RTL8762C RCU Design Spec

V1.2

2020/02/15

Revision History

Date	Version	Comments	Author	Reviewer
2018/06/15	Draft v1.0	First Bee2 RCU spec version for beta version of Bee2 RCU SDK	Chenjie	Rui
2018/09/12	V 1.1	Bee2 RCU design spec for RCU SDK V1.0.2	Chenjie	Rui
2020/01/15	V 1.2	Bee2 RCU design spec for RCU SDK V1.2.2	Mandy	Livingston

Contents

Revision History 2
1 Overview 8
2 Function & Feature 9
3 GPIO Configuration 10
3.1 System GPIO 10
3.2 Keyscan GPIO 10
3.3 Voice MIC GPIO 10
3.4 IR GPIO 11
3.5 LED GPIO 11
3.6 MP Test GPIO 11
4 System Block Diagram 12
5 Initialization Process 13
5.1 System Startup Flow 13
5.2 Normal Startup Flow 15
6 APP TASK 16
7 RCU Status 18
7.1 RCU Status 18
7.2 RCU State Transformation Diagram 19
7.3 RCU State Transformation Conditions 19
7.4 RCU IDLE STATUS DESCRIPTION 20
8 SW Timer Application Case and Purpose 22
9 Advertising Packet 23
9.1 Advertising Packet Format 23
9.2 Advertising Packet Type 24
9.2.1 Pairing Advertising Packet 24
9.2.2 Reconnection Advertising Packet 25
9.2.3 Prompt Advertising Packet 26
9.2.4 Power-on Advertising Packet 27

10 Connection and Pairing 28
11 BLE Service 29
12 Key 30
12.1 Keyscan Scheme 30
12.2 Keyscan Configuration 31
12.3 Key Behavior Rules 32
12.3.1 Single Key Behavior 32
12.3.2 Double-Key Behaviors 35
12.3.3 Three and more keys Behaviors 36
13 Voice 39
13.1 Voice Module Introduction 39
13.2 Voice Module Initialization 39
13.2.1 Voice PAD Initialization 40
13.2.2 Voice I2S Initialization 41
13.2.3 Voice CODEC Initialization 42
13.2.4 Voice GDMA Initialization 44
13.3 Voice Buffer Introduction 45
13.3.1 GDMA Ping-Pong Buffer 45
13.3.2 Voice Data Loop Buffer 45
13.4 Voice Encoding Algorithm 46
13.5 Voice Interaction Flow 46
13.5.1 IFLYTEK VOICE FLOW 46
13.5.2 HIDS Google VOICE FLOW 47
13.5.3 ATV Google VOICE FLOW 50
13.5.4 RTK GATT VOICE FLOW 51
14 IR 53
14.1 IR Module Configuration 53
14.2 IR module operation 54
15 OTA 55
15.1 Normal OTA 55

15.2 Silent OTA ... 56
16 Battery Voltage Module ... 57
16.1 Battery Module Configuration ... 57
16.2 Battery Module Workflow ... 57
17 LED ... 59
17.1 LED Configuration ... 59
17.2 LED Interface ... 59
18 MP Test ... 61
19 The Reference ... 62

Figure List

Figure 1 System Block Diagram ... 12
Figure 2 System Startup Flow ... 13
Figure 3 Normal Startup Flow ... 15
Figure 4 App task flow ... 16
Figure 5 APP IO Message Handler ... 17
Figure 6 RCU Status ... 18
Figure 7 RCU State Transformation Diagram ... 19
Figure 8 RCU State Transformation Conditions ... 20
Figure 9 STOP ADV REASON ... 20
Figure 10 DISCONNECT REASON ... 21
Figure 11 Software timers ... 22
Figure 12 ADV_IND PDU Payload ... 23
Figure 13 Advertising and Scan Response data format ... 23
Figure 14 ADV_DIRECT_IND PDU Payload ... 24
Figure 15 Pairing Advertising Packet ... 25
Figure 16 Prompt Advertising Packet Format ... 26
Figure 17 Power-On Advertising Packet Format ... 27
Figure 18 BLE Services ... 29
Figure 19 Keyscan Flow ... 31
Figure 20 RCU_STATUS_IDLE Single Key Processing Flow ... 33
Figure 21 RCU_STATUS_ADVERTISING Single Key Processing Flow ... 34
Figure 22 RCU_STATUS_PAIRED Single Key Processing Flow ... 34
Figure 23 Other States Single Key Processing Flow ... 35
Figure 24 Combined Key Definition ... 36
Figure 25 Key Code Table ... 38
Figure 26 Voice Module Block Diagram ... 39
Figure 27 Voice Module Initialization Flow ... 40
Figure 28 Voice Buffer Queue Diagram ... 45
Figure 29 IFLYTEK_VOICE_FLOW ... 47
Figure 30 HIDS Google VOICE FLOW – 1 ... 48
Figure 31 HIDS Google VOICE FLOW – 2 ... 49
Figure 32 ATV Google VOICE FLOW ... 51
Figure 33 RTK GATT VOICE FLOW ... 52

Figure 34 Switching to MP test mode flow 61

1 Overview

This document presents technical design and implementation information related to RTL8762C/RTL8752C remote control (Bee2 RCU), including advertising packets definition, standby and wakeup mechanism, pairing procedure, key operations and voice function, IR transmission and learning function, so as to provide instructions on development and to trace the problems encountered in software testing. The software supports two modes: R_DEMO_RCU and H_DEMO_RCU, this paper takes R_DEMO_RCU as an example to explain the detail.

2 Function & Feature

- Support BLE 4.2 (RTL8762C/RTL8752C)

- Support BLE 5.0 (RTL8762C)

- Support Automatic Pairing

- Support Power ON/OFF via Bluetooth

- Support Voice Input and Transmission

- Support 12 keys (5x4 Matrix) default; Support extension up to 208 keys (8x26) with embedded hardware Keyscan circuit

- Support LED Indication

- Support IR Transmission

- Support IR Learning

- Support Battery Voltage Detection

- Support Low-power protection mode

- Support OTA

- Support MP Mode

- Power Supply：1.8~3V

3 GPIO Configuration

3.1 System GPIO

P0_3 --- LOG

P1_0 --- SWDIO

P1_1 --- SWDCLK

P3_0 --- HCI UART TX

P3_1 —- HCI UART RX

3.2 Keyscan GPIO

P4_3 --- Keyscan ROW 0

P4_2 --- Keyscan ROW 1

P4_1 --- Keyscan ROW 2

P4_0 --- Keyscan ROW 3

P0_6 --- Keyscan ROW 4

P3_0 --- Keyscan COLUMN 0

P3_1 --- Keyscan COLUMN 1

P3_2 --- Keyscan COLUMN 2

P3_3 --- Keyscan COLUMN 3

3.3 Voice MIC GPIO

Digital MIC

P1_6 --- DMIC CLK

P1_7 --- DMIC DAT

Analog MIC

P2_6 --- AMIC_N

P2_7 --- AMIC_P

3.4 IR GPIO

P2_3 --- IR Transmit Pin

P2_2 --- IR Learn Pin

3.5 LED GPIO

P2_4 --- LED_1

3.6 MP Test GPIO

P3_2 --- MP Test Trigger Pin 1

P3_3 --- MP Test Trigger Pin 2

P3_0 --- MP Test UART TX

P3_1 --- MP Test UART RX

4 System Block Diagram

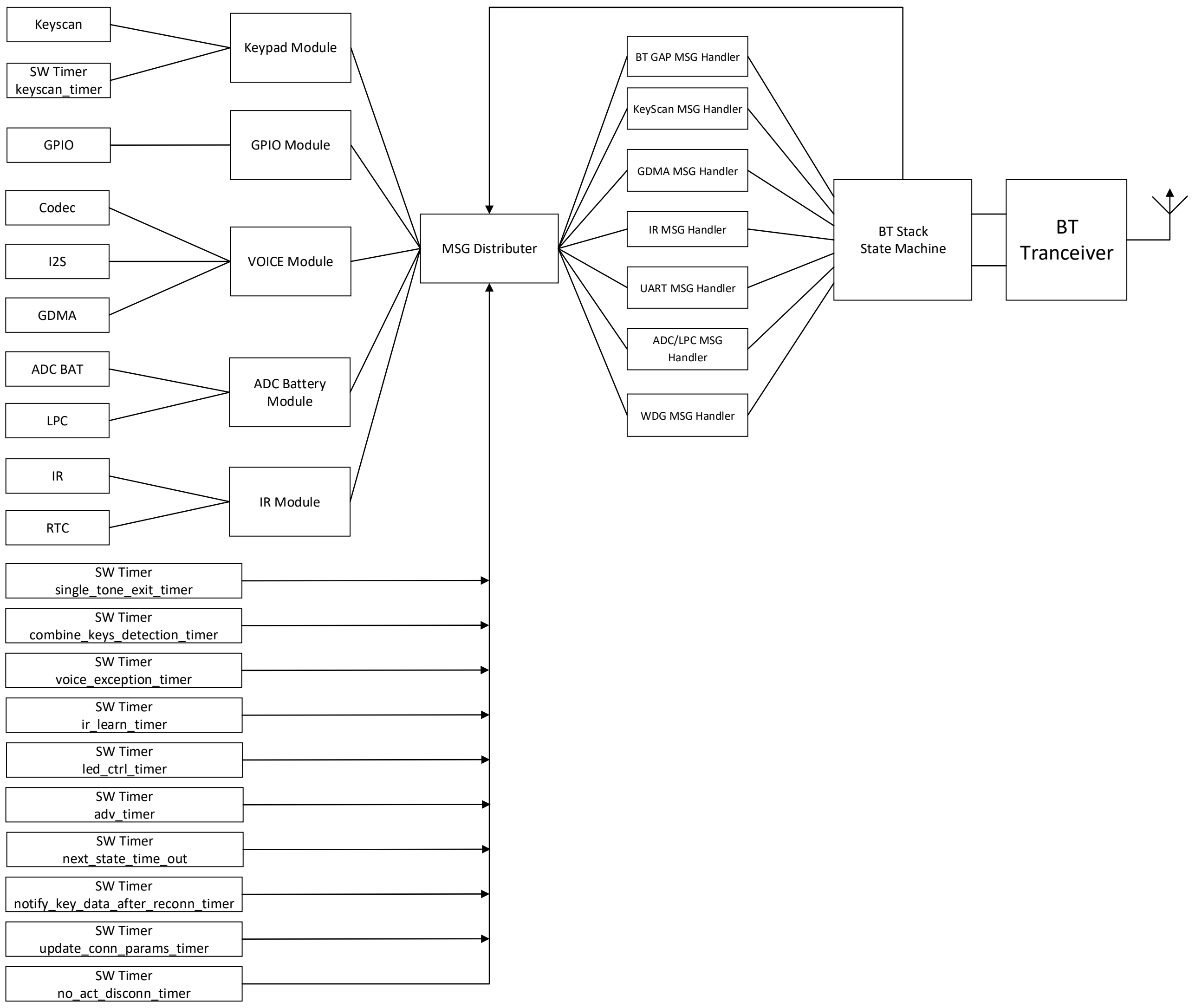


Figure 1 System Block Diagram

5 Initialization Process

5.1 System Startup Flow

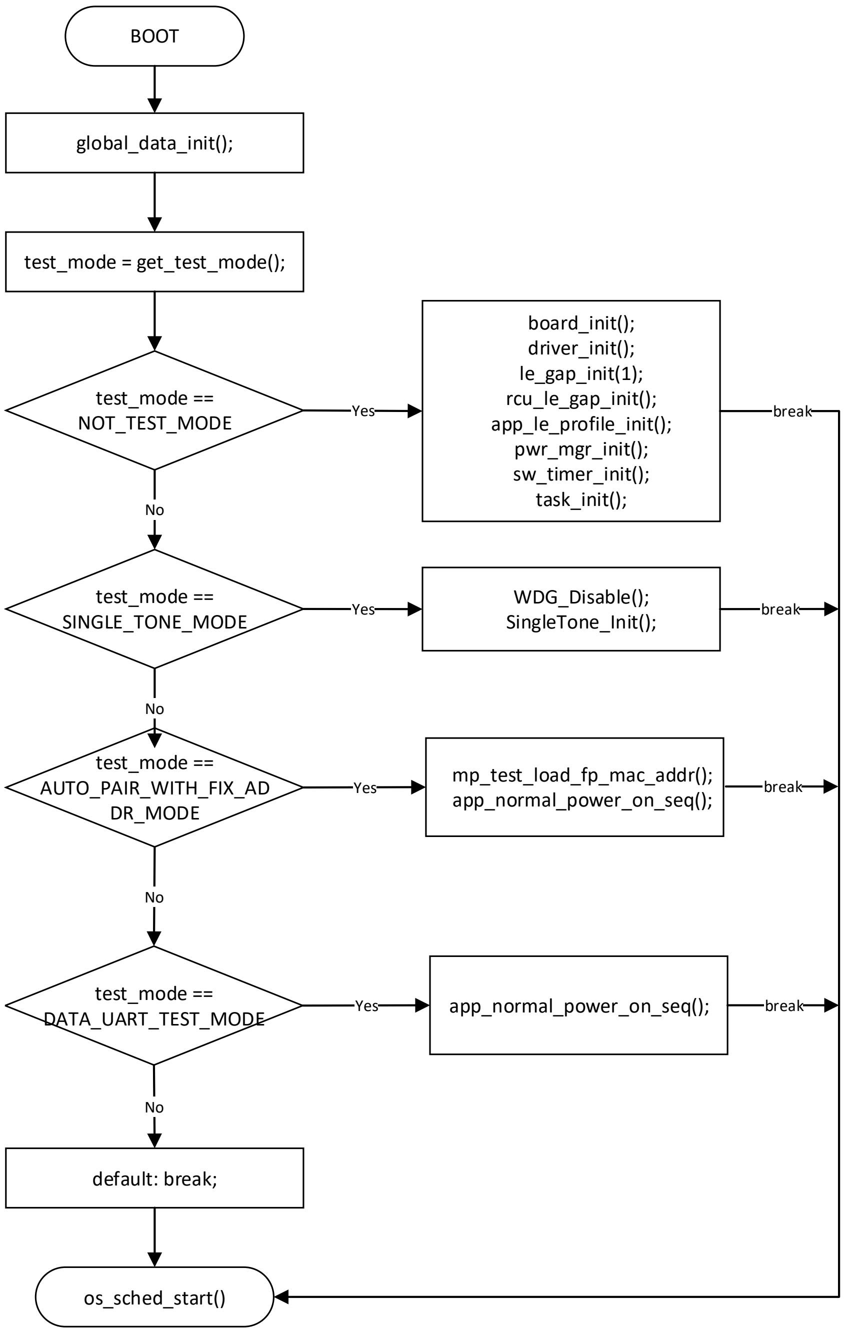

BOOT
global_data_init();
test_mode = get_test_mode();
test_mode == NOT_TEST_MODE
Yes
No
board_init(); driver_init(); le_gap_init(1); rcu_le_gap_init(); app_le_profile_init(); pwr_mgr_init(); sw_timer_init(); task_init();
break
test_mode == SINGLE_TONE_MODE
Yes
No
WDG_Disable(); SingleTone_Init();
break
test_mode == AUTO_PAIR_WITH_FIX_ADDR_MODE
Yes
No
mp_test_load_fp_mac_addr(); app_normal_power_on_seq();
break
test_mode == DATA_UART_TEST_MODE
Yes
No
app_normal_power_on_seq();
break
default: break;
os_sched_start()

Figure 2 System Startup Flow

System Mode

- Normal Mode

- Single Tone Mode
- Auto Pair with Fixed Address Mode
- Data UART Test Mode

5.2 Normal Startup Flow

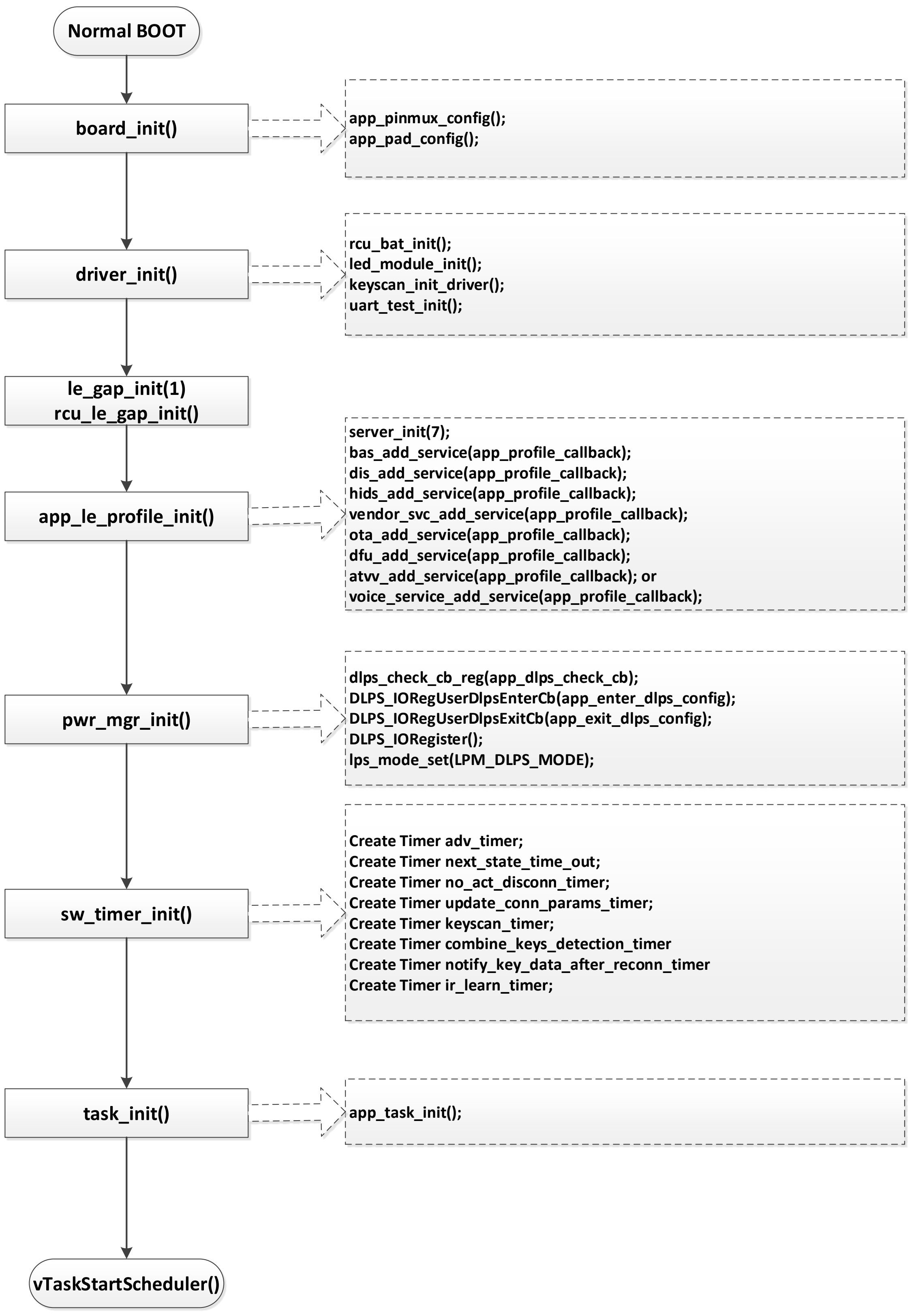

Normal BOOT
board_init()
app_pinmux_config();
app_pad_config();
driver_init()
rcu_bat_init();
led_module_init();
keyscan_init_driver();
uart_test_init();
le_gap_init(1)
rcu_le_gap_init()
app_le_profile_init()
server_init(7);
bas_add_service(app_profile_callback);
dis_add_service(app_profile_callback);
hids_add_service(app_profile_callback);
vendor_svc_add_service(app_profile_callback);
ota_add_service(app_profile_callback);
dfu_add_service(app_profile_callback);
atvv_add_service(app_profile_callback); or
voice_service_add_service(app_profile_callback);
pwr_mgr_init()
dlps_check_cb_reg(app_dlps_check_cb);
DLPS_IORegUserDlpsEnterCb(app_enter_dlps_config);
DLPS_IORegUserDlpsExitCb(app_exit_dlps_config);
DLPS_IORegister();
lps_mode_set(LPM_DLPS_MODE);
sw_timer_init()
Create Timer adv_timer;
Create Timer next_state_time_out;
Create Timer no_act_disconn_timer;
Create Timer update_conn_params_timer;
Create Timer keyscan_timer;
Create Timer combine_keys_detection_timer
Create Timer notify_key_data_after_reconn_timer
Create Timer ir_learn_timer;
task_init()
app_task_init();
vTaskStartScheduler()

Figure 3 Normal Startup Flow

6 APP TASK

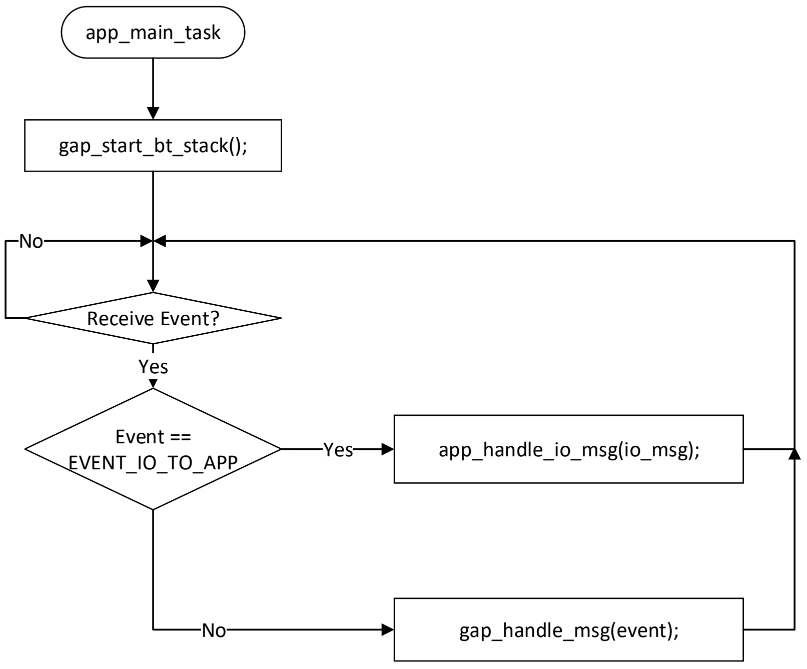


Figure 4 App task flow

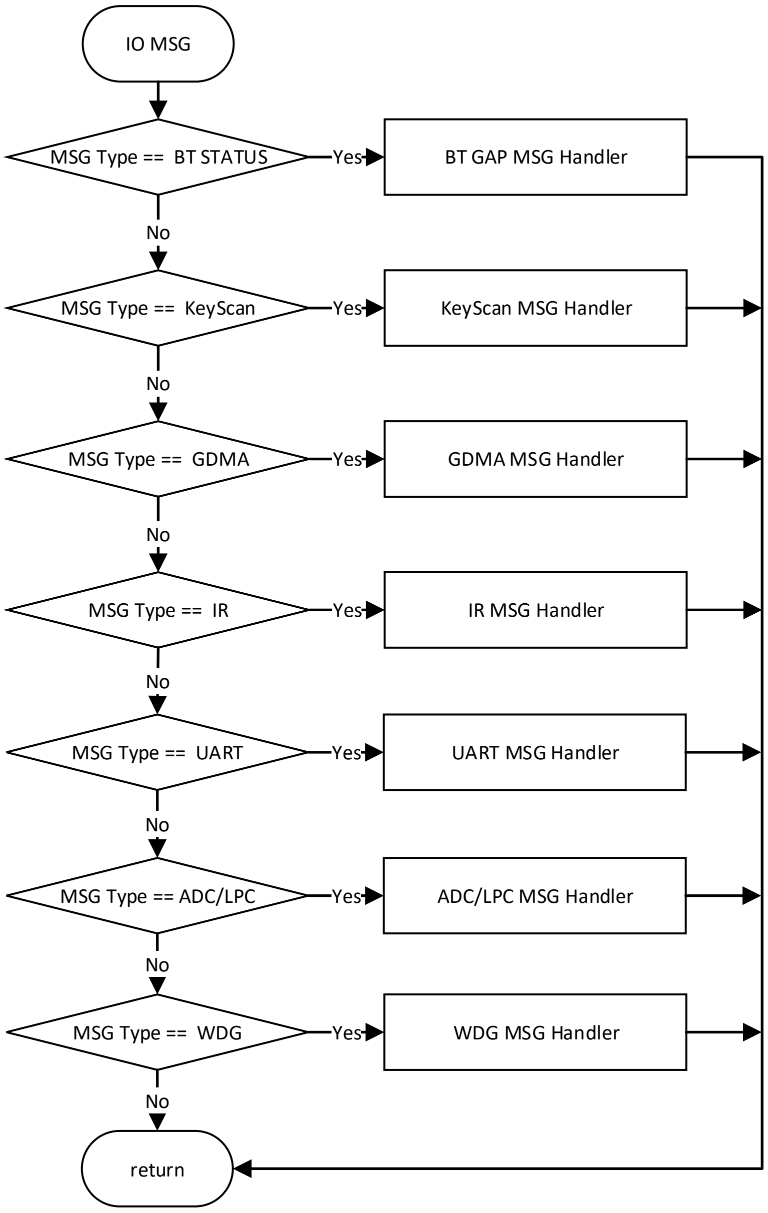


Figure 5 APP IO Message Handler

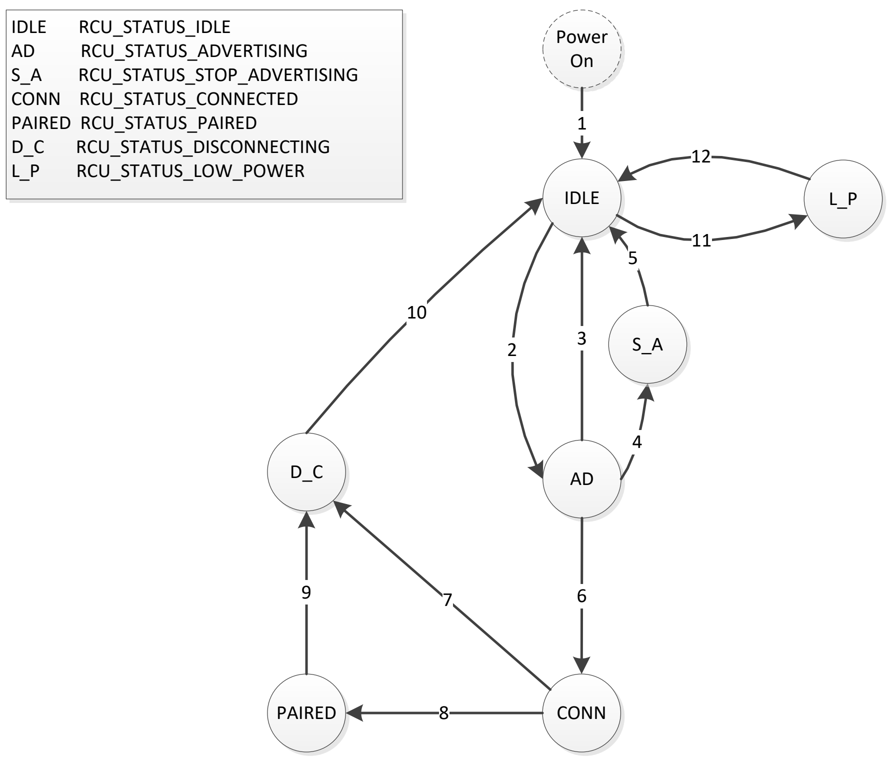
7 RCU Status

7.1 RCU Status

Number	Status	Description
1	RCU_STATUS_IDLE	IDLE state
2	RCU_STATUS_ADVERTISING	Advertising state
3	RCU_STATUS_STOP_ADVERTISING	Stop advertising command has been issued to stack, while GAP state is advertising (temporary state)
4	RCU_STATUS_CONNECTED	Connection has been established, while pairing is not started yet
5	RCU_STATUS_PAIRED	Pairing successfully state
6	RCU_STATUS_DISCONNECTING	Disconnecting command has been issued to stack, while GAP state is connected (temporary state)
7	RCU_STATUS_LOW_POWER	Low power mode state

Figure 6 RCU Status

7.2 RCU State Transformation Diagram



IDLE	RCU_STATUS_IDLE
AD	RCU_STATUS_ADVERTISING
S_A	RCU_STATUS_STOP_ADVERTISING
CONN	RCU_STATUS_CONNECTED
PAIRED	RCU_STATUS_PAIRED
D_C	RCU_STATUS_DISCONNECTING
L_P	RCU_STATUS_LOW_POWER

Figure 7 RCU State Transformation Diagram

7.3 RCU State Transformation Conditions

Number	Description
1	Power On after GAP ready
2	When APP calls le_adv_start in idle status
3	High duty cycle direct advertising time out, no connect request received
4	When APP calls le_adv_stop in advertising status
5	When BT stack sends GAP state change callback message from advertising to idle status
6	When connection is established
7	When connection is terminated in connected status
8	When pairing successfully in connected status
9	When connection is terminated in paired status
10	When BT stack sends GAP state change callback message from connection to idle status

11	When low power voltage detected in idle status
12	When normal power voltage detected in low power status

Figure 8 RCU State Transformation Conditions

7.4 RCU IDLE STATUS DESCRIPTION

Besides the case of system initialized upon first powered up, there are 2 scenarios in which system will switch to idle status.

A) No connection is established after an advertising event ends.

B) Connection is terminated under connection status.

For scenario A, the program issues a reason list for advertising termination, as enumerated below:

```
typedef enum
{
    STOP_ADV_REASON_IDLE = 0,
    STOP_ADV_REASON_PAIRING,
    STOP_ADV_REASON_TIMEOUT,
    STOP_ADV_REASON_POWERKEY,
     STOP_ADV_REASON_LOWPOWER,
     STOP_ADV_REASON_UART_CMD,
} T_STOP_ADV_REASON;
```

No.	STOP_ADV_REASON	Description
1	STOP_ADV_REASON_IDLE	When receiving stack callback message for ADV_DIRECT_HDC timeout
2	STOP_ADV_REASON_PAIRING	Stop current advertisement to start pairing advertisement
3	STOP_ADV_REASON_TIMEOUT	When APP calls le_adv_stop after software advertising timer timeout
4	STOP_ADV_REASON_POWERKEY	Stop current advertisement to start power advertisement
5	STOP_ADV_REASON_LOWPOWER	Stop current advertisement to enter Low-Power mode
6	STOP_ADV_REASON_UART_CMD	When receiving UART stop advertisement command

Figure 9 STOP ADV REASON

For scenario B, the program issues a reason list for disconnection, as enumerated below:

```
typedef enum
```

```
{
    DISCONN_REASON_IDLE = 0,
    DISCONN_REASON_PAIRING,
    DISCONN_REASON_TIMEOUT,
    DISCONN_REASON_OTA,
     DISCONN_REASON_PAIR_FAILED,
     DISCONN_REASON_LOW_POWER,
     DISCONN_REASON_UART_CMD,
     DISCONN_REASON_SILENT_OTA,
} T_DISCONN_REASON;
```

No.	DISCONN_REASON	Description
1	DISCONN_REASON_IDLE	Default value
2	DISCONN_REASON_PAIRING	Terminate connection to start pairing advertisement
3	DISCONN_REASON_TIMEOUT	Terminate connection, if no action detected for a certain period (FEATURE_SUPPORT_NO_ACTION_DISCONN is enabled)
4	DISCONN_REASON_OTA	Terminate connection to start OTA process
5	DISCONN_REASON_PAIR_FAILED	Terminate connection for pairing failed
6	DISCONN_REASON_LOW_POWER	Terminate connection to enter low power mode
7	DISCONN_REASON_UART_CMD	Terminate connection to receive UART command
8	DISCONN_REASON_SILENT_OTA	Need to restart the silent upgrade, disconnect the BLE connection

Figure 10 DISCONNECT REASON

8 SW Timer Application Case and Purpose

Ten software timers below are created in RCU SDK.

No.	SW Timer	Description
1	adv_timer	Timer is used to stop advertising after timeout
2	next_state_time_out	Timer is used to capture timeout for pairing
3	no_act_disconn_timer	Timer is used to terminate connection after timeout if there is no action under connection
4	update_conn_params_timer	Timer is used to update desired connection parameters after timeout
5	keyscan_timer	Timer is used for keyscan DLPS
6	combine_keys_detection_timer	Timer is used to detect combined keys after timeout
7	notify_key_data_after_reconn_timer	Timer is used to notify key data after timeout
8	ir_learn_timer	Timer is used for IR learning timeout
9	led_ctrl_timer	Timer is used to control LED timing
10	single_tone_exit_timer	Timer is used to exit Single Tone test mode after timeout
11	voice_exception_timer	Timer is used to limit the maximum working time of a single ATV voice

Figure 11 Software timers

9 Advertising Packet

9.1 Advertising Packet Format

RCU uses advertising packets in two formats:

1. Undirected advertising event: AdvA field is an address of the device sending advertising packets, and AdvData is in the format as is shown in Figure 13.

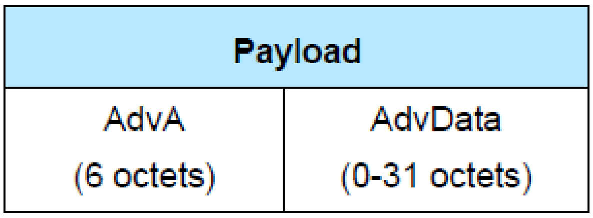


Figure 12 ADV_IND PDU Payload

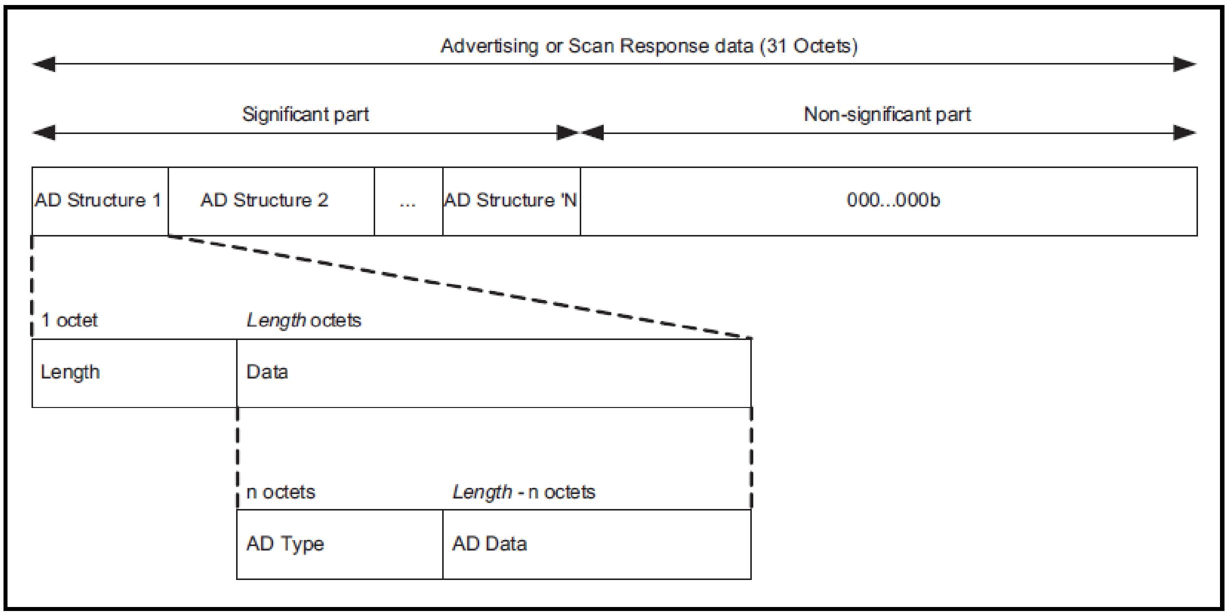


Figure 13 Advertising and Scan Response data format

2. Directed advertising event: AdvA field is an address of the device sending advertising packets, while InitA is an address of the remote device.

Payload	
AdvA (6 octets)	InitA (6 octets)

Figure 14 ADV_DIRECT_IND PDU Payload

9.2 Advertising Packet Type

Bee2 RCU shall call rcu_start_adv to set advertisement type and start advertising, and the advertising type is enumerated below：

```
typedef enum
{
    ADV_IDLE = 0,
    ADV_DIRECT_HDC,
    ADV_UNDIRECT_RECONNECT,
    ADV_UNDIRECT_PAIRING,
    ADV_UNDIRECT_PROMPT,
    ADV_UNDIRECT_POWER,
} T_ADV_TYPE;
```

9.2.1 Pairing Advertising Packet

RCU sends pairing advertising packets in pairing mode to activate automatic pairing flow initiated by peer device. Pairing advertising packet is Undirected Advertising Packet, which will last for 20s with recommended Advertising Interval of 0x20-0x30 (20ms-30ms). Pairing mode will be activated in one of the following 2 cases:

1. Press and hold pairing combined keys (Volume+ and OK) for 2s, Link Key stored in RCU will be cleared and the RCU will enter pairing mode.

2. If RCU is not paired before and macro FEATURE_SUPPORT_ANY_KEY_TRIG_PAIRING_ADV is set to 1, any key event will activate RCU to enter pairing mode.

   If FEATURE_SUPPORT_ANY_KEY_TRIG_PAIRING_ADV is set to 0, only pairing combined keys will trigger paring, other keys are invalid. In RCU SDK, default value of FEATURE_SUPPORT_ANY_KEY_TRIG_PAIRING_ADV is 0.

Pairing advertising packet is in the following format:

Flag field (byte0~2)	Service Field (byte3~6)	Appearance Field – Device type (byte7~10)	Local Name Field (byte11~20)	Vendor Information Field (byte21~26)
0x02,0x01,0x05	0x03,0x03, 0x12,0x18	0x03,0x19,0x80,0x01	0x09,0x09, 'B', 'e', 'e', '2', ' ', 'R', 'C', 'U'	0x05,0xff,0x5d,0x00,0x04, 0x00

Figure 15 Pairing Advertising Packet

9.2.2 Reconnection Advertising Packet

RCU sends reconnection advertising packets and connection can be established quickly if link key has been stored.

Reconnection advertising packet is Direct Advertising High Duty Packet, and sent at 1.28s each time. RCU will enter reconnection mode in one of the following two cases.

1. RCU enters reconnection mode when it is powered on, if it has ever been paired successfully and Link Key is stored in it.

2. After RCU connects to the peer device, any key event will activate it to enter reconnection mode when the connection is terminated in case that the peer device is shutdown, or the RCU is out of range, or it actively disconnects when entering DLPS mode.

Bee2 RCU implements Direct Advertising High Duty Packet as reconnection advertising packet, this type of reconnection advertising packet will obtain the least latency for reconnection. On the other side, this High Duty Packet will require more power consumption. If in other solution, which is required for long time reconnection, other two types can be implemented: Low Duty Cycle Direct Advertising or Undirect Advertising with white list. Set the broadcast Packet format of Undirect Advertising of white list to Undirected Advertising Packet, and Advertising Interval is recommended to be set as 0x20, which means 20ms. In addition, the duration should be set to 3s. When the macro FEATURE_SUPPORT_PRIVACY is set to 1, the reconnect mode sends Undirected

Advertising Packet, and when the macro FEATURE_SUPPORT_PRIVACY is set to 0, the reconnect mode sends Direct Advertising High Duty Packet. By default, FEATURE_SUPPORT_PRIVACY is 0, which means that PRIVACY features are not supported.

9.2.3 Prompt Advertising Packet

Bee2 RCU transmits non-connectable advertising packets to activate the peer device to pop up a connection prompt, prompting user to perform pairing operations by the following the instructions. Prompt advertising packet is Undirected Advertising Packet, which will last for 5s with recommended Advertising Interval of 0x20-0x30 (20ms-30ms).

In SDK, prompt advertising can be enabled or disabled by FEATURE_SUPPORT_UNDIRECT_PROMPT_ADV. In default, FEATURE_SUPPORT_UNDIRECT_PROMPT_ADV is set to 0, which means RCU shall not send prompt advertising packets. If FEATURE_SUPPORT_UNDIRECT_PROMPT_ADV is set to 1, prompting pairing mode will be activated in the following cases:

In case of available link key, press any key, except for Power key, to enter reconnection mode. If connection cannot be established with the following three successive reconnection advertising packets, prompting pairing mode is activated to send prompt advertising packet.

Prompt advertising packet format: (the difference with pairing advertising packet is that the Limited Discoverable Mode bit is 0 and non-connectable)

Flag field (byte0~2)	Service Field (byte3~6)	Appearance Field – Device type (byte7~10)	Local Name Field (byte11~20)	Vendor Information Field (byte21~26)
0x02,0x01,0x04	0x03,0x19,0xc1,0x03	0x03,0x19,0xc1,0x03	0x09,0x09, 'B', 'e', 'e', '2', ' ', 'R', 'C', 'U'	0x05,0xff,0x5d,0x00,0x04,0x00

Figure 16 Prompt Advertising Packet Format

9.2.4 Power-on Advertising Packet

RCU sends power-on advertising packet to wake up the peer device in sleep, while RCU must sends HIDS key value to enter sleep mode from start mode. Power-on advertising packet is Undirected Advertising Packet, which will last for 12s with Advertising Interval of 0x20-0x30 (20ms-30ms). Power-on Advertising Packet just contains BD Address of the peer device, so RCU can only wake up the peer device which has been paired.

Power-on mode will be activated in the following 2 cases:

1. In the case where link key is already exists, press the power button to enter the power-on mode;

2. In reconnection mode and prompt pairing mode, press the power button to enter the power-on mode;

Power-On advertising packet format:

Flag field （byte0~2）	**Service Field （byte3~6）**	**Appearance Field – Device type (byte7~10)**	**Vendor Information Field (byte11~24)**
0x02,0x01,0x04	0x03,0x03, 0x12,0x18	0x03,0x19,0x80,0x01	0x0D,0xFF,0x5D,0x00,0x03,0x00,0x01,0x01(index), 0xXX,0xXX, 0xXX, 0xXX, 0xXX,0xXX,(TV BD Address)

Figure 17 Power-On Advertising Packet Format

10 Connection and Pairing

The connection and pairing scenarios are different according to the pairing information status on both RCU and Master device:

1. Previous pairing information has been stored on both RCU and Master device: any key pressed on RCU will trigger reconnection process and connection will be established quickly without pairing process.

2. Previous pairing information has been removed on both RCU and Master device: as reconnection process cannot work, RCU shall press pairing combined keys to trigger pairing advertisement. Then master device shall attempt to establish connection and start pairing process.

3. Previous pairing information has been stored on RCU side, but it has been removed on master device: Similar to scenario 2, reconnection process cannot work, so RCU shall press pairing combined keys to trigger pairing advertisement. Then master device shall attempt to establish connection and start pairing process.

4. Previous pairing information has been stored on master device, but it has been removed on RCU: reconnection process cannot work, so RCU shall press pairing combine keys to trigger pairing advertisement, but master device will start reconnection process this time. Pairing will failed due to "Key Missing". To work around this failure, RCU will terminate current connection and start pairing advertisement. Meanwhile, master device shall start paring process.

After pairing successfully, master device will start GATT services finding process and enable claimed CCCD bits.

In order to achieve lower power consumption, Bee2 RCU will send connection parameter update request with desired connection parameters.

In bee2 SDK, the default connection parameters are as follows:

- Connection Interval: 15ms
- Slave Latency: 14
- Supervision Timeout Period: 2s

11 BLE Service

Bee2 RCU supports the following BLE Services:

No.	Service	Description
1	GAP	Generic Access Profile
2	DIS	Device Information Service
3	HIDS	HID Service
4	BAS	Battery Access Service
5	OTA Service	Realtek OTA Service
6	DFU Service	Realtek Silent OTA Service
7	Vendor Service	Realtek RCU Test Tool Service
8	ATV Voice Service	Google Vendor ATV Voice Service
9	RTK GATT Voice Service	RTK GATT Voice Service

Figure 18 BLE Services

12 Key

12.1 Keyscan Scheme

To achieve the minimum power consumption of the Keyscan module, RCU uses the hardware Keyscan + SW Timer + HW debounce solution. The Keyscan is divided into ***Platform always Active*** mode and ***Platform Sleep at scan interval*** mode. The difference between the two is the usage of the debounce. In the Platform always Active mode, debounce normally uses the debounce of Keyscan module. In the Platform Sleep at scan interval mode, debounce uses the PAD’s debounce to shorten the interruption time.

HW performs a debounce and scans a time when key is pressed. After the end of the scan, the software generates scan interrupt, reads Keyscan FIFO data in the interrupt handler and sends IO_MSG_KEYSCAN_RX_PKT message to the APP task for processing. Meanwhile, start a SW scan timer of scan interval, and then enter DLPS. The system wakes up when the SW scan timer expires. Re-initialize the Keyscan module with debounce disabled, and start a 10ms SW release timer for key status detection.

If all keys are released, the scan interrupt is not triggered. When the SW scan timer expires, send IO_MSG_KEYSCAN_ALLKEYRELEASE message to the APP task for processing and re-initialize the Keyscan module with debounce enabled.

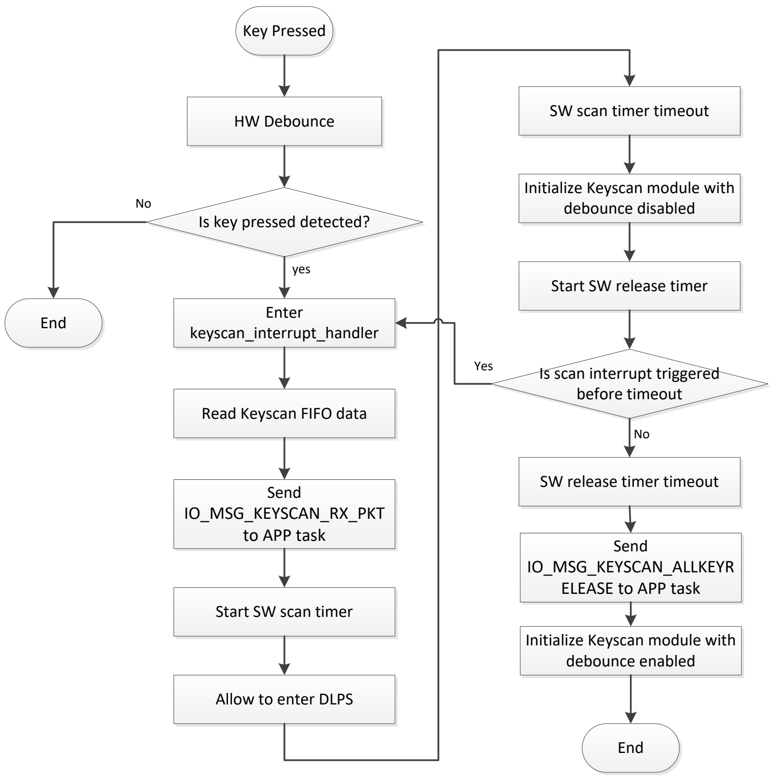

Key Pressed
HW Debounce
Is key pressed detected?
No
yes
End
Enter keyscan_interrupt_handler
Read Keyscan FIFO data
Send IO_MSG_KEYSCAN_RX_PKT to APP task
Start SW scan timer
Allow to enter DLPS
SW scan timer timeout
Initialize Keyscan module with debounce disabled
Start SW release timer
Is scan interrupt triggered before timeout
Yes
No
SW release timer timeout
Send IO_MSG_KEYSCAN_ALLKEYRELEASE to APP task
Initialize Keyscan module with debounce enabled
End

Figure 19 Keyscan Flow

12.2 Keyscan Configuration

In the RCU application software, the default configuration of the keyscan matrix is 5 rows and 4 columns. The pin of the rows and columns are defined in the board.h and can be modified according to the actual project requirements.

```
/************************************************************
*                   RCU Keyscan Config
************************************************************/
/* keypad row and column */
#define KEYPAD_ROW_SIZE        5
#define KEYPAD_COLUMN_SIZE     4

#define ROW0                   P4_3
#define ROW1                   P4_2
#define ROW2                   P4_1
#define ROW3                   P4_0
#define ROW4                   P0_6

#define COLUMN0                 P3_0
```

```
#define COLUMN1             P3_1
#define COLUMN2             P3_2
#define COLUMN3             P3_3
```

The rows and columns of the Keyscan are defined in key_handle.c which can be modified according to the actual project requirements.

```
/* Key Mapping Table Definiton */
static const T_KEY_INDEX_DEF KEY_MAPPING_TABLE[KEYPAD_ROW_SIZE][KEYPAD_COLUMN_SI
ZE] =
{
    {VK_TV_POWER,       VK_EXIT,           VK_ENTER,     VK_MOUSE_EN},
    {VK_POWER,          VK_VOLUME_UP,      VK_DOWN,      VK_RIGHT},
    {VK_TV_SIGNAL,      VK_VOLUME_DOWN,   VK_VOICE,     VK_MENU},
    {VK_UP,             VK_PAGE_DOWN,      VK_HOME,      VK_PAGE_UP},
    {VK_LEFT,           VK_NC,             VK_NC,        VK_NC},
}
```

12.3 Key Behavior Rules

This section mainly introduces key behavior rules and provides a scheme for key processing, which can be adjusted according to specific project requirements. There are different key behaviors based on the number of keys detected and the current state of the RCU.

12.3.1 Single Key Behavior

After a single key is pressed, RCU application software receives IO_MSG_KEYSCAN_RX_PKT sent by the Keyscan module. According to the current state of the APP, the following processing is carried out.

- RCU_STATUS_IDLE

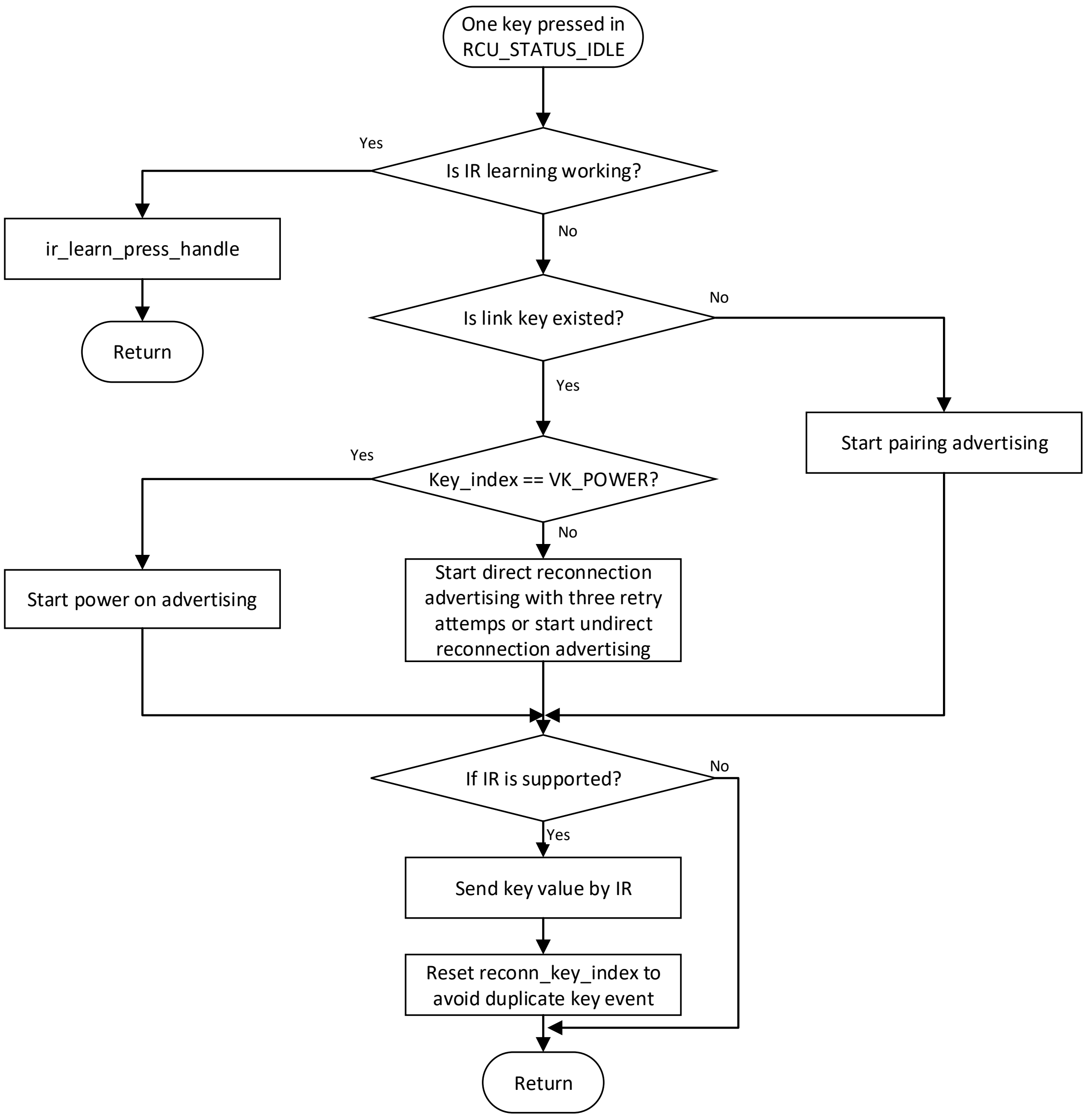

One key pressed in RCU_STATUS_IDLE
Is IR learning working?
Yes
No
ir_learn_press_handle
Return
Is link key existed?
No
Yes
Start pairing advertising
Key_index == VK_POWER?
Yes
No
Start power on advertising
Start direct reconnection advertising with three retry attemps or start undirect reconnection advertising
If IR is supported?
No
Yes
Send key value by IR
Reset reconn_key_index to avoid duplicate key event
Return

Figure 20 RCU_STATUS_IDLE Single Key Processing Flow

- RCU_STATUS_ADVERTISING

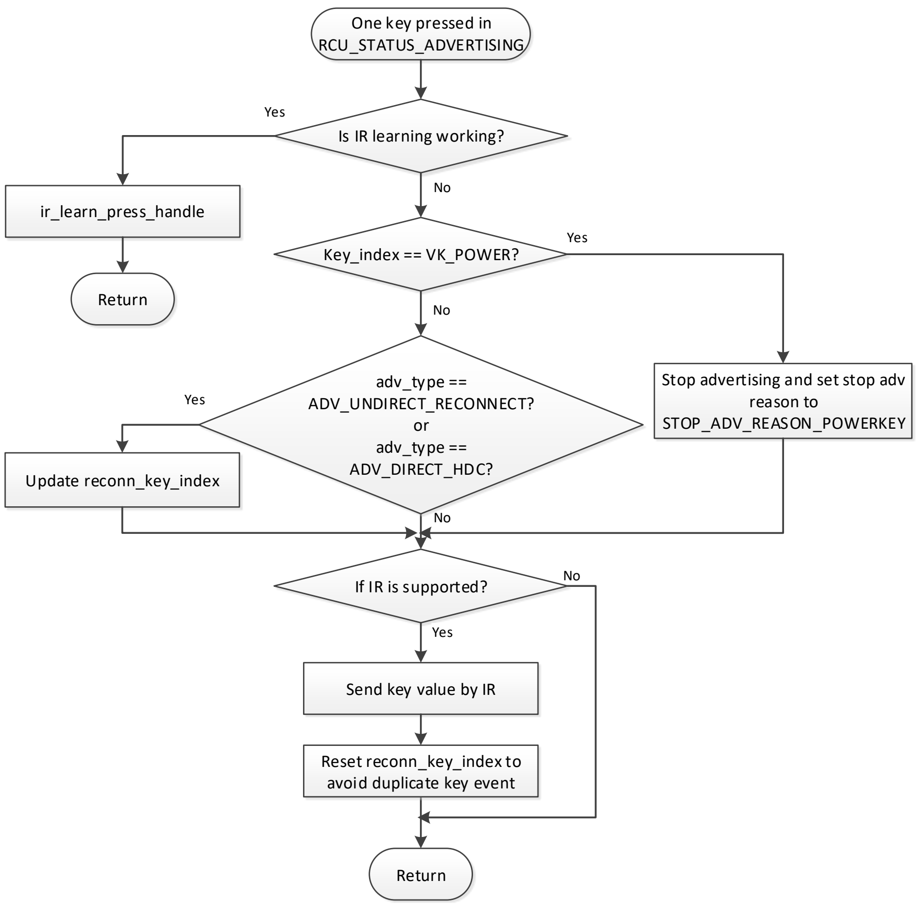

One key pressed in RCU_STATUS_ADVERTISING
Is IR learning working?
Yes
ir_learn_press_handle
Return
No
Key_index == VK_POWER?
Yes
Stop advertising and set stop adv reason to STOP_ADV_REASON_POWERKEY
No
adv_type == ADV_UNDIRECT_RECONNECT? or adv_type == ADV_DIRECT_HDC?
Yes
Update reconn_key_index
No
If IR is supported?
No
Yes
Send key value by IR
Reset reconn_key_index to avoid duplicate key event
Return

Figure 21 RCU_STATUS_ADVERTISING Single Key Processing Flow

- RCU_STATUS_PAIRED

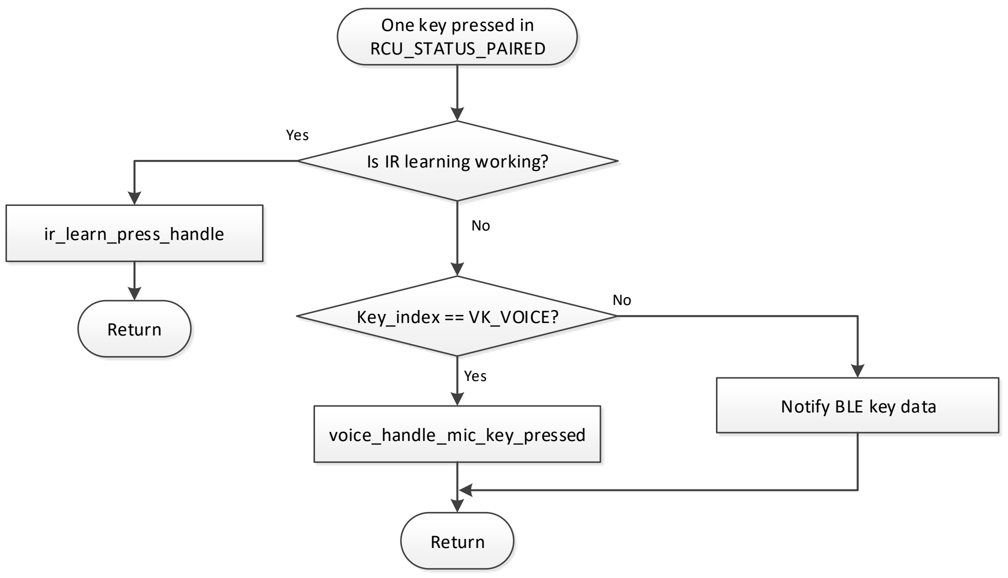


Figure 22 RCU_STATUS_PAIRED Single Key Processing Flow

- Other status

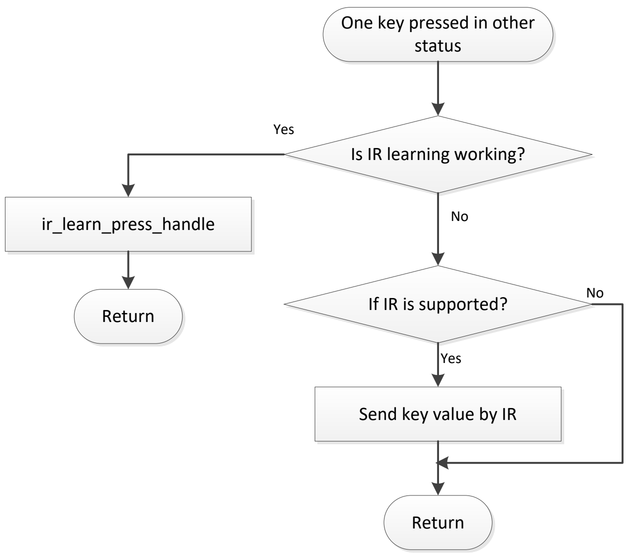


Figure 23 Other States Single Key Processing Flow

12.3.2 Double-Key Behaviors

When two keys are pressed at the same time, the RCU APP detects whether they are combined keys defined. If it is not combined keys, the RCU APP will send BLE Notification of Key Up or stop the previous infrared transmission. If they are combined keys, the SW Timer is turned on to verify that whether the combined keys meet the requirement of press time.

RCU APP uses a 32-bit global variable to distinguish different combination of keys, and each key combination corresponds to a bit. Currently, there are several combinations of keys as below. Actual adjustments can be made according to specific project needs.

No.	Bit Mask	Combined key	instructions
1	0x0001	VK_ENTER + VK_VOLUME_UP	Enter pairing mode
2	0x0002	VK_ENTER + VK_LEFT	Enter infrared learning mode （IR_LEARN_MODE effective）
3	0x0004	VK_POWER + VK_VOLUME_UP	Enter HCI UART testing mode （MP_TEST_MODE_SUPPORT_HCI_UART_TEST effective）

4	0x0008	VK_POWER + VK_VOLUME_DOWN	Enter Data UART testing mode （MP_TEST_MODE_SUPPORT_DATA_UART_TEST effective）
5	0x0010	VK_POWER + VK_EXIT	Enter Single Tone testing mode （MP_TEST_MODE_SUPPORT_SINGLE_TONE_TEST effective）
6	0x0020	VK_POWER + VK_UP	Enter fast pairing mode 1 （MP_TEST_MODE_SUPPORT_FAST_PAIR_TEST effective）
7	0x0040	VK_POWER + VK_DOWN	Enter fast pairing mode 2 （MP_TEST_MODE_SUPPORT_FAST_PAIR_TEST effective）
8	0x0080	VK_POWER + VK_LEFT	Enter fast pairing mode 3 （MP_TEST_MODE_SUPPORT_FAST_PAIR_TEST effective）
9	0x0100	VK_POWER + VK_RIGHT	Enter fast pairing mode 4 （MP_TEST_MODE_SUPPORT_FAST_PAIR_TEST effective）
10	0x0200	VK_POWER + VK_ENTER	Enter fast pairing mode 5 （MP_TEST_MODE_SUPPORT_FAST_PAIR_TEST effective）

Figure 24 Combined Key Definition

12.3.3 Three and more keys Behaviors

When three or more buttons are pressed, the RCU APP will send the BLE Notification of Key Up or stop the previous infrared transmission.

12.4 Key HID Code Value Table

RCU APP supports two HID Usage Page key values, Keyboard/Keypad Page (0x07) and Consumer Page (0x0C). For the Keyboard/Keypad Page keys, their key values are defined in the hids_report_descriptor in hids_rmc.c. When sending key notification, send HID key Value directly.

For the Consumer Page keys, their key values are defined in the hids_report_descriptor in hids_rmc.c. The BLE_KEY_CODE_TABLE in key_handle.c is its corresponding Key Index. When sending key notification, key Index is sent. The host gets the corresponding Key Value according to Key Index and hids_report_descriptor.

Key No.	HID Usage Page	HID Value
VK_POWER	Keyboard/Keypad Page(0x07)	0x66
VK_PAGE_UP	Keyboard/Keypad Page(0x07)	0x4B
VK_PAGE_DOWN	Keyboard/Keypad Page(0x07)	0x4E
VK_MENU	Keyboard/Keypad Page(0x07)	0x76
VK_HOME	Keyboard/Keypad Page(0x07)	0x4A
VK_VOICE	Keyboard/Keypad Page(0x07)	0x3E
VK_ENTER	Keyboard/Keypad Page(0x07)	0x28
VK_EXIT	Keyboard/Keypad Page(0x07)	0x29
VK_LEFT	Keyboard/Keypad Page(0x07)	0x50
VK_RIGHT	Keyboard/Keypad Page(0x07)	0x4F
VK_UP	Keyboard/Keypad Page(0x07)	0x52
VK_DOWN	Keyboard/Keypad Page(0x07)	0x51
VK_VOLUME_MUTE	Keyboard/Keypad Page(0x07)	0x7F
VK_VOLUME_UP	Keyboard/Keypad Page(0x07)	0x80
VK_VOLUME_DOWN	Keyboard/Keypad Page(0x07)	0x81
VK_VOICE_STOP	Keyboard/Keypad Page(0x07)	0x3F
MM_ScanNext	Consumer Page(0x0C)	0xB5
MM_ScanPrevious	Consumer Page(0x0C)	0xB6
MM_Stop	Consumer Page(0x0C)	0xB7
MM_Play_Pause	Consumer Page(0x0C)	0xCD
MM_Mute	Consumer Page(0x0C)	0xE2
MM_BassBoost	Consumer Page(0x0C)	0xE5
MM_Loudness	Consumer Page(0x0C)	0xE7
MM_VolumeIncrement	Consumer Page(0x0C)	0xE9
MM_VolumeDecrement	Consumer Page(0x0C)	0xEA

MM_BassIncrement	Consumer Page(0x0C)	0x0152
MM_BassDecrement	Consumer Page(0x0C)	0x0153
MM_TrebleIncrement	Consumer Page(0x0C)	0x0154
MM_TrebleDecrement	Consumer Page(0x0C)	0x0155
MM_AL_ConsumerControl	Consumer Page(0x0C)	0x0183
MM_AL_EmailReader	Consumer Page(0x0C)	0x018A
MM_AL_Calculator	Consumer Page(0x0C)	0x0192
MM_AL_LocalMachineBrowser	Consumer Page(0x0C)	0x0194
MM_AC_Search	Consumer Page(0x0C)	0x0221
MM_AC_Home	Consumer Page(0x0C)	0x0223
MM_AC_Back	Consumer Page(0x0C)	0x0224
MM_AC_Forward	Consumer Page(0x0C)	0x0225
MM_AC_Stop	Consumer Page(0x0C)	0x0226
MM_AC_Refresh	Consumer Page(0x0C)	0x0227
MM_AC_Bookmarks	Consumer Page(0x0C)	0x022A

Figure 25 Key Code Table

13 Voice

13.1 Voice Module Introduction

RCU supports voice function, which recording voice information by Digital Microphone (DMIC) or Analog Microphone (AMIC). Then voice data is encoded and sent to the peer device by Bluetooth. The voice module block diagram is shown in Figure 26.

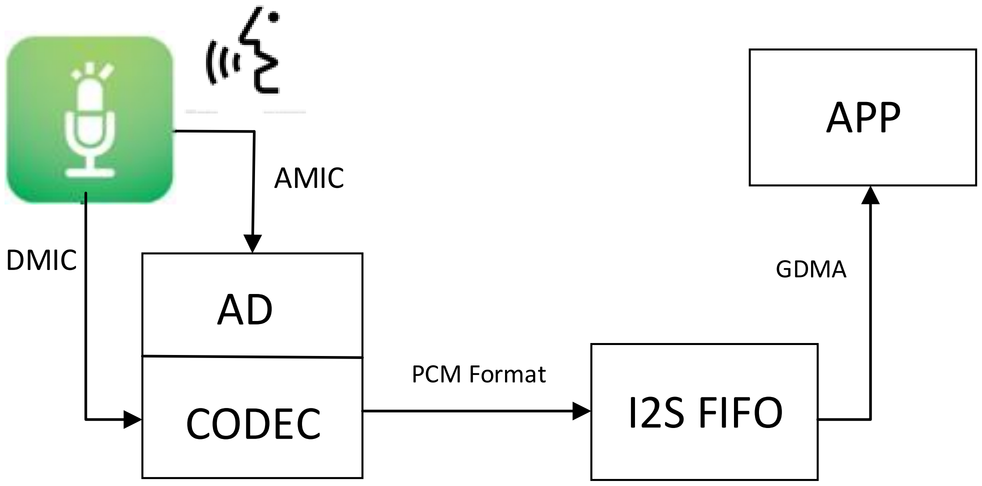


Figure 26 Voice Module Block Diagram

When using DMIC, RCU directly move the digital voice data to CODEC. Then COEDC encodes raw data into PCM format and put it into I2S FIFO. After that, GDMA automatically moves the encoded data into RAM which can be used by App.

When using AMIC, RCU uses AD converter to process input analogy voltage signal firstly, then CODEC encodes converted data into PCM format and put it into I2S FIFO. After that, as same as DMIC, GDMA automatically moves the encoded data into RAM, which can be used by App.

13.2 Voice Module Initialization

Voice module initialization includes, voice module data initialization and I/O initialization such as Pad/Pinmux/I2S/CODEC/GDMA. In program, the initialization of voice module calls voice_handle_start_mic function.

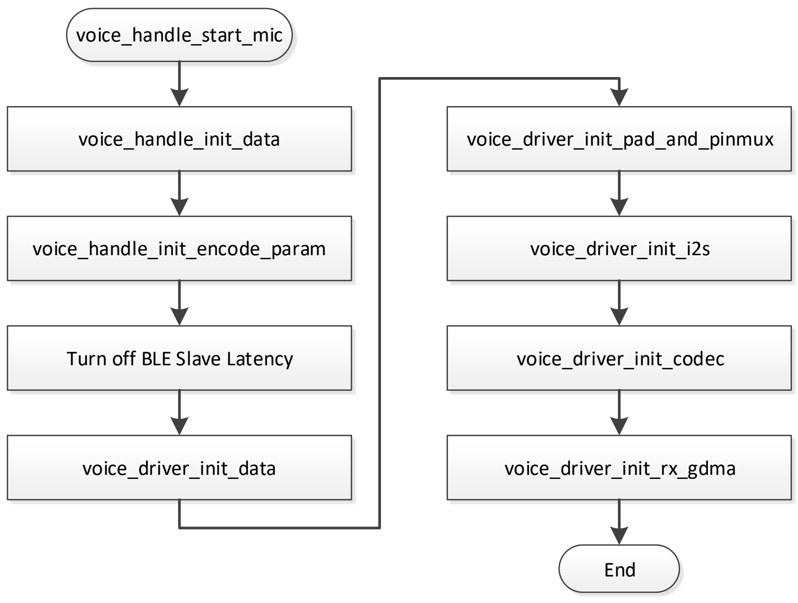

voice_handle_start_mic
voice_handle_init_data
voice_handle_init_encode_param
Turn off BLE Slave Latency
voice_driver_init_data
voice_driver_init_pad_and_pinmux
voice_driver_init_i2s
voice_driver_init_codec
voice_driver_init_rx_gdma
End

Figure 27 Voice Module Initialization Flow

13.2.1 Voice PAD Initialization

If Bee2 RCU uses AMIC, the three PAD MIC_N, MIC_P and MICBIAS which corresponding to P2_6, p2_7 and H0 need to be used. If useing DMIC, the two PAD DMIC_CLK and DMIC_DAT which use P1_6 and P1_7 by default need to be used. You can use PINPUX to switch to other GPIO. The specific initialization code is as below.

```
static void voice_driver_init_pad_and_pinmux(void)
{
#if (VOICE_MIC_TYPE == AMIC_TYPE)
    Pad_Config(H_0, PAD_SW_MODE, PAD_NOT_PWRON, PAD_PULL_NONE, PAD_OUT_ENABLE, PAD_OUT_HIGH);
    Pad_Config(AMIC_PIN1, PAD_SW_MODE, PAD_NOT_PWRON, PAD_PULL_NONE, PAD_OUT_DISABLE, PAD_OUT_HIGH);
    Pad_Config(AMIC_PIN2, PAD_SW_MODE, PAD_NOT_PWRON, PAD_PULL_NONE, PAD_OUT_DISABLE, PAD_OUT_LOW);
#elif (VOICE_MIC_TYPE == DMIC_TYPE)
    Pad_Config(DMIC_MSBC_CLK, PAD_PINMUX_MODE, PAD_IS_PWRON, PAD_PULL_NONE, PAD_OUT_DISABLE,  PAD_OUT_LOW);
    Pad_Config(DMIC_MSBC_DAT, PAD_PINMUX_MODE, PAD_IS_PWRON, PAD_PULL_NONE, PAD_OUT_DISABLE,  PAD_OUT_LOW);
    Pinmux_Deinit(DMIC_MSBC_CLK);
    Pinmux_Deinit(DMIC_MSBC_DAT);
    Pinmux_Config(DMIC_MSBC_CLK, DMIC1_CLK);
    Pinmux_Config(DMIC_MSBC_DAT, DMIC1_DAT);
#endif
}
```

13.2.2 Voice I2S Initialization

Bee2 RCU uses I2S to transfer PCM data of CODEC to APP. In I2S configuration, it is necessary to note that the BCLK configuration should be consistent with the sampling rate of CODEC. The specific initialization code is as below.

```
static void voice_driver_init_i2s(void)
{
    RCC_PeriphClockCmd(APBPeriph_I2S0, APBPeriph_I2S0_CLOCK, ENABLE);

    I2S_InitTypeDef I2S_InitStruct;
    I2S_StructInit(&I2S_InitStruct);
    I2S_InitStruct.I2S_ClockSource      = I2S_CLK_40M;
#if (CODEC_SAMPLE_RATE_SEL == CODEC_SAMPLE_RATE_8K)
    I2S_InitStruct.I2S_BClockMi        = 0x271;/* <!BCLK = 8K */
    I2S_InitStruct.I2S_BClockNi        = 0x08;
#elif (CODEC_SAMPLE_RATE_SEL == CODEC_SAMPLE_RATE_16K)
    I2S_InitStruct.I2S_BClockMi        = 0x271;/* <!BCLK = 16K */
    I2S_InitStruct.I2S_BClockNi        = 0x10;
#endif
    I2S_InitStruct.I2S_DeviceMode       = I2S_DeviceMode_Master;
    I2S_InitStruct.I2S_ChannelType      = I2S_Channel_Mono;
    I2S_InitStruct.I2S_DataFormat       = I2S_Mode;
    I2S_InitStruct.I2S_RxWaterlevel       = 0x4;
    I2S_InitStruct.I2S_TxWaterlevel       = 0x8;
    I2S_Init(I2S0, &I2S_InitStruct);

    I2S_Cmd(I2S0, I2S_MODE_TX | I2S_MODE_RX, ENABLE);
```

}

13.2.3 Voice CODEC Initialization

The sampling rates of Bee2 RCU Codec can be set to 8 KHz or 16 KHz.

For AMIC, four parameters MicBIAS, MICBstMode, MICBstGain and AdGain need to be configured during initialization.

- MICBIAS is the bias voltage of the MIC and needs to be set according to the parameters of the MIC.
- MICBstMode is the voice input mode selection. It can be divided into MICBST_Mode_Single and MICBST_Mode_Differential. It must be selected according to the connection mode of the hardware circuit. The default value MICBST_Mode_Differential is recommended.
- MICBstGain can adjust Codec's Analog Gain. The default value is MICBST_Gain_20dB. AdGain can adjust Codec's Digital Gain and the default value is 0x2F. These values need to be adjusted according to the required pickup distance and the housing of RCU. The adjustment principle is: In the case that the MICBstGain can be adjusted to meet the demand, AdGain is kept as the default value as much as possible so that the signal to noise of the amplified signal can be better.

For DMIC, three parameters DmicClock, DmicDataLatch and BoostGain need to be configured during initialization.

- DmicClock sets the DMIC clock. The default value is DMIC_Clock_2MHz.
- DmicDataLatch sets the Data Latch Type of the DMIC. The default value is DMIC_Rising_Latch.
- BoostGain sets Codec's Digital Gain. The default value is Boost_Gain_0dB.

The specific initialization code is as below.

```
static void voice_driver_init_codec(void)
{
    RCC_PeriphClockCmd(APBPeriph_CODEC, APBPeriph_CODEC_CLOCK, ENABLE);

    CODEC_InitTypeDef CODEC_InitStruct;
    CODEC_StructInit(&CODEC_InitStruct);

    /* set codec common parameters */
    CODEC_InitStruct.CODEC_SampleRate = voice_driver_codec_params.codec_sample_rate;
    CODEC_InitStruct.CODEC_MicType = voice_driver_codec_params.mic_type;

    if (voice_driver_codec_params.mic_type == CODEC_AMIC)
    {
        CODEC_InitStruct.CODEC_MicBIAS = voice_driver_codec_params.amic_bias_voltage;
        CODEC_InitStruct.CODEC_MICBstMode = voice_driver_codec_params.amic_input_mode;
        CODEC_InitStruct.CODEC_MICBstGain = voice_driver_codec_params.amic_analog_gain;
        CODEC_InitStruct.CODEC_AdGain = voice_driver_codec_params.amic_digital_gain;
    }
    else if (voice_driver_codec_params.mic_type == CODEC_AMIC)
    {
        CODEC_InitStruct.CODEC_DmicClock = voice_driver_codec_params.dmic_clock;
        CODEC_InitStruct.CODEC_DmicDataLatch = voice_driver_codec_params.dmic_data_latch;
        CODEC_InitStruct.CODEC_BoostGain = voice_driver_codec_params.dmic_boost_gain;
    }

    CODEC_InitStruct.CODEC_I2SFormat          = CODEC_I2S_DataFormat_I2S;
    CODEC_InitStruct.CODEC_I2SDataWidth       = CODEC_I2S_DataWidth_16Bits;
    CODEC_Init(CODEC, &CODEC_InitStruct);

#if SUPPORT_CODEC_EQ_CONFIG_FEATURE
    /* init codec EQ parameters */
    for (uint8_t eq_index = 0; eq_index < CODEC_EQ_MAX_NUM; eq_index++)
    {
        CODEC_EQInitTypeDef codec_eq_params;
        CODEC_EQStructInit(&codec_eq_params);
        if (voice_driver_codec_params.codec_eq_params[eq_index].CODEC_EQChCmd == EQ_CH_Cmd_ENABLE)
        {
            codec_eq_params.CODEC_EQCoefH0 = voice_driver_codec_params.codec_eq_params[eq_index].CODEC
_EQCoefH0;
            codec_eq_params.CODEC_EQCoefB1 = voice_driver_codec_params.codec_eq_params[eq_index].CODEC
_EQCoefB1;
            codec_eq_params.CODEC_EQCoefB2 = voice_driver_codec_params.codec_eq_params[eq_index].CODEC
_EQCoefB2;
            codec_eq_params.CODEC_EQCoefA1 = voice_driver_codec_params.codec_eq_params[eq_index].CODEC
_EQCoefA1;
            codec_eq_params.CODEC_EQCoefA2 = voice_driver_codec_params.codec_eq_params[eq_index].CODEC
_EQCoefA2;
        }
        CODEC_EQInit(CODEC_EQ_TABLE[eq_index], &codec_eq_params);
    }
#endif
}
```

13.2.4 Voice GDMA Initialization

Voice GDMA mainly transfers data in I2S FIFO to RCU APP. The specific initialization code is as below.

```
static void voice_driver_init_rx_gdma(void)
{
    /* Enable GDMA clock */
    RCC_PeriphClockCmd(APBPeriph_GDMA, APBPeriph_GDMA_CLOCK, ENABLE);

    /* Initialize GDMA peripheral */
    GDMA_InitTypeDef GDMA_InitStruct;
    GDMA_StructInit(&GDMA_InitStruct);
    GDMA_InitStruct.GDMA_ChannelNum        = GDMA_Channel_RX_NUM;
    GDMA_InitStruct.GDMA_DIR               = GDMA_DIR_PeripheralToMemory;
    GDMA_InitStruct.GDMA_BufferSize        = VOICE_GDMA_FRAME_SIZE / 4;
    GDMA_InitStruct.GDMA_SourceInc         = DMA_SourceInc_Fix;
    GDMA_InitStruct.GDMA_DestinationInc    = DMA_DestinationInc_Inc;
    GDMA_InitStruct.GDMA_SourceDataSize    = GDMA_DataSize_Word;
    GDMA_InitStruct.GDMA_DestinationDataSize = GDMA_DataSize_Byte;
    GDMA_InitStruct.GDMA_SourceMsize       = GDMA_Msize_4;
    GDMA_InitStruct.GDMA_DestinationMsize  = GDMA_Msize_16;
    GDMA_InitStruct.GDMA_SourceAddr        = (uint32_t)(&(I2S0->RX_DR));
    GDMA_InitStruct.GDMA_DestinationAddr   = (uint32_t)voice_driver_global_data.gdma_buffer.buf0;
    GDMA_InitStruct.GDMA_SourceHandshake   = GDMA_Handshake_SPORT0_RX;
    GDMA_InitStruct.GDMA_DestHandshake     = GDMA_Handshake_SPIC_RX;
    GDMA_Init(GDMA_Channel_RX, &GDMA_InitStruct);
    GDMA_INTConfig(GDMA_Channel_RX_NUM, GDMA_INT_Transfer, ENABLE);

    NVIC_InitTypeDef NVIC_InitStruct;
    NVIC_InitStruct.NVIC_IRQChannel = GDMA0_Channel_RX_IRQn;
    NVIC_InitStruct.NVIC_IRQChannelPriority = 3;
    NVIC_InitStruct.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStruct);

    GDMA_Cmd(GDMA_Channel_RX_NUM, ENABLE);
}
```

13.3 Voice Buffer Introduction

The Bee2 RCU voice module uses two kinds of buffer: GDMA ping-pong buffer and voice data loop buffer.

13.3.1 GDMA Ping-Pong Buffer

The GDMA ping-pong buffer is designed to let GDMA carry voice data from the I2S FIFO to the APP task. Ping-Pong buffer is a buffer of two GDMA buffer sizes, one of which is used as a GDMA transport destination cache, and the other is a data cache that GDMA interrupt to send message to APP task. After each GDMA interruption, alternately use them.

13.3.2 Voice Data Loop Buffer

Bee2 RCU designs a Loop Buffer of voice data for voice delay and instantaneous RF interference to store the compressed voice data.

The voice buffer queue is a buffer area with a continuous memory address. When the buffer area is full, the data continues to be stored over the old data. The voice buffer queue is a 7200 Bytes memory block by default that can store 900ms of voice data. The enqueue pointer is in_queue_index in figure 28 and the dequeue pointer is out_queue_index in figure 28. As for the following diagram, in_queue_index moves forward if enqueue, and out_queue_index moves forward if dequeue.

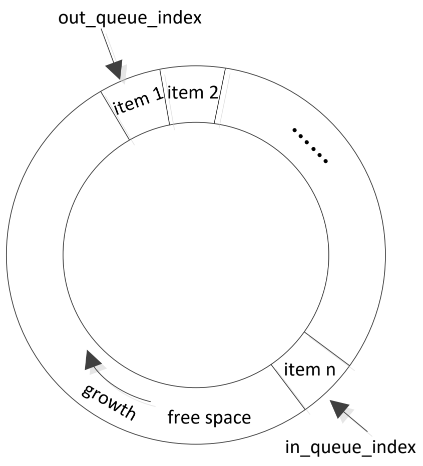


Figure 28 Voice Buffer Queue Diagram

13.4 Voice Encoding Algorithm

Currently, the sample rate of voice data is 16Ksample /s and bit resolution is 16 bit. The program uses voice compression encoding to improve Bluetooth transmission efficiency.

Bee2 RCU provides several methods of compression encoding: SW mSCB encoder, SW SBC encoder, SW IMA/DVI ADPCM encoder and SW 16:3 ADPCM encoder. Bee2 RCU can use the VOICE_ENC_TYPE macro defined in board.h to select the software encoding algorithm. The default is SW mSCB encoder.

```
/* voice encode type */
#define SW_MSBC_ENC     1  /* software msbc encode */
#define SW_SBC_ENC      2  /* software sbc encode */
#define SW_IMA_ADPCM_ENC 3  /* software IMA/DVI adpcm encode */
#define SW_OPT_ADPCM_ENC 4  /* software optimized adpcm encode */

#define VOICE_ENC_TYPE     SW_MSBC_ENC
```

13.5 Voice Interaction Flow

Bee2 RCU supports four voice interaction flows.

- IFLYTEK_VOICE_FLOW
- HIDS_GOOGLE_VOICE_FLOW
- ATV_GOOGLE_VOICE_FLOW
- RTK_GATT_VOICE_FLOW

Bee2 RCU can use the VOICE_FLOW_SEL macro defined in board.h to select the voice interaction flow.

13.5.1 IFLYTEK VOICE FLOW

IFLYTEK voice interaction flow is based on HID Service, and the voice interaction flow between RCU and the host is as below.

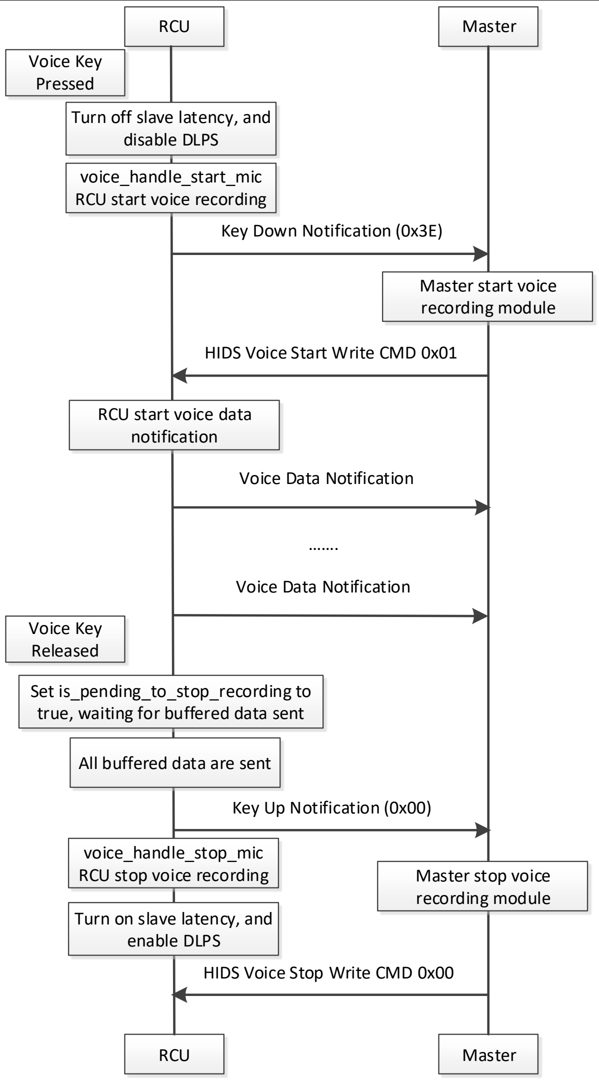

RCU
Master
Voice Key Pressed
Turn off slave latency, and disable DLPS
voice_handle_start_mic RCU start voice recording
Key Down Notification (0x3E)
Master start voice recording module
HIDS Voice Start Write CMD 0x01
RCU start voice data notification
Voice Data Notification
....
Voice Data Notification
Voice Key Released
Set is_pending_to_stop_recording to true, waiting for buffered data sent
All buffered data are sent
Key Up Notification (0x00)
voice_handle_stop_mic RCU stop voice recording
Master stop voice recording module
Turn on slave latency, and enable DLPS
HIDS Voice Stop Write CMD 0x00
RCU
Master

Figure 29 IFLYTEK_VOICE_FLOW

13.5.2 HIDS Google VOICE FLOW

The HIDS Google Voice interaction flow is similar to the IFLYTEK speech interaction flow. Both of them are based on HID Service. The difference is as below.

1. Voice Data Notification does not need to send after the host has received the ***voice start write*** cmd. Start to send Voice Data Notification directly after Voice Start Notification by default.

2. Two ways to end the voice.
   1) After the voice data transmission, the host-side voice recognition server will automatically determine

whether the current voice flow is completed. If it is completed, then ***voice stop write*** cmd will be used to stop the RCU voice recording.

2) If the user releases the voice key before receiving the voice stop write cmd from the host, stop the voice recording actively and send a Voice Stop Notification to notify the host to end the voice after waiting for the voice buffer data to be sent.

The voice interaction flow between the RCU and the host is as below.

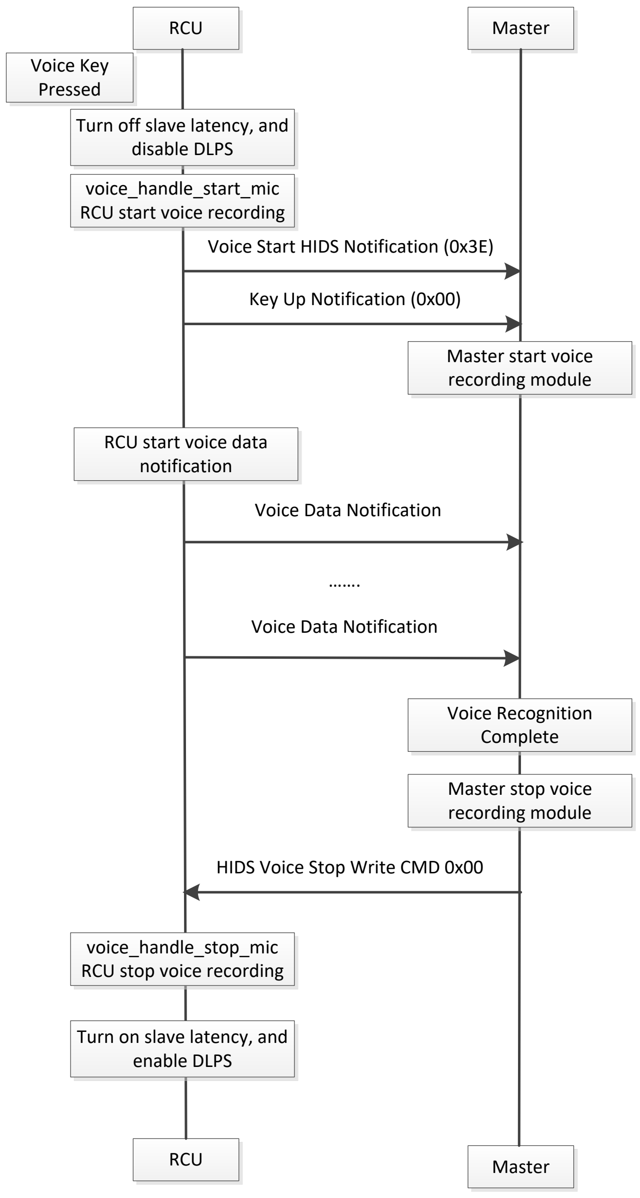

RCU
Master
Voice Key Pressed
Turn off slave latency, and disable DLPS
voice_handle_start_mic RCU start voice recording
Voice Start HIDS Notification (0x3E)
Key Up Notification (0x00)
Master start voice recording module
RCU start voice data notification
Voice Data Notification
....
Voice Data Notification
Voice Recognition Complete
Master stop voice recording module
HIDS Voice Stop Write CMD 0x00
voice_handle_stop_mic RCU stop voice recording
Turn on slave latency, and enable DLPS
RCU
Master

Figure 30 HIDS Google VOICE FLOW – 1

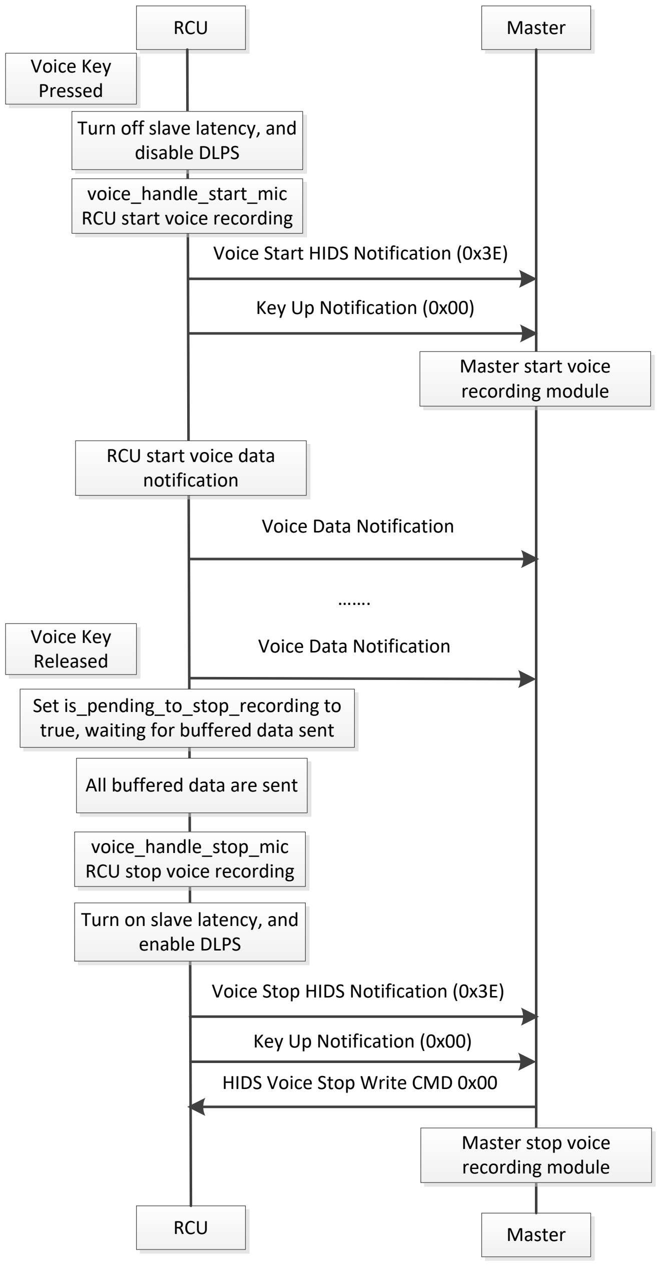

RCU
Master
Voice Key Pressed
Turn off slave latency, and disable DLPS
voice_handle_start_mic RCU start voice recording
Voice Start HIDS Notification (0x3E)
Key Up Notification (0x00)
Master start voice recording module
RCU start voice data notification
Voice Data Notification
....
Voice Key Released
Voice Data Notification
Set is_pending_to_stop_recording to true, waiting for buffered data sent
All buffered data are sent
voice_handle_stop_mic RCU stop voice recording
Turn on slave latency, and enable DLPS
Voice Stop HIDS Notification (0x3E)
Key Up Notification (0x00)
HIDS Voice Stop Write CMD 0x00
Master stop voice recording module
RCU
Master

Figure 31 HIDS Google VOICE FLOW – 2

13.5.3 ATV Google VOICE FLOW

Starting from Android version 8.0, Google has defined a set of standard Voice Solutions, which are based on Google's standard ATV Voice Service for voice transmission. The latest spec is draft V0.5.

RCU uses the ATV Voice Service for voice data interaction. See Google ATV Voice documentation for details. The ATV Voice flow of Bee2 RCU basically follows the above spec. On this basis, the efficiency of speech transmission is optimized. RCU will actively initiate the request of the MTU Exchange after connecting and pairing. Then use the larger MTU size for voice interaction, and the Notification size of the Voice Data is 134bytes.

The voice interaction flow between the RCU and the host is as below.

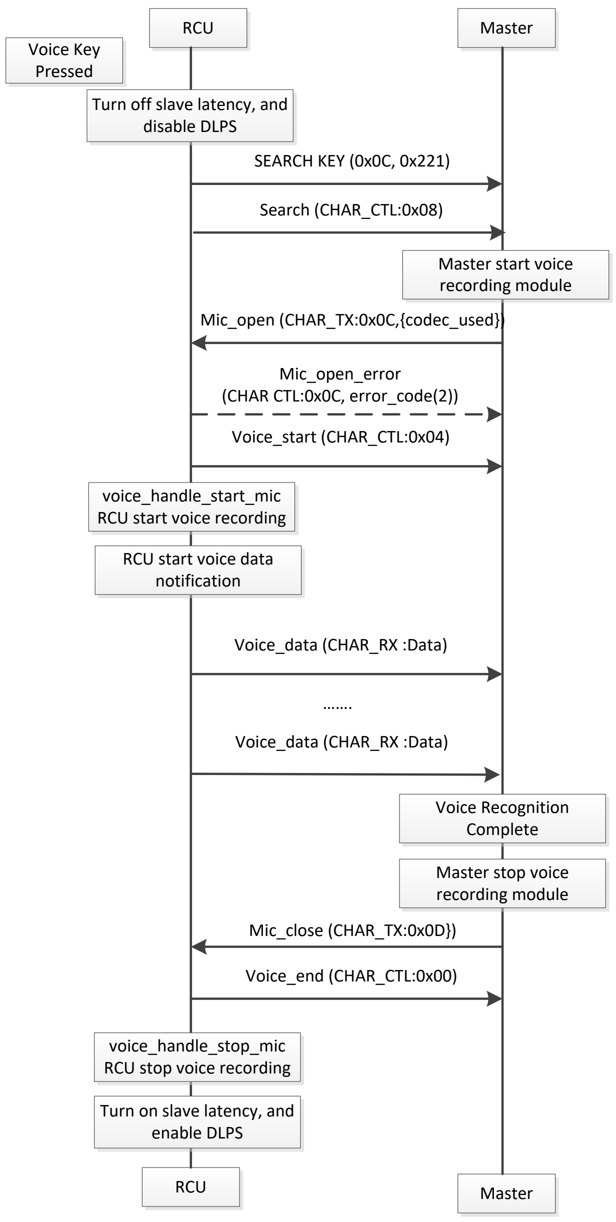

RCU
Master
Voice Key Pressed
Turn off slave latency, and disable DLPS
SEARCH KEY (0x0C, 0x221)
Search (CHAR_CTL:0x08)
Master start voice recording module
Mic_open (CHAR_TX:0x0C,{codec_used})
Mic_open_error (CHAR CTL:0x0C, error_code(2))
Voice_start (CHAR_CTL:0x04)
voice_handle_start_mic RCU start voice recording
RCU start voice data notification
Voice_data (CHAR_RX :Data)
....
Voice_data (CHAR_RX :Data)
Voice Recognition Complete
Master stop voice recording module
Mic_close (CHAR_TX:0x0D})
Voice_end (CHAR_CTL:0x00)
voice_handle_stop_mic RCU stop voice recording
Turn on slave latency, and enable DLPS
RCU
Master

Figure 32 ATV Google VOICE FLOW

13.5.4 RTK GATT VOICE FLOW

RTK Voice interaction process is based on Voice Service. Please refer to AI headset Voice transmission protocol (v0.3) for relevant content of Voice Service.

The voice interaction process between the RCU and the host is as follows:

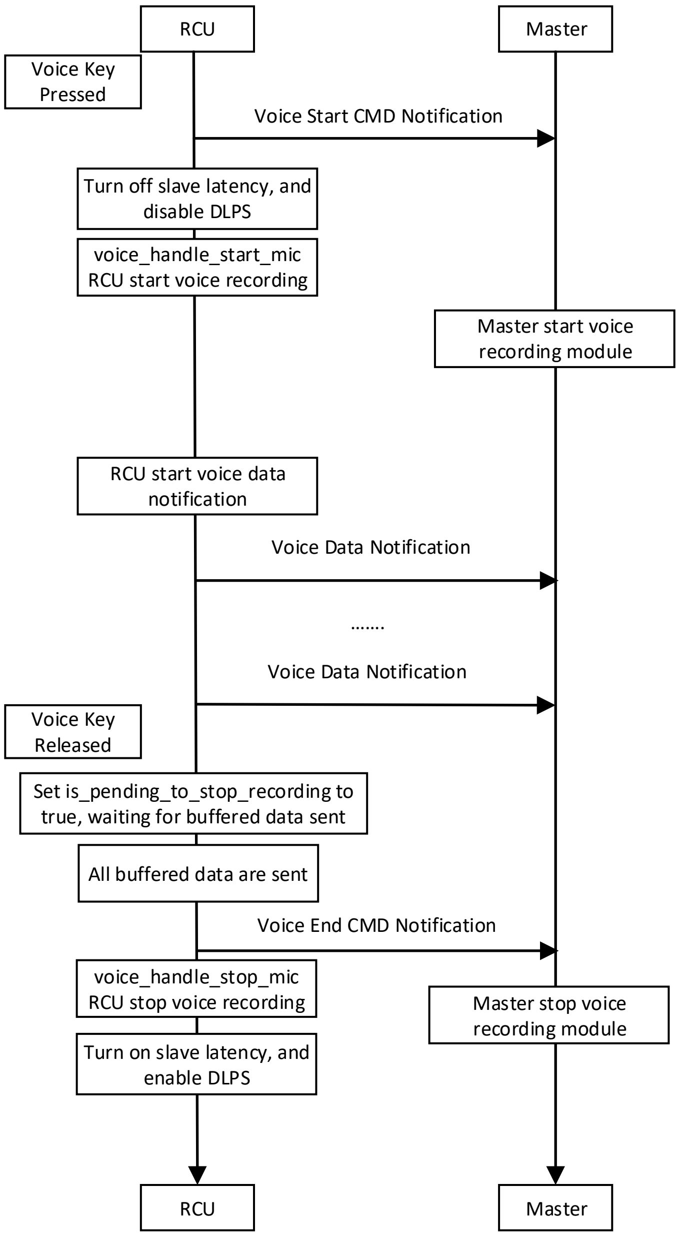

RCU
Master
Voice Key Pressed
Voice Start CMD Notification
Turn off slave latency, and disable DLPS
voice_handle_start_mic RCU start voice recording
Master start voice recording module
RCU start voice data notification
Voice Data Notification
....
Voice Data Notification
Voice Key Released
Set is_pending_to_stop_recording to true, waiting for buffered data sent
All buffered data are sent
Voice End CMD Notification
voice_handle_stop_mic RCU stop voice recording
Master stop voice recording module
Turn on slave latency, and enable DLPS
RCU
Master

Figure 33 RTK GATT VOICE FLOW

14IR

The RCU SDK supports IR function including IR learning and IR transmission.IR learning supports 6 keys learning and storage IR learning wave data. IR can send the learned IR wave data, or send user-defined IR local data and address. If there is a learning wave data in the FTL memory, the IR learning wave data will be sent first. If the local data is invalid or there is no IR learning data, the preset infrared data will be sent.

14.1 IR Module Configuration

If the IR module works, it needs to turn on the control macro IR_FUN that defines the IR module in board.h file. IR learning function can be opened with micro IR_LEARN_MODE enable, when IR_FUN is already turned on.

```
/*********************************************************
*             IR Module Config
*********************************************************/
#define IR_FUN        0

#if  IR_FUN

/*IR  send  config*/
#define  IR_SEND_PIN                  P2_3

#define  IR_SEND_WAVE_MAX_SIZE     70

/*IR  repeat  code  period  config  */
#define  IR_REPEAT_CODE_PERIOD  108   /*  108  ms,  unit  ms  */

/*IR  learn  module  config*/
#define  IR_LEARN_MODE       0

#if  IR_LEARN_MODE

#define  IR_LEARN_PIN                 P2_2
#endif

/*  IR_LEARN_TRIG_MODE  need  to  be  configured  according  to  hardware  */
#define  IR_LEARN_TRIG_RISING_EDGE  0
#define  IR_LEARN_TRIG_FALL_EDGE     1
#define  IR_LEARN_TRIG_MODE                 IR_LEARN_TRIG_FALL_EDGE
```

```

/*max key number can be storged*/
#define  IR_LEARN_MAX_KEY_NUM          0x06
#define  IR_WAVE_CONFIG_BASE_ADDR      IR_MODULE_FTL_PARAMS_BASE_ADDR
#define  IR_WAVE_DATA_BASE_ADDR        (IR_WAVE_CONFIG_BASE_ADDR + sizeof(uint32_t)/*config CRC value*/
+ \
                                      sizeof(IR_LearnStorgeInfo)*IR_LEARN_MAX_KEY_NUM)
#define  IR_LEARN_WAVE_MAX_SIZE        IR_SEND_WAVE_MAX_SIZE

#endif
#endif
```

For now, the beta version of the RCU SDK IR learning example is an IR protocol with wave data length below 70 data, which can be adjusted through micro IR_LEARN_WAVE_MAX_SIZE. IR learning wave data is stored in FTL memory, and the storage base address of IR wave data can be adjusted by macros IR_WAVE_CONFIG_BASE_ADDR and IR_WAVE_DATA_BASE_ADDR, but be careful not to exceed the memory size of the FTL. The storage format of the data is the configuration information of IR learning data and CRC check information. Infrared data storage is the beginning of address IR_WAVE_DATA_BASE_ADDR. When IR learning data is sent, it reads the specified infrared data from the stored address and sends it directly.

14.2 IR module operation

The entry of IR learning mode is triggered by a combination of keys. Keep pressing buttons ENTER and LEFT for 2s will enter the IR learn mode. At this time, the indicator lights will constantly blink, indicating the infrared learning state, and the timeout timer will start to start timing. It will exit the IR learning state after the 20s timeout, the indicator light will also exit the blinking state. In the IR learning mode, RCU needs to keep a single button pressed to capture input IR wave date. If the IR learning is successful, the indicator turns from constant to rapid flashing several times, then extinguishes.

If you want to exit from the IR learning mode quickly, you can long press the same combination buttons directly in this mode, and LED will exit from the flicker to other states.

15 OTA

RCU device supports Realtek vendor OTA protocol through OTA service and DFU service. The OTA on RCU device has two types, normal OTA and silent OTA. Normal OTA needs RCU reboot to enter OTA mode, in which RCU focus on OTA flow and RCU keeps in unused state. After OTA updated successfully, new RCU image can work. Silent OTA can be executed in RCU normal state. When OTA updated successfully, RCU device will reboot and run new image.

For OTA's detailed principles, please refer to the document of "RTL8762C OTA User Manual"

15.1 Normal OTA

Normal OTA needs OTA service only in application layer to work. Peer device writes specific command to allow RCU device to enter OTA mode. OTA mode is a section of Independent code on the rom, so the peer device needs to search for RCU device in OTA mode and establish connection, then executes OTA flow according to OTA service information. After new image updated successfully, RCU device reboots and executes new image code.

OTA service UUID is as follows.

```
const uint8_t GATT_UUID_OTA_SERVICE[16] = { 0x12, 0xA2, 0x4D, 0x2E, 0xFE, 0x14, 0x48, 0x8e, 0x93, 0xD2, 0x17, 0x3C, 0
xFF, 0xD0, 0x00, 0x00};
```

OTA Service includes the following Characteristics.

```
#define GATT_UUID_CHAR_OTA                    0xFFD1
#define GATT_UUID_CHAR_MAC                    0xFFD2
#define GATT_UUID_CHAR_PATCH                  0xFFD3
#define GATT_UUID_CHAR_APP_VERSION            0xFFD4
#define GATT_UUID_CHAR_DEVICE_INFO            0xFFF1
#define GATT_UUID_CHAR_IMAGE_VERSION          0xFFE0
```

0xFFD1 is used for OTA command, Write no response. Peer device writes 0x01 data into 0xFFD1, and then RCU device reboots and enters normal OTA mode.

0xFFD2 is used to read MAC address (BD Address) by peer device.

0xFFD3 is used to read Patch version by peer device.

0xFFD4 is used to read application version by peer device.

0xFFF1 is used to read device information by peer device, such as AES encryption and buffer check.

0xFFE0 is used to read device RCU image address and versions by peer device.

15.2 Silent OTA

Silent OTA is a way to update when the remote controller is in normal state, and it requires to be supported in both OTA service and DFU service. Before updating, peer devices read RCU device information through OTA service; when updating, peer devices can control updating process and data transmission through DFU service.

DFU service UUID is as follows：

```
const uint8_t   SILENCE_GATT_UUID128_DFU_SERVICE[16] = {0x12, 0xA2, 0x4D, 0x2E, 0xFE, 0x14, 0x48, 0x8e, 0x93, 0xD2
, 0x17, 0x3C, 0x87, 0x62, 0x00, 0x00};
```

DFU service has two characteristics, data characteristic and Control point characteristic. Data Characteristic, responsible for data transmission, is writable attributes. Control Point characteristic, responsible for OTA time series control, is writable and notification attributes.

The UUID of the two attributes is as follows.

```
#define  GATT_UUID128_DFU_PACKET  0x12, 0xA2, 0x4D, 0x2E, 0xFE, 0x14, 0x48, 0x8e, 0x93, 0xD2, 0x17, 0x3C, 0x87, 0x63, 0x00, 0x00
#define  GATT_UUID128_DFU_CONTROL_POINT  0x12, 0xA2, 0x4D, 0x2E, 0xFE, 0x14, 0x48, 0x8e, 0x93, 0xD2, 0x17, 0x3C, 0x87, 0x64, 0x00, 0x00
```

16 Battery Voltage Module

Bee2 RCU supports detecting battery voltage and low-power mode protection. After pairing, master device can read Bee2 RCU battery voltage or battery level via BLE BAS service.

16.1 Battery Module Configuration

Bee2 RCU SDK use macros in board.h to set battery module related configuration. BAT_LOW_POWER_THRESHOLD and BAT_ADC_DETECT_THRESHOLD can set the threshold for LPC and ADC battery low power voltage. BAT_LOW_POWER_INDICATE is used to control LED behavior in low power mode. BAT_LPC_EN is used to choose whether or not to use the LPC hardware module. BAT_LPC_CONFIG is used to set the voltage threshold of the LPC comparator.

```
/*******************************************************
*          Battery Module Config
*******************************************************/
#define BAT_EN      1

#if BAT_EN

#define BAT_LPC_EN      0 /* 1 enable lpc, 0 disable */
#if BAT_LPC_EN
#define BAT_LPC_CONFIG                  LPC_2000_mV /*lpc detect bat threshold value*/
#endif

#define BAT_LOW_POWER_THRESHOLD      2000 /*2.0v threshold for rcu system*/
#define BAT_ADC_DETECT_THRESHOLD     (BAT_LOW_POWER_THRESHOLD + 200)
#define BAT_LOW_POWER_INDICATE       1 /*1 led indicate enable, 0 disable */

#endif
```

16.2 Battery Module Workflow

Low power mode can be triggered by LPC hardware module, or by button mode, and the default way is button mode. The default threshold of low power is 2.0v. If the low power mode is triggered by LPC hardware module, when the power is higher than this threshold, RCU is in normal working state. When the voltage is lower than this threshold, LPC interruption will be triggered to perform a low power detection operation. If low power mode is triggered by key, VBAT_DETECT_KEY_CNT times key press will trigger a low-power detection operation. In the low-voltage detection operation, ADC will conduct accurate voltage sampling, and remove the maximum and minimum values for 4 times, then average the remaining values. If the average value falls below the set threshold, the system is required to enter low power mode.

In Low Power mode, Bee2 RCU will keep BLE status in IDLE and stop running IO modules except Keyscan module. Any key pressed action will trigger Bee2 RCU to detect battery voltage again. If battery voltage is above BAT_ADC_DETECT_THRESHOLD, Bee2 RCU will return to normal mode. This mechanism is designed to handle unexpected scenario, such as if user changes new battery very quickly.

17 LED

The SDK supports a multi-function LED module. The module supports simultaneous flashing of multiple LEDs, and supports multiple events of a single LED flashing by priority.

17.1 LED Configuration

```
/*******************************************************
*          LED Module Config
*******************************************************/

#define  LED_EN       1

#if  LED_EN

#define   LED_NUM_MAX     0x01
#define   LED_INDEX(n)     (n<<8)
/*uint16_t, first byte led index, last byte led pin*/
#define   LED_1               (LED_INDEX(0) | P2_4)

/* voltage level to trigger LED On action */
#define  LED_ON_LEVEL_HIGH  0
#define  LED_ON_LEVEL_LOW    1

#define  LED_ON_LEVEL_TRIG  LED_ON_LEVEL_HIGH

#endif
```

The entire LED module is controlled by the macro LED_EN. The number of LED is controlled by LED_NUM_MAX, which currently supports one by default. LED_ON_LEVEL_TRIG is used to select the corresponding driving elecrtical level state when LED is on.

17.2 LED Interface

There are four interfaces for LED, which are defined in led_driver.h. These include LED_ON, LED_OFF, LED_BLINK, and LED_BLINK_EXIT.

```
#if LED_EN
#define LED_ON(index)              led_blink_start(index, LED_TYPE_ON, 0)
#define LED_OFF(index)             led_blink_exit(index, LED_TYPE_ON)
#define LED_BLINK(index, type, n)      led_blink_start(index, type, n)
#define LED_BLINK_EXIT(index, type)    led_blink_exit(index, type)
#else
#define LED_ON(index)
#define LED_OFF(index)
#define LED_BLINK(index, type, n)
#define LED_BLINK_EXIT(index, type)
#endif
```

LED_ON and LED_OFF mainly indicate the status of the key.

The two macros LED_BLINK and LED_BLINK_EXIT mainly control the blinking and exit of the LED. LED_BLINK has three parameters. The first is the index of the LED, which is defined in board.h. The second parameter is the type of LED flashing, which is defined in led_driver.h. The third is the number of LED flashes. If you fill in 0, flashing does not stop. If you fill in the specified value, the number of flashes depends on the data you filled in. The maximum number of flashes is 255.

Currently there are 7 types of flashes added by default, distributed in four modules: infrared, low battery, Bluetooth and buttons. If you need to add a new LED type, you need to add three places. One is the LED type. One is the LED blinking bitmap and the last is to add the execution branch in led_driver.c.

18 MP Test

Bee2 RCU supports MP test modes, including: HCI UART Test Mode, Data UART Test Mode, Single Tone Test Mode, and Fast Pair Test Mode. Please refer to document "RTL8752C/RTL8762C RCU MP Test Mode Design Spec" for more information.

In Bee2 RCU SDK, the following macros are defined to support MP 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_SUPPORT_AUTO_K_RF                 0   /*  set  1  to  support  Auto  K  RF  */
#define  MP_TEST_MODE_SUPPORT_DATA_UART_DOWNLOAD  0   /*  set  1  to  support  Data  UART  download  */

#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_K
EYS)
```

Bee2 RCU supports switching to MP test mode from normal mode by Test Mode AON register and Watchdog reboot. The flow of switching to MP test mode is as follows:

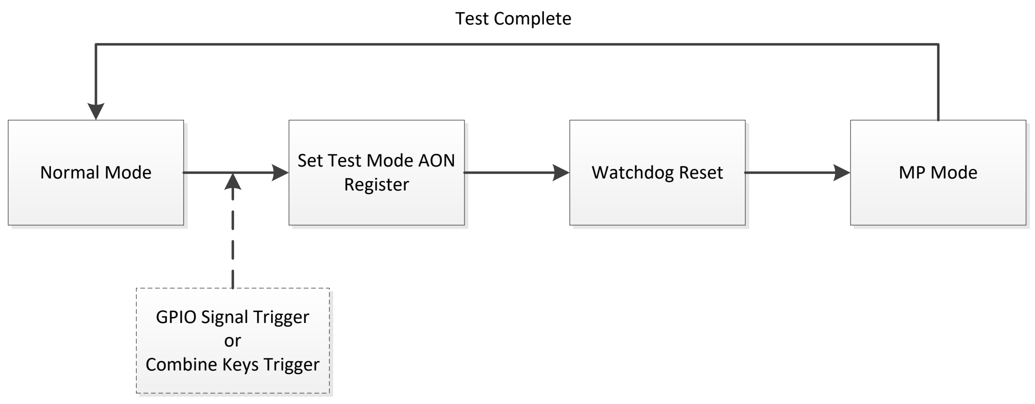


Figure 34 Switching to MP test mode flow

Bee2 RCU SDK provides sample of MP flow, including: Image Download, PCBA Level Test and Product Function Test. Please refer to document "RTL8752C/RTL8762C RCU MP Test Sample Flow" for more information.

19 The Reference

[1] RTL8762C OTA User Manual

[2] RTL8762C RCU MP Test Mode Design Spec

[3] RTL8762C RCU MP Test Sample Flow

[4] voice transmission protocol for AI headphones (v0.3)