**发布时间：2022/09/19**
**此文档适用于 wireless_mic_sdk-v1.4.0_duplex SDK 及以上版本**

说明：

此文档用于说明如何添加按键实现双工 SDK 的绑定配对功能，以及如何实现一端绑定配对功能

## 1. 功能说明

绑定配对用于实现 earphone 端和 dongle 端的绑定连接。以下有两种实现场景：

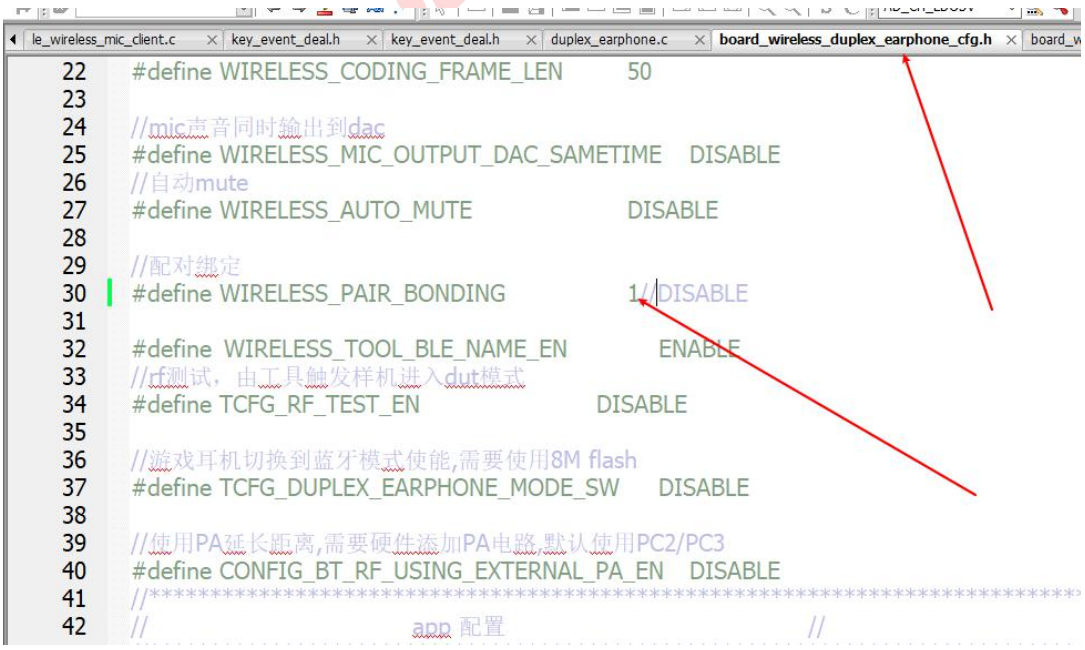
**场景一**：实现两端绑定配对，dongle 端和 earphone 端绑定之后将不能与其他设备进行连接，除非调用用接口清除配对记录；

**场景二**：实现 earphone 端添加绑定配对，dongle 端不添加绑定配对（注：dongle 只记录一个 earphone 端地址，连上新的 earphone 端，旧的将被覆盖无法回连）；

## 2. 场景一

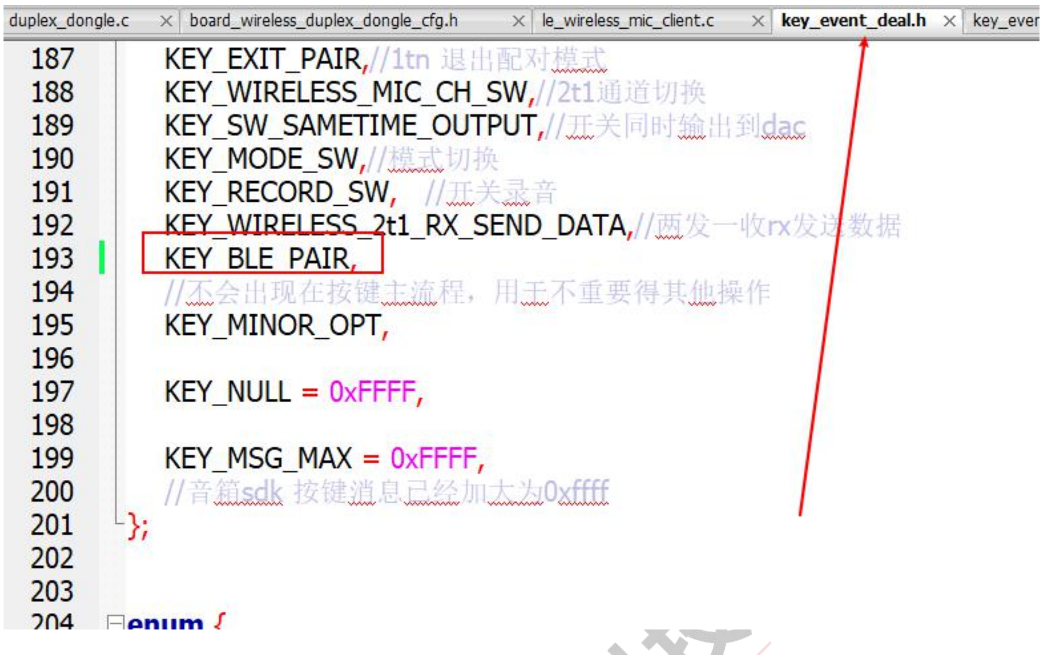
### 2.1. earphone 端修改

（1）开启绑定配对的宏

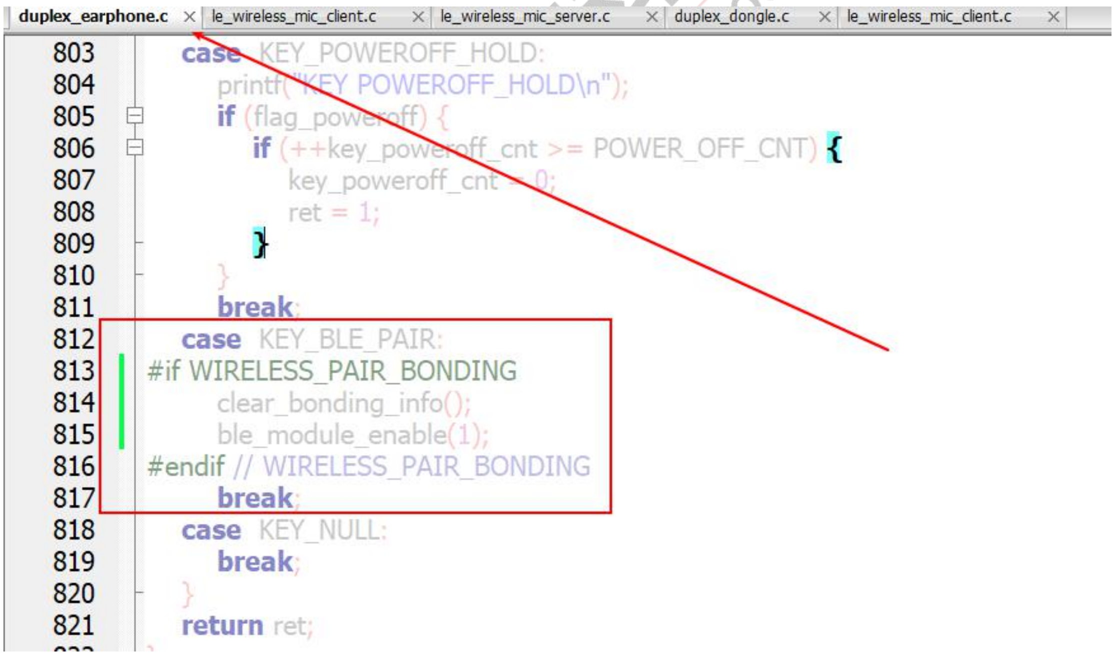


```
#define WIRELESS_CODING_FRAME_LEN        50

//mic声音同时输出到dac
#define WIRELESS_MIC_OUTPUT_DAC_SAMETIME    DISABLE
//自动mute
#define WIRELESS_AUTO_MUTE                  DISABLE

//配对绑定
#define WIRELESS_PAIR_BONDING               1//DISABLE

#define  WIRELESS_TOOL_BLE_NAME_EN          ENABLE
//rf测试，由工具触发样机进入dut模式
#define TCFG_RF_TEST_EN                  DISABLE

//游戏耳机切换到蓝牙模式使能,需要使用8M flash
#define TCFG_DUPLEX_EARPHONE_MODE_SW        DISABLE

//使用PA延长距离,需要硬件添加PA电路,默认使用PC2/PC3
#define CONFIG_BT_RF_USING_EXTERNAL_PA_EN    DISABLE
//***********************************************************************
//                          app 配置                          //
```

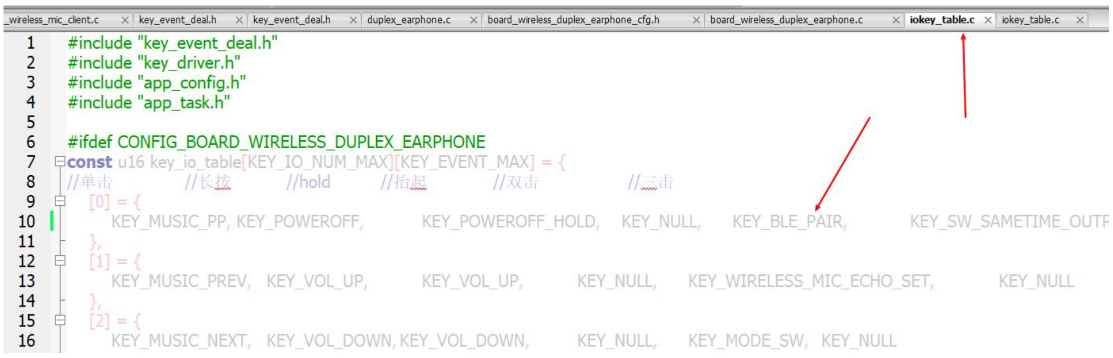
（2）添加按键清除绑定记录



```
    KEY_EXIT_PAIR,//1tn 退出配对模式
    KEY_WIRELESS_MIC_CH_SW,//2t1通道切换
    KEY_SW_SAMETIME_OUTPUT,//开关同时输出到dac
    KEY_MODE_SW,//模式切换
    KEY_RECORD_SW,  //开关录音
    KEY_WIRELESS_2t1_RX_SEND_DATA,//两发一收rx发送数据
    KEY_BLE_PAIR,
    //不会出现在按键主流程，用于不重要得其他操作
    KEY_MINOR_OPT,

    KEY_NULL = 0xFFFF,

    KEY_MSG_MAX = 0xFFFF,
    //音箱sdk 按键消息已经加大为0xffff
};


enum {
```

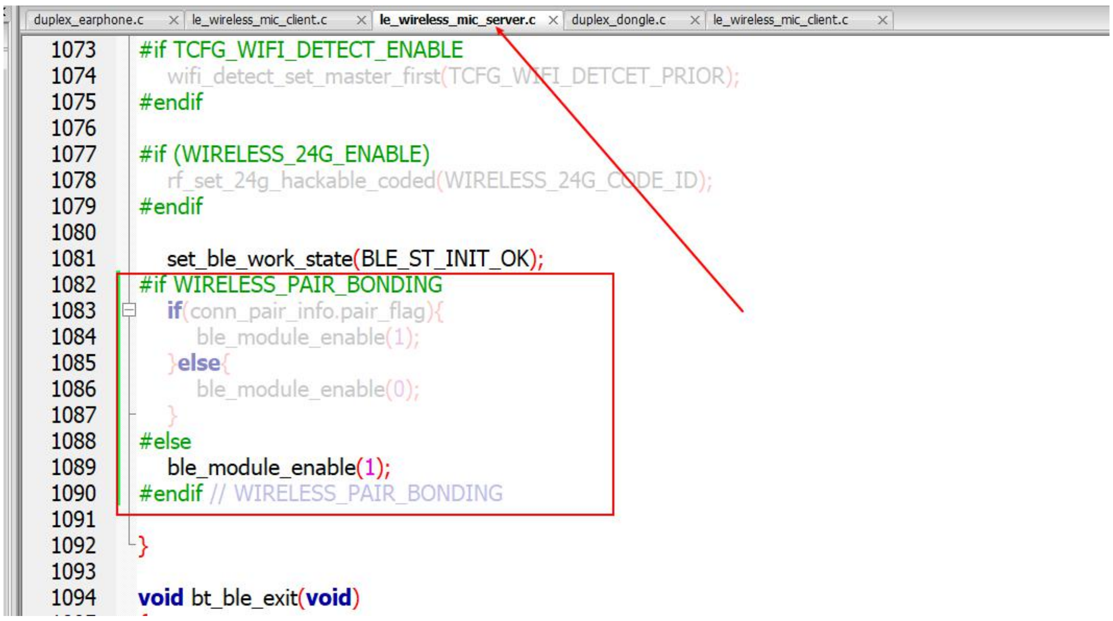


```
    case KEY_POWEROFF_HOLD:
        printf("KEY POWEROFF_HOLD\n");
        if (flag_poweroff) {
            if (++key_poweroff_cnt >= POWER_OFF_CNT) {
                key_poweroff_cnt = 0;
                ret = 1;
            }
        }
        break;
    case KEY_BLE_PAIR:
#if WIRELESS_PAIR_BONDING
        clear_bonding_info();
        ble_module_enable(1);
#endif // WIRELESS_PAIR_BONDING
        break;
    case KEY_NULL:
        break;
    }
    return ret;
```



```
#include "key_event_deal.h"
#include "key_driver.h"
#include "app_config.h"
#include "app_task.h"

#ifdef CONFIG_BOARD_WIRELESS_DUPLEX_EARPHONE
const u16 key_io_table[KEY_IO_NUM_MAX][KEY_EVENT_MAX] = {
//单击          //长按          //hold          //抬起          //双击          //三击
    [0] = {
        KEY_MUSIC_PP, KEY_POWEROFF,         KEY_POWEROFF_HOLD,  KEY_NULL,    KEY_BLE_PAIR,          KEY_SW_SAMETIME_OUTF
    },
    [1] = {
        KEY_MUSIC_PREV,  KEY_VOL_UP,         KEY_VOL_UP,         KEY_NULL,    KEY_WIRELESS_MIC_ECHO_SET,         KEY_NULL
    },
    [2] = {
        KEY_MUSIC_NEXT,  KEY_VOL_DOWN, KEY_VOL_DOWN,        KEY_NULL,    KEY_MODE_SW,  KEY_NULL
```

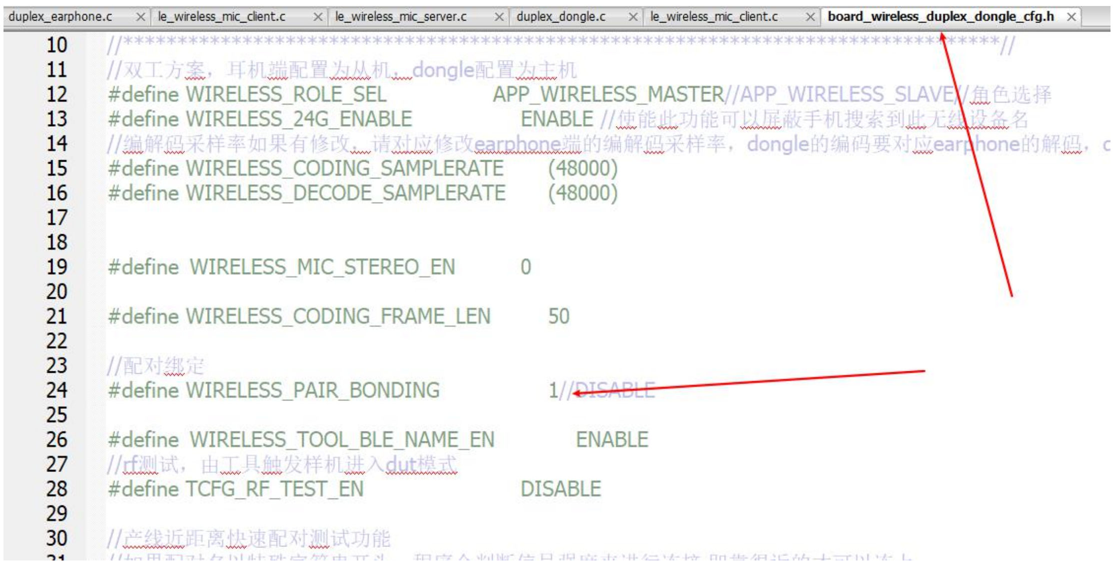
（3）控制第一次上电连接状态



```
#if TCFG_WIFI_DETECT_ENABLE
    wifi_detect_set_master_first(TCFG_WIFI_DETCET_PRIOR);
#endif

#if (WIRELESS_24G_ENABLE)
    rf_set_24g_hackable_coded(WIRELESS_24G_CODE_ID);
#endif

    set_ble_work_state(BLE_ST_INIT_OK);
#if WIRELESS_PAIR_BONDING
    if(conn_pair_info.pair_flag){
        ble_module_enable(1);
    }else{
        ble_module_enable(0);
    }
#else
    ble_module_enable(1);
#endif // WIRELESS_PAIR_BONDING

}

void bt_ble_exit(void)
```

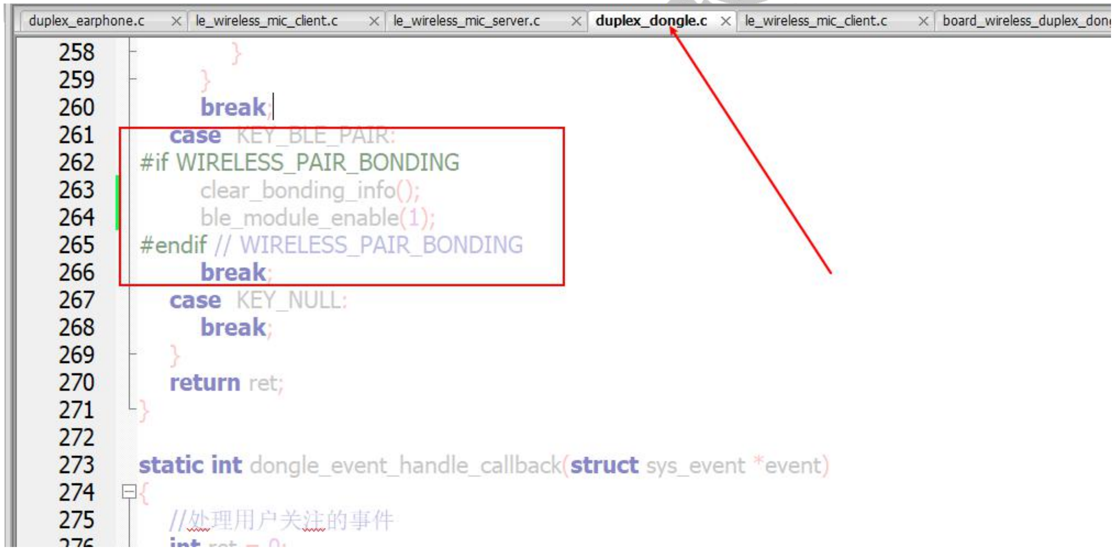
## 2.2. dongle 端相关修改

（1）开启绑定配对的宏

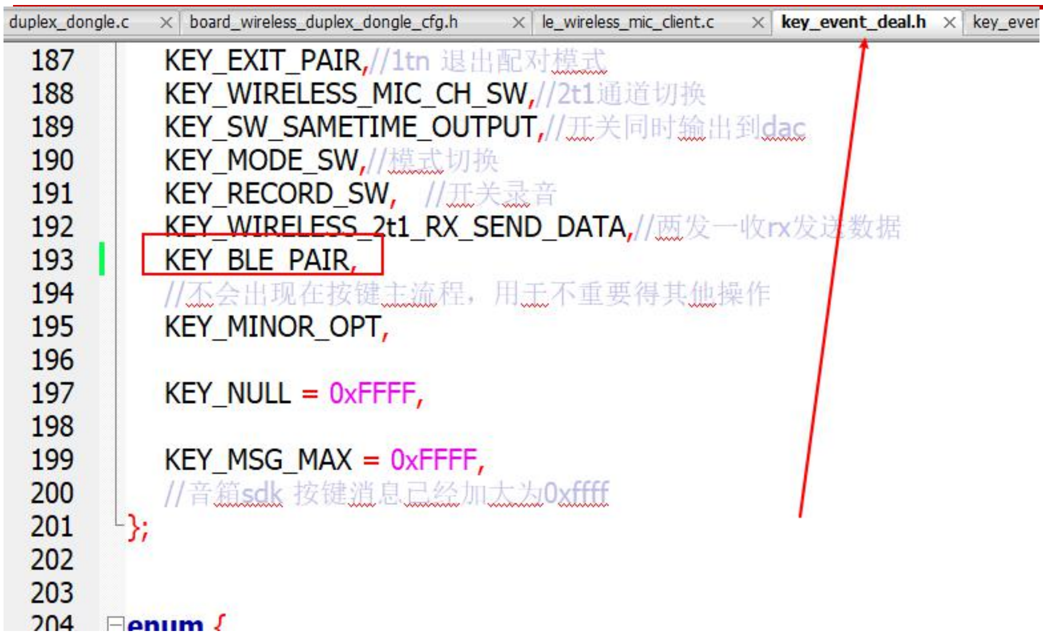

duplex_earphone.c | le_wireless_mic_client.c | le_wireless_mic_server.c | duplex_dongle.c | le_wireless_mic_client.c | board_wireless_duplex_dongle_cfg.h

```
//*************************************************************************************//
//双工方案，耳机端配置为从机，dongle配置为主机
#define WIRELESS_ROLE_SEL              APP_WIRELESS_MASTER//APP_WIRELESS_SLAVE//角色选择
#define WIRELESS_24G_ENABLE              ENABLE //使能此功能可以屏蔽手机搜索到此无线设备名
//编解码采样率如果有修改，请对应修改earphone端的编解码采样率，dongle的编码要对应earphone的解码，c
#define WIRELESS_CODING_SAMPLERATE      (48000)
#define WIRELESS_DECODE_SAMPLERATE      (48000)


#define  WIRELESS_MIC_STEREO_EN         0

#define WIRELESS_CODING_FRAME_LEN        50

//配对绑定
#define WIRELESS_PAIR_BONDING             1//DISABLE

#define  WIRELESS_TOOL_BLE_NAME_EN          ENABLE
//rf测试，由工具触发样机进入dut模式
#define TCFG_RF_TEST_EN                  DISABLE

//产线近距离快速配对测试功能
```

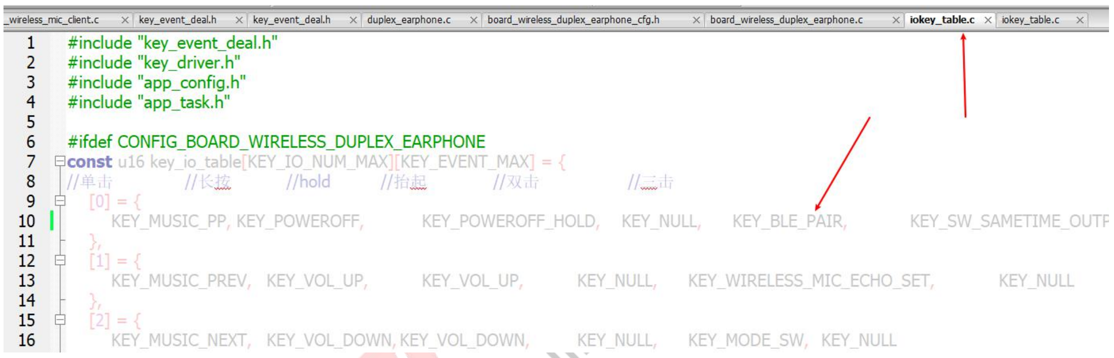
（2）添加按键清除绑定配对


duplex_earphone.c | le_wireless_mic_client.c | le_wireless_mic_server.c | duplex_dongle.c | le_wireless_mic_client.c | board_wireless_duplex_don

```
                }
            }
            break;
        case KEY_BLE_PAIR:
#if WIRELESS_PAIR_BONDING
            clear_bonding_info();
            ble_module_enable(1);
#endif // WIRELESS_PAIR_BONDING
            break;
        case KEY_NULL:
            break;
        }
        return ret;
}

static int dongle_event_handle_callback(struct sys_event *event)
{
        //处理用户关注的事件
```

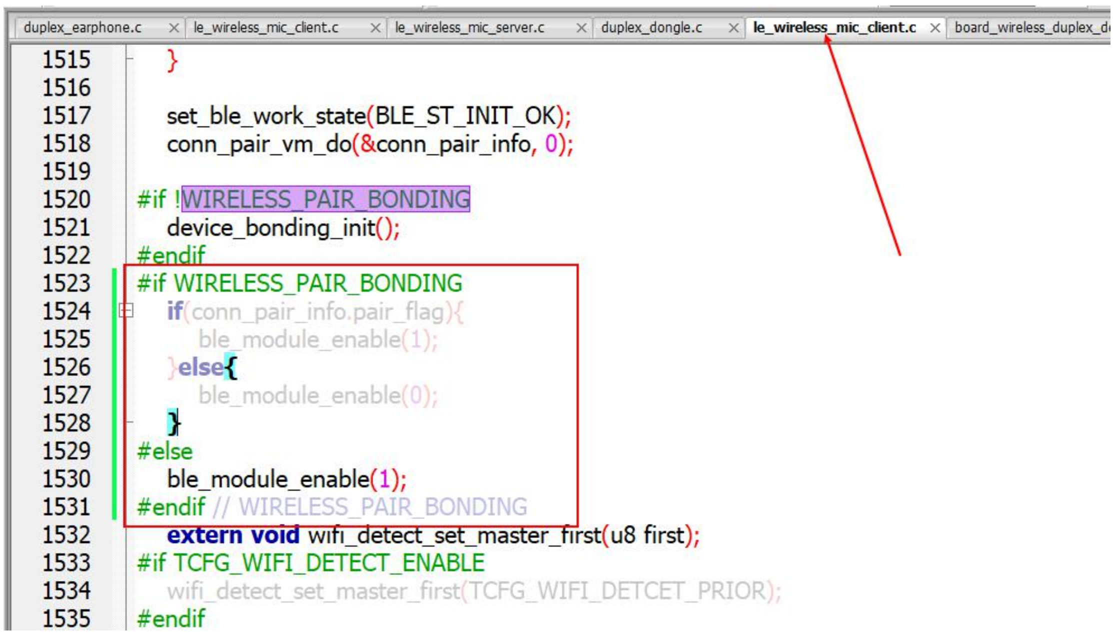


```
    KEY_EXIT_PAIR,//1tn 退出配对模式
    KEY_WIRELESS_MIC_CH_SW,//2t1通道切换
    KEY_SW_SAMETIME_OUTPUT,//开关同时输出到dac
    KEY_MODE_SW,//模式切换
    KEY_RECORD_SW,  //开关录音
    KEY_WIRELESS_2t1_RX_SEND_DATA,//两发一收rx发送数据
    KEY_BLE_PAIR,
    //不会出现在按键主流程，用于不重要得其他操作
    KEY_MINOR_OPT,

    KEY_NULL = 0xFFFF,

    KEY_MSG_MAX = 0xFFFF,
    //音箱sdk 按键消息已经加大为0xffff
};

enum {
```



```
#include "key_event_deal.h"
#include "key_driver.h"
#include "app_config.h"
#include "app_task.h"

#ifdef CONFIG_BOARD_WIRELESS_DUPLEX_EARPHONE
const u16 key_io_table[KEY_IO_NUM_MAX][KEY_EVENT_MAX] = {
//单击          //长按          //hold          //抬起          //双击          //三击
    [0] = {
        KEY_MUSIC_PP, KEY_POWEROFF,         KEY_POWEROFF_HOLD,   KEY_NULL,      KEY_BLE_PAIR,           KEY_SW_SAMETIME_OUTP
    },
    [1] = {
        KEY_MUSIC_PREV,   KEY_VOL_UP,          KEY_VOL_UP,          KEY_NULL,      KEY_WIRELESS_MIC_ECHO_SET,          KEY_NULL
    },
    [2] = {
        KEY_MUSIC_NEXT,   KEY_VOL_DOWN, KEY_VOL_DOWN,           KEY_NULL,      KEY_MODE_SW,   KEY_NULL
```

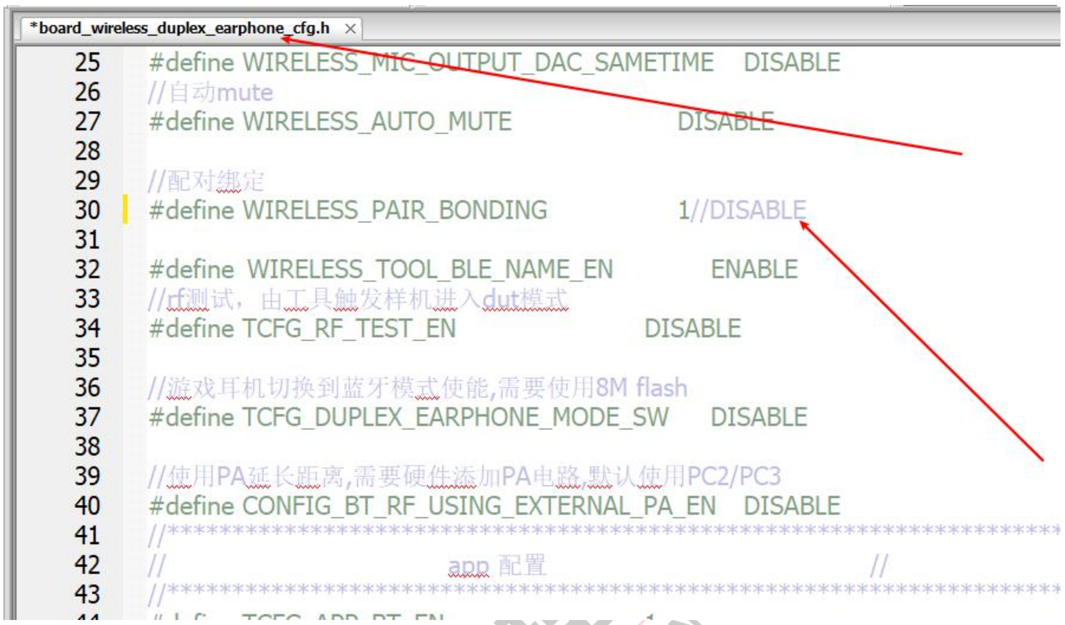
（3）控制第一次上电连接状态



```
    }

    set_ble_work_state(BLE_ST_INIT_OK);
    conn_pair_vm_do(&conn_pair_info, 0);

#if !WIRELESS_PAIR_BONDING
    device_bonding_init();
#endif
#if WIRELESS_PAIR_BONDING
    if(conn_pair_info.pair_flag){
        ble_module_enable(1);
    }else{
        ble_module_enable(0);
    }
#else
    ble_module_enable(1);
#endif // WIRELESS_PAIR_BONDING
    extern void wifi_detect_set_master_first(u8 first);
#if TCFG_WIFI_DETECT_ENABLE
    wifi_detect_set_master_first(TCFG_WIFI_DETCET_PRIOR);
#endif
```

## 3. 场景二

### 3.1. earphone 端开启绑定配对



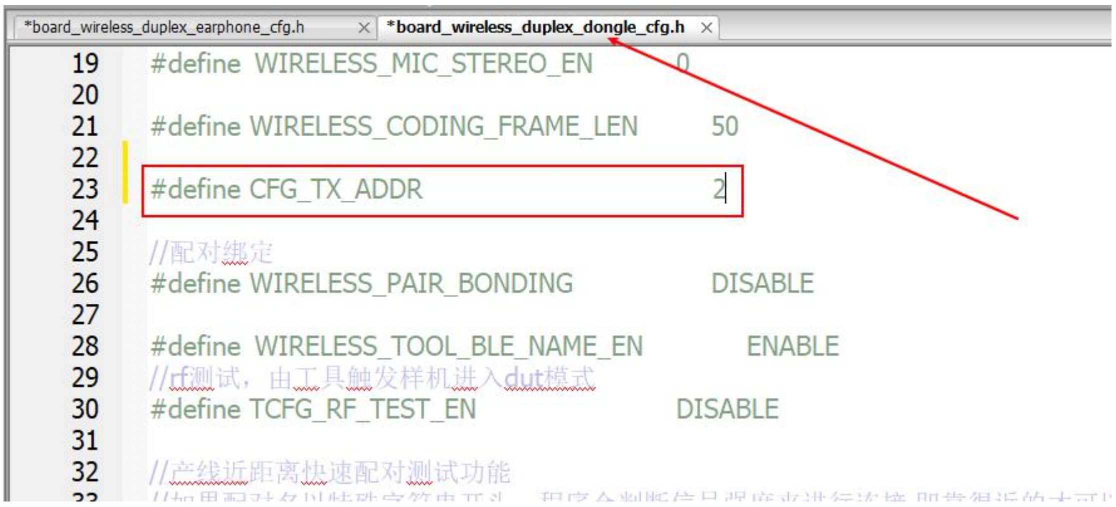
*board_wireless_duplex_earphone_cfg.h

```
#define WIRELESS_MIC_OUTPUT_DAC_SAMETIME    DISABLE
//自动mute
#define WIRELESS_AUTO_MUTE                DISABLE

//配对绑定
#define WIRELESS_PAIR_BONDING             1//DISABLE

#define  WIRELESS_TOOL_BLE_NAME_EN          ENABLE
//rf测试，由工具触发样机进入dut模式
#define TCFG_RF_TEST_EN                   DISABLE

//游戏耳机切换到蓝牙模式使能,需要使用8M flash
#define TCFG_DUPLEX_EARPHONE_MODE_SW        DISABLE

//使用PA延长距离,需要硬件添加PA电路,默认使用PC2/PC3
#define CONFIG_BT_RF_USING_EXTERNAL_PA_EN   DISABLE
//*************************************************************************
//                          app 配置                                 //
//*************************************************************************
```

### 3.2. earphone 添加按键清除配对消息

参考案例一的 earphone 端修改的（2）；

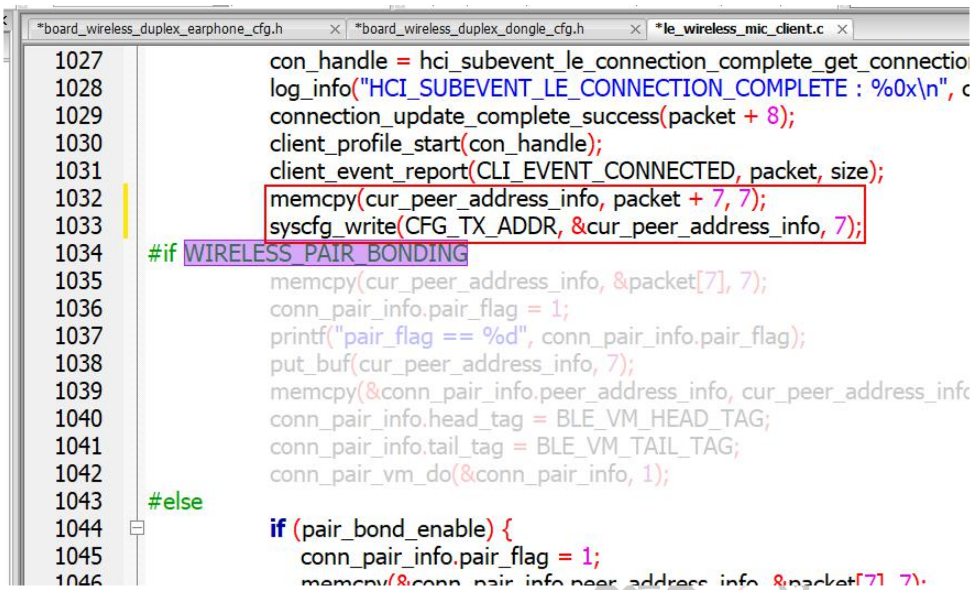
### 3.3. dongle 端添加用来保存地址的 VM 区



*board_wireless_duplex_earphone_cfg.h   *board_wireless_duplex_dongle_cfg.h

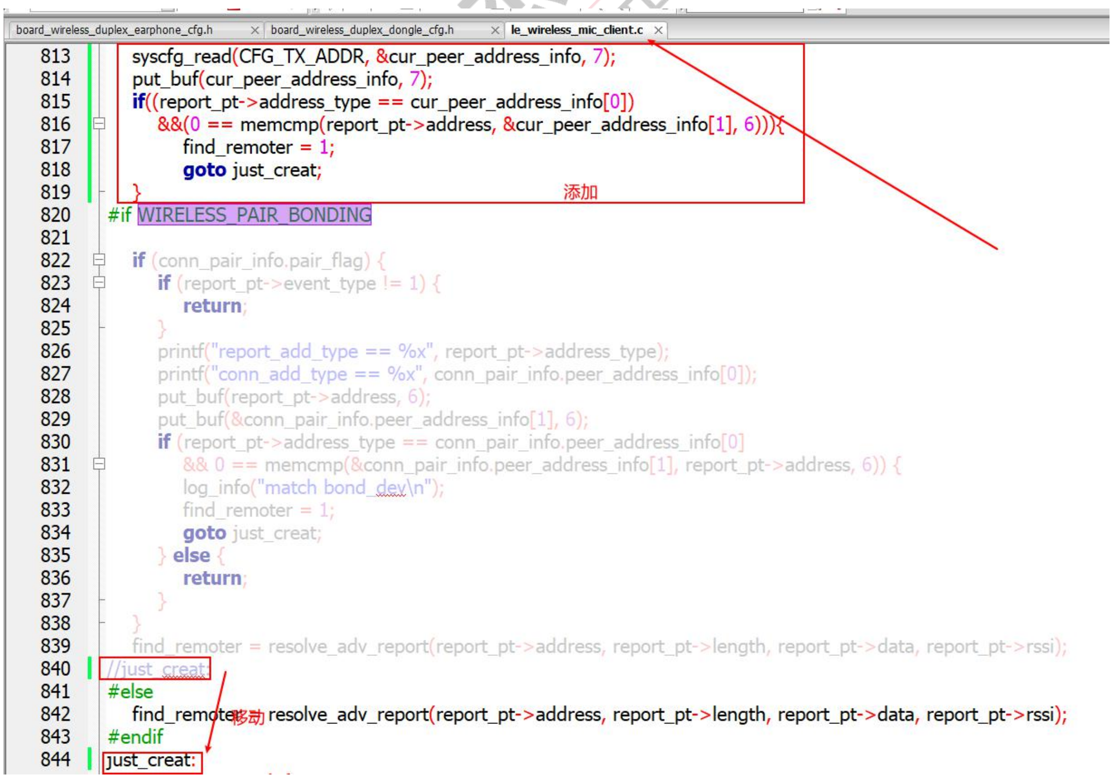
```
#define  WIRELESS_MIC_STEREO_EN           0

#define WIRELESS_CODING_FRAME_LEN          50

#define CFG_TX_ADDR                        2

//配对绑定
#define WIRELESS_PAIR_BONDING              DISABLE

#define  WIRELESS_TOOL_BLE_NAME_EN           ENABLE
//rf测试，由工具触发样机进入dut模式
#define TCFG_RF_TEST_EN                    DISABLE

//产线近距离快速配对测试功能
```

## 3.4. dongle 端记录连接的地址


*board_wireless_duplex_earphone_cfg.h
*board_wireless_duplex_dongle_cfg.h
*le_wireless_mic_client.c

```
                con_handle = hci_subevent_le_connection_complete_get_connectio
                log_info("HCI_SUBEVENT_LE_CONNECTION_COMPLETE : %0x\n", c
                connection_update_complete_success(packet + 8);
                client_profile_start(con_handle);
                client_event_report(CLI_EVENT_CONNECTED, packet, size);
                memcpy(cur_peer_address_info, packet + 7, 7);
                syscfg_write(CFG_TX_ADDR, &cur_peer_address_info, 7);
    #if WIRELESS_PAIR_BONDING
                memcpy(cur_peer_address_info, &packet[7], 7);
                conn_pair_info.pair_flag = 1;
                printf("pair_flag == %d", conn_pair_info.pair_flag);
                put_buf(cur_peer_address_info, 7);
                memcpy(&conn_pair_info.peer_address_info, cur_peer_address_infc
                conn_pair_info.head_tag = BLE_VM_HEAD_TAG;
                conn_pair_info.tail_tag = BLE_VM_TAIL_TAG;
                conn_pair_vm_do(&conn_pair_info, 1);
    #else
                if (pair_bond_enable) {
                    conn_pair_info.pair_flag = 1;
                    memcpy(&conn_pair_info.peer_address_info, &packet[7], 7);
```

## 3.5. dongle 判断到记录的地址，直接走连接流程


board_wireless_duplex_earphone_cfg.h
board_wireless_duplex_dongle_cfg.h
le_wireless_mic_client.c
添加
移动

```
    syscfg_read(CFG_TX_ADDR, &cur_peer_address_info, 7);
    put_buf(cur_peer_address_info, 7);
    if((report_pt->address_type == cur_peer_address_info[0])
       &&(0 == memcmp(report_pt->address, &cur_peer_address_info[1], 6))){
        find_remoter = 1;
        goto just_creat;
    }
#if WIRELESS_PAIR_BONDING

    if (conn_pair_info.pair_flag) {
        if (report_pt->event_type != 1) {
            return;
        }
        printf("report_add_type == %x", report_pt->address_type);
        printf("conn_add_type == %x", conn_pair_info.peer_address_info[0]);
        put_buf(report_pt->address, 6);
        put_buf(&conn_pair_info.peer_address_info[1], 6);
        if (report_pt->address_type == conn_pair_info.peer_address_info[0]
            && 0 == memcmp(&conn_pair_info.peer_address_info[1], report_pt->address, 6)) {
            log_info("match bond_dev\n");
            find_remoter = 1;
            goto just_creat;
        } else {
            return;
        }
    }
    find_remoter = resolve_adv_report(report_pt->address, report_pt->length, report_pt->data, report_pt->rssi);
//just_creat:
#else
    find_remoter = resolve_adv_report(report_pt->address, report_pt->length, report_pt->data, report_pt->rssi);
#endif
just_creat:
```