

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：

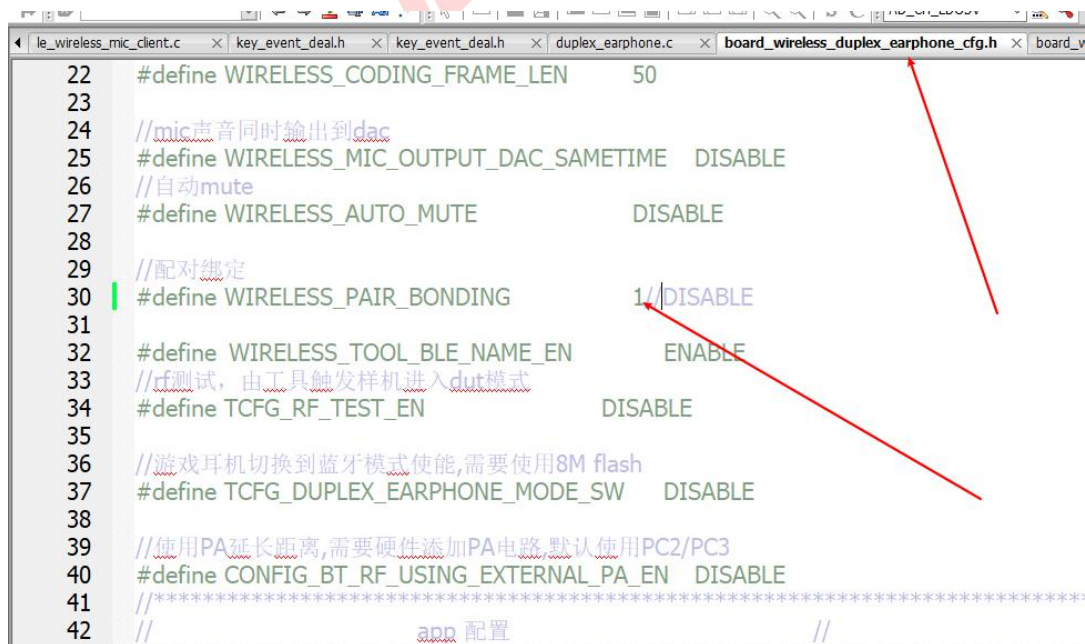
场景一：实现两端绑定配对，dongle 端和 earphone 端绑定之后将不能与其他设备进行连接，除非调用接口清除配对记录；

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



(2) 添加按键清除绑定记录

```
duplex_dongle.c x board_wireless_duplex_dongle_cfg.h x le_wireless_mic_client.c x key_event_deal.h x key_event_deal.c
187 KEY_EXIT_PAIR, //1tn 退出配对模式
188 KEY_WIRELESS_MIC_CH_SW, //2t1通道切换
189 KEY_SW_SAMETIME_OUTPUT, //开关同时输出到dac
190 KEY_MODE_SW, //模式切换
191 KEY_RECORD_SW, //开关录音
192 KEY_WIRELESS_2t1_RX_SEND_DATA, //两发一收rx发送数据
193 KEY_BLE_PAIR,
194 //不会出现在按键主流程, 用于不重要得其他操作
195 KEY_MINOR_OPT,
196
197 KEY_NULL = 0xFFFF,
198
199 KEY_MSG_MAX = 0xFFFF,
200 //音箱sdk 按键消息已经加大为0xffff
201 };
202
203
204 enum
```

```
duplex_earphone.c x le_wireless_mic_client.c x le_wireless_mic_server.c x duplex_dongle.c x le_wireless_mic_client.c x
803 case KEY_POWEROFF_HOLD:
804     printf("KEY POWEROFF_HOLD\n");
805     if (flag_poweroff) {
806         if (++key_poweroff_cnt >= POWER_OFF_CNT) {
807             key_poweroff_cnt = 0;
808             ret = 1;
809         }
810     }
811     break;
812 case KEY_BLE_PAIR:
813     #if WIRELESS_PAIR_BONDING
814         clear_bonding_info();
815         ble_module_enable(1);
816     #endif // WIRELESS_PAIR_BONDING
817     break;
818 case KEY_NULL:
819     break;
820 }
821 return ret;
822
```

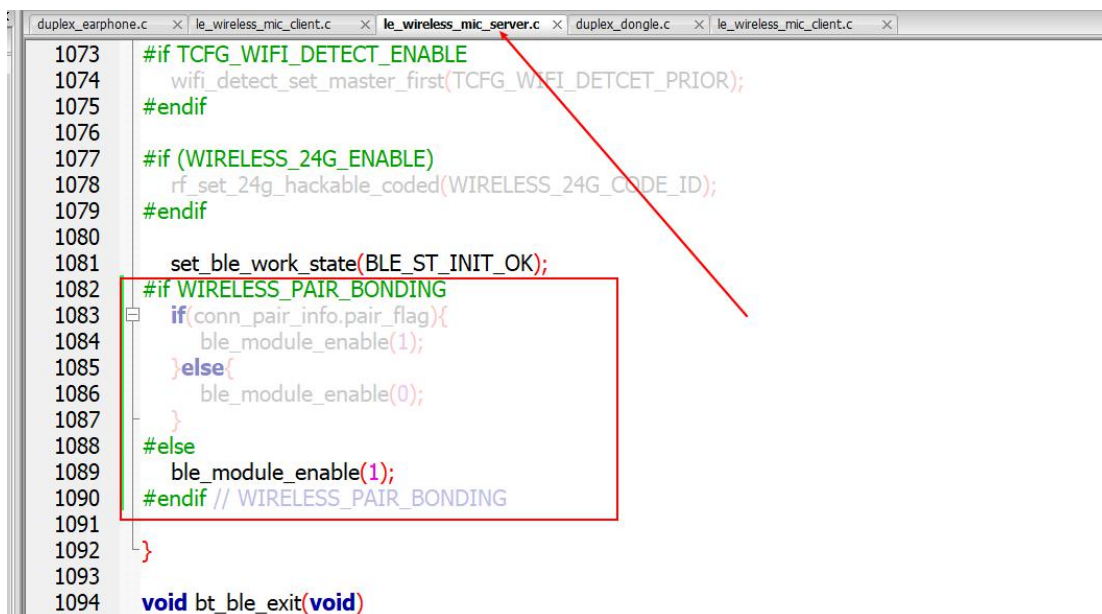


```

1  #include "key_event_deal.h"
2  #include "key_driver.h"
3  #include "app_config.h"
4  #include "app_task.h"
5
6  #ifdef CONFIG_BOARD_WIRELESS_DUPLEX_EARPHONE
7  const u16 key_io_table[KEY_IO_NUM_MAX][KEY_EVENT_MAX] = {
8      //单击      //长按      //hold      //抬起      //双击      //三击
9      [0] = {
10         KEY_MUSIC_PP, KEY_POWEROFF, KEY_POWEROFF_HOLD, KEY_NULL, KEY_BLE_PAIR, KEY_SW_SAMETIME_OUTF
11     },
12     [1] = {
13         KEY_MUSIC_PREV, KEY_VOL_UP, KEY_VOL_UP, KEY_NULL, KEY_WIRELESS_MIC_ECHO_SET, KEY_NULL
14     },
15     [2] = {
16         KEY_MUSIC_NEXT, KEY_VOL_DOWN, KEY_VOL_DOWN, KEY_NULL, KEY_MODE_SW, KEY_NULL
17     }
18 }
19 #endif

```

(3) 控制第一次上电连接状态



```

1073 #if TCFG_WIFI_DETECT_ENABLE
1074     wifi_detect_set_master_first(TCFG_WIFI_DETECT_PRIOR);
1075 #endif
1076
1077 #if (WIRELESS_24G_ENABLE)
1078     rf_set_24g_hackable_coded(WIRELESS_24G_CODE_ID);
1079 #endif
1080
1081     set_ble_work_state(BLE_ST_INIT_OK);
1082     #if WIRELESS_PAIR_BONDING
1083         if(conn_pair_info.pair_flag){
1084             ble_module_enable(1);
1085         }else{
1086             ble_module_enable(0);
1087         }
1088     #else
1089         ble_module_enable(1);
1090     #endif // WIRELESS_PAIR_BONDING
1091 }
1092
1093
1094 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 x le_wireless_mic_client.c x le_wireless_mic_server.c x duplex_dongle.c x le_wireless_mic_client.c x board_wireless_duplex_dongle_cfg.h x
10 //*****
11 //双工方案，耳端配置为从机，dongle配置为主机
12 #define WIRELESS_ROLE_SEL APP_WIRELESS_MASTER//APP_WIRELESS_SLAVE//角色选择
13 #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
14 //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
15 #define WIRELESS_CODING_SAMPLERATE (48000)
16 #define WIRELESS_DECODE_SAMPLERATE (48000)
17
18
19 #define WIRELESS_MIC_STEREO_EN 0
20
21 #define WIRELESS_CODING_FRAME_LEN 50
22
23 //配对绑定
24 #define WIRELESS_PAIR_BONDING 1//DISABLE
25
26 #define WIRELESS_TOOL_BLE_NAME_EN ENABLE
27 //测试，由工具触发样机进入dut模式
28 #define TCFG_RF_TEST_EN DISABLE
29
30 //产线近距离快速配对测试功能
31 //使用到该功能时，请对应修改耳端和dongle端的配置，如需要修改耳端配置，请参考耳端配置说明，如需要修改dongle配置，请参考dongle配置说明

```

(2) 添加按键清除绑定配对

```

duplex_earphone.c x le_wireless_mic_client.c x le_wireless_mic_server.c x duplex_dongle.c x le_wireless_mic_client.c x board_wireless_duplex_dongle_cfg.h x
258 }
259 }
260 break;
261 case KEY_BLE_PAIR:
262 #if WIRELESS_PAIR_BONDING
263 clear_bonding_info();
264 ble_module_enable(1);
265 #endif // WIRELESS_PAIR_BONDING
266 break;
267 case KEY_NULL:
268 break;
269 }
270 return ret;
271 }
272
273 static int dongle_event_handle_callback(struct sys_event *event)
274 {
275 //处理用户关注的事件
276

```



```

duplex_dongle.c  x  board_wireless_duplex_dongle_cfg.h  x  le_wireless_mic_client.c  x  key_event_deal.h  x  key_event_deal.h
187  KEY_EXIT_PAIR, //1tn 退出配对模式
188  KEY_WIRELESS_MIC_CH_SW, //2t1通道切换
189  KEY_SW_SAMETIME_OUTPUT, //开关同时输出到dac
190  KEY_MODE_SW, //模式切换
191  KEY_RECORD_SW, //开关录音
192  KEY_WIRELESS_2t1_RX_SEND_DATA, //两发一收rx发送数据
193  KEY_BLE_PAIR,
194  //不会出现在按键主流程, 用于不重要得其他操作
195  KEY_MINOR_OPT,
196
197  KEY_NULL = 0xFFFF,
198
199  KEY_MSG_MAX = 0xFFFF,
200  //音箱sdk 按键消息已经加大为0xffff
201  };
202
203
204  enum

```

```

wireless_mic_client.c  x  key_event_deal.h  x  key_event_deal.h  x  duplex_earphone.c  x  board_wireless_duplex_earphone_cfg.h  x  board_wireless_duplex_earphone.c  x  iokey_table.c  x  iokey_table.c
1  #include "key_event_deal.h"
2  #include "key_driver.h"
3  #include "app_config.h"
4  #include "app_task.h"
5
6  #ifdef CONFIG_BOARD_WIRELESS_DUPLEX_EARPHONE
7  #const u16 key_io_table[KEY_IO_NUM_MAX][KEY_EVENT_MAX] = {
8  //单击 //长按 //hold //抬起 //双击 //三击
9  [0] = {
10 KEY_MUSIC_PP, KEY_POWEROFF, KEY_POWEROFF_HOLD, KEY_NULL, KEY_BLE_PAIR, KEY_SW_SAMETIME_OUTP
11 },
12 [1] = {
13 KEY_MUSIC_PREV, KEY_VOL_UP, KEY_VOL_UP, KEY_NULL, KEY_WIRELESS_MIC_ECHO_SET, KEY_NULL
14 },
15 [2] = {
16 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

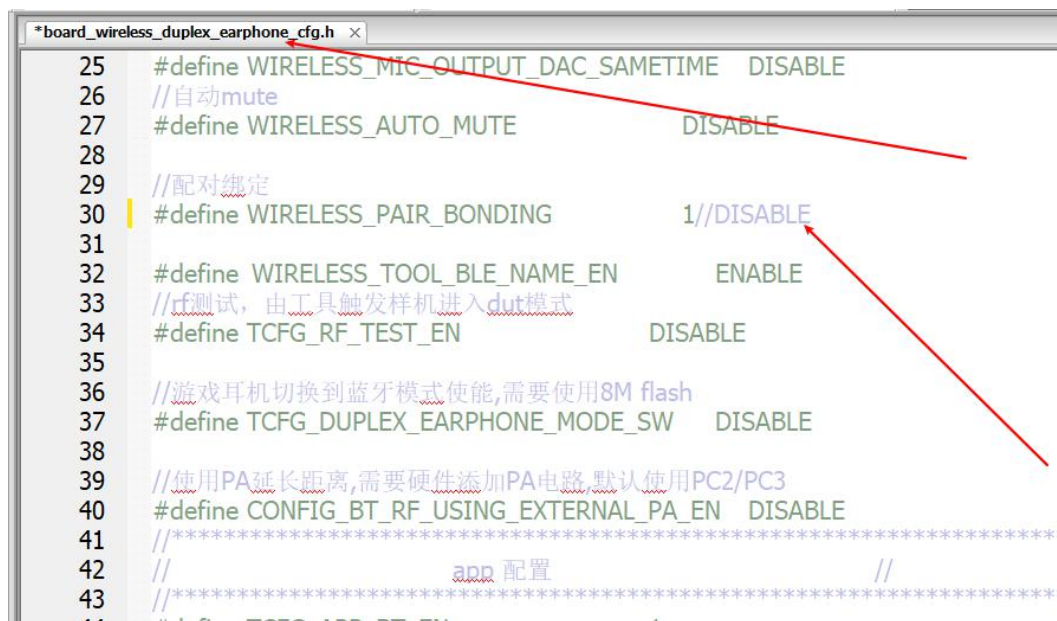
```

duplex_earphone.c  x  le_wireless_mic_client.c  x  le_wireless_mic_server.c  x  duplex_dongle.c  x  le_wireless_mic_client.c  x  board_wireless_duplex_d
1515  }
1516
1517  set_ble_work_state(BLE_ST_INIT_OK);
1518  conn_pair_vm_do(&conn_pair_info, 0);
1519
1520  #if !WIRELESS_PAIR_BONDING
1521  device_bonding_init();
1522  #endif
1523  #if WIRELESS_PAIR_BONDING
1524  if(conn_pair_info.pair_flag){
1525      ble_module_enable(1);
1526  }else{
1527      ble_module_enable(0);
1528  }
1529  #else
1530  ble_module_enable(1);
1531  #endif // WIRELESS_PAIR_BONDING
1532  extern void wifi_detect_set_master_first(u8 first);
1533  #if TCFG_WIFI_DETECT_ENABLE
1534  wifi_detect_set_master_first(TCFG_WIFI_DETECT_PRIOR);
1535  #endif

```

3. 场景二

3.1. earphone 端开启绑定配对



```

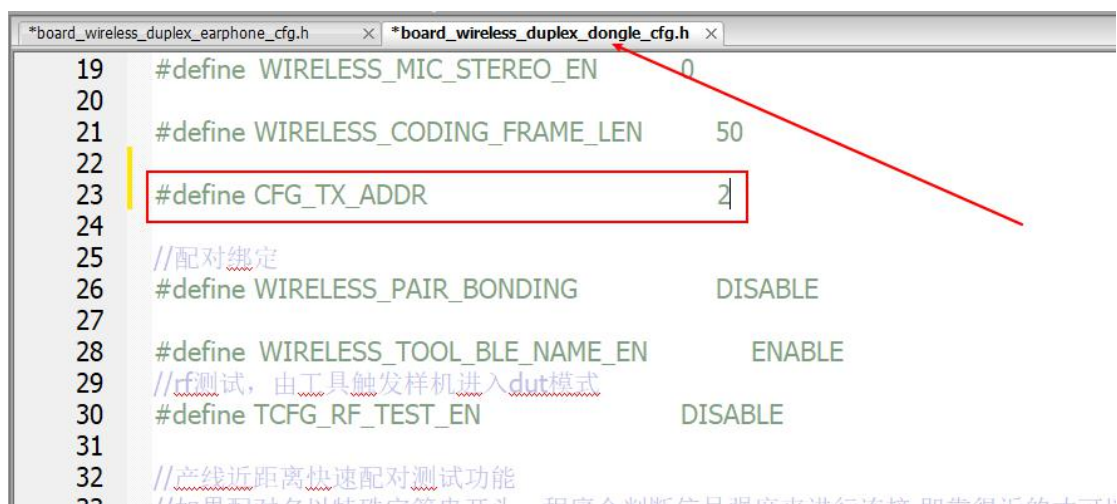
25 #define WIRELESS_MIC_OUTPUT_DAC_SAMETIME DISABLE
26 //自动mute
27 #define WIRELESS_AUTO_MUTE DISABLE
28
29 //配对绑定
30 #define WIRELESS_PAIR_BONDING 1//DISABLE
31
32 #define WIRELESS_TOOL_BLE_NAME_EN ENABLE
33 //rf测试, 由工具触发样机进入dut模式
34 #define TCFG_RF_TEST_EN DISABLE
35
36 //游戏耳机切换到蓝牙模式使能,需要使用8M flash
37 #define TCFG_DUPLEX_EARPHONE_MODE_SW DISABLE
38
39 //使用PA延长距离,需要硬件添加PA电路,默认使用PC2/PC3
40 #define CONFIG_BT_RF_USING_EXTERNAL_PA_EN DISABLE
41 //*****
42 // app 配置 //
43 //*****
44 #define TCFG_APP_BT_EN

```

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

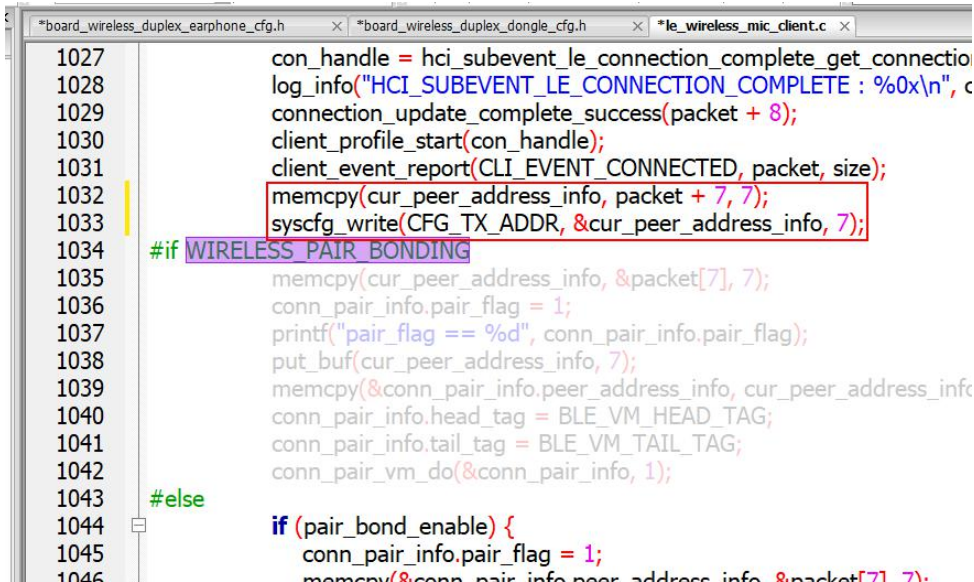


```

19 #define WIRELESS_MIC_STEREO_EN 0
20
21 #define WIRELESS_CODING_FRAME_LEN 50
22
23 #define CFG_TX_ADDR 4
24
25 //配对绑定
26 #define WIRELESS_PAIR_BONDING DISABLE
27
28 #define WIRELESS_TOOL_BLE_NAME_EN ENABLE
29 //rf测试, 由工具触发样机进入dut模式
30 #define TCFG_RF_TEST_EN DISABLE
31
32 //产线近距离快速配对测试功能
33 //使用配对工具触发样机进入dut模式, 和产线测试设备连接, 即产线测试设备

```

3.4. dongle 端记录连接的地址

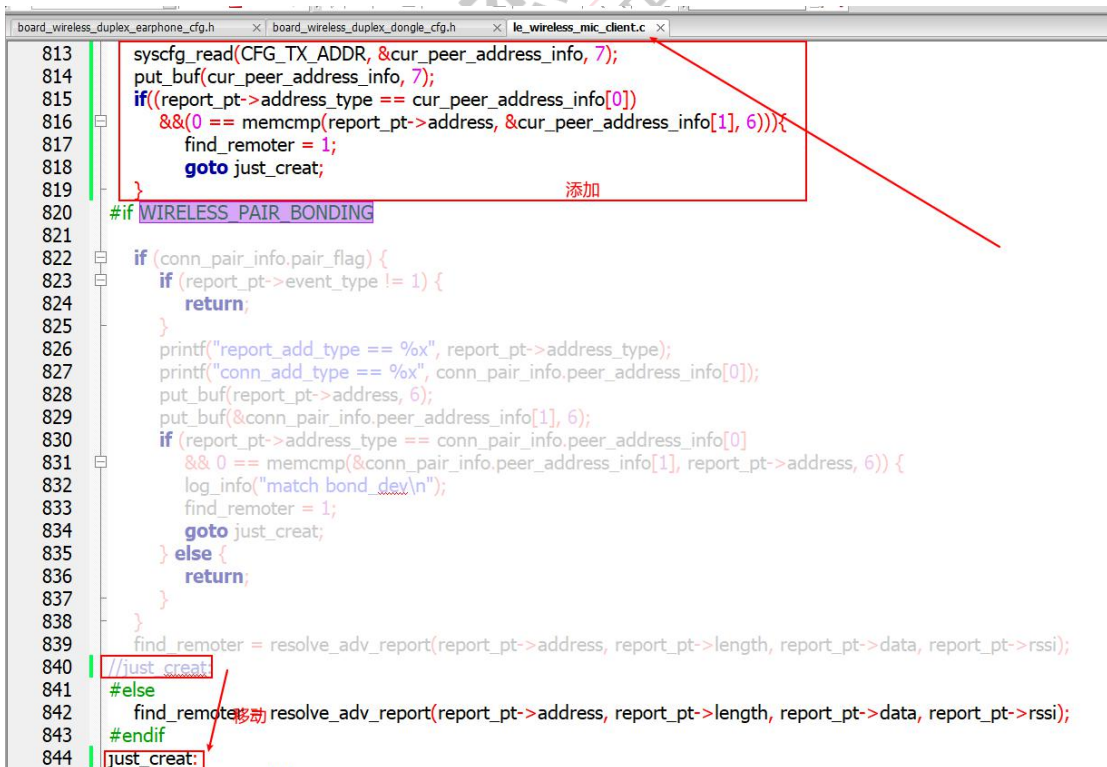


```

1027 con_handle = hci_subevent_le_connection_complete_get_connection
1028 log_info("HCI_SUBEVENT_LE_CONNECTION_COMPLETE : %0x\n", c
1029 connection_update_complete_success(packet + 8);
1030 client_profile_start(con_handle);
1031 client_event_report(CLI_EVENT_CONNECTED, packet, size);
1032 memcpy(cur_peer_address_info, packet + 7, 7);
1033 syscfg_write(CFG_TX_ADDR, &cur_peer_address_info, 7);
1034 #if WIRELESS_PAIR_BONDING
1035 memcpy(cur_peer_address_info, &packet[7], 7);
1036 conn_pair_info.pair_flag = 1;
1037 printf("pair_flag == %d", conn_pair_info.pair_flag);
1038 put_buf(cur_peer_address_info, 7);
1039 memcpy(&conn_pair_info.peer_address_info, cur_peer_address_info
1040 conn_pair_info.head_tag = BLE_VM_HEAD_TAG;
1041 conn_pair_info.tail_tag = BLE_VM_TAIL_TAG;
1042 conn_pair_vm_do(&conn_pair_info, 1);
1043 #else
1044 if (pair_bond_enable) {
1045     conn_pair_info.pair_flag = 1;
1046     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



```

813 syscfg_read(CFG_TX_ADDR, &cur_peer_address_info, 7);
814 put_buf(cur_peer_address_info, 7);
815 if((report_pt->address_type == cur_peer_address_info[0])
816     &&(0 == memcmp(report_pt->address, &cur_peer_address_info[1], 6))) {
817     find_remoter = 1;
818     goto just_creat;
819 }
820 #if WIRELESS_PAIR_BONDING
821
822 if (conn_pair_info.pair_flag) {
823     if (report_pt->event_type != 1) {
824         return;
825     }
826     printf("report_add_type == %x", report_pt->address_type);
827     printf("conn_add_type == %x", conn_pair_info.peer_address_info[0]);
828     put_buf(report_pt->address, 6);
829     put_buf(&conn_pair_info.peer_address_info[1], 6);
830     if (report_pt->address_type == conn_pair_info.peer_address_info[0]
831         && 0 == memcmp(&conn_pair_info.peer_address_info[1], report_pt->address, 6)) {
832         log_info("match bond_dev\n");
833         find_remoter = 1;
834         goto just_creat;
835     } else {
836         return;
837     }
838 }
839 find_remoter = resolve_adv_report(report_pt->address, report_pt->length, report_pt->data, report_pt->rssi);
840 //just_creat
841 #else
842 find_remoter = resolve_adv_report(report_pt->address, report_pt->length, report_pt->data, report_pt->rssi);
843 #endif
844 just_creat:

```