RTL8762C RCU MP Test Mode Design Spec

v1.1 by Realtek

2018/09/11

Revision History

Data	Version	Description
2018/06/23	V1.0	Initial Version
2018/09/11	V1.1	Minor changes

Contents

Figure

1 Overview

This paper mainly introduces the related test modes and behavior specifications of the mass production testing software of the RTL8762C voice remote control program, including the HCI UART test mode, Data UART test mode, Single Tone test mode and Fast Pair test mode. This article is mainly used to guide the problems encountered in the test with the RCU mass production test software.

In the following part, the RTL8762C voice remote control is referred to as Bee2 RCU.

Test mode:

- HCI UART test mode
- Single Tone test mode
- Data UART test mode
- Fast Pair test mode

2 Test Mode Switching Method

When the specific GPIO signals or combined keys are triggered, the Bee2 RCU switches from the normal mode to the mass production test mode through the Test Mode AON register and the watchdog reboot. The flow of switching the mass production test mode is as follows.

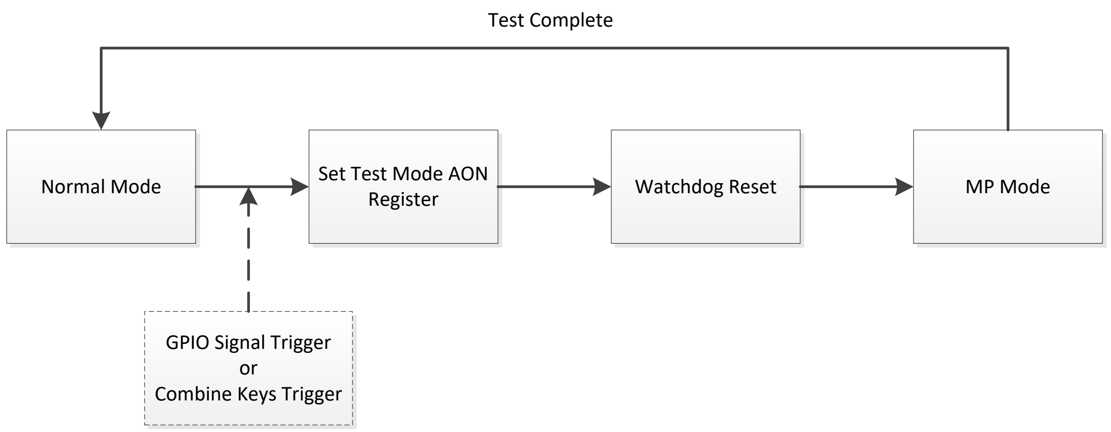

Figure 1 Switch the Mass Production Test Mode Flow

The Bee2 RCU SDK supports two ways to trigger test mode, including GPIO signal and combined key mode. RCU needs to prevent false triggering in user mode. According to the combined key mode, the value of flag bit in FLASH is used to determine whether the combined key is permitted to trigger the test mode.

1) When the value of flag bit in Flash is MP_TEST_MODE_FLG_ENABLE_VALUE, Bee2 RCU is permitted to enter the mass production test mode by specifying combined keys.
2) When the value of flag bit in Flash is MP_TEST_MODE_FLG_DISABLE_VALUE, Bee2 RCU is prohibited to enter the mass production test mode by specifying combined keys.

In the final stage of mass production testing, set the flag bit in FLASH to MP_TEST_MODE_FLG_DISABLE_VALUE, which means that it is prohibited to enter the mass production mode through the combined keys. The code for MP Test Mode is shown in the mp_test.c in the SDK. Part of the reference code is as below.

```
#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 en
abled!");
        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);
}
```

In the Bee2 RCU SDK, there are related macro definitions in board.h to support the mass production test mode.

```
#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 Test Mode

3.1 Introduction

The HCI UART test mode allows the BLE RCU device in normal APP mode to expose the HCI layer through UART temporarily when it is triggered by an external trigger (GPIO signals or combined keys mode). The purpose of exposing HCI layer is to allow RCUs that have been programmed with the final product firmware to be connected directly to the Bluetooth tester via the UART and run the Direct Test Mode (DTM) command to perform production line test. At the same time, ensure that the UART can be used for other purposes in normal mode.

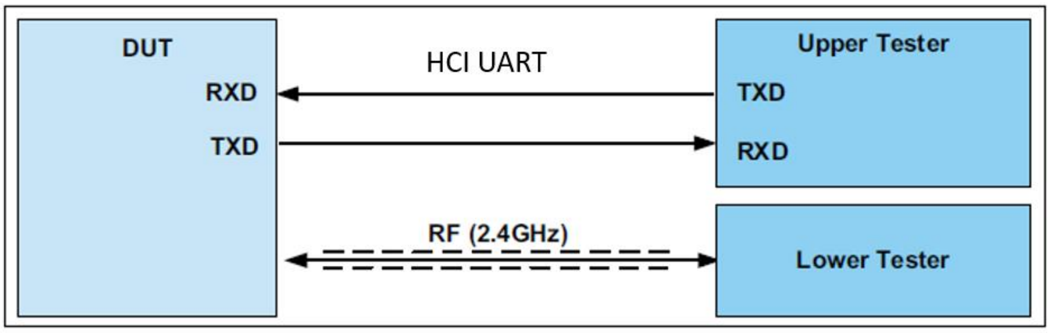

Figure 2 System Block Diagram of HCI UART Test Mode

The figure above is the system block diagram of HCI UART test mode. Like standard BLE test mode DTM, RCU supports a series of standard universal HCI commands in HCI UART test mode. With Bluetooth test equipment (such as Anritsu MT8852B), it can verify the RF performance of BLE RCU, including output power, modulation characteristics, carrier frequency drift, sensitivity, etc. Specific test command description can refer to the relevant chapter of the Bluetooth Core Specification.

3.2 Software

The main code and logic of the HCI UART test mode are implemented in the Patch part. APP code provides SwitchToHciMode API to switch to the HCI UART test mode. Part of the reference code is as below.

```
static inline void switch_to_hci_mode(void)
{
    set_hci_mode_flag(true);
    WDG_SystemReset(RESET_ALL_EXCEPT_AON, SWITCH_HCI_MODE);
}
```

4 Single Tone Test Mode

4.1 Introduction

Although the HCI UART test mode and professional Bluetooth test instruments can perform a more comprehensive and detailed analysis of the Bluetooth performance, the actual test on the production line also has the following limitations.

- Professional Bluetooth test equipment is generally expensive and the equipment investment cost is high.
- Need to be performed in the environment of wireless shielded room.
- The test item is detailed, but also takes time.

If the production line wants to simplify the RF performance test, Single Tone test mode of RCU can be applied. The RCU enters the Single Tone test mode by external trigger (GPIO signal mode or combined key mode), and it will transmit a single carrier signal through a certain set of channels. By using a spectrum analyzer to observe and measure the spectrum waveform of a single carrier, the RCU RF transmit power and frequency offset values can be determined. The Single Tone waveform seen on the spectrum analyzer is similar to the one shown below.

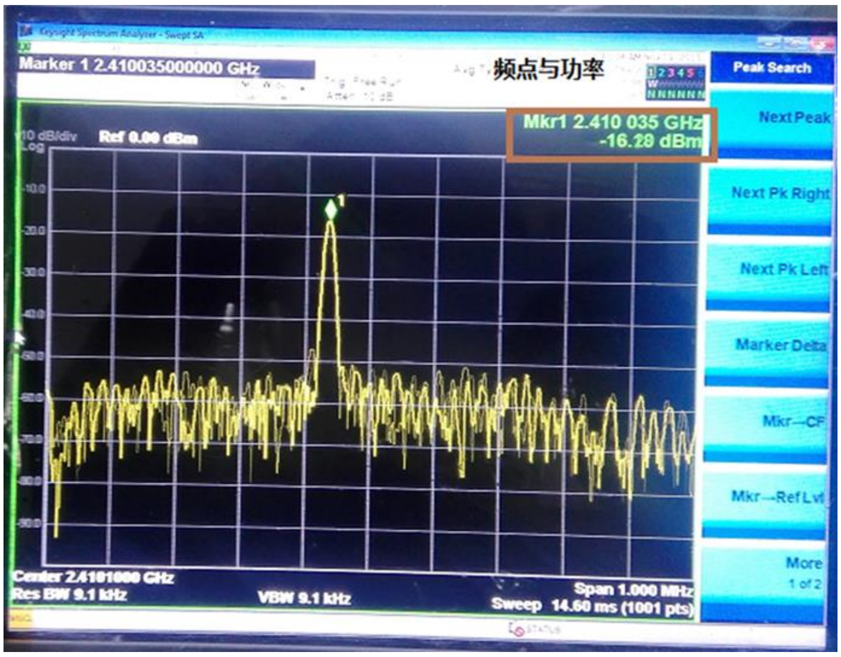

Figure 3 Single Tone Waveform

4.2 Software

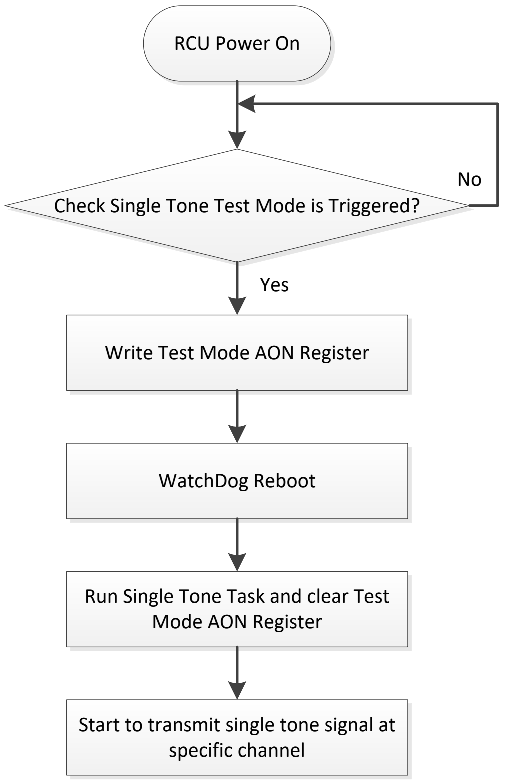
RCU Power On
Check Single Tone Test Mode is Triggered?
No
Yes
Write Test Mode AON Register
WatchDog Reboot
Run Single Tone Task and clear Test Mode AON Register
Start to transmit single tone signal at specific channel

Figure 4 Single Tone Test Mode Flow Chart

1) When the RCU detects an external trigger condition (if a particular key combination is pressed), write the flag to AON Register.
2) Software reset.
3) Reboot and enter APP Main () and enter single tone test mode after confirming AON Register contains the flag.
4) Clear AON Register so that the next reboot is normal remote control mode.
5) Start to use single carrier signal at fixed frequency.

Part of the reference code is as below.

```
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!");

    T_SINGLE_TONE_VEND_CMD_PARAMS *p_vend_cmd_params = os_mem_alloc(RAM_TYPE_DATA_ON,
                                                       sizeof(T_SINGLE_TONE_VEND_CMD_PARAMS));

    if (p_vend_cmd_params)
    {
        p_vend_cmd_params->pkt_type = 1;
        p_vend_cmd_params->opcode = 0xfc78;
        p_vend_cmd_params->length = 4;
        p_vend_cmd_params->start = 1;
        p_vend_cmd_params->channle = channel_num;
        p_vend_cmd_params->tx_power = 8;
        p_vend_cmd_params->is_le = 1;

        hci_if_write((uint8_t *)p_vend_cmd_params, sizeof(T_SINGLE_TONE_VEND_CMD_PARAMS));

        single_tone_is_sent_start_cmd = true;
    }
}
```

Single Tone can be controlled by modifying the parameters of p_vend_cmd_params, among which P_vend_cmd_params->channel can specify Single Tone frequency, and p_vend_cmd_params->tx_power can specify Tx Power.

5 Data UART Test Mode

5.1 Introduction

The Data UART test mode allows the BLE RCU device in normal APP mode to be temporarily cut out of the Data UART by an external trigger (such as by pulling down a GPIO pin in reboot process). The purpose of this operation is to enable the RCU that has already programmed the firmware of the final product to control the remote control directly by the Data UART command for various tests. At the same time, ensure that the UART can be used for other purposes in normal mode, and does not affect the normal use of the user.

5.2 Data UART Command Format

Currently, Data UART test mode supports reading software version number, MAC Address, voice MIC test and so on. See DATA_UART_MP_Command.xlsx for more information. Users can modify and add/subtract orders according to actual production line requirements. The DATA UART command and Status return format is as below.

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

Figure 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	

Figure 6 Data UART Response Command Format

5.3 Software

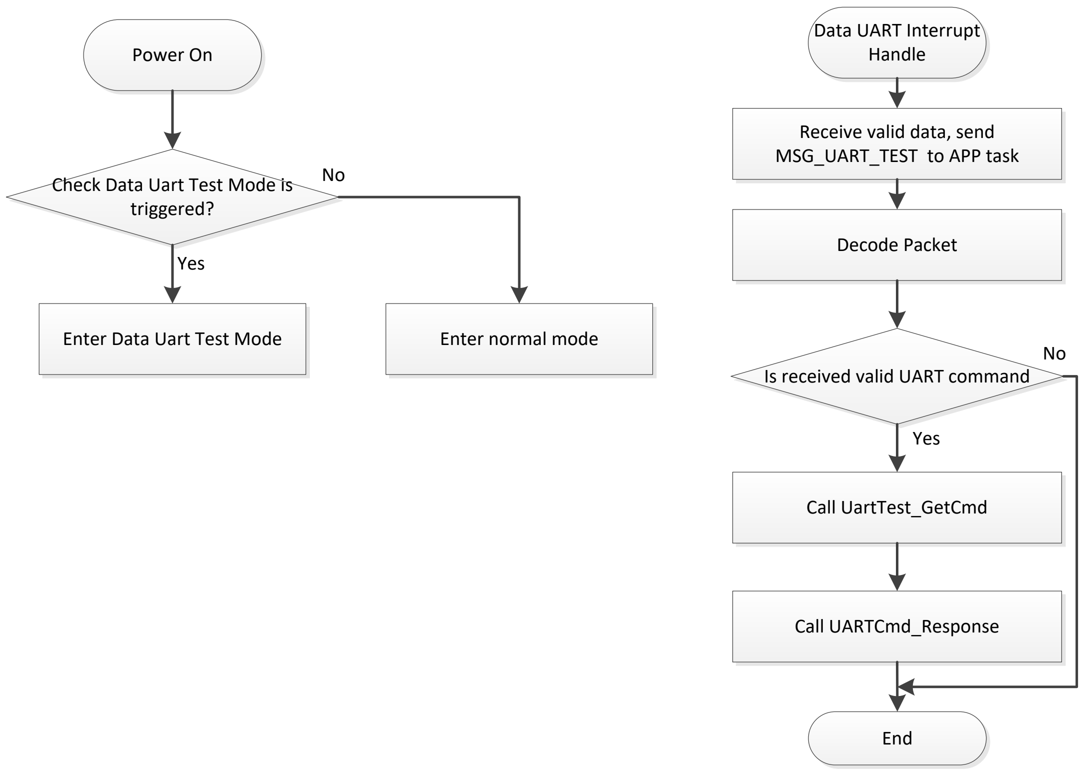
Power On
Check Data Uart Test Mode is triggered?
No
Yes
Enter Data Uart Test Mode
Enter normal mode
Data UART Interrupt Handle
Receive valid data, send MSG_UART_TEST to APP task
Decode Packet
Is received valid UART command
No
Yes
Call UartTest_GetCmd
Call UARTCmd_Response
End

Figure 7 Data UART Test Mode Flow Chart

Part of the Bee2 RCU SDK related code in the file data_uart_test.c is shown as below:

```c
/**
  * @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 Test Mode

6.1 Introduction

RTL8762A remote control uses Windows RCU Tool and Windows RCU test dongle for functional testing during mass production.

At present, functional testing includes Bluetooth connection, key test and voice test. In order to improve the production efficiency, multiple stations can be arranged for functional testing on the production line. At the same time, the RCU also supports a variety of combined keys to enter the measurement mode to establish connection with the corresponding test dongle. When the various combinations of keys are pressed, the RCU will reboot and start direct advertisement, which uses the pre-defined Mac Address and bonding information, to attempt to establish a connection with the corresponding test dongle.

6.2 Configure Bluetooth Address and Pairing Information

In order to realize the requirement to complete the connection and encrypt the link with the RCU Tool by pressing the combined key, the pairing information must be preconfigured in the RCU Tool and the remote control. The method is as follows (taking 5 stations as an example).

1) After prior defined, the consistent pairing information is solidified into RCU and Tool, including virtual MAC address of remote control, virtual MAC address of Test Dongle, Link key and CCCD information. The pairing information of RCU Tool will be recorded in the windows registry. The pairing information of the remote control will be recorded in the specified location in flash.
2) Manually pair 5 stations' RCU Tool (named RCU Tool 1, RCU Tool 2, RCU Tool 3, RCU Tool 4, and RCU Tool 5) with 5 remote controls (RCU1, RCU2, RCU3, RCU4, RCU5).
3) After RCU program starts, RCU will restart according to different combined keys. Different configuration information is initialized according to different combined keys. After RCU starts, RCU Tool can connect the designated remote control according to the pairing device information stored in the windows registry. Once the connection is successfully established, the link key is used to encrypt the link directly. This procedure reduces time consumption by eliminating the need for a pairing process.

6.3 Software

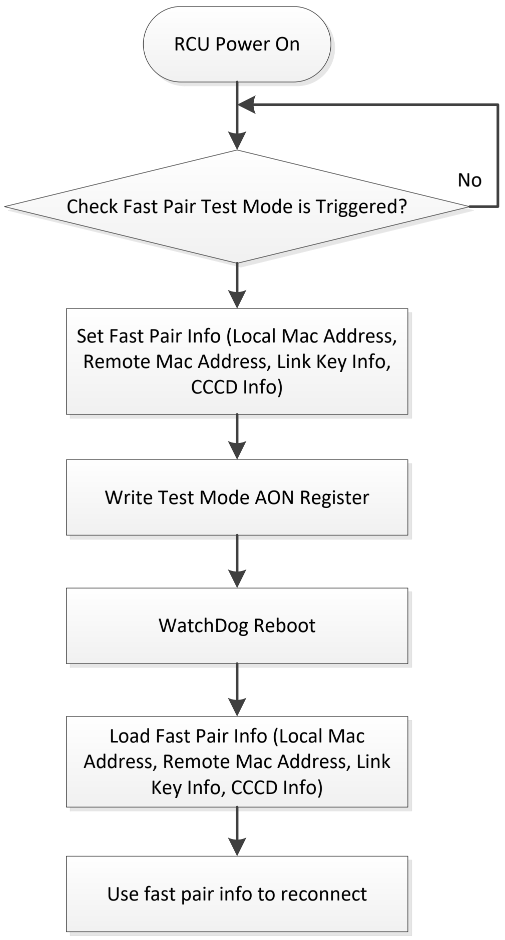
RCU Power On
Check Fast Pair Test Mode is Triggered?
No
Set Fast Pair Info (Local Mac Address, Remote Mac Address, Link Key Info, CCCD Info)
Write Test Mode AON Register
WatchDog Reboot
Load Fast Pair Info (Local Mac Address, Remote Mac Address, Link Key Info, CCCD Info)
Use fast pair info to reconnect

Figure 8 Fast Pair Test Mode Flow

Specific implementation of code can refer to mp_test.c, part of the reference code is as below.

```
/* 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)
    {
        return false;
    }

    /* set local bd addr info */
    memcpy(mp_local_bd, mp_fp_local_addr_config_list[index], FP_MAC_ADDR_LEN);
    result = ftl_save(mp_local_bd, MP_TEST_FTL_PARAMS_LOCAL_BD_ADDR_OFFSET,
                      MP_TEST_FTL_PARAMS_LOCAL_BD_ADDR_LEN);
    if (result != 0)
    {
        return false;  /* ftl save failed */
    }

    /* set remote mac addr info */
    remote_addr_type = GAP_REMOTE_ADDR_LE_PUBLIC;
    memcpy(remote_mac_addr, mp_fp_remote_addr_config_list[index], FP_MAX_LINE_NUM);

    /* set link key info */
    ltk_length = FP_LINK_KEY_LEN;
    memcpy(local_ltk, mp_fp_link_key_config_list[index], FP_LINK_KEY_LEN);

    /* set cccd info */
    ccc_bits_count = FP_CCCD_BITS_CNT;
    p_cccd_data = os_mem_alloc(RAM_TYPE_DATA_ON, 4 + ccc_bits_count * 4);
    p_cccd_data->data_length = FP_CCCD_DATA_LEN;
    memcpy(p_cccd_data->data, mp_fp_cccd_info, FP_CCCD_DATA_LEN);

    /* generate bond dev info */
    result = le_gen_bond_dev(remote_mac_addr, remote_addr_type, GAP_LOCAL_ADDR_LE_PUBLI
C,
                             ltk_length, local_ltk, link_key_type, p_cccd_data);

    os_mem_free(p_cccd_data);

    return result;
}

/*****************************************************************
* @brief    MP Test load fast pair mac addr
*/
bool mp_test_load_fp_mac_addr(void)
{
    bool result = false;
    uint8_t mp_mac[8] = {0};

    if (0 == ftl_load(mp_mac, MP_TEST_FTL_PARAMS_LOCAL_BD_ADDR_OFFSET, sizeof(mp_mac)))
    {
        memcpy((uint8_t *)0x00200197, mp_mac, 6);
        APP_PRINT_INFO1("[mp_test_load_fp_mac_addr] MP MAC Addr: %b", TRACE_BINARY(6, mp_mac));
        result = true;
    }
    else
    {
        APP_PRINT_ERROR0("[mp_test_load_fp_mac_addr] load mp mac addr failed!");
        result = false;
    }

    return result;
}
```

7 References

[1] DATA_UART_MP_Command.xlsx

[2] Realtek RCU Test Tool User Guide