RTL8762C RCU MP Test Mode Design Spec

v1.0 by Realtek

2018/06/23

修订历史

日期	版本	修改
2018/06/23	V1.0	Initial Version

目录

图表

1 概述

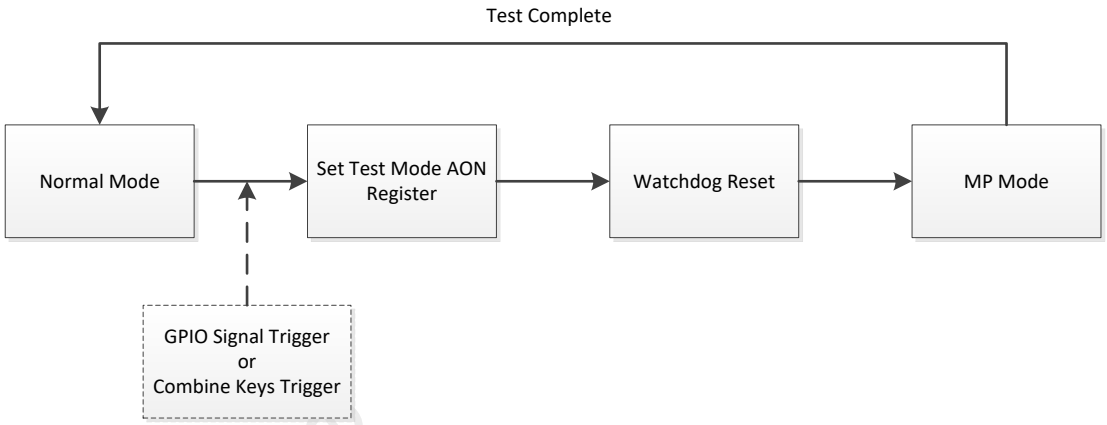
本文主要介绍 RTL8762C 语音遥控器方案的量产测试软件相关测试模式和行为规范，包括 HCI UART 测试模式，Data UART 测试模式，Single Tone 测试模式，及 Fast Pair 测试模式。本文主要用以指导遥控器的开发量产测试软件测试中遇到的问题。下文简称 RTL8762C 语音遥控器为 Bee2 RCU。

测试模式：

- HCI UART 测试模式
- Single Tone 测试模式
- Data UART 测试模式
- Fast Pair 测试模式

2 测试模式切换方式

当设定的 GPIO 信号或组合按键触发后，Bee2 RCU 通过 Test Mode AON 寄存器和看门狗重启的方式，从正常模式切换成量产测试模式，切换量产测试模式的流程如下：



图表 1 切换测试模式流程

Bee2 RCU SDK 支持两种方式触发测试模式：GPIO 信号和组合按键方式。

考虑到 RCU 在用户模式下，需要防止误触发操作，针对组合按键方式，使用了 FLASH 中的标志位的值来判断是否允许组合按键触发测试模式：

1. 当 Flash 中标志位的值为 MP_TEST_MODE_FLG_ENABLE_VALUE 时，Bee2 RCU 允许通过指定组合按键，进入量产测试模式；
2. 当 Flash 中标志位的值为 MP_TEST_MODE_FLG_DISABLE_VALUE 时，Bee2 RCU 不允许通过指定组合按键，进入量产测试模式；

在量产测试最后阶段，将 FLASH 中标志位设为 MP_TEST_MODE_FLG_DISABLE_VALUE，即不允许通过组合按键进入量产模式。MP Test Mode 相关代码，请参见 SDK 中 mp_test.c 文件，部分参考代码如下：

```
#define MP_TEST_MODE_FLG_ENABLE_VALUE      0x74657374
#define MP_TEST_MODE_FLG_DISABLE_VALUE     0x50245150

/*****************************************************************
* @brief    MP Test enable test mode flag
*/
bool mp_test_enable_test_mode_flag(void)
{
    uint32_t result = false;
    uint32_t test_mode_value = MP_TEST_MODE_FLG_ENABLE_VALUE;

    result = ftl_save(&test_mode_value, MP_TEST_FTL_PARAMS_TEST_MODE_FLG_OFFSET,
                      MP_TEST_FTL_PARAMS_TEST_MODE_FLG_LEN);

    return (result == 0);
```

```
}

/*****************************************************************
* @brief    MP Test disable test mode flag
*/
bool mp_test_disable_test_mode_flag(void)
{
    uint32_t result = false;
    uint32_t test_mode_value = MP_TEST_MODE_FLG_DISABLE_VALUE;

    result = ftl_save(&test_mode_value, MP_TEST_FTL_PARAMS_TEST_MODE_FLG_OFFSET,
                      MP_TEST_FTL_PARAMS_TEST_MODE_FLG_LEN);

    return (result == 0);
}

/*****************************************************************
* @brief    MP Test check if test mode is enabled or not
*/
bool mp_test_is_test_mode_flag_en(void)
{
    uint32_t ftl_res = 0;
    uint32_t test_mode_value = 0;

    ftl_res = ftl_load(&test_mode_value, MP_TEST_FTL_PARAMS_TEST_MODE_FLG_OFFSET,
                       MP_TEST_FTL_PARAMS_TEST_MODE_FLG_LEN);

    if (ftl_res == FTL_READ_ERROR_READ_NOT_FOUND)
    {
        APP_PRINT_WARN0("[mp_test_is_test_mode_flag_en] test mode ftl flag is invalid, reset to enabled!");
        mp_test_enable_test_mode_flag();
        ftl_load(&test_mode_value, MP_TEST_FTL_PARAMS_TEST_MODE_FLG_OFFSET,
                 MP_TEST_FTL_PARAMS_TEST_MODE_FLG_LEN);
    }

    APP_PRINT_INFO2("[mp_test_is_test_mode_flag_en] ftl_res is %d, value is 0x%08X", ftl_res,
                    test_mode_value);

    return (test_mode_value == MP_TEST_MODE_FLG_ENABLE_VALUE);
}
```

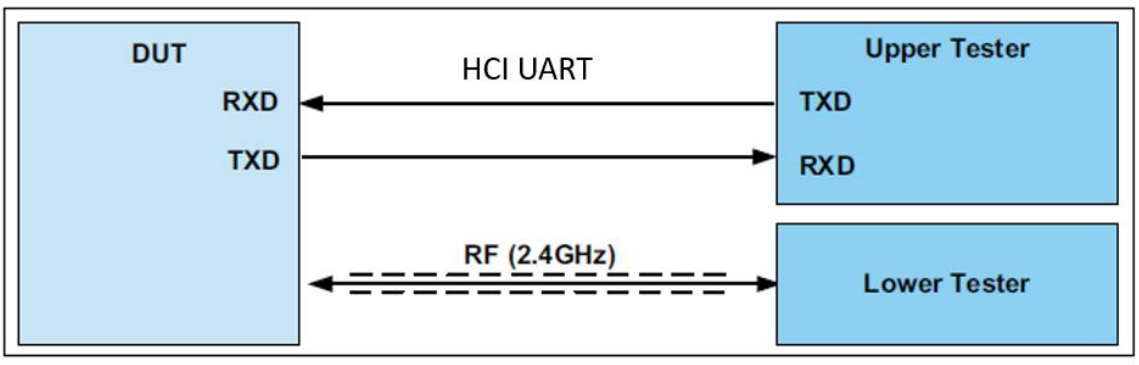
Bee2 RCU SDK 中，在 board.h 中有相关的宏定义来支持量产测试模式。

```
#define FEATURE_SUPPORT_MP_TEST_MODE 1  /* set 1 to enable MP test */

#define MP_TEST_MODE_SUPPORT_HCI_UART_TEST     1  /* set 1 to support HCI Uart Test Mode */
#define MP_TEST_MODE_SUPPORT_DATA_UART_TEST    1  /* set 1 to support Data Uart Test Mode */
#define MP_TEST_MODE_SUPPORT_SINGLE_TONE_TEST  1  /* set 1 to support SingleTone Test Mode */
#define MP_TEST_MODE_SUPPORT_FAST_PAIR_TEST    1  /* set 1 to support Fast Pair Test */

#define MP_TEST_MODE_TRIG_BY_GPIO          0x0001  /* GPIO signal while power on to trigger MP test mode */
#define MP_TEST_MODE_TRIG_BY_COMBINE_KEYS  0x0002  /* Combine keys to trigger MP test mode */

#define MP_TEST_MODE_TRIG_SEL              (MP_TEST_MODE_TRIG_BY_GPIO | MP_TEST_MODE_TRIG_BY_COMBINE_KEYS)
```


3 HCI UART 测试模式

3.1 简介

HCI UART 测试模式允许在正常 APP 模式下的 BLE RCU 设备通过外部触发（GPIO 信号或组合按键方式），来临时地把 HCI 层通过 UART 暴露出来。这么做的目的是能让 RCU 在产线上做测试的时候，在已经烧录了最终产品固件的前提下，还能直接通过 UART，和蓝牙测试仪器进行连接，运行“直接测试模式”（Direct Test Mode, DTM）的命令进行产线测试，同时保证这个 UART 在普通模式下能被用作于其他用途。



图表 2 HCI UART Test Mode 系统框图

上图是 HCI 测试模式下的测试系统框图。和标准的 BLE 测试模式 DTM 一样，RCU 在 HCI UART 测试模式支持一系列标准通用的 HCI 命令，配合蓝牙测试仪器（如 Anritsu MT8852B），可以验证 BLE RCU 的射频性能，包括输出功率、调制特性、载波频率漂移、灵敏度等。具体的测试命令描述可以参见 Bluetooth Core Specification 相关章节。

3.2 程序实现

HCI UART 测试模式的主要代码和逻辑是在 Patch 部分实现，APP 代码中提供了切换到 HCI UART 测试模式的 API SwitchToHciMode。部分参考代码：

```
1.  static inline void switch_to_hci_mode(void)
2.  {
3.      set_hci_mode_flag(true);
4.      WDG_SystemReset(RESET_ALL_EXCEPT_AON, SWITCH_HCI_MODE);
5.  }
```

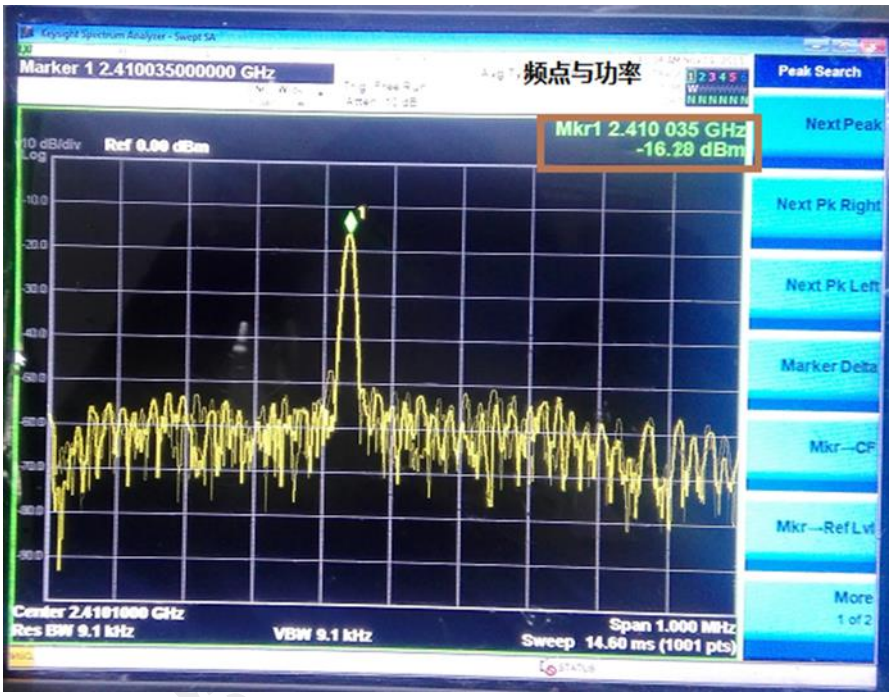
4 Single Tone 测试模式

4.1 简介

使用 HCI UART 测试模式及专业的蓝牙测试仪器，虽然可以对蓝牙性能进行比较全面详细的分析，但实际在产线上测试也会有一些限制：

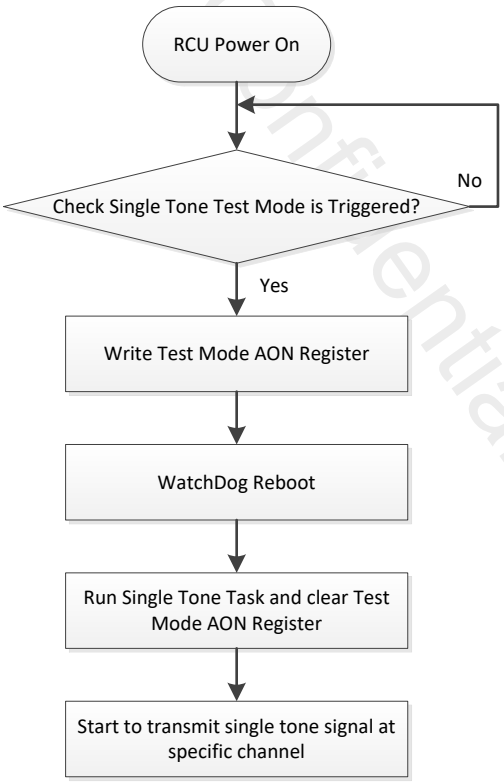
- 专业蓝牙测试仪器一般设备较昂贵，设备投入成本较大；
- 需要在无线屏蔽室环境下进行；
- 测试项目详细，但也比较耗时间；

因此，如果产线上只是想对 RF 性能测试进行简化的话，可以使用遥控器的 Single Tone 测试模式。遥控器通过外部触发（GPIO 信号或组合按键方式），进入到 Single Tone 测试模式下，会在某一设定的 Channel 上打单载波信号。使用频谱仪，通过观察和测量单载波的频谱波形，可以对 RCU RF 的发射功率及频偏值进行判断。频谱仪看到的 Single Tone 波形类似下图所示。



图表 3 Single Tone 波形图

4.2 程序实现



图表 4 Single Tone Test Mode 流程图

1. 当遥控器检测到外部触发条件时（如特定组合键被按下），写标志位到 AON Register；

2. 然后软件复位；
3. 重启进入 APP Main()时再通过判断 AON Register，进入 single tone 测试模式；
4. 清除 AON Register，这样下一次重启就是正常的遥控器模式；
5. 开始在固定频点上打单载波信号；

部分参考代码：

```
switch_to_test_mode(SINGLE_TONE_MODE);

static inline void switch_to_test_mode(T_TEST_MODE test_mode)
{
    T_BTAON_FAST_TEST_MODE_TYPE aon;
    aon.d8 = btaon_fast_read_safe(BTAON_FAST_TEST_MODE);
    aon.s.test_mode = test_mode;
    btaon_fast_write_safe(BTAON_FAST_TEST_MODE, aon.d8);

    WDG_SystemReset(RESET_ALL_EXCEPT_AON, SWITCH_TEST_MODE);
}
```

```
void single_tone_init(void)
{
    APP_PRINT_INFO0("Single Tone Init");

#ifdef EXIT_SINGLE_TONE_TEST_WHEN_TIMEOUT
    if (true == os_timer_create(&single_tone_exit_timer, "single_tone_exit_timer",  1,
                                EXIT_SINGLE_TONE_TIME, false, single_tone_exit_timeout_cb))
    {
        os_timer_start(&single_tone_exit_timer);
    }
#endif

    os_task_create(&single_tone_task_handle, "single_tone", single_tone_task, 0, 256, 1);
}
```

```
/**
  * @brief  start singletone
  * @param  channel_num: channel of singletone
  * @retval none
  */
void single_tone_start(uint8_t channel_num)
{
    APP_PRINT_INFO0("Single Tone Start!");

```

```
10.   T_SINGLE_TONE_VEND_CMD_PARAMS *p_vend_cmd_params = os_mem_alloc(RAM_TYPE_DATA
    _ON,
11.                                                   sizeof(T_SINGLE_TONE_VEND_CMD_PARAMS));
12.
13.   if (p_vend_cmd_params)
14.   {
15.       p_vend_cmd_params->pkt_type = 1;
16.       p_vend_cmd_params->opcode = 0xfc78;
17.       p_vend_cmd_params->length = 4;
18.       p_vend_cmd_params->start = 1;
19.       p_vend_cmd_params->channle = channel_num;
20.       p_vend_cmd_params->tx_power = 8;
21.       p_vend_cmd_params->is_le = 1;
22.
23.       hci_if_write((uint8_t *)p_vend_cmd_params, sizeof(T_SINGLE_TONE_VEND_CMD_PARAM
    S));
24.
25.       single_tone_is_sent_start_cmd = true;
26.   }
27. }
```

通过修改 p_vend_cmd_params 的参数可以对 Single Tone 进行控制，其中 p_vend_cmd_params->channle 可以指定的 Single Tone 频点， p_vend_cmd_params->tx_power 指定 Tx Power。

5 Data UART 测试模式

5.1 简介

Data UART 测试模式允许在正常 APP 模式下的 BLE RCU 设备通过外部触发（比如说通过在 reboot 过程中拉低某个 GPIO 管脚），来临时地把 Data UART 切出来。这么做的目的是能让 RCU 在产线上做测试的时候，在已经烧录了最终产品固件的前提下，还能直接通过 Data UART 命令控制遥控器执行各项测试，同时保证这个 UART 在普通模式下能被用作于其他用途，且不影响用户的正常使用。

5.2 Data UART 命令格式

目前，Data UART 测试模式下，支持读取软件版本号，MAC Address，语音 MIC 测试等，详细信息可以参考 DATA_UART_MP_Command.xlsx。用户可以根据实际产线要求进行命令的修改及增减。DATA UART 命令及 Status 返回格式如下：

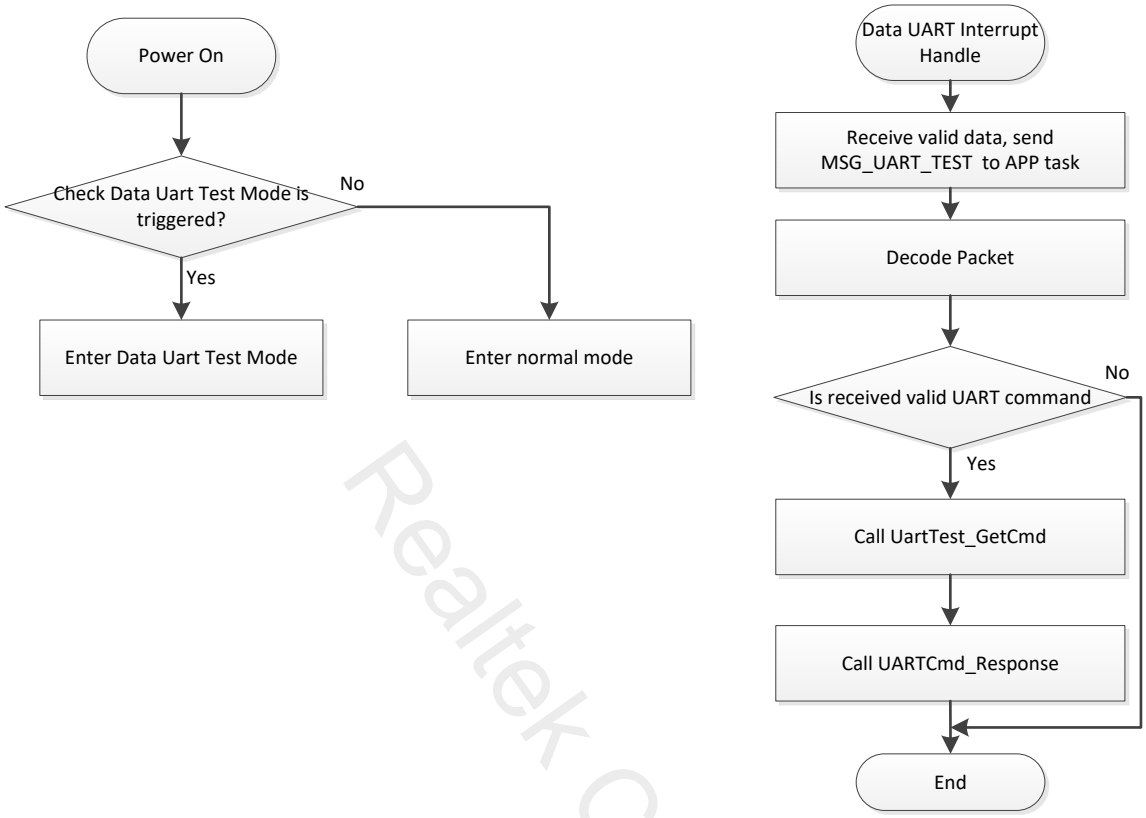
Request Format	Size in Byte	Comment
Start Byte	1	Must be 0x87
Request Opcode	2	
Payload Data	N	
CRC16	2	

图表 5 Data UART Request Command Format

Response Format	Size in Byte	Comment
Start Byte	1	Must be 0x87
Response Opcode	2	
Response Status	1	0 – Success, 1 - Fail
Payload Size	4	
Payload Data	N	
CRC16	2	

图表 6 Data UART Response Command Format

5.3 程序实现



图表 7 Data UART Test Mode 流程图

Bee2 RCU SDK 相关代码在文件 data_uart_test.c 中，部分参考代码：

```
/**
 * @brief initialize uart test function.
 * @param None.
 * @retval None.
 */
void uart_test_init(void)
{
    UART_DBG_BUFFER(MODULE_APP, LEVEL_INFO, "[uart_test_init] initialize uart test mode", 0);
    UartTransport_Init();
    uart_test_is_dlps_allowed = false;
}

/**
 * @brief handle UART message
 * @param io_driver_msg_recv - recieved io message
 * @return none
 * @retval void
 */
```

```
void uart_test_handle_uart_msg(T_IO_MSG io_driver_msg_recv)
{
    UART_PacketTypeDef *pUartTestPacket = (UART_PacketTypeDef *)(io_driver_msg_recv.u.buf);

    if (Packet_Decode(pUartTestPacket))
    {
        uart_test_get_cmd_func(pUartTestPacket);
    }
}

/**<  Array of all used test function informations */
const T_UART_TEST_PROTOCOL uart_test_func_map[UART_TEST_SUPPORT_NUM] =
{
    /* Opcode, Parameter Length, Function */
    {READ_PATCH_VERSION_CMD, 0, uart_test_read_patch_version},
    {READ_APP_VERSION_CMD, 0, uart_test_read_app_version},
    {READ_MAC_ADDR_CMD, 0, uart_test_read_mac_addr},
    {ENTER_FAST_PAIR_MODE_CMD, 1, uart_test_enter_fast_pair_mode},
    {GET_DEVICE_STATE_CMD, 0, uart_test_get_dev_state},
    {VOICE_TEST_START_CMD, 0, uart_test_voice_test_start},
    {VOICE_TEST_STOP_CMD, 0, uart_test_voice_test_stop},
    {SET_VOICE_CONFIG_CMD, 0, NULL},
    {GET_VOICE_CONFIG_CMD, 0, NULL},
    {ENTER_DLPS_TEST_MODE_CMD, 0, uart_test_enter_dlps},
    {START_STOP_ADV_CMD, 1, uart_test_start_stop_adv},
    {START_IR_TEST_MODE_CMD, 0, NULL},
    {ENTER_HCI_TEST_MODE_CMD, 0, uart_test_enter_hci_mode},
    {DISABLE_TEST_MODE_FLG_CMD, 0, uart_test_disable_test_mode_flag},
    {ENABLE_TEST_MODE_FLG_CMD, 0, uart_test_enable_test_mode_flag},
    {ERASE_PAIR_INFO_CMD, 0, uart_test_erase_pair_info},
    {CHANGE_BAUDRATE_CMD, 1, uart_test_change_baudrate},
    {TERMINATE_CONNECT_CMD, 0, uart_test_terminate_connect},
    {GET_GLODEN_INFO_CMD, 0, NULL},
    {GET_DUT_INFO_CMD, 0, NULL},
    {VERIFY_DUT_INFO_CMD, 0, NULL},
    {AUTO_K_RF_FREQ_CMD, 0, NULL},
    {FIND_DEVICE_TYPE_CMD, 0, NULL},
    {REBOOT_DEVICE_CMD, 0, uart_test_reboot_device},
    {UPDATE_MAC_ADDR_CMD, 0, NULL},
    {ENTER_SINGLE_TONE_MODE_CMD, 0, uart_test_enter_single_tone},
    {READ_HARDWARE_VERSION_CMD, 0, NULL},
    //Add more command here,Please store in order according to opcode!
};
```

6 Fast Pair 测试模式

6.1 简介

RTL8762C 遥控器使用 Windows RCU Tool 和 Windows RCU test dongle 进行量产时功能测试。

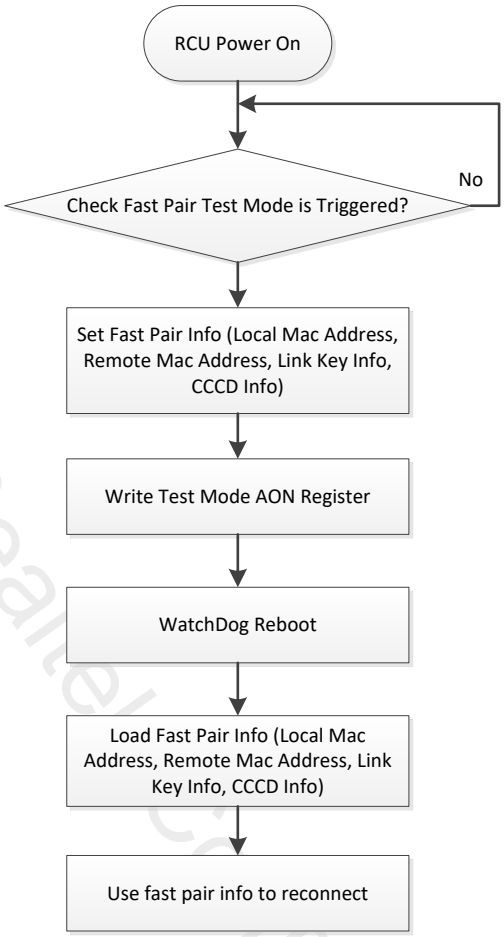
目前功能测试包括蓝牙连接、按键测试和语音测试。为提高产测效率，产线上可以同时安排多个工位进行功能测试，同时遥控器端也支持多种组合按键分别进入产测模式和对应的测试 dongle 建立连接。在上述多种组合键被按下时，遥控器会重启，并使用事先约定的 Mac Address 和 Link Key 发送 direct 回连广播，尝试和对应的测试 dongle 建立连接。

6.2 配置蓝牙地址以及配对信息

为了实现按组合键就能够和 RCU Tool 完成建立连线并加密链路的要求，配对信息必须预先配置到 RCU Tool 和遥控器中，方法如下（以 5 个工位为例）：

1. 通过事先约定的方式，将一致的配对信息固化到 RCU 及 Tool 中，包括遥控器的虚拟 MAC 地址、Test Dongle 的虚拟 MAC 地址、Link key 及 CCCD 信息；RCU Tool 的配对信息会记录在 windows 注册表中；遥控器端的配对信息会记录在 flash 中的指定位置；
2. 通过手动方式将 5 个工位的 RCU Tool（命名为 RCU Tool 1，RCU Tool 2，RCU Tool 3，RCU Tool 4，RCU Tool 5）分别和 5 个遥控器（RCU1，RCU2，RCU3，RCU4，RCU5）配对；
3. RCU 程序启动之后，按不同的组合键，RCU 会重新启动，并根据不同的组合键初始化加载不同的配置信息，这样 RCU 启动之后，RCU Tool 就可以根据 windows 注册表中存放的配对设备信息来连接指定的遥控器，连接成功之后直接使用 link key 加密链路就可以了，不需要配对流程，节省了时间。

6.3 程序实现



图表 8 快速配对测试模式流程图

具体实现代码可以参考 mp_test.c，部分参考代码如下：

```
/* Fast Pairing Local Mac Address Config List */
static const uint8_t mp_fp_local_addr_config_list[FP_MAX_LINE_NUM][FP_MAC_ADDR_LEN] =
{
    {0x01, 0x88, 0x23, 0x4c, 0xe0, 0x00},
    {0x02, 0x88, 0x23, 0x4c, 0xe0, 0x00},
    {0x03, 0x88, 0x23, 0x4c, 0xe0, 0x00},
    {0x04, 0x88, 0x23, 0x4c, 0xe0, 0x00},
    {0x05, 0x88, 0x23, 0x4c, 0xe0, 0x00},
};

/* Fast Pairing Remote Mac Address Config List */
static const uint8_t mp_fp_remote_addr_config_list[FP_MAX_LINE_NUM][FP_MAC_ADDR_LEN] =
{
    {0x87, 0x99, 0x23, 0x4c, 0xe0, 0x00},
    {0x87, 0x99, 0x23, 0x4c, 0xe0, 0x00},
```

```
    {0x87, 0x99, 0x23, 0x4c, 0xe0, 0x00},
    {0x87, 0x99, 0x23, 0x4c, 0xe0, 0x00},
    {0x87, 0x99, 0x23, 0x4c, 0xe0, 0x00},
};

/* Fast Pairing Link Key Config List */
static const uint8_t mp_fp_link_key_config_list[FP_MAX_LINE_NUM][FP_LINK_KEY_LEN] =
{
    {0xcb, 0x84, 0xaa, 0x4d, 0x66, 0x42, 0xd5, 0xa2, 0x33, 0xa1, 0x6a, 0x51, 0x9a, 0x50, 0xb5, 0xac},
    {0xcb, 0x84, 0xaa, 0x4d, 0x66, 0x42, 0xd5, 0xa2, 0x33, 0xa1, 0x6a, 0x51, 0x9a, 0x50, 0xb5, 0xac},
    {0xcb, 0x84, 0xaa, 0x4d, 0x66, 0x42, 0xd5, 0xa2, 0x33, 0xa1, 0x6a, 0x51, 0x9a, 0x50, 0xb5, 0xac},
    {0xcb, 0x84, 0xaa, 0x4d, 0x66, 0x42, 0xd5, 0xa2, 0x33, 0xa1, 0x6a, 0x51, 0x9a, 0x50, 0xb5, 0xac},
    {0xcb, 0x84, 0xaa, 0x4d, 0x66, 0x42, 0xd5, 0xa2, 0x33, 0xa1, 0x6a, 0x51, 0x9a, 0x50, 0xb5, 0xac},
};

/* Fast Pairing CCCD Information */
static const uint8_t mp_fp_cccd_info[FP_CCCD_DATA_LEN] =
{
    0x40, 0x00, 0x01, 0x00, 0x30, 0x00, 0x01, 0x00,
    0x34, 0x00, 0x01, 0x00, 0x3b, 0x00, 0x01, 0x00
};

/******************************************************************
* @brief    MP Test Set Fast Pair Info
*/
bool mp_test_set_fast_pair_info(uint8_t index)
{
    bool result = false;
    uint8_t ltk_length;
    uint8_t mp_local_bd[8] = {0};
    uint8_t remote_mac_addr[6] = {0};
    T_LE_KEY_TYPE link_key_type = LE_KEY_UNAUTHEN;
    uint8_t local_ltk[FP_LINK_KEY_LEN];
    T_GAP_REMOTE_ADDR_TYPE remote_addr_type;
    uint8_t ccc_bits_count;
    T_LE_CCCD *p_cccd_data;

    if (index >= FP_MAX_LINE_NUM)
    {
```

```
55.         return false;
56.     }
57.
58.     /* set local bd addr info */
59.     memcpy(mp_local_bd, mp_fp_local_addr_config_list[index], FP_MAC_ADDR_LEN);
60.     result = ftl_save(mp_local_bd, MP_TEST_FTL_PARAMS_LOCAL_BD_ADDR_OFFSET,
61.                       MP_TEST_FTL_PARAMS_LOCAL_BD_ADDR_LEN);
62.     if (result != 0)
63.     {
64.         return false;  /* ftl save failed */
65.     }
66.
67.     /* set remote mac addr info */
68.     remote_addr_type = GAP_REMOTE_ADDR_LE_PUBLIC;
69.     memcpy(remote_mac_addr, mp_fp_remote_addr_config_list[index], FP_MAX_LINE_NUM);
70.
71.     /* set link key info */
72.     ltk_length = FP_LINK_KEY_LEN;
73.     memcpy(local_ltk, mp_fp_link_key_config_list[index], FP_LINK_KEY_LEN);
74.
75.     /* set cccd info */
76.     ccc_bits_count = FP_CCCD_BITS_CNT;
77.     p_cccd_data = os_mem_alloc(RAM_TYPE_DATA_ON, 4 + ccc_bits_count * 4);
78.     p_cccd_data->data_length = FP_CCCD_DATA_LEN;
79.     memcpy(p_cccd_data->data, mp_fp_cccd_info, FP_CCCD_DATA_LEN);
80.
81.     /* generate bond dev info */
82.     result = le_gen_bond_dev(remote_mac_addr, remote_addr_type, GAP_LOCAL_ADDR_LE_PUBLI
    C,
83.                              ltk_length, local_ltk, link_key_type, p_cccd_data);
84.
85.     os_mem_free(p_cccd_data);
86.
87.     return result;
88. }
89.
90. /*****************************************************************
91. * @brief    MP Test load fast pair mac addr
92. */
93. bool mp_test_load_fp_mac_addr(void)
94. {
95.     bool result = false;
96.     uint8_t mp_mac[8] = {0};
97.
```

```
98.    if (0 == ftl_load(mp_mac, MP_TEST_FTL_PARAMS_LOCAL_BD_ADDR_OFFSET, sizeof(mp_ma
    c)))
99.    {
100.        memcpy((uint8_t *)0x00200197, mp_mac, 6);
101.        APP_PRINT_INFO1("[mp_test_load_fp_mac_addr] MP MAC Addr: %b", TRACE_BINARY(6, m
    p_mac));
102.        result = true;
103.    }
104.    else
105.    {
106.        APP_PRINT_ERROR0("[mp_test_load_fp_mac_addr] load mp mac addr failed!");
107.        result = false;
108.    }
109.
110.    return result;
111. }
```


7 参考文献

[1] DATA_UART_MP_Command.xlsx

[2] Realtek RCU Test Tool User Guide