through the packaging tool beken_packager.



Name	FlashName	Offset	Length
bootloader	beken_onchip_crc	0x00000000	60K
app	beken_onchip_crc	0x00010000	1152K
download	beken_onchip	0x00143000	748K

20.6 Generate all.bin

After you get the bootloader.bin file with partitioned tables, you can generate a complete bin file through the packaging tool. Execute beken_packager.exe to generate the complete bin file (all_2M.1220.bin) and rtthread_uart_2M.1220.bin which is used for serial port upgrade.

Description of config.json file for generating all.bin

field	description
firmware	bin files packaged by partitions
version	version
partition	the partition name is the same as the name of the corresponding partition table in bootloader. bin
start_addr	the partition start address, which is a physical address and is represented in hexadecimal, is the same as the physical start address of the corresponding partition in the partition table
size	the actual size of the partition, expressed in decimal, is the same as the actual length of the corresponding partition in the partition table

20.7 Bootloader Sample Code

20.7.1 2M partition table information profile partition_audio_2M.json sample

```
{
    "part_table": [
        {
            "name": "bootloader",
            "flash_name": "beken_onchip_crc",
            "offset": "0x00000000",
            "len": "60K"
        },
        {
            "name": "app",
            "flash_name": "beken_onchip_crc",
            "offset": "0x00010000",
            "len": "1152K"
        },
        {
            "name": "download",
            "flash_name": "beken_onchip",
            "offset": "0x00143000",
            "len": "748K"
        }
    ]
}
```

22.7.2 UP_boot sample

```
/*
* program         list: this is a up_boot sample code
* program function:  The program implements OTA encryption, decompression, copy partition, etc.
*/
int ota_main(UINT32 * ex)
{
        int result = 0;
        size_t i, part_table_size;
        const struct fal_partition *dl_part = NULL;
        const struct fal_partition *part_table = NULL;
        const char *dest_part_name = NULL;

        if (rt_ota_init() >= 0)
        {
                /* verify bootloader partition
                  * 1. Check if the BL partition exists
                  * 2. CRC BL FW HDR
                  * 3. HASH BL FW
                  * */
                if (rt_ota_part_fw_verify_header(fal_partition_find(RT_BK_BL_PART_NAME)) < 0)
                {
                        //TODO upgrade bootloader to safe image
                        // firmware HDR crc failed or hash failed. if boot verify failed, may not jump to app
running
#if !BOOT_OTA_DEBUG // close debug
                        return -1;
#endif
                }

                // 4. Check if the download partition exists
                dl_part = fal_partition_find(RT_BK_DL_PART_NAME);
                if (!dl_part)
                {
                        log_e("download partition is not exist, please check your configuration!");
                        return -1;
                }
        /* 5. Check if the target partition name is bootloader, skip ota upgrade if yes */
        dest_part_name = rt_ota_get_fw_dest_part_name(dl_part);
        if (dest_part_name && !strncmp(dest_part_name, RT_BK_BL_PART_NAME,
strlen(RT_BK_BL_PART_NAME)))
        {
            log_e("Can not upgrade bootloader partition!");
            goto _app_check;
        }

        /* do upgrade when check upgrade OK
         * 5. CRC DL FW HDR
         * 6. Check if the dest partition exists
         * 7. CRC APP FW HDR
         * 8. Compare DL and APP HDR, containning fw version
         */
        log_d("check upgrade...");
        if ((result = rt_ota_check_upgrade()) == 1) // need to upgrade
        {
            if((rt_ota_get_fw_algo(dl_part) & RT_OTA_CRYPT_STAT_MASK) ==
RT_OTA_CRYPT_ALGO_NONE)
            {
                log_e("none encryption Not allow!");
                goto _app_check;
            }

            /* verify OTA download partition
            * 9. CRC DL FW HDR
            * 10. CRC DL FW
            */
            if (rt_ota_part_fw_verify(dl_part) == 0)
            {
                // 11. rt_ota_custom_verify
                // 12. upgrade
                set_flash_protect(NONE);
                if (rt_ota_upgrade() < 0)
                {
                    log_e("OTA upgrade failed!");
                    /*
                     *   upgrade failed, goto app check. If success, jump to app to run, otherwise
goto recovery factory firmware.
                     **/
                    goto _app_check;
                }
                ota_erase_dl_rbl();
            }
            else
            {
                goto _app_check;
            }
        }
        else if (result == 0)
        {
            log_d("No firmware upgrade!");
        }
        else if (result == -1)
        {
            goto _app_check;
        }
        else
        {
            log_e("OTA upgrade failed! Need to recovery factory firmware.");
            return -1;
        }

_app_check:
        part_table = fal_get_partition_table(&part_table_size);
        /* verify all partition */
        for (i = 0; i < part_table_size; i++)
        {
            /* ignore bootloader partition and OTA download partition */
            if (!strncmp(part_table[i].name, RT_BK_APP_NAME, FAL_DEV_NAME_MAX))
            {
                // verify app firmware
                if (rt_ota_part_fw_verify_header(&part_table[i]) < 0)
                {
                    // TODO upgrade to safe image
                    log_e("App verify failed! Need to recovery factory firmware.");
                    return -1;
                }
```

```
                else
                {
                    *ex = part_table[i].offset;
                    result = 0;
                }
            }
        }
    }
    else
    {
        result = -1;
    }

    return result;
}
```

22.7.3 Configuration file config_sample.json example

```
{
    "magic": "RT-Thread",
    "version": "0.1",
    "count": 2,
    "section": [
        {
            "firmware": "bootloader.bin",
            "version": "2M.1220",
            "partition": "bootloader",
            "start_addr": "0x00000000",
            "size": "65280"
        },
        {
            "firmware": "../../rtthread.bin",
            "version": "2M.1220",
            "partition": "app",
            "start_addr": "0x00011000",
            "size": "1224K"
        }
    ]
}
```

21 Mixer

21.1 Mixer Introduction

BK7251 supports connecting network to play music, line in interface to access audio as background audio. It support that play two kinds of audio data at the same time, and can also eliminate the background audio from line in.

21.2 Mixer Related API

mixer related API refer to \fuction\mixer.h .APIs are as follows：

function	description
mixer_init()	mixer init
mixer_pause()	pause background music
mixer_replay()	replay

21.2.1 mixer initialization

The initialization function of mixing module includes audio delay initialization, semaphore, mutex and mq creation.

```
uint32_t mixer_init(void)；
```

parameters	description
void	null
return	0:success; 1:fail

21.2.2 pause background music

```
void mixer_pause(void)；
```

parameters	description
void	null
return	null

21.2.3 replay

```
void mixer_replay(void)；
```

parameters	description
void	null
return	null

21.3 Mixer Macros

#define	**CONFIG_SOUND_MIXER**	must open this macro start mixer

#define	**MIXER_FAILURE 1**	fail
#define	**MIXER_SUCCESS 0**	success

21.4 Mixer Sample Code

Mixer sample code refers to \samples\Mixer\ mixer_demo.c. Openning macro definitions : MIXER_DEMO to start test.

```
/*
* program      list: This is a sample code about mixer. It supports play 2 kinds of music.The one
  played by cloud, the others played from line in.
* command format: input command after connecting network. command:mixer_set_value 1 stop-
  background sound    mixer_set_value 0 –replay background sound
* program function: The sample code controls the playing and stopping of background music by
 calling commands.
*/
#include "rtconfig.h"
#if CONFIG_SOUND_MIXER
#include "mixer.h"
void mixer_set_value(int argc, char** argv)
{
    int val;
    val = atoi(argv[1]);
    if(val == 1) {
         rt_kprintf("mixer_set_value:%d pause\r\n", val);
        mixer_pause();                                        /*pause*/
    } else if(val == 0) {
         rt_kprintf("mixer_set_value:%d replay\r\n", val);
        mixer_replay();                                       /*replay*/
     }
}
MSH_CMD_EXPORT(mixer_set_value, mixer_set_value test);
```

22 Vad

22.1 Vad Introduction

Vad function is sound boundary detection, which detects the beginning and end of sound. When there is audio data in the chip, the function detects the presence of data and prints the detected sound.

22.2 Vad Related API

Vad related APIs refer to \beken378\func\vad.h. APIs are as follows：

function	description
wb_vad_enter()	enter into vad mode
wb_vad_get_frame_len()	get frame length
wb_vad_entry()	vad entry function
wb_vad_deinit()	close vad

22.2.1 enter into vad mode

```
int wb_vad_enter(void);
```

parameters	description
void	null
return	0:success;　others: fail

22.2.2 get frame length

```
int wb_vad_get_frame_len(void);
```

parameters	description
void	null
return	frame length

22.2.3 vad entry function

```
int wb_vad_entry(char *buffer, int len)；
```

parameters	description
buffer	test buffer
len	test buffer length
return	vad_flag

22.2.4 close vad

```
void wb_vad_deinit(void);
```

parameters	description
void	null
return	null

22.3 Vad Sample Code

Vad sample code refers to \test\mic_record.c.

```
/*
* program     list: this is a sample code about using vad function.
* command format: record_and_play 1
* program function: the sample code verifies the accuracy of vad by recording and playing functions
*/
#include <rtthread.h>
#include <rtdevice.h>
#include <finsh.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include "board.h"
#include "audio_device.h"

#define MICPHONE_TEST
#ifdef MICPHONE_TEST

#define TEST_BUFF_LEN 60*1024
#define READ_SIZE 1024

static uint8_t *test_buf;

void record_and_play(int argc,char *argv[])
{
    int mic_read_len = 0;
    int actual_len,i;
    int dac_wr_len=0;
    uint16_t *buffer = NULL;

    int vad_on;
```

```
#if CONFIG_SOUND_MIXER
    mixer_pause();
#endif

    vad_on = atoi(argv[1]);

     test_buf = sdram_malloc(TEST_BUFF_LEN);
     if(test_buf == NULL)
     {
          rt_kprintf("===not enough memory===\r\n");
          return;
     }

     audio_device_init();                                    /*sound mic device init*/

     audio_device_mic_open();                               /*open mic*/
     audio_device_mic_set_channel(1);                        /*set adc channel*/
     audio_device_mic_set_rate(16000);                       /*set adc sampling rate*/

     if (vad_on)
     {
         rt_kprintf("Vad is ON !!!!!!!!\r\n");               /*enter into vad detection*/
          wb_vad_enter();
     }

     while(1)
     {
         if (vad_on)
              rt_thread_delay(5);
          else
               rt_thread_delay(20);

          int chunk_size = wb_vad_get_frame_len();//320
          char *val = NULL;

          if(mic_read_len > TEST_BUFF_LEN - READ_SIZE)
               break;

          if (!vad_on)
```

```
        {
              actual_len = audio_device_mic_read(test_buf+mic_read_len,READ_SIZE);
        }
        else
        {
            /*mic collect data*/
            actual_len = audio_device_mic_read(test_buf+mic_read_len,chunk_size);
            if(wb_vad_entry(test_buf+mic_read_len, actual_len))
              {
                    rt_kprintf("Vad Detected !!!!!!!!\r\n");                   /*detected voice*/
                     break;
              }
        }

        mic_read_len += actual_len;
}

if (vad_on)
{
        wb_vad_deinit();                                                    /* close vad detection */
}

rt_kprintf("mic_read_len is %d\r\n", mic_read_len);
audio_device_mic_close();                                                   /*close mic*/

audio_device_open();                                                        /*open dac*/
audio_device_set_rate(8000);                                               /*set dac sampling rate */

while(1)
{
        buffer = (uint16_t *)audio_device_get_buffer(RT_NULL);
        if(dac_wr_len >= mic_read_len)
        {
               audio_device_put_buffer(buffer);
               break;
        }

        memcpy(buffer,test_buf+dac_wr_len,READ_SIZE);
        dac_wr_len += READ_SIZE;
```

```
            audio_device_write((uint8_t *)buffer, READ_SIZE);               /*dac play audio*/
        }
        audio_device_close();                                                /*close dac*/

        if(test_buf)
                sdram_free(test_buf);                                        /*free ram*/

#if CONFIG_SOUND_MIXER
       mixer_replay();
#endif
}
MSH_CMD_EXPORT(record_and_play, record play);
#endif
```

23 AMR Encoder

23.1 AMR Encoder Introduction

AMR encoding encodes the received voice information into an AMR format audio file.

23.2 AMR Encoder Related API

AMR encoder related APIs refer to \components\codec\lib_amr_encode\amrnb_encoder.h. APIs are as follows：

function	description
amrnb_encoder_init()	AMR-NB encoder initialize
amrnb_encoder_encode()	AMR-NB encoder encode one frame
amrnb_encoder_deinit()	de-initialize encoder

23.2.1 AMR-NB encoder initialization

```
int32_t amrnb_encoder_init(void** amrnb, uint32_t dtx, void* pmalloc, void* pfree);
```

parameters	description
amrnb	AMR-NB encoder point
dtx	AMR-NB discontinus transmission 0: transmission continuous 1: transmission discontinuous
pmalloc	malloc function pointer
pfree	free function pointer
return	0:success ; others:fail

23.2.2 AMR-NB encoder encode one frame

```
int32_t amrnb_encoder_encode(void* amrnb, uint32_t mode, const int16_t
in[AMRNB_ENCODER_SAMPLES_PER_FRAME], uint8_t
out[AMRNB_ENCODER_MAX_FRAME_SIZE])
```

parameters	description
amrnb	AMR-NB encoder point
mode	AMR-NB mode
in	input frame PCM buffer
out	output encoded AMR buffer
return	>=0: number of encoded bytes ; others:fail

23.2.3 de-initialize encoder

```
int32_t amrnb_encoder_deinit(void** amrnb);
```

parameters	description
amrnb	AMR-NB encoder point
return	0:success ; others:fail

24.3 AMR Encoder Macros

Define the size of data in each frame of AMR encoder

```
#define     AMRNB_ENCODER_SAMPLES_PER_FRAME     (160)
```

Define the maximum frame size of AMR encoder

```
#define     AMRNB_ENCODER_MAX_FRAME_SIZE        (32)
```

23.4 AMR Encoder Enumeration

AMR encoding rate enumeration type:

```
enum Mode {
        AMRNB_MODE_MR475 = 0,/* 4.75 kbps */
        AMRNB_MODE_MR515,      /* 5.15 kbps */
        AMRNB_MODE_MR59,       /* 5.90 kbps */
        AMRNB_MODE_MR67,       /* 6.70 kbps */
        AMRNB_MODE_MR74,       /* 7.40 kbps */
        AMRNB_MODE_MR795,      /* 7.95 kbps */
        AMRNB_MODE_MR102,      /* 10.2 kbps */
        AMRNB_MODE_MR122,      /* 12.2 kbps */
        AMRNB_MODE_MRDTX,      /* DTX       */
        AMRNB_MODE_N_MODES     /* Not Used */
};
```

23.5 AMR Encoder Sample Code

AMR encoder sample code refers to \test\ record_amr_tcp.c. Sample code describe how to use related APIs. Openning macro definitions : RECORD_AMR _TCP_TEST to start test.

```
/*
* program       list: this is a sample code about amr encoder
* command format: start command : record_amr_tcp   start samplerate address   port(After the
   distribution network is successful, the network serial debugging assistant receives the encoding
   data from the network end.)
```

```
                            stop command : record_amr_tcp    stop
* program function: the sample code converts recorded audio signals into AMR format stream by
  calling commands
*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include <rtthread.h>
#include <rtdevice.h>
#include <finsh.h>

#include <drivers/audio.h>
#include <rtthread.h>
#include <sys/socket.h> /* with BSD socket, you need to include the sockets.h */
#include "netdb.h"
#include "record_common.h"
#include <interf_enc.h>
#include "amrnb_encoder.h"

#define RECORD_SAVE_BUF_SIZE (60 * 20 * 5)

struct record_manager
{
      struct net_worker *net;
      rt_mq_t msg;

      int action;
      struct rt_mempool mp;
      int sample_rate;
      int mp_block_size; /* sample / 50 * 2 ==> 8k:320    16k:640*/
      int mp_cnt;

      char *save_buf;
      int save_len;

      rt_sem_t ack;
};

static struct record_manager *recorder;
static short in_short[320*2]; /* NB:8K 160, WB:16K 320 */
#define AMR_MAGIC_NUMBER              "#!AMR\n"

static int record_msg_send(struct record_manager *record, void *buffer, int type, int len)
{
        int ret = RT_EOK;
        struct record_msg msg;

        msg.type = type;
        msg.arg = (uint32_t)buffer;
        msg.len = len;

        ret = rt_mq_send(record->msg, (void *)&msg, sizeof(struct record_msg));
        if (ret != RT_EOK)
                rt_kprintf("[record]:send msg failed \n");
}

static void record_thread_entry(void *parameter)    // entry function for recording and encoding
                                                    //function threads
{
        rt_device_t device = RT_NULL;
        int ret = RT_ERROR;
        uint8_t *mempool;
        rt_uint8_t *buffer;
        rt_uint32_t read_bytes = 0;
         void*amr = NULL;

        int     amr_enc_dtx = 0, frame_size, tmp;
        enum Mode amr_enc_mode = MR122;

/* initialize mempool */
        recorder->mp_block_size = 120;
        recorder->mp_cnt = 10;
        mempool = rt_malloc(recorder->mp_block_size * recorder->mp_cnt);
        rt_mp_init(&(recorder->mp), "record_mp", mempool, recorder->mp_block_size *
recorder->mp_cnt, recorder->mp_block_size);

        /* initialize msg queue */
        recorder->msg = rt_mq_create("net_msg", sizeof(struct record_msg), 12, RT_IPC_FLAG_FIFO);
```

```
/* initialize tcp client */
ret = tcp_client_init(recorder->net);
if (ret != RT_EOK)
{
    return;
}

device = rt_device_find("mic");
if (!device)
{
    rt_kprintf("sound device not found \n");
    return;
}

rt_device_open(device, RT_DEVICE_OFLAG_RDONLY);

/* set samplerate */
{
    int rate = recorder->sample_rate;
    rt_device_control(device, CODEC_CMD_SAMPLERATE, (void *)&rate);
}

{
    /* Initial Encoder */
      ret = amrnb_encoder_init(&amr, amr_enc_dtx, rt_malloc, rt_free);
      if(ret!= RT_EOK)
      {
          rt_kprintf("Encoder_Interface_init failed ====%d",ret);
          return 0;
      }
    frame_size = recorder->sample_rate / 50;
    rt_kprintf("frame_size = %d \n", frame_size);
}

rt_kprintf("[record]:start record, tick %d \n", rt_tick_get());
/* write amr head */
buffer = rt_mp_alloc(&(recorder->mp), RT_WAITING_NO);
memcpy(buffer, AMR_MAGIC_NUMBER, strlen(AMR_MAGIC_NUMBER));
record_msg_send(recorder, buffer, RECORD_MSG_DATA, strlen(AMR_MAGIC_NUMBER));
```

```
    while (1)
    {
        buffer = rt_mp_alloc(&(recorder->mp), RT_WAITING_NO);
        if(!buffer)
        {
            rt_kprintf("[record]：malloc memory for mempool failed \n");
            rt_thread_mdelay(20);
        }
        else
        {
            /* read data from sound device */
            read_bytes = rt_sound_read(device, 0, in_short, frame_size * 2);
            /*encode ....*/
            {
                tmp = amrnb_encoder_encode(amr, amr_enc_mode, in_short, buffer);
            }
            record_msg_send(recorder, buffer, RECORD_MSG_DATA, tmp);
        }

        /* send stop cmd */
        if (recorder->action == 0)
        {
            int cmd;
            cmd = 0;
            record_msg_send(recorder, 0, RECORD_MSG_CMD, 1);
            /* wait ack */
            rt_kprintf("[record]:stop record, tick = %d \n", rt_tick_get());
            break;
        }
    }
    rt_device_close(device);
    rt_mp_detach(&(recorder->mp));
    rt_free(mempool);
    rt_mq_delete(recorder->msg);
    {
        amrnb_encoder_deinit(&amr);
    }
    rt_kprintf("[record]:exit record thread, tick = %d \n", rt_tick_get());
}
```

```
static void net_transmit_thread_entry(void *parameter)      // entry function of AMR format stream after
                                                             //network transmission coding
{
    int ret, cmd;
    struct record_msg msg;

    rt_thread_mdelay(100);
    recorder->save_len = 0;
    while(1)
    {
        if (rt_mq_recv(recorder->msg, &msg, sizeof(struct record_msg), RT_WAITING_FOREVER)
== RT_EOK)
        {
            if(msg.type == RECORD_MSG_DATA)
            {
                memcpy(recorder->save_buf + recorder->save_len, (void *)msg.arg, msg.len);
                recorder->save_len += msg.len;
                rt_mp_free((void *)msg.arg);

                if(recorder->save_len >= RECORD_SAVE_BUF_SIZE - recorder->mp_block_size)
                {
                    /*send data*/
                    send(recorder->net->sock, recorder->save_buf, recorder->save_len, 0);
                    recorder->save_len = 0;
                }
            }
            else if(msg.type = RECORD_MSG_CMD)
            {
                cmd = *(int *)msg.arg;
                if(cmd == 0)
                {
                    /* send remain data, and send ack */
                }
            }
        }
    }
}
static int record_amr_tcp(int argc, char **argv)
{
    rt_thread_t tid = RT_NULL;
    int result;

    if(recorder == RT_NULL)
    {
        recorder = rt_malloc(sizeof(struct record_manager));
        if(!recorder)
        {
            rt_kprintf("[record]:malloc memory for recorder manager \n");
            return -RT_ERROR;
        }
        memset(recorder, 0, sizeof(struct record_manager));

        {
            struct net_worker *net = RT_NULL;
            net = rt_malloc(sizeof(struct net_worker));
            if(!net)
            {
                rt_kprintf("[record]:malloc memory for net worker \n");
                return -RT_ERROR;
            }
            memset(net, 0, sizeof(struct net_worker));
            recorder->net = net;

            recorder->save_buf = rt_malloc(RECORD_SAVE_BUF_SIZE);
            memset(recorder->save_buf, 0, RECORD_SAVE_BUF_SIZE);
        }

        rt_kprintf("L%d, recorder_create done \n", __LINE__);
    }

    rt_kprintf("L%d, record enter \n", __LINE__);
    if (strcmp(argv[1], "stop") == 0)
    {
        recorder->action = 0;                                   //stop mic and encoder
    }
    else if (strcmp(argv[1], "start") == 0)
    {
        /* record start format samplerate url port */
        recorder->action = 1;
```

```
            if(recorder->net->url)
            {
                  rt_free(recorder->net->url);
                  recorder->net->url = RT_NULL;
            }
            recorder->sample_rate = atoi(argv[2]);                    //set mic sample rate
            recorder->net->url = rt_strdup(argv[3]);                  //set url address
            recorder->net->port = atoi(argv[4]);                      //set ports for network connections

            rt_kprintf("[record]:samplerate = %d \n", recorder->sample_rate);
            rt_kprintf("[record]:url = %s \n", recorder->net->url);
            rt_kprintf("[record]:port = %d \n", recorder->net->port);
            /* creat a recording thread */
            tid = rt_thread_create("record",
                                    record_thread_entry,
                                    RT_NULL,
                                    1024 * 32,
                                    27,
                                    10);
            if (tid != RT_NULL)
                  rt_thread_startup(tid);

            /* create net send thread */
            tid = rt_thread_create("net_send",
                                    net_transmit_thread_entry,
                                    RT_NULL,
                                    1024 * 8,
                                    25,
                                    10);
            if (tid != RT_NULL)
                  rt_thread_startup(tid);
      }
      else
      {
            // print_record_usage();
      }
}
FINSH_FUNCTION_EXPORT_ALIAS(record_amr_tcp, __cmd_record_amr_tcp, record amr tcp);
```

24 Opus Encoder

24.1 Opus Encoder Introduction

Opus encoding encodes the received voice information into an audio file in opus format.

24.2 Opus Encoder Related API

Opus encoder APIs refer to \components\codec\lib_opus\include\opus.h. APIs are as follows：

function	description
opus_encoder_create()	allocate and initializes an encoder state
opus_encoder_get_size()	get the size of an opusdecoder structure
opus_encoder_set_complexity()	set complexity of opus encoder
opus_encoder_get_bitrate()	get bitrate of opus encoder
opus_encoder_get_final_range()	get final range of opus encoder
opus_encode()	opus encode
opus_encoder_destroy()	destroy opus encoder

24.2.1 allocate and initialize an encoder state

```
OpusEncoder *opus_encoder_create(opus_int32 Fs, int channels, int application, int *error
)；
```

parameters	description
Fs	sampling rate of input signal (Hz)(8K,16K)
channels	number of channels (1 or 2) in input signal
application	coding mode,refer to 3 modes
error	error of type
return	a pointer to opus encoder struct

24.2.2 get the size of an opusdecoder structure

```
int opus_encoder_get_size(int channels);
```

parameters	description
channels	number of channels(must be 1 or 2)
return	the size in bytes

24.2.3 set complexity of opus encoder

```
opus_encoder_set_complexity(opus_enc, complexity)；
```

parameters	description
opus_enc	encoder state
complexity	encoder complexity :0 -10
return	a pointer to opus encoder struct

24.2.4 get bitrate of opus encoder

```
opus_encoder_get_bitrate(opus_enc, bitrate_bps)；
```

parameters	description
opus_enc	encoder state
bitrate_bps	encoder bitrate
return	a pointer to opus encoder struct

24.2.5 get final range of opus encoder

```
opus_encoder_get_final_range(opus_enc, enc_final_range)；
```

parameters	description
opus_enc	encoder state
enc_final_range	entropy coder state
return	a pointer to opus encoder struct

24.2.6 opus encode

```
opus_int32 opus_encode (OpusEncoder *st,
                        const opus_int16 *pcm,
                        int frame_size,
                        unsigned char *data,
                        opus_int32 max_data_bytes)；
```

parameters	description
st	encoder state
pcm	input signal
frame_size	number of samples per channel in the input signal
data	output payload
max_data_bytes	size of the allocated memory for the output

	payload
return	the length of the encoded packet:success others:fail

24.2.7 destroy opus encoder

```
void opus_encoder_destroy(OpusEncoder *st);
```

parameters	description
st	encoder state
return	null

24.3 Opus Encoder Macros

There are 3 coding modes:

1. gives best quality at a given bitrate for voice signals.

```
#define     OPUS_APPLICATION_VOIP                    2048
```

2. gives best quality at a given bitrate for most non-voice signals like music

```
#define     OPUS_APPLICATION_AUDIO                   2049
```

3. configures low-delay mode that disables the speech-optimized mode in exchange for slightly reduced delay.

```
#define     OPUS_APPLICATION_RESTRICTED_LOWDELAY     2051
```

24.4 Opus Encoder Sample Code

Opus encoder sample code refers to \test\record_opus_tcp.c. Sample code describe how to use related APIs. Openning macro definitions : RECORD_OPUS _TCP_TEST to start test.

```
/*
* program       list: this is a sample code about opus encoder
* command format: start command : record_opus_tcp   start samplerate address   port(After the
   distribution network is successful, the network serial debugging assistant receives the encoding
   data from the network end. Use tools to convert PCM files and play the generated PCM format files
   through Cool Edit Pro)
                         stop command : record_opus_tcp   stop
* program function: the sample code converts recorded audio signals into opus format stream by
  calling commands
*/
#include <stdio.h>
```

```c
#include <stdlib.h>
#include <string.h>

#include <rtthread.h>
#include <rtdevice.h>
#include <finsh.h>
#include <drivers/audio.h>

#include <rtthread.h>
#include <sys/socket.h> /* with BSD socket, you need to include the sockets.h */
#include "netdb.h"
#include "record_common.h"
#include <opus.h>
#define RECORD_SAVE_BUF_SIZE (60 * 20 * 5)

struct record_manager
{
    struct net_worker *net;
    rt_mq_t msg;

    int action;
    struct rt_mempool mp;
    int sample_rate;
    int mp_block_size; /* sample / 50 * 2 ==> 8k:320    16k:640*/
    int mp_cnt;

    char *save_buf;
    int save_len;

    rt_sem_t ack;
};

static struct record_manager *recorder;
static short in_short[320*2]; /* NB:8K 160, WB:16K 320 */

static int record_msg_send(struct record_manager *record, void *buffer, int type, int len)
{
    int ret = RT_EOK;
    struct record_msg msg;
    msg.type = type;
    msg.arg = (uint32_t)buffer;
    msg.len = len;

    ret = rt_mq_send(record->msg, (void *)&msg, sizeof(struct record_msg));
    if (ret != RT_EOK)
        rt_kprintf("[record]:send msg failed \n");
}

static void record_thread_entry(void *parameter)
{
    rt_device_t device = RT_NULL;
    int ret = RT_EOK;
    uint8_t *mempool;
    rt_uint8_t *buffer;
    rt_uint32_t read_bytes = 0;

    OpusEncoder *opus_enc = RT_NULL;
    int sample_rate, channels, errors, frame_size;
    int application, complexity;
    opus_int32 bitrate_bps;
    int enc_len;

    /* initialize mempool */
    recorder->mp_block_size = 120;
    recorder->mp_cnt = 10;
    mempool = rt_malloc(recorder->mp_block_size * recorder->mp_cnt);
    rt_mp_init(&(recorder->mp), "record_mp", mempool, recorder->mp_block_size *
recorder->mp_cnt, recorder->mp_block_size);

    /* initialize msg queue */
    recorder->msg = rt_mq_create("net_msg", sizeof(struct record_msg), 12, RT_IPC_FLAG_FIFO);

    /* initialize tcp client */
    ret = tcp_client_init(recorder->net);
    if (ret != RT_EOK)
    {
        return;
    }
```

```
device = rt_device_find("mic");
if (!device)
{
    rt_kprintf("mic device not found \n");
    return;
}

rt_device_open(device, RT_DEVICE_OFLAG_RDONLY);

/* set samplerate */
{
    int rate = recorder->sample_rate;
    rt_device_control(device, CODEC_CMD_SAMPLERATE, (void *)&rate);
}


{
    enc_len = opus_encoder_get_size(1);
    rt_kprintf("opus_encoder_get_size: 1 channel size: %d \n", enc_len);
    enc_len = opus_encoder_get_size(2);
    rt_kprintf("opus_encoder_get_size: 2 channel size: %d \n", enc_len);

    sample_rate = recorder->sample_rate;
    channels = 1;
    application = OPUS_APPLICATION_VOIP;
    complexity = 1; // 1 to 10

    opus_enc = opus_encoder_create(sample_rate, channels, application, &errors);
    if(errors != OPUS_OK)
    {
        rt_kprintf("[opus]:create opus encoder failed : %d! \n", errors);
    }

    frame_size = sample_rate / 50; // 20ms ==>
    opus_encoder_set_complexity(opus_enc, complexity);
    opus_encoder_get_bitrate(opus_enc,bitrate_bps );
    rt_kprintf("[opus]:default bitrate %d\n", bitrate_bps);
    rt_kprintf("frame_size = %d \n", frame_size);
}

rt_kprintf("[record]:start record, tick %d \n", rt_tick_get());
```

```
    while (1)
    {
        buffer = rt_mp_alloc(&(recorder->mp), RT_WAITING_NO);
        if(!buffer)
        {
            rt_kprintf("[record]：malloc memory for mempool failed \n");
            rt_thread_mdelay(20);
        }
        else
        {
            /* read data from sound device */
            read_bytes = rt_sound_read(device, 0, in_short, frame_size * 2);
            /*encode ....*/
            {
                enc_len = opus_encode(opus_enc, in_short, frame_size, buffer + 8,
recorder->mp_block_size - 8);

                /* write head */
                {
                    opus_uint32 enc_final_range;
                    int_to_char_big_endian(enc_len, buffer);

                    opus_encoder_get_final_range(opus_enc, enc_final_range);
                    int_to_char_big_endian(enc_final_range, buffer+4);
                }

                enc_len += 8;
            }
            record_msg_send(recorder, buffer, RECORD_MSG_DATA, enc_len);
        }

        /* send stop cmd */
        if (recorder->action == 0)
        {
            int cmd;

            cmd = 0;
            record_msg_send(recorder, 0, RECORD_MSG_CMD, 1);
            /* wait ack */
            rt_kprintf("[record]:stop record, tick = %d \n", rt_tick_get());
```

```
                break;
            }
    }
    rt_device_close(device);
    rt_mp_detach(&(recorder->mp));
    rt_free(mempool);
    rt_mq_delete(recorder->msg);
    {
        opus_encoder_destroy(opus_enc);
    }
    rt_kprintf("[record]:exit record thread, tick = %d \n", rt_tick_get());
}

static void net_transmit_thread_entry(void *parameter)     // entry function of opus format stream after
                                                           //network transmission coding
{
    int ret, cmd;
    struct record_msg msg;

    recorder->save_len = 0;
    while(1)
    {
        if (rt_mq_recv(recorder->msg, &msg, sizeof(struct record_msg), RT_WAITING_FOREVER)
== RT_EOK)
        {
            if(msg.type == RECORD_MSG_DATA)
            {
                memcpy(recorder->save_buf + recorder->save_len, (void *)msg.arg, msg.len);
                recorder->save_len += msg.len;
                rt_mp_free((void *)msg.arg);

                if(recorder->save_len >= RECORD_SAVE_BUF_SIZE - recorder->mp_block_size)
                {
                    /*send data*/
                    send(recorder->net->sock, recorder->save_buf, recorder->save_len, 0);
                    recorder->save_len = 0;
                }
            }
            else if(msg.type = RECORD_MSG_CMD)
            {
                    cmd = *(int *)msg.arg;
                    if(cmd == 0)
                    {
                        /* send remain data, and send ack */
                    }
                }
            }
        }
}
static int record_opus_tcp(int argc, char **argv)
{
    rt_thread_t tid = RT_NULL;
    int result;

    if(recorder == RT_NULL)
    {
        recorder = rt_malloc(sizeof(struct record_manager));
        if(!recorder)
        {
            rt_kprintf("[record]:malloc memory for recorder manager \n");
            return -RT_ERROR;
        }
        memset(recorder, 0, sizeof(struct record_manager));

        {
            struct net_worker *net = RT_NULL;
            net = rt_malloc(sizeof(struct net_worker));
            if(!net)
            {
                rt_kprintf("[record]:malloc memory for net worker \n");
                return -RT_ERROR;
            }
            memset(net, 0, sizeof(struct net_worker));
            recorder->net = net;

            recorder->save_buf = rt_malloc(RECORD_SAVE_BUF_SIZE);
            memset(recorder->save_buf, 0, RECORD_SAVE_BUF_SIZE);
        }

        rt_kprintf("L%d, recorder_create done \n", __LINE__);
    }

    rt_kprintf("L%d, record enter \n", __LINE__);
    if (strcmp(argv[1], "stop") == 0)                                   //stop mic
    {
        recorder->action = 0;
    }
    else if (strcmp(argv[1], "start") == 0)                              //start mic and encoder
    {
        /* record start format samplerate url port */
        recorder->action = 1;

        if(recorder->net->url)
        {
            rt_free(recorder->net->url);
            recorder->net->url = RT_NULL;
        }

        recorder->sample_rate = atoi(argv[2]);
        recorder->net->url = rt_strdup(argv[3]);
        recorder->net->port = atoi(argv[4]);

        rt_kprintf("[record]:samplerate = %d \n", recorder->sample_rate);
        rt_kprintf("[record]:url = %s \n", recorder->net->url);
        rt_kprintf("[record]:port = %d \n", recorder->net->port);

        /* create net send thread */
        tid = rt_thread_create("record",
                                record_thread_entry,
                                RT_NULL,
                                1024 * 32,
                                27,
                                10);
        if (tid != RT_NULL)
            rt_thread_startup(tid);

        /* create net send thread */
        tid = rt_thread_create("net_send",
                                  net_transmit_thread_entry,
                                  RT_NULL,
                                                1024 * 8,
                                                28,
                                                10);
            if (tid != RT_NULL)
                  rt_thread_startup(tid);
      }
      else
      {
            // print_record_usage();
      }
}
FINSH_FUNCTION_EXPORT_ALIAS(record_opus_tcp, __cmd_record_opus_tcp, record opus tcp);
```

25 EasyFlash

25.1 EasyFlash Introduction

EasyFlash is an open source and lightweight embedded flash memory library. It can quickly save product parameters, supports write balance and power-down protect tion, reduces the difficulty of processing product parameters for developers, and ensures that the product has better scalability in later upgrades.

25.2 EasyFlash Related API

EasyFlash related APIs refer to\packages\EasyFlash\inc\easyflash.h. APIs are as follows：

function	description
easyflash_init()	easyflash init
ef_get_env()	get easyflash environment variable
ef_set_env()	write data to easyflash
ef_save_env()	save data to flash

25.2.1 easyflash initialization

```
EfErrCode easyflash_init(void);
```

parameters	description
void	null
return	0:success;　others: fail

25.2.2 get easyflash environment variable

```
char *ef_get_env(const char *key);
```

parameters	description
key	key
return	value: variable address

25.2.3 write data to easyflash

```
EfErrCode ef_set_env(const char *key, const char *value);
```

parameters	description
key	key

value	the data that will be write
return	0:success;　others: fail

25.2.4 save data to flash

```
EfErrCode ef_save_env(void);
```

parameters	**description**
void	null
return	0:success;　others: fail

25.3 EasyFlash Macros

#define　PKG_USING_EASYFLASH		must open if use EasyFlash
#define　EF_START_ADDR	**0x1FE000**	EasyFlash start address :0x1FE000
#define ENV_USER_SETTING_SIZE	**1 * 1024**	EasyFlash size

25.4 EasyFlash Sample Code

EasyFlash sample code refers to \test\ easyflash_test.c. Sample code describes how to use EasyFlash related APIs. Openning macro definitions : EASY_FLASH_ TEST to start test.

```
/*
* program        list: This is a sample code about easyflash
* command format: write data command : Easy_Flash_Write
                         read data command : Easy_Flash_Read
* program function: Save the string in flash and read it out.
*/
#define          EASY_FLASH_TEST
#ifdef           EASY_FLASH_TEST
#include "rtthread.h"
#include <dfs.h>
#include <dfs_fs.h>
#include "player.h"
#include "include.h"
#include "driver_pub.h"
#include "func_pub.h"
#include "app.h"
#include "ate_app.h"
#include "shell.h"
#include "flash.h"
```

```
#include <finsh.h>
#include "easyflash.h"

unsigned char read_buff[10*1024];
unsigned char write_buff[10*1024];

#define   test_data "AABBCCDDEEFFGGHHIIJJKKLLMMNNOOPPQQRRSSTTUUVVWWXXYYZZ"
#define   KEY "temp"

static void Easy_Flash_Write(void)
{
        int i ;

        easyflash_init();                                    /*easyflash init */
       ef_set_env(KEY,test_data);                  /*write the data to easy flash environment variable */
       ef_save_env();                                        /*save data */

       rt_kprintf("---Flash Write over \r\n");
}

static void Easy_Flash_Read(void)                          /*read data from easy flash */
{
       char *p_write_buff;

        easyflash_init();

        p_write_buff = ef_get_env(KEY);                  /*get the data that easy flash saved */

        rt_kprintf("%s",p_write_buff);
}

MSH_CMD_EXPORT(Easy_Flash_Write,set_or_read_Easy_Flash_Write test);
MSH_CMD_EXPORT(Easy_Flash_Read, set_or_read_Easy_Flash_Read            test);
#endif
```

26 Voice Changer

26.1 Voice Changer Introduction

Voice changer supports voice change function, can lengthen the voice, the latter shorten the voice.

26.2 Voice Changer Related API

Voice changer related APIs refer to \components\voice_changer\app_voice_changer.h. APIs are as follows:

function	description
voice_changer_initial()	voice changer init
voice_changer_exit()	exit voice changer mode
voice_changer_start()	start voice changer function
voice_changer_stop()	stop voice changer function
voice_changer_set_change_flag()	set voice changer flag
voice_changer_get_need_mic_data()	get data from mic
voice_changer_set_cost_data()	set cost data
voice_changer_data_handle()	voice changer handle

26.2.1 voice changer initialization

```
VC_ERR voice_changer_initial(uint32_t freq);
```

parameters	description
freg	frequency
return	0:success; others: fail

26.2.2 exit voice changer

```
void voice_changer_exit(void);
```

parameters	description
void	null
return	null

26.2.3 start voice changer

```
void voice_changer_start(void);
```

parameters	description
void	null
return	null

26.2.4 stop voice changer

```
void voice_changer_stop(void);
```

parameters	description
void	null
return	null

26.2.5 set voice changer flag

```
void voice_changer_set_change_flag(void);
```

parameters	description
void	null
return	null

26.2.6 get data from mic

```
int voice_changer_get_need_mic_data(void);
```

parameters	description
void	null
return	length of left

26.2.7 set cost data

```
int voice_changer_set_cost_data(int cost_len);
```

parameters	description
cost_len	length of cost data
return	length of left

26.2.8 voice changer handle

```
int voice_changer_data_handle(uint8_t *mic_in, int mic_len, uint8_t **vc_out);
```

parameters	description
mic_in	mic receive data

mic_len	length of mic receive data
vc_out	output data
return	0:success; others: fail

26.3 Voice Changer Macros

#define	**CONFIG_VOICE_CHANGER**	**1**	must open when using voice changer

26.4 Voice Changer Enumeration

```
typedef enum {
      VC_STOP,        //stop
      VC_FIRST,        //first
      VC_START,       //start
} VC_STA;
```

26.5 Voice Changer Sample Code

voice changer sample code refers to \components\voice_changer\voice_changer_task.c.

```
/*
* program        list: This is a sample code about voice changer
* command format: voice_changer_sample launch/shutoff/next
* program function: Change the collected sound
*/
#include <rtthread.h>
#include "vc_config.h"

#define VOICE_CHANGER_SOFT_TIMER_HANDLER            1
#define VOICE_CHANGER_THREADT_TASK_HANDLER        2

#define VOICE_CHANGER_HANDLER
VOICE_CHANGER_THREADT_TASK_HANDLER



#define VOICE_CHANGER_MIC_CFG                                   1
#define VOICE_CHANGER_MIC_INIT_CFG                              1

#define VOICE_CHANGER_DEFAULT_OUT_AUD                    1
#define VOICE_CHANGER_AUD_INIT_CFG                        1
#define VOICE_CHANGER_AUD_SINGLE_CH                       1
```

```
#ifndef min
#define min(x, y)                    (((x) < (y)) ? (x) : (y))
#endif

#if CONFIG_VOICE_CHANGER
#include "app_voice_changer.h"
#include "rtos_pub.h"
#include "audio_device.h"
#include "string.h"
#include "stdio.h"
#include "stdlib.h"

#define VC_BUFF_MAX_LEN      (256 * 4 *sizeof(unsigned int))
#define VC_HANDLER_INTERVAL_MS      5

beken_thread_t voice_changer_handler = NULL;
beken_timer_t vc_timer;

static void *vctimer = NULL;
static char *vcbuff = NULL;
static int g_running_flag;

#if VOICE_PCM_VC_AUD_OUTPUT_TEST
#define PCM_LENGTH              35254
extern const unsigned char acnumber_pcm[];                    ///35254
static unsigned int pc_offset = 0;
#endif
static int voice_changer_read_pcm(char*outbuf,int len)        /*read data from mic and save*/
{
    int out_len = 0;
#if VOICE_CHANGER_MIC_CFG
    out_len = audio_device_mic_read(outbuf,len);
#endif

#if VOICE_PCM_VC_AUD_OUTPUT_TEST
    out_len = min(len,(PCM_LENGTH - pc_offset));
    memcpy(outbuf,acnumber_pcm+pc_offset,out_len);
```

```
        pc_offset += out_len;
        if(pc_offset >= PCM_LENGTH)
        {
                pc_offset = 0;
                rt_kprintf("restart\r\n");
        }
#endif
        return out_len;
}

static int voice_changer_write_pcm(char*outbuf,int len)                 /*write data to pcm*/
{
        int input_len = 0;

#if VOICE_CHANGER_DEFAULT_OUT_AUD
        int bufsz;
        uint16_t* aud_buf = (uint16_t *)audio_device_get_buffer(&bufsz);
        if((bufsz == 0) || (aud_buf == NULL))
        {
                if(aud_buf)
                {
                        audio_device_put_buffer(aud_buf);
                }
                rt_kprintf("vc err L%d\r\n",__LINE__);
                return input_len;
        }

        input_len = min((bufsz>>1),len);
        if(len == 0)
        {
                goto exit;
        }
        #if VOICE_CHANGER_AUD_SINGLE_CH
                int16_t *src,*dst;
                int i;
                src = outbuf;
                dst = aud_buf;
                for(i=0;i<(len/2);i++)
                {
                        dst[2 * i] = src[i];
                        dst[2 * i + 1] = src[i];
                }

                audio_device_write((uint8_t *)aud_buf, input_len*2);
        #else
                memcpy(aud_buf,outbuf,input_len);
                audio_device_write((uint8_t *)aud_buf, input_len);
        #endif
#endif
        return input_len;
exit:
        if(aud_buf)
        {
                audio_device_put_buffer(aud_buf);
        }
        rt_kprintf("vc L%d err\r\n",__LINE__);
        return 0;
}

static int voice_changer_shutoff(void)                                              / *close voice changer*/
{
        g_running_flag = 0;
        if (vctimer != RT_NULL)
        {
        #if VOICE_CHANGER_HANDLER    == VOICE_CHANGER_SOFT_TIMER_HANDLER
                rt_timer_stop((rt_timer_t)vctimer);
        #elif VOICE_CHANGER_HANDLER    == VOICE_CHANGER_THREADT_TASK_HANDLER
                bk_rtos_delete_thread(vc_handler);
        #endif
        }

        return 0;}

static int voice_changer_launch(unsigned int freq)              /*start voice changer：lengthening voice */
{
        if(vcbuff == NULL)
        {
                vcbuff = (char*)rt_malloc(VC_BUFF_MAX_LEN);
        }
        if(vcbuff == NULL)
        {
                rt_kprintf("vcbuff == null\r\n");
                return -1;
        }
#if (VOICE_CHANGER_MIC_CFG && VOICE_CHANGER_MIC_INIT_CFG)
        audio_device_init();

        audio_device_mic_open();
        audio_device_mic_set_channel(1);
        audio_device_mic_set_rate(freq);
#endif

#if VOICE_CHANGER_DEFAULT_OUT_AUD && VOICE_CHANGER_AUD_INIT_CFG
        audio_device_init();

        audio_device_open();
        audio_device_set_rate(freq);
        audio_device_set_volume(100);
#endif
       g_running_flag = 1;
        voice_changer_initial(freq);
        if (vctimer != RT_NULL)
        {
        #if VOICE_CHANGER_HANDLER    == VOICE_CHANGER_SOFT_TIMER_HANDLER
                rt_timer_start((rt_timer_t)vctimer);
                voice_changer_start();
        #elif VOICE_CHANGER_HANDLER    == VOICE_CHANGER_THREADT_TASK_HANDLER
                rt_thread_startup((rt_thread_t)vctimer);
        #endif
                rt_kprintf("vc start\r\n");
        }

        return 0;
}

static int voice_changer_handler(void)                                    /*voice data handle*/
{
        unsigned char* vc_out;
         int vc_out_len;
        int len;
```

```
 if(vcbuff == NULL)
 {
        rt_kprintf("vcbuff err\r\n");
        return -1;
 }

 len = voice_changer_get_need_mic_data();
 if(len > 0) {
        len = (len > (VC_BUFF_MAX_LEN/4))?(VC_BUFF_MAX_LEN/4) : len;
 }
 else if(len < 0)
 {
        return -1;
 }
 else if(len == 0)
 {
        return 0;
 }

 len = voice_changer_read_pcm(vcbuff,len);
 if(len <= 0)
 {
        rt_kprintf("origin pcm empty\r\n");
        return 0;
 }

 vc_out_len = voice_changer_data_handle((uint8*)vcbuff, len, &vc_out);
if(vc_out_len == 0)
 {
      // no enough data for vc, so vc return 0, no need do sm_playing
      return 0;
}
 else if(vc_out_len > 0)
 {
#if 1
 len = voice_changer_write_pcm((char*)vc_out,vc_out_len);
 #else
        voice_changer_write_pcm(vcbuff,len);
        len = vc_out_len;
        #endif
                if(len > 0)
                {
                voice_changer_set_cost_data(len);
                }
       }
        return 0;
}
static int app_voice_changer_init(void)                                  /* voice changer init*/
{
#if VOICE_CHANGER_HANDLER    == VOICE_CHANGER_SOFT_TIMER_HANDLER
        if(vctimer == NULL)
        {
                vctimer = (void*)rt_timer_create("vc",
                                                        voice_changer_timer_handler,
                                                        NULL,
                                                        VC_HANDLER_INTERVAL_MS,
                                                        RT_TIMER_FLAG_PERIODIC |
RT_TIMER_FLAG_SOFT_TIMER);
        }
#elif VOICE_CHANGER_HANDLER    == VOICE_CHANGER_THREADT_TASK_HANDLER
        if(vctimer == NULL)
        {
                vctimer = (void*)rt_thread_create("vc",
                                                                        voice_changer_task_handler,
                                                                        NULL,
                                                                        4*1024,
                                                                        15,
                                                                        20);
                rt_kprintf("vctimer = %p\r\n",vctimer);
        }
#endif
        return 0;
}
INIT_APP_EXPORT(app_voice_changer_init);
static int voice_changer_sample(int argc, char *argv[])
{
        rt_err_t ret = RT_EOK;
        unsigned int freq = 16000;
        if(argc == 2)
        {
                if(strcmp(argv[1],"launch") == 0)
                {
                        rt_kprintf("voice changer freq = %d\r\n",freq);
                        voice_changer_launch(freq);
                        app_voice_changer_init();
                }
                else if(strcmp(argv[1],"shutoff") == 0)
                {
                        rt_kprintf("voice changer shutoff\r\n");
                        voice_changer_shutoff();
                }
                else if(strcmp(argv[1],"next") == 0)
                {
                        rt_kprintf("voice changer set next\r\n");
                        voice_changer_set_change_flag();
                }
        }
        else if(argc == 3)
        {
                if(strcmp(argv[1],"launch") == 0)
                {
                        freq = atoi(argv[2]);
                        rt_kprintf("voice changer freq = %d\r\n",freq);
                        voice_changer_launch(freq);
                }
        }
        return ret;
}

MSH_CMD_EXPORT(voice_changer_sample,vc sample);
#endif
```

27 Image Transmission

27.1 Image Transmission Introduction

BK7251 supporting high-speed spi-slave interface, speed up to 50Mbps, can be attached to other MCU cameras. It supports DCMI standard camera interface, PCLK up to 24M. It supports cameras such as: PAS6329/6375,OV_7670,GC0328C/0308C. BK7251 currently supports hardware Jpeg compression module and maximum resolution 600*800.

27.2 Image Transmission Related API

Image Transmission related APIs refer to\beken378\func\video_transfer\video_transfer.h. APIs are as follows:

function	description
video_transfer_init()	open video_transfer
video_transfer_deinit()	close video_transfe

27.2.1 open video_transfer

UINT32 video_transfer_init(TVIDEO_SETUP_DESC_PTR setup_cfg)；

parameters	description
setup_cfg	video_transfer configure parameters
return	0: success;　others: fail

27.2.2 close video_transfer

UINT32 video_transfer_deinit(void)；

parameters	description
void	null
return	0: success;　others: fail

27.3 Image Transmission Struct Reference

TVIDEO_SETUP_DESC_PTR ：	
UINT32 send_type	TVIDEO_SND_TYPE enumeration type,send_type determines the size of each image data packet
send_func	function of transmitting image data
start_cb	indicates the start of transmission when open spi or after camera_intf

end_cb	indicates the end of transmission when close spi or before camera_intf
pkt_header_size	pkt_header_size indicates the size of "header information" when add "header information" to image data.pkt_header_size must be an integer multiple of 4. when do not use it ,set it to 0.
add_pkt_header	adding"header information" callback. this function calls back every time when an image data packet is received. when do not use it ,set it to NULL.

TV_HDR_PARAM_PTR：

UINT8* ptk_ptr	header information can be filled in from memory pointed to by this pointer
UINT32 ptklen	this data data contained in image packages
UINT32 frame_id	this data packet belongs to the frame_id image frame of this image transmission
UINT32 is_eof	1:the end of frame; 0: not
UINT32 frame_len	only is_eof is valid when it is 1, indicating the data of the whole frame, which occupies a total of frame_len packets

27.4 Image Transmission Macros

#define CFG_USE_CAMERA_INTF	using camera and jpeg
#define CFG_USE_HSLAVE_SPI	using High-spi-slave spi interface
#deine CFG_USE_APP_DEMO_VIDEO_TRANSFER	using video_transfer demo
#define CFG_USE_SPIDMA	using High-spi-slave spidma moudle

27.5 Image Transmission Enumeration

send type:

```
typedef enum
{
    TVIDEO_SND_UDP，                /*upload via UDP */
    TVIDEO_SND_TCP，                /* upload via TCP */
    TVIDEO_SND_INTF，              /* upload via others*/
} TVIDEO_SND_TYPE；
```

27.6 Image Transmission Sample Code

Image transmission sample code refers to \beken378\app\app_demo.

1.not using "header information"

```
int app_video_intf_send_packet (UINT8 *data, UINT32 len)
{
        //os_printf("voide send:%p, %p\r\n", data, len);
         return len;
}
void app_video_intf_open (void)
{
        os_printf("voide open\r\n");
        /*spi or camera_intf */
        #if (CFG_USE_SPIDMA || CFG_USE_CAMERA_INTF)
        TVIDEO_SETUP_DESC_ST setup;
        /* TVIDEO_SND_INTF      the way of sendig: send intf*/
        setup.send_type = TVIDEO_SND_INTF;
        setup.send_func = app_video_intf_send_packet;

        setup.start_cb = NULL;
        setup.end_cb = NULL;
        /* not using "header information"*/
        setup.pkt_header_size = 0;
        setup.add_pkt_header = NULL;

        video_transfer_init(&setup);
        #endif
}

void app_video_intf_close (void)
{
        os_printf("voide close\r\n");
        #if (CFG_USE_SPIDMA || CFG_USE_CAMERA_INTF)
        video_transfer_deinit();
        #endif
}
```

2.using "header information"

```
/*define "header information"*/
typedef struct tvideo_hdr_st
{
        UINT8 id;
        UINT8 is_eof;
        UINT8 pkt_cnt;
        UINT8 size;
}HDR_ST, *HDR_PTR;
/*"header information" callback. */
void app_demo_add_pkt_header(TV_HDR_PARAM_PTR param)
{
        HDR_PTR elem_tvhdr = (HDR_PTR)param->ptk_ptr;
        elem_tvhdr->id = (UINT8)param->frame_id;
        elem_tvhdr->is_eof = param->is_eof;
        elem_tvhdr->pkt_cnt = param->frame_len;
        elem_tvhdr->size = 0;
}
/*send function ,using UDP*/
int app_demo_udp_send_packet (UINT8 *data, UINT32 len)
{
        int send_byte = 0;
        if(!app_demo_udp_romote_connected)
                return 0;
        send_byte = sendto(app_demo_udp_img_fd, data, len, MSG_DONTWAIT|MSG_MORE,
                (struct sockaddr *)app_demo_remote, sizeof(struct sockaddr_in));
        if (send_byte < 0) {
                /* err */
                //APP_DEMO_UDP_PRT("send return fd:%d\r\n", send_byte);
                send_byte = 0;
        }
         return send_byte;
}
/*indicate start transmission */
static void app_demo_udp_app_connected(void)
{
        app_demo_softap_send_msg(DMSG_APP_CONECTED);
}
/* indicate stop transmission */
static void app_demo_udp_app_disconnected(void)
{
        app_demo_softap_send_msg(DMSG_APP_DISCONECTED);
}
void app_video_intf_open (void)
{
        TVIDEO_SETUP_DESC_ST setup;
        setup.send_type = TVIDEO_SND_UDP;
```

```
    setup.send_func = app_demo_udp_send_packet;
    setup.start_cb = app_demo_udp_app_connected;
    setup.end_cb = app_demo_udp_app_disconnected;
    setup.pkt_header_size = sizeof(HDR_ST);
    setup.add_pkt_header = app_demo_add_pkt_header;
    video_transfer_init(&setup);
}

void app_video_intf_close (void)
{
    os_printf("voide close\r\n");
    #if (CFG_USE_SPIDMA || CFG_USE_CAMERA_INTF)
    video_transfer_deinit();
    #endif
}
```