

模式管理接口

珠海市杰理科技股份有限公司

Zhuhai Jieli Technologyco.,LTD

版权所有，未经许可，禁止外传

2020 年 08 月

Table of contents

1. Mode management instructions1

2. Interface Introduction1

3. Mode Switching Table1

4. Button mapping2

5. Mode Messaging Interface2

key message sending2

key message response3

Message Extensions3

6. Detailed interface notes4

void app_task_switch_prev().....4 void

app_task_switch_next()..... 4 int

app_task_switch_to(u8 app_task)4 int

app_task_switch_back()..... 5 u8

app_task_exitting()//..... 5 u8

app_get_curr_task()..... 5 u8

app_check_curr_task(u8 app).....5 7. Added

mode description5

All rights reserved, infringement must be investigated

Address: Building 9, No. 107 Shihua West Road, Jida,

Zuhai Tel: 0756-6313088 Website: www.zh-jieli.com

Postcode:

519015 Fax: 0756-6313081

1. Mode management instructions

The mode management interface provides switching, query and other operations for the application layer app mode to ensure orderly switching and response of each application scenario.

2. Interface introduction

```
//Switch to the previous valid
mode void app_task_switch_prev(); //
Switch to the next valid mode void
app_task_switch_next(); //Return
to the previous mode int
app_task_switch_back(); //Switch
to the specified mode
int app_task_switch_to(u8 app_task); //Get
the current mode id u8 app_get_curr_task(); //
Check whether it is the current mode by id
u8 app_check_curr_task(u8 app); //Mode
switch exit detection u8 app_task_exitting();
```

3. Mode switching table

///Mode configuration table, where the order of switching modes can be configured, and the scheme is defined according to requirements

```
6 ///模式配置表, 这里可以配置切换模式的顺序, 方案根据需求定义
7 static const u8 app_task_list[] = {
8 #if TCFG_APP_BT_EN
9     APP_BT_TASK,
10 #endif
11 #if TCFG_APP_MUSIC_EN
12     APP_MUSIC_TASK,
13 #endif
14 #if TCFG_APP_FM_EN
15     APP_FM_TASK,
16 #endif
17 #if TCFG_APP_RECORD_EN
18     APP_RECORD_TASK,
19 #endif
20 #if TCFG_APP_LINEIN_EN
21     APP_LINEIN_TASK,
22 #endif
23 #if TCFG_APP_RTC_EN
24     APP_RTC_TASK,
25 #endif
26 #if TCFG_APP_PC_EN
27     APP_PC_TASK,
28 #endif
29 #if TCFG_APP_SPDIF_EN
30     APP_SPDIF_TASK,
31 #endif
32 };
```

4. Button mapping

After the button driver detects the button, it will map the button before the notify button event. The mapping process is as follows (map according to different button types):

```

46 int key_event_remap(struct sys_event *e)
47 {
48     struct key_event *key = &e->u.key;
49     int msg = -1;
50     switch (key->type) {
51     case KEY_DRIVER_TYPE_IO:
52         msg = iokey_event_to_msg(app_curr_task, key);
53         break;
54     case KEY_DRIVER_TYPE_AD:
55     case KEY_DRIVER_TYPE_RTCVDD_AD:
56         msg = adkey_event_to_msg(app_curr_task, key);
57         break;
58     case KEY_DRIVER_TYPE_IR:
59         msg = irkey_event_to_msg(app_curr_task, key);
60         break;
61     case KEY_DRIVER_TYPE_TOUCH:
62         msg = touch_key_event_to_msg(app_curr_task, key);
63         break;
64     case KEY_DRIVER_TYPE_RDEC:
65         msg = rdec_key_event_to_msg(app_curr_task, key);
66         break;
67
68     case KEY_DRIVER_TYPE_SOFTKEY:
69         msg = key->event;
70         break;
71     default:
72         break;
73     }
74     e->u.key.event = msg;
75     e->u.key.value = 0; //
76     return TRUE; // notify 数据
77 }

```

5. Mode messaging interface

//app custom message sending

interface int app_task_put_usr_msg(int msg, int arg_num, ...); //app

message acquisition interface (block parameter 0 means internal pend, 1 returns directly)

void app_task_get_msg(int *msg, int msg_size, int block); //app key message sending

interface int app_task_put_key_msg(int msg, int value);

The application process message **app_task_put_key_msg**, the message enumeration is defined in key_event_deal.h

sends **the response in the interface key event (SYS_KEY_EVENT)**, such as:

key message sending

app_task_put_key_msg(KEY_MUSIC_PLAYER_START, 0);

key message response

```
static int music_key_event_opn(struct sys_event *event)
{
    int ret = true;
    int err = MUSIC_PLAYER_ERR_NULL;
    u8 vol;
    int mode;
    char *logo = NULL;

    int msg[2];
    msg[0] = event->u.key.event; ← 获取msg
    msg[1] = event->u.key.value; ← 获取msg中参数
    static int msg_demo = 0;

    log_i("music task msg = %d\n", msg[0]);

    switch (msg[0]) {
    case KEY_MUSIC_PLAYER_START: ← 断点播放活动设备
        log_i("KEY_MUSIC_PLAYER_START !!\n");
        logo = dev_manager_get_logo(dev_manager_find_active(1));
        if (true == breakpoint_vm_read(breakpoint, logo)) {
            err = music_player_play_by_breakpoint(logo, breakpoint);
        } else {
            err = music_player_play_first_file(logo);
        }
        break;
    }
```

message extension

In the absence of special requirements, it is not recommended to use `app_task_put_usr_msg`. It is used only when multiple parameters need to be transmitted. The message

enumeration is defined in `app_task.h` as follows:

```
25 enum {
26     APP_MSG_SYS_EVENT = Q_EVENT,
27
28     /* 用户自定义消息 */
29     APP_MSG_SWITCH_TASK = Q_USER + 1,
30     APP_MSG_USER        = Q_USER + 2,
31
32 };
33
```

← 往后增加自定义消息

To customize the message acquisition process, add a case to the message acquisition in the mode you are in to respond:

```
while (1) {
    app_task_get_msg(msg, ARRAY_SIZE(msg), 1);
    switch (msg[0]) {
    case APP_MSG_SYS_EVENT:
        if (music_sys_event_handler((struct sys_event *)(&msg[1])) == false) {
            app_default_event_deal((struct sys_event *)(&msg[1]));
        }
        break;
    default: ← 此处增加case处理自定义消息
        break;
    }
    if (app_task_exitting()) {
        music_task_close();
        return;
    }
}
```

6. Detailed interface notes

```
/**-----*/ /**@brief
```

switch to previous mode @param @return @note

```
*/
```

```
/*-----*/
```

void app_task_switch_prev()

```
/**-----*/ /**@brief
```

switch to next mode @param @return @note

```
*/
```

```
/*-----*/
```

void app_task_switch_next()

```
/**-----*/ /**@brief
```

switch to the specified mode

 @param app_task: specified mode

 @return @note

```
*/
```

```
/*-----*/
```

int app_task_switch_to(u8 app_task)

```
/**-----*/
```

/**@brief jump back to the original mode

 @param @return @note

```
*/
```

```
/*-----*/
```

int app_task_switch_back()

```
/*-----*/  
/**@brief mode switch exit detection  
    @param @return 1: respond to exit  
    mode, 0: do not respond to @note  
  
*/  
/*-----*/
```

u8 app_task_exitting()

```
/*-----*/  
/**@brief get current mode @param  
    @return current mode id @note  
  
*/  
/*-----*/
```

u8 app_get_curr_task()

```
/*-----*/  
/**@brief Check whether it is the current mode by specifying  
    the id @param @return true: is the current mode, false:  
    not the current mode @note  
  
*/  
/*-----*/
```

u8 app_check_curr_task(u8 app)

7. Add mode description

(1) Add the mode id in app_task.h (take music as an example)


```

9 enum {
10     APP_POWERON_TASK = 1,
11     APP_POWEROFF_TASK = 2,
12     APP_BT_TASK = 3,
13     APP_MUSIC_TASK = 4,
14     APP_FM_TASK = 5,
15     APP_RECORD_TASK = 6,
16     APP_LINEIN_TASK = 7,
17     APP_RTC_TASK = 8,
18     APP_PC_TASK = 9,
19     APP_SPDIF_TASK = 10,
20     APP_IDLE_TASK = 11,
21     APP_TASK_MAX_INDEX,
22 };

```

(2) Add the newly added mode id to the mode configuration table app_task_list in app_task_switch.c

```

6 ///模式配置表，这里可以配置切换模式的顺序，方案根据需求定义
7 static const u8 app_task_list[] = {
8     #if TCFG_APP_BT_EN
9         APP_BT_TASK,
10    #endif
11    #if TCFG_APP_MUSIC_EN
12        APP_MUSIC_TASK,
13    #endif
14    #if TCFG_APP_FM_EN
15        APP_FM_TASK,
16    #endif
17    #if TCFG_APP_RECORD_EN
18        APP_RECORD_TASK,
19    #endif
20    #if TCFG_APP_LINEIN_EN
21        APP_LINEIN_TASK,
22    #endif
23    #if TCFG_APP_RTC_EN
24        APP_RTC_TASK,
25    #endif
26    #if TCFG_APP_PC_EN
27        APP_PC_TASK,
28    #endif
29    #if TCFG_APP_SPDIF_EN
30        APP_SPDIF_TASK,
31    #endif
32 };

```

(3) Refer to adding mode key conversion table (ad, io, ir, etc.) in task_key.c (4) Add the

corresponding mode directory (and corresponding header file directory) in task_manager (5)

Implement mode-related interfaces (refer to Existing mode, the following takes music as an example to illustrate)

① Implement the following basic

necessary interfaces: void

app_music_task() int

music_app_check(void) static int music_sys_event_handler(struct sys_event

*event) static int music_key_event_opr(struct sys_event *event) Complete the

② following basic operations in the 2 -mode main loop (app_music_task) Get message

response messages and events

Response mode Internal messages and events

Response to public messages and events

```

void app_music_task()
{
    int res;
    int msg[32];
    music_task_start(); // ← 初始化, 非必要, 根据具体情景定义

    int err = tone_play_with_callback_by_name(tone_table[IDEX_TONE_MUSIC], 1, music_tone_play_end_callback, (void *)IDEX_TONE_MUSIC);
    if (err) {
        music_player_play_start(); // ← 播放模式提示音, 非必要, 如果要的话参考music.c的案例实现
    }

    while (1) { // ← 模式主循环
        app_task_get_msg(msg, ARRAY_SIZE(msg), 1); // ← 获取消息
        switch (msg[0]) {
            case APP_MSG_SYS_EVENT: // ← 处理系统case消息
                if (music_sys_event_handler((struct sys_event *)&msg[1])) // ← 模式内部消息拦截, 如果不拦截, 给公共消息处理响应
                    app_default_event_deal((struct sys_event *)&msg[1]); // ← 必要, 响应公共消息处理
                break;
            default:
                break;
        }
        if (app_task_exitting()) { // ← 必要, 模式退出检测处理
            music_task_close(); // ← 模式退出内部释放处理, 非必要, 如果有初始化, 就一定一定要注意释放操作
            return;
        }
    }
}

```

3 Call the corresponding mode main loop interface (app_music_task) in app_main.c

```


38 void app_task_loop()
39 {
40     while (1) {
41         switch (app_curr_task) {
42             case APP_POWERON_TASK:
43                 log_info("APP_POWERON_TASK \n");
44                 app_poweron_task();
45                 break;
46             case APP_POWEROFF_TASK:
47                 log_info("APP_POWEROFF_TASK \n");
48                 app_poweroff_task();
49                 break;
50             case APP_BT_TASK:
51                 log_info("APP_BT_TASK \n");
52                 app_bt_task();
53                 break;
54             case APP_MUSIC_TASK:
55                 log_info("APP_MUSIC_TASK \n");
56                 app_music_task();
57                 break;
58             case APP_FM_TASK:
59                 log_info("APP_FM_TASK \n");
60                 app_fm_task();
61                 break;
62             case APP_RECORD_TASK:
63                 log_info("APP_RECORD_TASK \n");
64                 app_record_task();
65                 break;

```

4 Implementation of the app_check interface (for example, music) app_check

is actually when switching modes, whether the conditions are met to enter the mode, and the entry condition of music mode is to judge whether there is a device that can be played online, so the interface is implemented as follows:

```
634 int music_app_check(void)
635 {
636     if (dev_manager_get_total(1)) {
637         return true;
638     }
639     return false;
640 }
```



⑤ Call app_check in app_task_switch_check (music as an example)

```
107 static int app_task_switch_check(u8 app_task)
108 {
109     int ret = false;
110     switch (app_task) {
111 #if TCFG_APP_MUSIC_EN
112         case APP_MUSIC_TASK:
113             ret = music_app_check();
114             break;
115 #endif
116 #if TCFG_APP_LINEIN_EN
117         case APP_LINEIN_TASK:
118             ret = linein_app_check();
119             break;
120 #endif
121 #if TCFG_APP_PC_EN
```

