SIG BLE Mesh and Zigbee

Dual Mode Introduction

AN-20011600-E1

Ver.0.2.2

2020/02/18

Keyword

SIG BLE MESH, ZigBee, Dual Mode, Concurrent, Switching

Published by

Telink Semiconductor

Bldg 3, 1500 Zuchongzhi Rd,

Zhangjiang Hi-Tech Park, Shanghai, China

Legal Disclaimer

This document is provided as-is. Telink Semiconductor reserves the right to make improvements without further notice to this document or any products herein. This document may contain technical inaccuracies or typographical errors. Telink Semiconductor disclaims any and all liability for any errors, inaccuracies or incompleteness contained herein.

Information:

For further information on the technology, product and business term, please contact Telink Semiconductor Company (www.telink-semi.com).

For sales or technical support, please send email to the address of:

telinkcnsales@telink-semi.com

telinkcnsupport@telink-semi.com

Revision History

Version 0.2.4 (2021-01-06)

Section	Change Description
2.1.1	Add ZigBee configuration parameter
2.1.2	Update selected type of “MESH_USER_DEFINE_MODE”
3.1	Update flash map to normal mode

Version 0.2.3 (2020-03-27)

Section	Change Description
1.1	Updated section 1.1 Configuration Parameter.
3.1	Updated section 3.1 Flash Map.

Version 0.2.2 (2020-02-18)

Section	Change Description
1.1, 1.2	Updated section 1.1 Bootloader Flow with OTA Check and section 1.2 Dual Mode Selection Flow.
2.8	Updated section 2.8 OTA Flow.
3.1	Updated section 3.1 Flash Map.

Updated the formatting of the document including adding titles to figures and tables.

Version 0.1.2 (2020-02-05)

Updated the formatting of the document.

Version 0.1.1 (2020-02-04)

Section	Change Description
1.1	Added Figure 1-1 Bootloader Flow with OTA Check.
1.4.5	Added section 1.4.5 Zigbee/BLE Concurrent.
2.8	Added Figure 2-1 OTA Flow Chart.
3	Added chapter 3 Flash and Memory Usage Information.

Version 0.1.0 (2020-01-16)

This is the Initial release.

Contents

Revision History 2

Contents........ 4

Contents of Figures 6

Contents of Tables 7

1. Dual Mode Introduction 8

1.1 Bootloader Flow with OTA Check 8

1.2 Dual Mode Selection Flow 9

1.3 Dual Mode Recover Flow........ 14

1.3.1 Reset to Dual Mode Selection 14

1.3.2 In BLE Mode, Reset to Dual Mode Selection Flow........ 14

1.3.3 In Zigbee Mode, Reset to Dual Mode Selection Flow 15

1.4 Zigbee Network Detection Introduction........ 16

1.4.1 Flow-Chart of Beacon Detection........ 16

1.4.2 Condition of the Valid Beacon........ 17

1.4.3 Beacon-Request Format 18

1.4.4 Beacon Format 18

1.4.5 Zigbee/BLE Concurrent........ 18

2. Instruction of Misc. 20

2.1 Configuration Parameter 20

2.1.1 ZigBee configuration parameter........ 20

2.1.2 Mesh and BootLoader configuration parameter 20

2.2 LED Indication........ 22

2.3 Mesh APP Network Creation (None Remote Provision)........ 23

2.4 Factory Reset........ 23

2.5 Switch to Zigbee Mode........ 23

2.6 Remote Provision....24

2.7 Mesh OTA 25

2.8 OTA Flow....27

3. Flash and Memory Usage Information....29

3.1 Flash Map....29

3.2 Firmware Flash and SRAM Usage 31

3.2.1 SIG Mesh and Zigbee/BLE Image Size 31

3.2.2 SIG Mesh Flash Major Module Resource Information 32

3.2.3 Zigbee_BLE Flash Resource Information 32

4. SIG MESH Features and Configurations 33

4.1 How Many Buffers Would Proxy Role Support？ 33

4.2 How Many Replay Protections? 33

4.3 Number of Lightbulbs....33

4.4 Switching Time Overhead....33

Contents of Figures

Figure 1-1 Bootloader Flow with OTA Check........8

Figure 1-2 Dual Mode Selection Flow........9

Figure 1-3 Dual Mode Recover Flow........14

Figure 1-4 Zigbee Network Detection Flow........16

Figure 1-5 Flow-Chart of Beacon Detection........16

Figure 1-6 Condition of the Valid Zigbee Beacon........17

Figure 1-7 Beacon-Request Format........18

Figure 1-8 Beacon Format........18

Figure 1-9 BLE/ZigBee Stack Architecture........19

Figure 1-10 BLE/ZigBee Band Sharing Timing Chart........19

Figure 2-1 OTA Flow Chart........28

Figure 3-1 Flash Map........29

Contents of Tables

Table 3-1 SIG Mesh and Zigbee/BLE Image Size 31

Table 3-2 SIG Mesh Flash Major Module Resource Information 32

Table 3-3 Zigbee_BLE Flash Resource Information 32

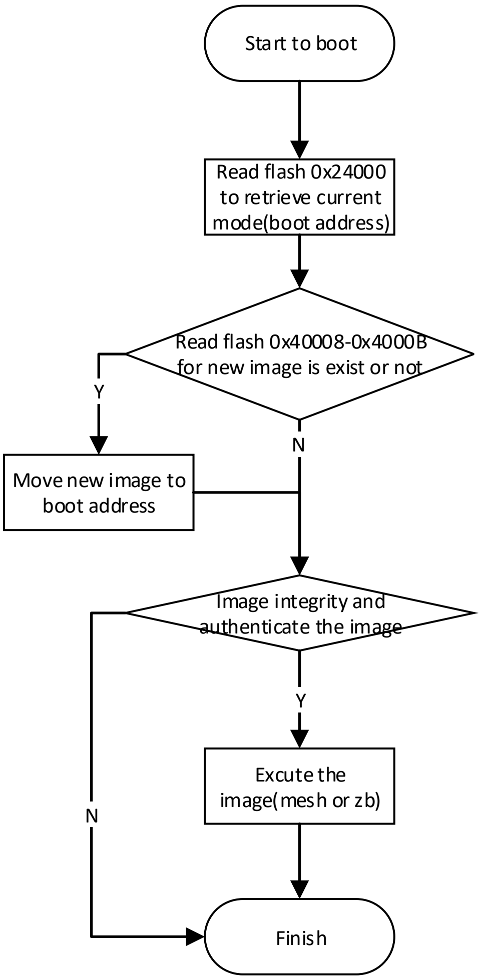
1. Dual Mode Introduction

This document is used to introduce the Telink SIG BLE MESH and Zigbee provisioning concurrent solution, including the design and concepts, it also provides guidance to modify some key parameters to adapt to your application scenarios.

In the following context, you'll learn how the Telink SIG BLE MESH and Zigbee provisioning concurrent design concepts and the details of how the concurrent mode switching decision making.

1.1 Bootloader Flow with OTA Check

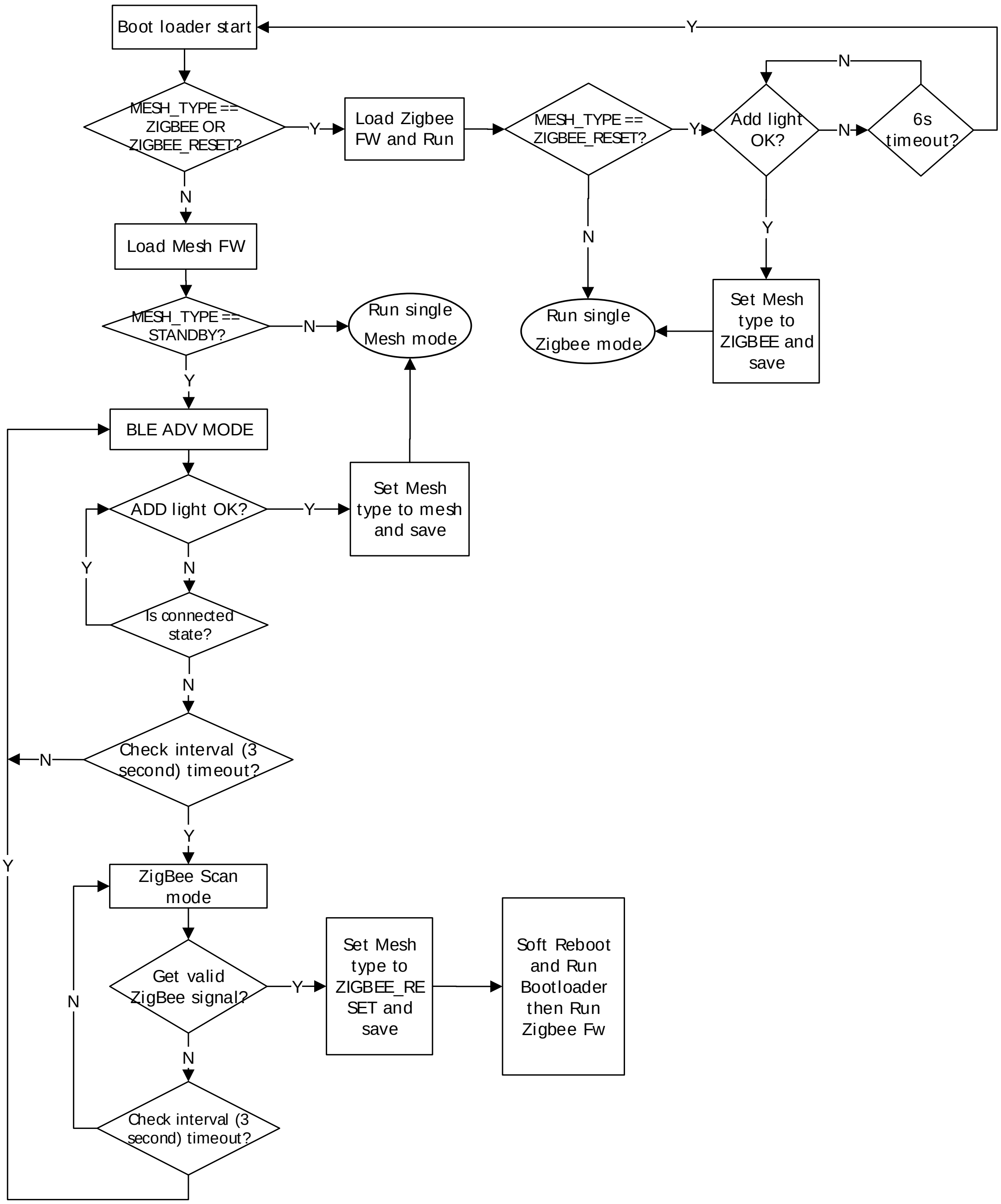
Figure 1-1 Bootloader Flow with OTA Check


Start to boot
Read flash 0x24000 to retrieve current mode(boot address)
Read flash 0x40008-0x4000B for new image is exist or not
Y
N
Move new image to boot address
Image integrity and authenticate the image
Y
N
Excute the image(mesh or zb)
Finish

Note: *In SDK demo, it only did firmware integrity check, but no authentication.*

1.2 Dual Mode Selection Flow

Figure 1-2 Dual Mode Selection Flow


Boot loader start
MESH_TYPE == ZIGBEE OR ZIGBEE_RESET?
Load Zigbee FW and Run
MESH_TYPE == ZIGBEE_RESET?
Add light OK?
6s timeout?
Load Mesh FW
MESH_TYPE == STANDBY?
Run single Mesh mode
Run single Zigbee mode
Set Mesh type to ZIGBEE and save
BLE ADV MODE
ADD light OK?
Set Mesh type to mesh and save
Is connected state?
Check interval (3 second) timeout?
ZigBee Scan mode
Get valid ZigBee signal?
Set Mesh type to ZIGBEE_RESET and save
Soft Reboot and Run Bootloader then Run Zigbee Fw
Check interval (3 second) timeout?
Y
N

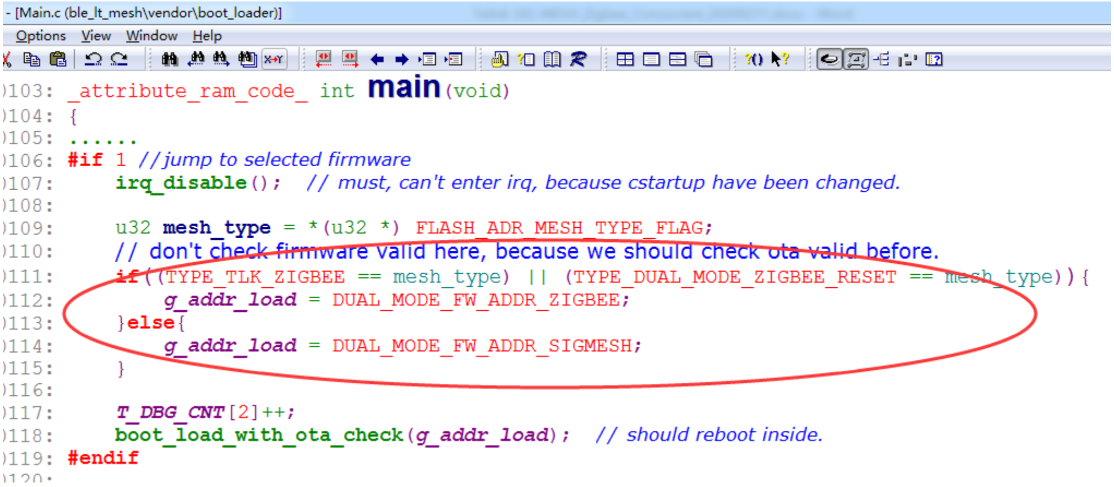
“Bootloader start”: Run 8258_bootloader.bin at flash 0x00000, and will read the Image Type from offset 0x24000 to decide which type of image/offset should continue.

Following are the definition of the image type definition, you can find the definition from

${SDK_WORKSPACE}/proj_lib/ble/blt_config.h

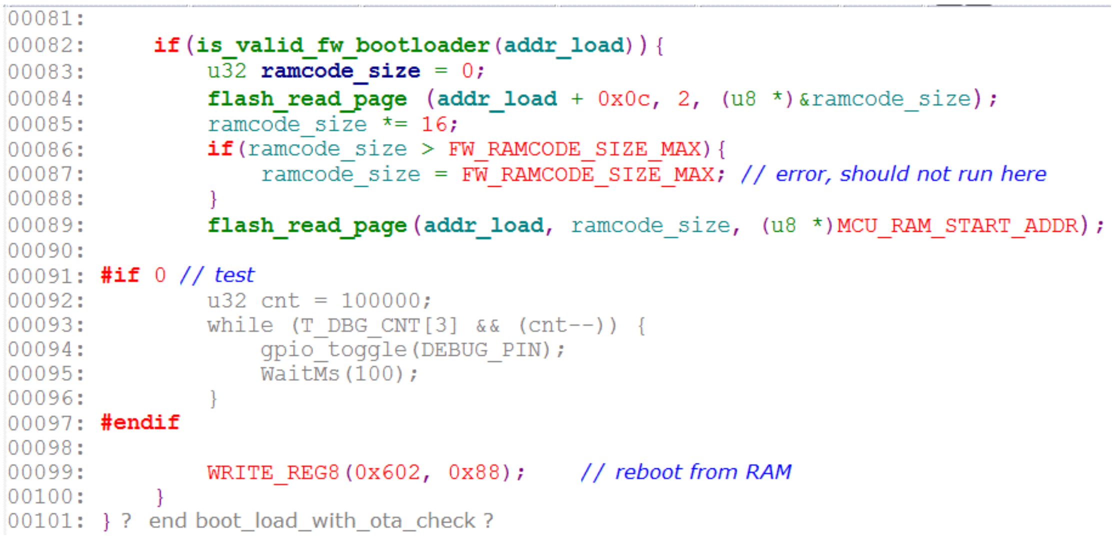
```
    enum{
   TYPE_TLK_MESH                   = 0x000000A3,
   TYPE_SIG_MESH                   = 0x0000003A,
   TYPE_TLK_BLE_SDK                = 0x000000C3,
   TYPE_TLK_ZIGBEE                 = 0x0000003C,
   TYPE_DUAL_MODE_STANDBY     = 0x00000065, // dual mode switch mode
   TYPE_DUAL_MODE_RECOVER     = 0x00000056, // recover from ZigBee.
   TYPE_DUAL_MODE_ZIGBEE_RESET    = 0x00000053, // switch to ZigBee
};
```

Below is the snapshot of the bootloader where it read the image type flag and check the type of the image and set the corresponding offset to load the image.



```
_attribute_ram_code_ int main(void)
{
......
#if 1 //jump to selected firmware
    irq_disable();  // must, can't enter irq, because cstartup have been changed.

    u32 mesh_type = *(u32 *) FLASH_ADR_MESH_TYPE_FLAG;
    // don't check firmware valid here, because we should check ota valid before.
    if((TYPE_TLK_ZIGBEE == mesh_type) || (TYPE_DUAL_MODE_ZIGBEE_RESET == mesh_type)){
        g_addr_load = DUAL_MODE_FW_ADDR_ZIGBEE;
    }else{
        g_addr_load = DUAL_MODE_FW_ADDR_SIGMESH;
    }

    T_DBG_CNT[2]++;
    boot_load_with_ota_check(g_addr_load);  // should reboot inside.
#endif
```

“Load mesh FW”: \vendor\boot_loader\main.c -- boot_load_with_ota_check()



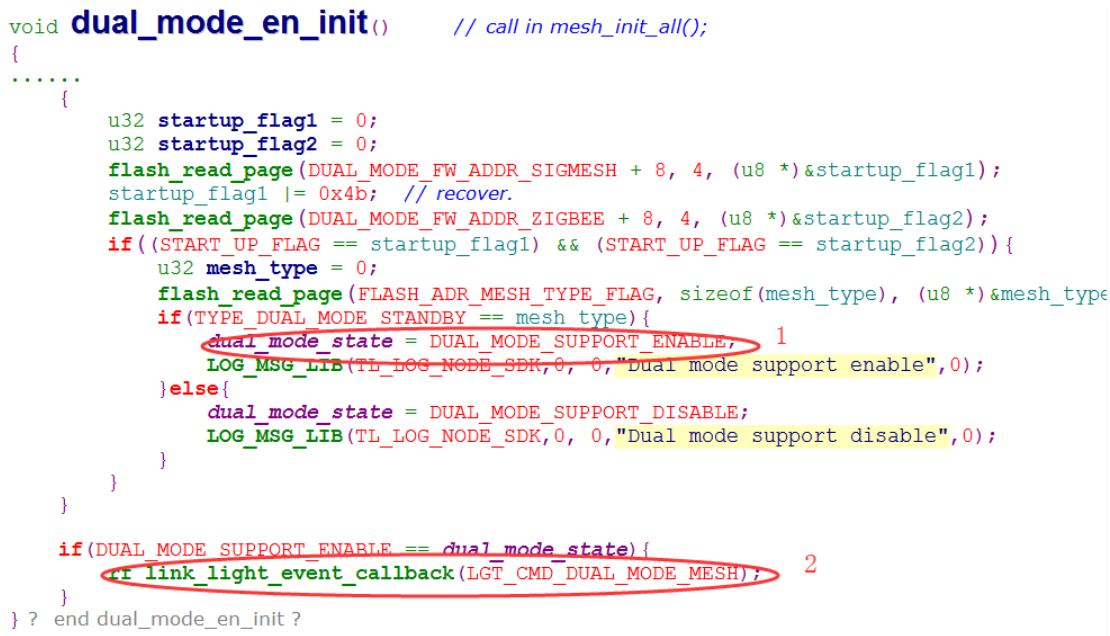
```

    if(is_valid_fw_bootloader(addr_load)){
        u32 ramcode_size = 0;
        flash_read_page (addr_load + 0x0c, 2, (u8 *)&ramcode_size);
        ramcode_size *= 16;
        if(ramcode_size > FW_RAMCODE_SIZE_MAX){
            ramcode_size = FW_RAMCODE_SIZE_MAX; // error, should not run here
        }
        flash_read_page(addr_load, ramcode_size, (u8 *)MCU_RAM_START_ADDR);

#if 0 // test
        u32 cnt = 100000;
        while (T_DBG_CNT[3] && (cnt--)) {
            gpio_toggle(DEBUG_PIN);
            WaitMs(100);
        }
#endif

        WRITE_REG8(0x602, 0x88);    // reboot from RAM
    }
} ? end boot_load_with_ota_check ?
```

“MESH_TYPE == STANDBY ?”

TYPE_DUAL_MODE_STANDBY

After load mesh firmware: During the first power cycle, the call flow is user_init()->proc_telink_mesh_to_sig_mesh(), and then firmware will set the boot type as TYPE_DUAL_MODE_STANDBY

From user_init()->dual_mode_en_init(), it’ll validate if device is has ZigBee firmware, if yes, the dual_mode_state is set to DUAL_MODE_SUPPORT_ENABLE, it’ll flash the LED light once.



```
void dual_mode_en_init()       // call in mesh_init_all();
{
......
    {
        u32 startup_flag1 = 0;
        u32 startup_flag2 = 0;
        flash_read_page(DUAL_MODE_FW_ADDR_SIGMESH + 8, 4, (u8 *)&startup_flag1);
        startup_flag1 |= 0x4b;  // recover.
        flash_read_page(DUAL_MODE_FW_ADDR_ZIGBEE + 8, 4, (u8 *)&startup_flag2);
        if((START_UP_FLAG == startup_flag1) && (START_UP_FLAG == startup_flag2)){
            u32 mesh_type = 0;
            flash_read_page(FLASH_ADR_MESH_TYPE_FLAG, sizeof(mesh_type), (u8 *)&mesh_type
            if(TYPE_DUAL_MODE_STANDBY == mesh_type){
                dual_mode_state = DUAL_MODE_SUPPORT_ENABLE;
                LOG_MSG_LIB(TL_LOG_NODE_SDK,0, 0,"Dual mode support enable",0);
            }else{
                dual_mode_state = DUAL_MODE_SUPPORT_DISABLE;
                LOG_MSG_LIB(TL_LOG_NODE_SDK,0, 0,"Dual mode support disable",0);
            }
        }
    }

    if(DUAL_MODE_SUPPORT_ENABLE == dual_mode_state){
        rf_link_light_event_callback(LGT_CMD_DUAL_MODE_MESH);
    }
} ?  end dual_mode_en_init ?
```

There are 2 main conditions which is used to control the Dual Mode Selection:

- **Firmware Magic Patterns (Firmware offset + 0x8)**
    - **SIG_MESH_Offset** is 0x80000 - SIG BLE Mesh firmware
    - **Zigbee_Offset** is 0xC0000 - Zigbee firmware

  The firmware magic pattern is generated and appended to the firmware is generated, during the boot, it’ll validate the firmware by this pattern, and only boot if the pattern “START_UP_FLAG(0x544c4e4b)” existed at location offset+0x8

- **FLASH_ADR_MESH_TYPE_FLAG (0x24000)**
    - TYPE_DUAL_MODE_STANDBY - 0x65

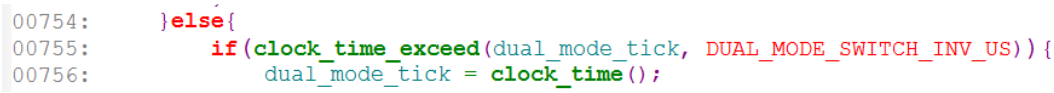
- TYPE_DUAL_MODE_RECOVER - 0x56 (it is set in Zigbee firmware when factory reset. And then it will be set to TYPE_DUAL_MODE_STANDBY in proc_telink_mesh_to_sig_mesh() in mesh firmware)

This flag is used to indicate what type of the MESH we're running, for SIG BLE MESH and Zigbee concurrent implementation, it is using 0x65 for Dual Mode, you can refer to proj_lib/ble/blt_config.h for other MESH mode Telink SDK could support

If it is DUAL_MODE_SUPPORT_DISABLE, then it'll continue BLE SIG MESH firmware, and check if there is a FACTORY reset event, if it does, it'll (based on CH1.2) reset back to the DUAL_MODE_SUPPORT_ENABLE mode.

If the mode configuration is DUAL_MODE_SUPPORT_ENABLE, then it'll run dual_mode_proc() and switch the mode to SIG BLE MESH and ZigBee every 3 seconds.

You can refer to the macro - DUAL_MODE_SWITCH_INV_US

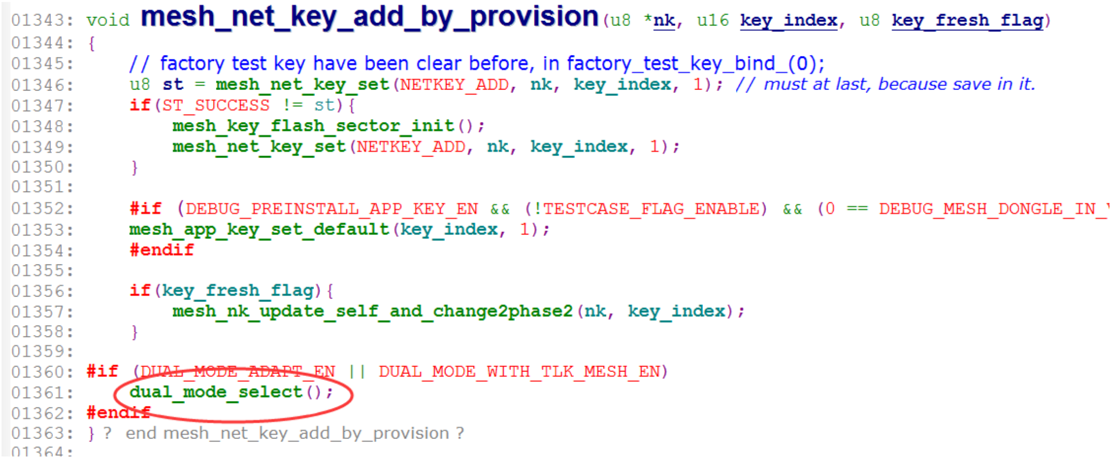


```
    }else{
        if(clock_time_exceed(dual_mode_tick, DUAL_MODE_SWITCH_INV_US)){
            dual_mode_tick = clock_time();
```

In BLE MESH mode, (rf_mode == RF_MODE_BLE)， it'll run

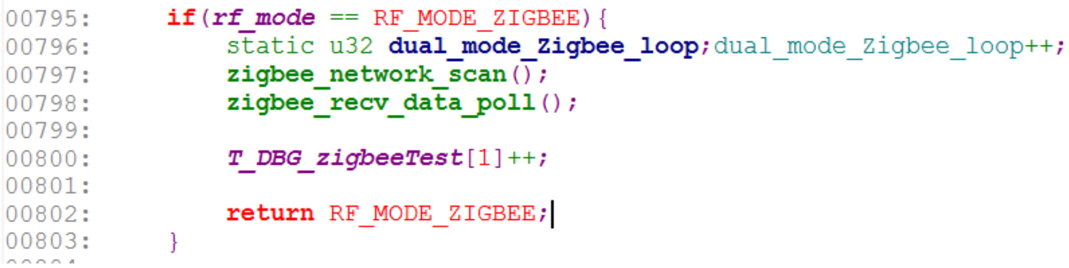
main_loop ()—>blt_sdk_main_loop (); and BLE mesh firmware functions.

The default time to stay in the mode is 3 seconds, during the time, it'll send out SIG BLE MESH unprovision beacon, and if there is a provisioner or SIG BLE Mesh gateway try to initiate adv provision (is_ble_found()=1), then the dual_mode_proc() will not switch to Zigbee mode any more but wait for the provisioning process to complete, once it joins the mesh network successfully, it'll invoke mesh_net_key_add_by_provision()->dual_mode_select() to choose SIG BLE Mesh and exit the Dual Mode Selection mode.



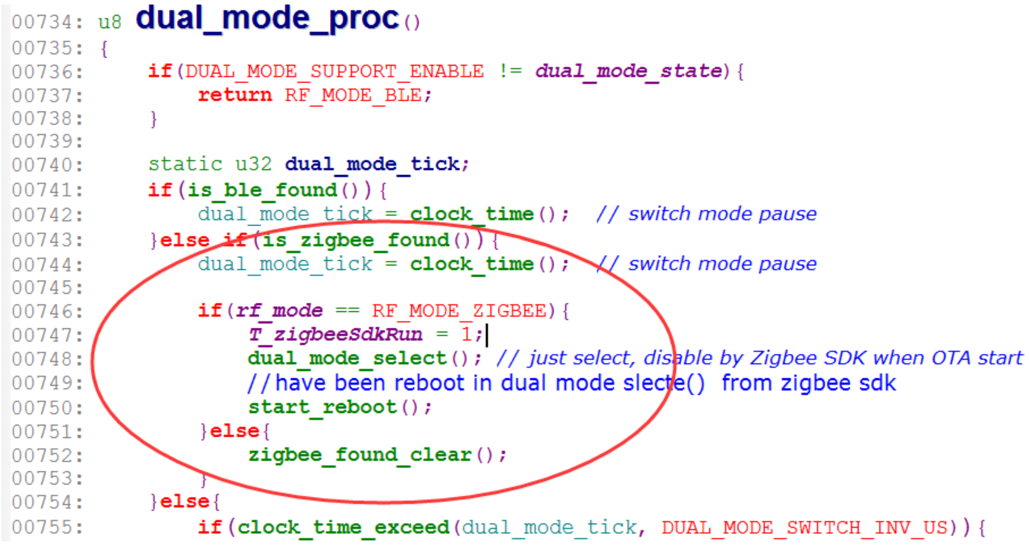
```
void mesh_net_key_add_by_provision(u8 *nk, u16 key_index, u8 key_fresh_flag)
{
    // factory test key have been clear before, in factory_test_key_bind_(0);
    u8 st = mesh_net_key_set(NETKEY_ADD, nk, key_index, 1); // must at last, because save in it.
    if(ST_SUCCESS != st){
        mesh_key_flash_sector_init();
        mesh_net_key_set(NETKEY_ADD, nk, key_index, 1);
    }

    #if (DEBUG_PREINSTALL_APP_KEY_EN && (!TESTCASE_FLAG_ENABLE) && (0 == DEBUG_MESH_DONGLE_IN_
    mesh_app_key_set_default(key_index, 1);
    #endif

    if(key_fresh_flag){
        mesh_nk_update_self_and_change2phase2(nk, key_index);
    }

#if (DUAL_MODE_ADAPT_EN || DUAL_MODE_WITH_TLK_MESH_EN)
    dual_mode_select();
#endif
} ? end mesh_net_key_add_by_provision ?
```

When 3 seconds is due, it'll switch to ZigBee mode (rf_mode == RF_MODE_ZIGBEE), and send out ZigBee scan and receive the network poll or beacon, implementation is like below, in dual_mode_proc():



```
    if(rf_mode == RF_MODE_ZIGBEE){
        static u32 dual_mode_Zigbee_loop;dual_mode_Zigbee_loop++;
        zigbee_network_scan();
        zigbee_recv_data_poll();

        T_DBG_zigbeeTest[1]++;

        return RF_MODE_ZIGBEE;
    }
```

And then it’ll check if there is any ZigBee coordinator enable the Permit_to_Join, , if there is, then zigbee_recv_data_poll() will set zigbeeNetworkFound=1 (is_zigee_found()=1), and invokes dual_mode_select() to set 0x24000 to TYPE_DUAL_MODE_ZIGBEE_RESET and then restart from 0xC0000 as ZigBee full function for the rest of network joining process.



```
u8 dual_mode_proc()
{
    if(DUAL_MODE_SUPPORT_ENABLE != dual_mode_state){
        return RF_MODE_BLE;
    }

    static u32 dual_mode_tick;
    if(is_ble_found()){
        dual_mode_tick = clock_time();   // switch mode pause
    }else if(is_zigbee_found()){
        dual_mode_tick = clock_time();   // switch mode pause

        if(rf_mode == RF_MODE_ZIGBEE){
            T_zigbeeSdkRun = 1;
            dual_mode_select(); // just select, disable by Zigbee SDK when OTA start
            //have been reboot in dual mode slecte()  from zigbee sdk
            start_reboot();
        }else{
            zigbee_found_clear();
        }
    }else{
        if(clock_time_exceed(dual_mode_tick, DUAL_MODE_SWITCH_INV_US)){
```

After switching to Zigbee mode, if the device couldn't joint the network after 6 seconds, the device will reboot from SIG Mesh firmware and resume the Dual Mode selection mode.

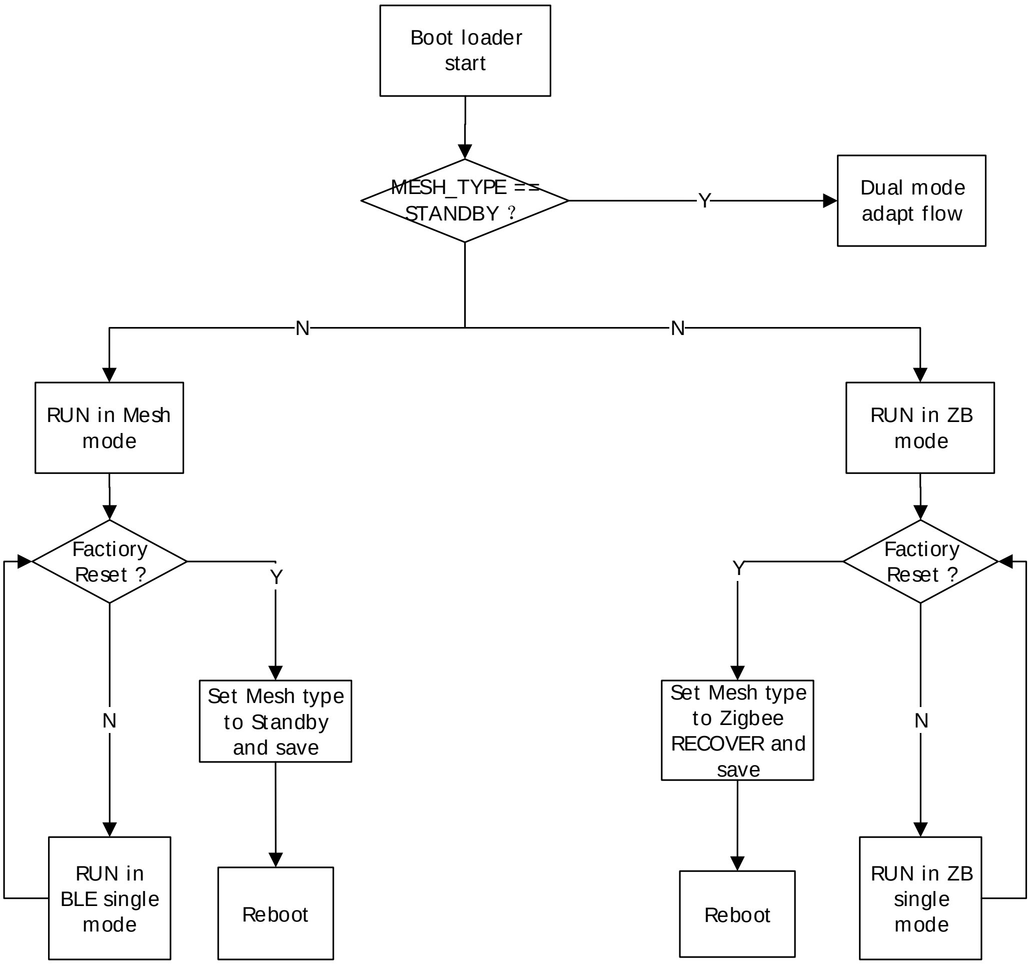
1.3 Dual Mode Recover Flow

1.3.1 Reset to Dual Mode Selection

When choosing one of the modes (SIG MESH or Zigbee), it'll continue to run under the mode. We could either factory reset or delete node to reset back to Dual Mode Selection state.

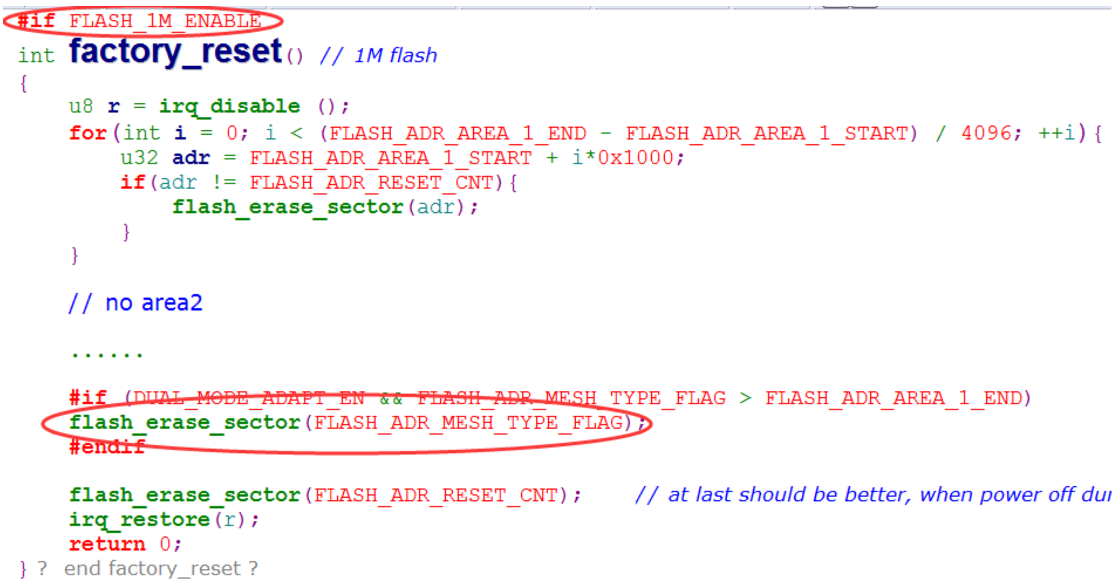
***Note:** It can keep dual mode after OTA.*

Figure 1-3 Dual Mode Recover Flow



1.3.2 In BLE Mode, Reset to Dual Mode Selection Flow

If invoking the Factory Reset, it'll check Zigbee firmware existence, and then clear the parameters area, and recover the firmware flag to **0xFFFFFFFF** (default value):



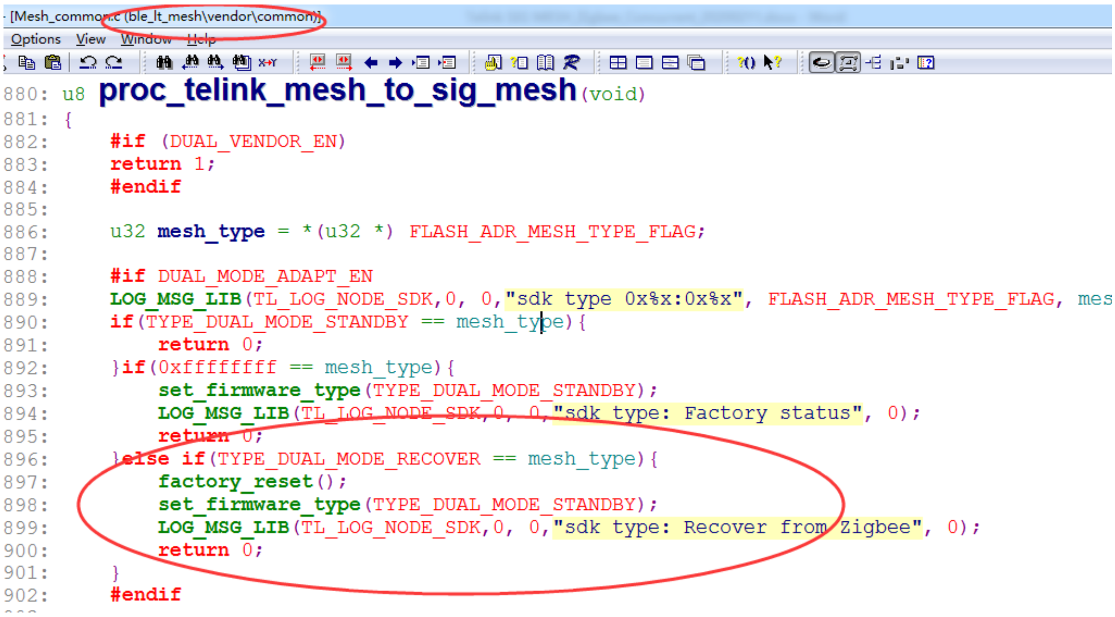
```
#if FLASH_1M_ENABLE
int factory_reset() // 1M flash
{
    u8 r = irq_disable ();
    for(int i = 0; i < (FLASH_ADR_AREA_1_END - FLASH_ADR_AREA_1_START) / 4096; ++i){
        u32 adr = FLASH_ADR_AREA_1_START + i*0x1000;
        if(adr != FLASH_ADR_RESET_CNT){
            flash_erase_sector(adr);
        }
    }

    // no area2

    ......

    #if (DUAL_MODE_ADAPT_EN && FLASH_ADR_MESH_TYPE_FLAG > FLASH_ADR_AREA_1_END)
    flash_erase_sector(FLASH_ADR_MESH_TYPE_FLAG);
    #endif

    flash_erase_sector(FLASH_ADR_RESET_CNT);     // at last should be better, when power off dui
    irq_restore(r);
    return 0;
} ? end factory_reset ?
```

After resetting the firmware type and restart the device, and it will return back to the Dual Mode Selection state as described in section 1.2 Dual Mode Selection Flow.

1.3.3 In Zigbee Mode, Reset to Dual Mode Selection Flow

When running Zigbee mode, if factory reset is triggered it will set the image type flag 0x24000 to TYPE_DUAL_MODE_RECOVER, and then reboot.

After reboot, it will run mesh firmware, and then set the flag 0x24000 to TYPE_DUAL_MODE_STANDBY to back to dual mode.

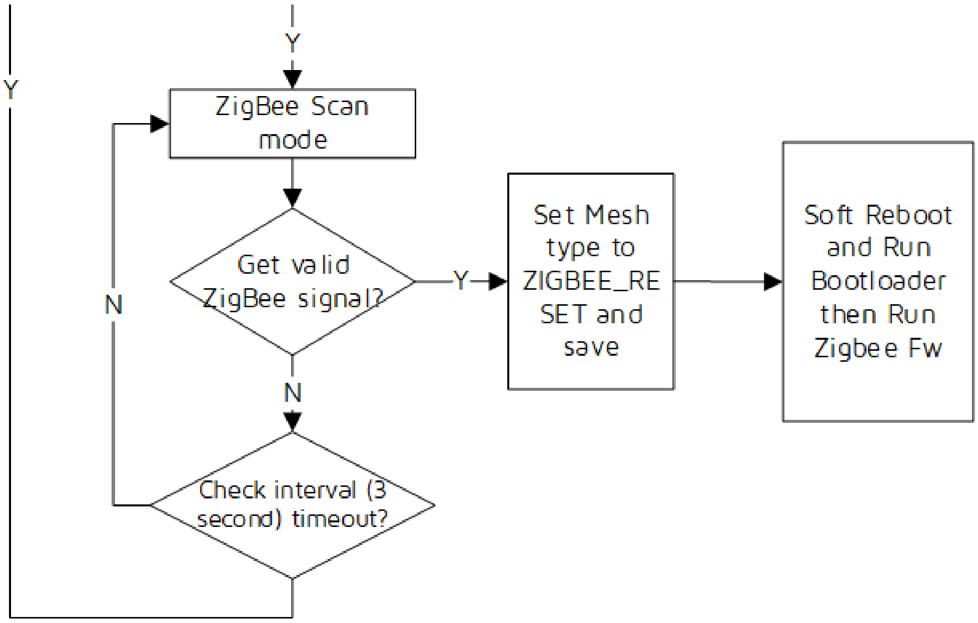


```
u8 proc_telink_mesh_to_sig_mesh(void)
{
    #if (DUAL_VENDOR_EN)
    return 1;
    #endif

    u32 mesh_type = *(u32 *) FLASH_ADR_MESH_TYPE_FLAG;

    #if DUAL_MODE_ADAPT_EN
    LOG_MSG_LIB(TL_LOG_NODE_SDK,0, 0,"sdk type 0x%x:0x%x", FLASH_ADR_MESH_TYPE_FLAG, mes
    if(TYPE_DUAL_MODE_STANDBY == mesh_type){
        return 0;
    }if(0xffffffff == mesh_type){
        set_firmware_type(TYPE_DUAL_MODE_STANDBY);
        LOG_MSG_LIB(TL_LOG_NODE_SDK,0, 0,"sdk type: Factory status", 0);
        return 0;
    }else if(TYPE_DUAL_MODE_RECOVER == mesh_type){
        factory_reset();
        set_firmware_type(TYPE_DUAL_MODE_STANDBY);
        LOG_MSG_LIB(TL_LOG_NODE_SDK,0, 0,"sdk type: Recover from Zigbee", 0);
        return 0;
    }
    #endif
```

1.4 Zigbee Network Detection Introduction

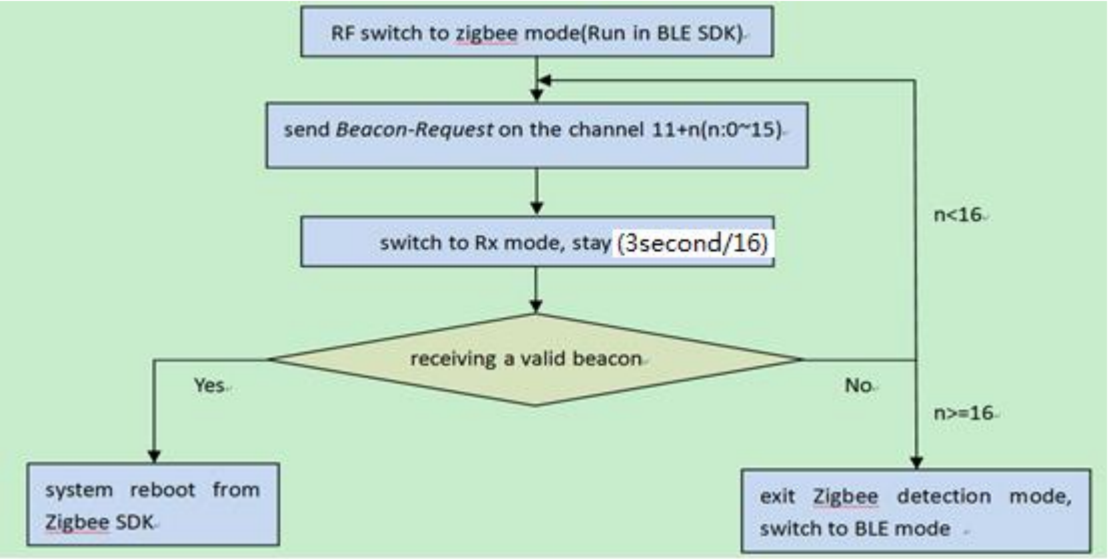
This section will introduce the details of Zigbee network detection and mode switching

Figure 1-4 Zigbee Network Detection Flow


Y
Y
ZigBee Scan mode
Get valid ZigBee signal?
Y
Set Mesh type to ZIGBEE_RESET and save
Soft Reboot and Run Bootloader then Run Zigbee Fw
N
N
Check interval (3 second) timeout?

1.4.1 Flow-Chart of Beacon Detection

Figure 1-5 Flow-Chart of Beacon Detection


RF switch to zigbee mode(Run in BLE SDK)
send Beacon-Request on the channel 11+n(n:0~15)
switch to Rx mode, stay (3second/16)
receiving a valid beacon
Yes
No
n<16
n>=16
system reboot from Zigbee SDK
exit Zigbee detection mode, switch to BLE mode

During the 3 seconds period, it'll seend to send the beacon from channel 0~15 and listen on the channel for 3s/16 = 187ms.

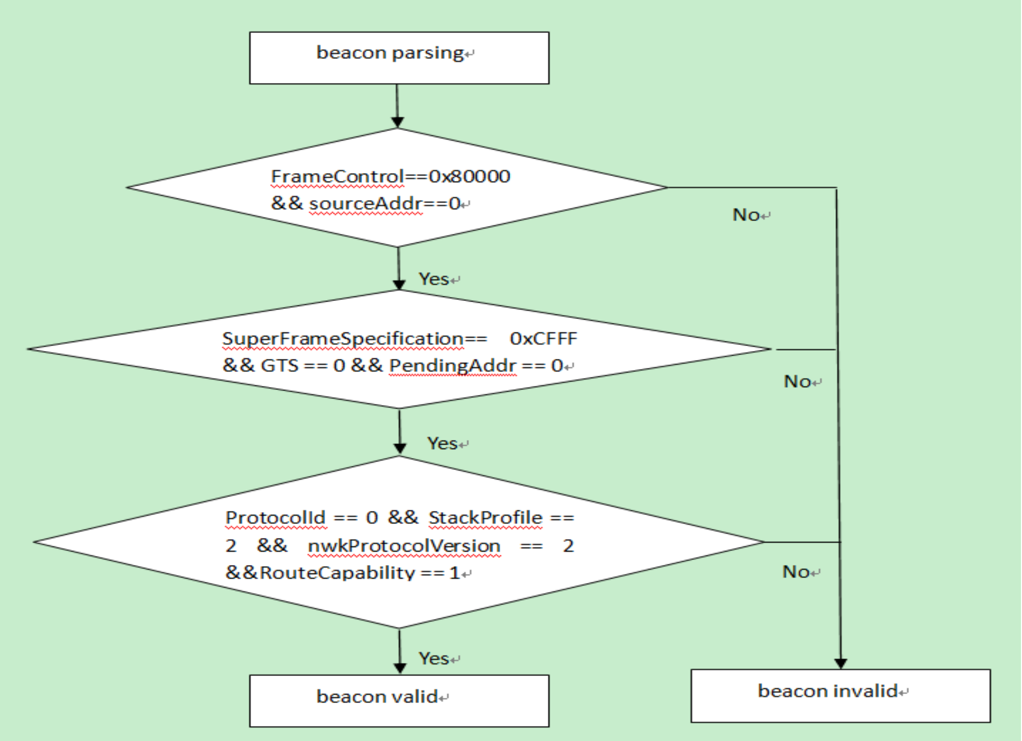
If there is a valid beacon received as the diagram showed "***receiving a valid beacon***" , then the SIG BLE MESH will execute ***"system reboot from ZigBee SDK***" and continue to finish joining the network.

If there is no valid beacon received, then it'll continue on next channel and wait for the valid beacon until it runs through all the 16 channels and then switch back to SIG BLE MESH for next 3 seconds.

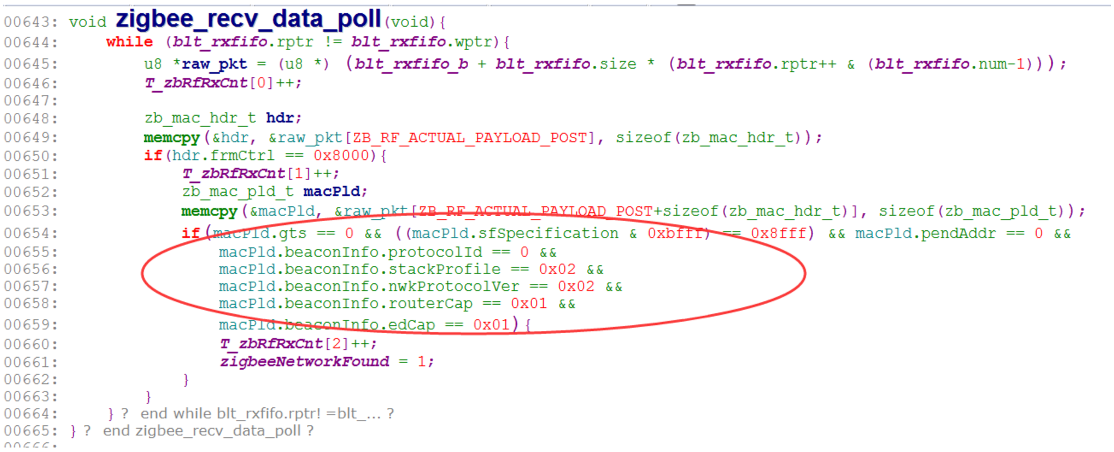
1.4.2 Condition of the Valid Beacon

After receiving the beacon, it'll validate the beacon before decides if this is a valid beacon:

Figure 1-6 Condition of the Valid Zigbee Beacon


beacon parsing
FrameControl==0x80000
&& sourceAddr==0
No
Yes
SuperFrameSpecification== 0xCFFF
&& GTS == 0 && PendingAddr == 0
No
Yes
ProtocolId == 0 && StackProfile ==
2 && nwkProtocolVersion == 2
&&RouteCapability == 1
No
Yes
beacon valid
beacon invalid

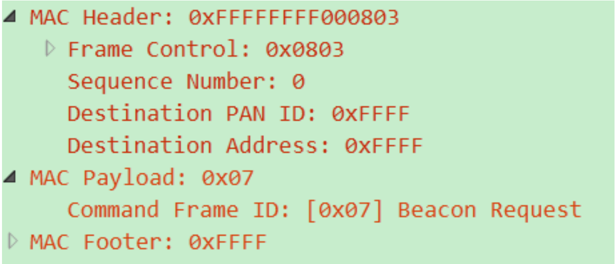
Snapshot of the implementation in mesh firmware:



```
void zigbee_recv_data_poll(void){
    while (blt_rxfifo.rptr != blt_rxfifo.wptr){
        u8 *raw_pkt = (u8 *)  (blt_rxfifo_b + blt_rxfifo.size * (blt_rxfifo.rptr++ & (blt_rxfifo.num-1)));
        T_zbRfRxCnt[0]++;

        zb_mac_hdr_t hdr;
        memcpy(&hdr, &raw_pkt[ZB_RF_ACTUAL_PAYLOAD_POST], sizeof(zb_mac_hdr_t));
        if(hdr.frmCtrl == 0x8000){
            T_zbRfRxCnt[1]++;
            zb_mac_pld_t macPld;
            memcpy(&macPld, &raw_pkt[ZB_RF_ACTUAL_PAYLOAD_POST+sizeof(zb_mac_hdr_t)], sizeof(zb_mac_pld_t));
            if(macPld.gts == 0 && ((macPld.sfSpecification & 0xbfff) == 0x8fff) && macPld.pendAddr == 0 &&
                macPld.beaconInfo.protocolId == 0 &&
                macPld.beaconInfo.stackProfile == 0x02 &&
                macPld.beaconInfo.nwkProtocolVer == 0x02 &&
                macPld.beaconInfo.routerCap == 0x01 &&
                macPld.beaconInfo.edCap == 0x01){
                T_zbRfRxCnt[2]++;
                zigbeeNetworkFound = 1;
            }
        }
    } ?  end while blt_rxfifo.rptr! =blt_... ?
} ?  end zigbee_recv_data_poll ?
```

1.4.3 Beacon-Request Format

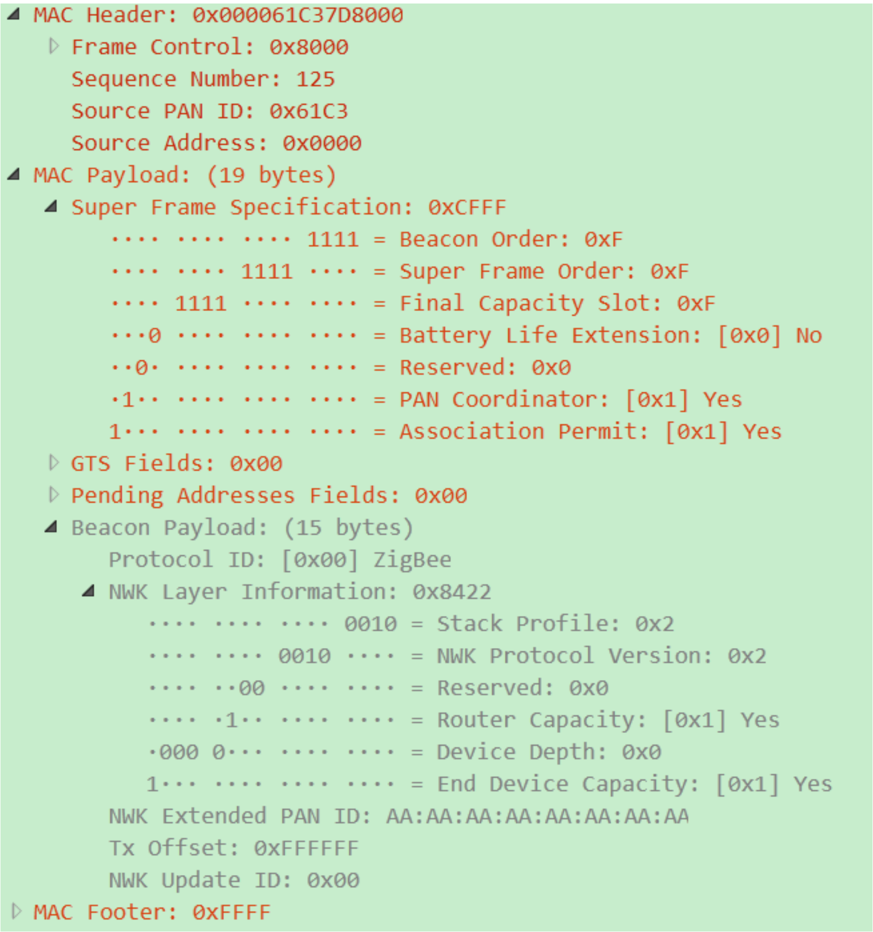
Figure 1-7 Beacon-Request Format



```
◢ MAC Header: 0xFFFFFFFF000803
   ▷ Frame Control: 0x0803
     Sequence Number: 0
     Destination PAN ID: 0xFFFF
     Destination Address: 0xFFFF
◢ MAC Payload: 0x07
     Command Frame ID: [0x07] Beacon Request
▷ MAC Footer: 0xFFFF
```

1.4.4 Beacon Format

Figure 1-8 Beacon Format



```
◢ MAC Header: 0x000061C37D8000
   ▷ Frame Control: 0x8000
     Sequence Number: 125
     Source PAN ID: 0x61C3
     Source Address: 0x0000
◢ MAC Payload: (19 bytes)
   ◢ Super Frame Specification: 0xCFFF
          ···· ···· ···· 1111 = Beacon Order: 0xF
          ···· ···· 1111 ···· = Super Frame Order: 0xF
          ···· 1111 ···· ···· = Final Capacity Slot: 0xF
          ···0 ···· ···· ···· = Battery Life Extension: [0x0] No
          ··0· ···· ···· ···· = Reserved: 0x0
          ·1·· ···· ···· ···· = PAN Coordinator: [0x1] Yes
          1··· ···· ···· ···· = Association Permit: [0x1] Yes
   ▷ GTS Fields: 0x00
   ▷ Pending Addresses Fields: 0x00
   ◢ Beacon Payload: (15 bytes)
        Protocol ID: [0x00] ZigBee
      ◢ NWK Layer Information: 0x8422
             ···· ···· ···· 0010 = Stack Profile: 0x2
             ···· ···· 0010 ···· = NWK Protocol Version: 0x2
             ···· ··00 ···· ···· = Reserved: 0x0
             ···· ·1·· ···· ···· = Router Capacity: [0x1] Yes
             ·000 0··· ···· ···· = Device Depth: 0x0
             1··· ···· ···· ···· = End Device Capacity: [0x1] Yes
        NWK Extended PAN ID: AA:AA:AA:AA:AA:AA:AA:AA
        Tx Offset: 0xFFFFFF
        NWK Update ID: 0x00
▷ MAC Footer: 0xFFFF
```

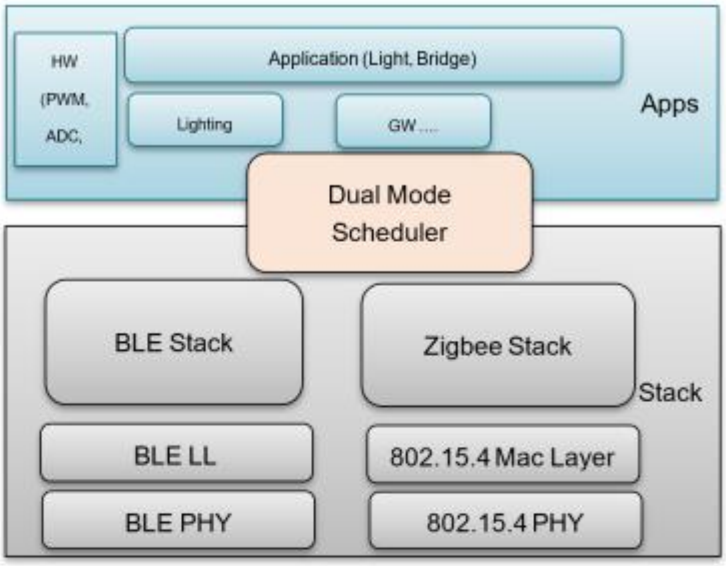
1.4.5 Zigbee/BLE Concurrent

1. Zigbee/BLE concurrent SDK is composed with the following components:

   - Application Layer
   - BLE Protocol Stack layer
   - BLE Link Layer

- BLE PHY Layer
- Zigbee Protocol Stack Layer
- 802.15.4 MAC layer
- 802.15.4 PHY Layer

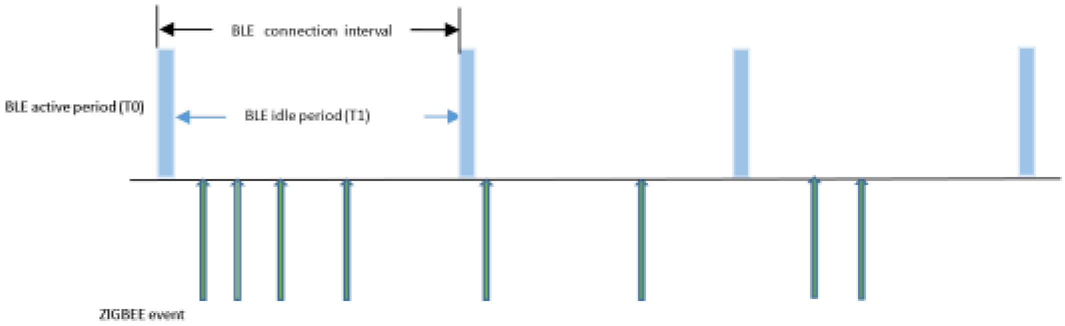
Figure 1-9 BLE/ZigBee Stack Architecture



2. Zigbee/BLE concurrent concepts:

Zigbee/BLE concurrent is based on the Time Division Multiplex (TDM), using the BLE clock as the base, to time slicing the radio for BLE and Zigbee, the time slot management will be based on the BLE Adv Interval and Connection Interval, and sharing with 802.15.4 radio.

Figure 1-10 BLE/ZigBee Band Sharing Timing Chart



2. Instruction of Misc.

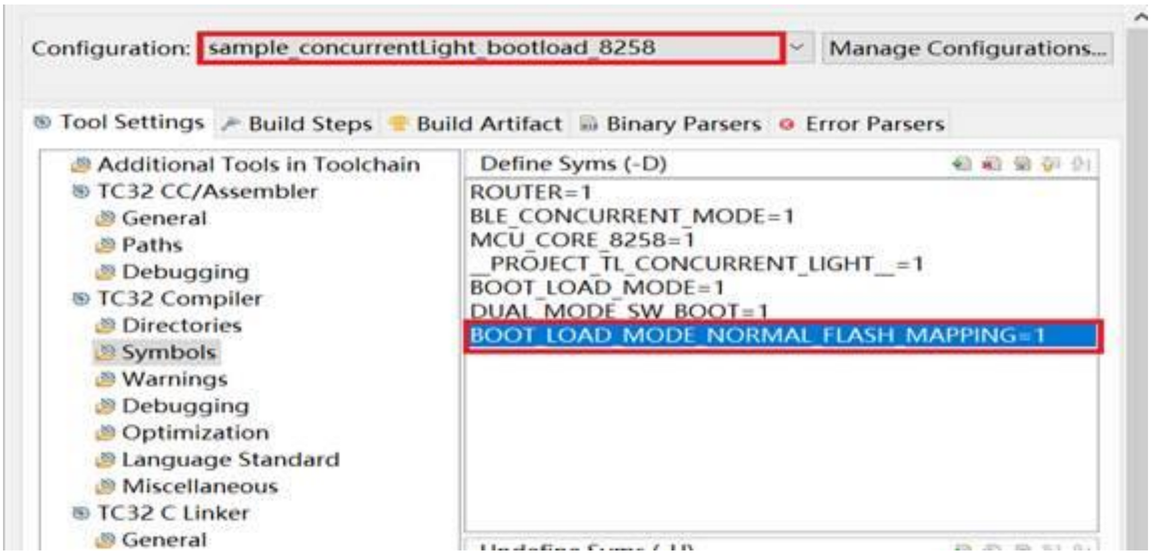
2.1 Configuration Parameter

2.1.1 ZigBee configuration parameter

SDK download: http://wiki.telink-semi.cn/tools_and_sdk/Dual_Mode/Zigbee_BLE_Concurrent_SDK.zip

set BOOT_LOAD_MODE_NORMAL_FLASH_MAPPING to 1, and select the project, sample_concurrentLight_bootload_8258, to build the sample_concurrentLight_bootload_8258 binary.

as shown in the figure below:



Configuration: sample_concurrentLight_bootload_8258

```
ROUTER=1
BLE_CONCURRENT_MODE=1
MCU_CORE_8258=1
__PROJECT_TL_CONCURRENT_LIGHT__=1
BOOT_LOAD_MODE=1
DUAL_MODE_SW_BOOT=1
BOOT_LOAD_MODE_NORMAL_FLASH_MAPPING=1
```

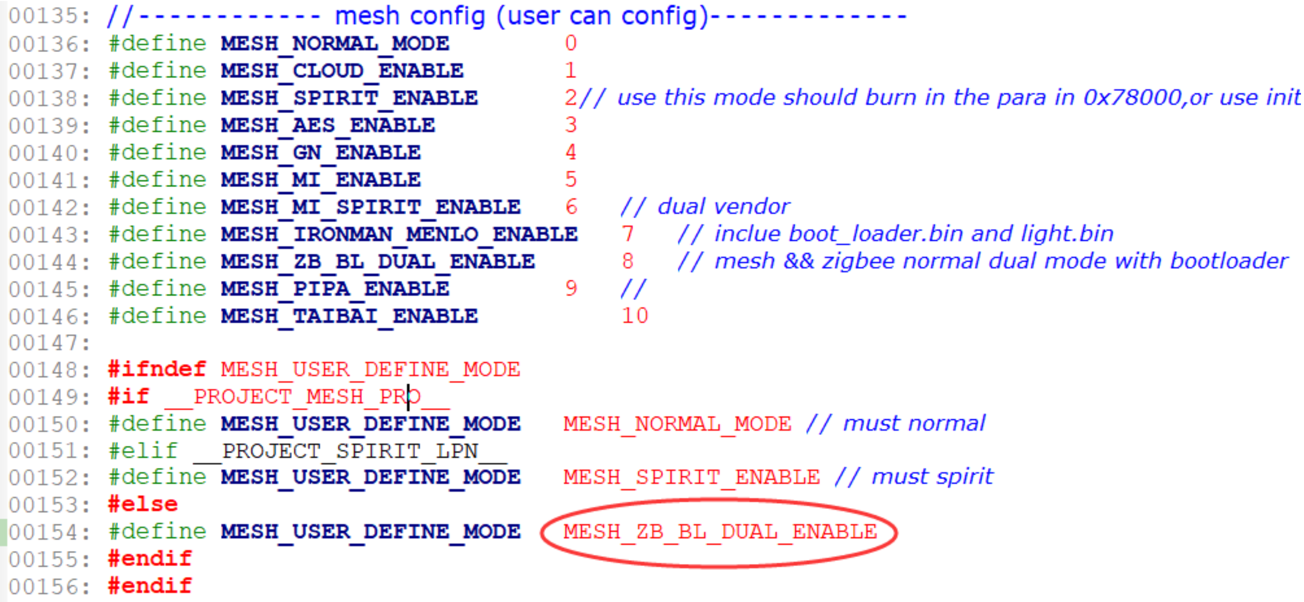
2.1.2 Mesh and BootLoader configuration parameter

SDK download: http://wiki.telink-semi.cn/tools_and_sdk/BLE_Mesh/SIG_Mesh/sig_mesh_sdk.zip

SIG MESH SDK firmware disables the DUAL Mode by default, you could set MESH_USER_DEFINE_MODE to MESH_ZB_BL_DUAL_ENABLE in ***vendor/common/mesh_config.h.***

After setting MESH_ZB_BL_DUAL_ENABLE , the SIG MESH/ZigBee switch mode will be enabled, which will also enable the related features like, **1MB flash** support, Static OOB, Remote Provision and Mesh OTA.

For bootloader build, the MESH_ZB_BL_DUAL_ENABLE will also need to be configured, this will enable the 1MB flash map layout so that the corresponding location will be mapped correctly.

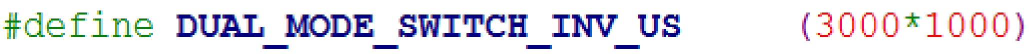


```
//------------ mesh config (user can config)-------------
#define MESH_NORMAL_MODE        0
#define MESH_CLOUD_ENABLE       1
#define MESH_SPIRIT_ENABLE      2// use this mode should burn in the para in 0x78000,or use init
#define MESH_AES_ENABLE         3
#define MESH_GN_ENABLE          4
#define MESH_MI_ENABLE          5
#define MESH_MI_SPIRIT_ENABLE   6   // dual vendor
#define MESH_IRONMAN_MENLO_ENABLE   7   // inclue boot_loader.bin and light.bin
#define MESH_ZB_BL_DUAL_ENABLE      8   // mesh && zigbee normal dual mode with bootloader
#define MESH_PIPA_ENABLE        9   //
#define MESH_TAIBAI_ENABLE          10

#ifndef MESH_USER_DEFINE_MODE
#if __PROJECT_MESH_PRO__
#define MESH_USER_DEFINE_MODE   MESH_NORMAL_MODE // must normal
#elif __PROJECT_SPIRIT_LPN__
#define MESH_USER_DEFINE_MODE   MESH_SPIRIT_ENABLE // must spirit
#else
#define MESH_USER_DEFINE_MODE   MESH_ZB_BL_DUAL_ENABLE
#endif
#endif
```

In the SIG MESH SDK, select the 8258_bootloader project to build the SIG boot loader binary.

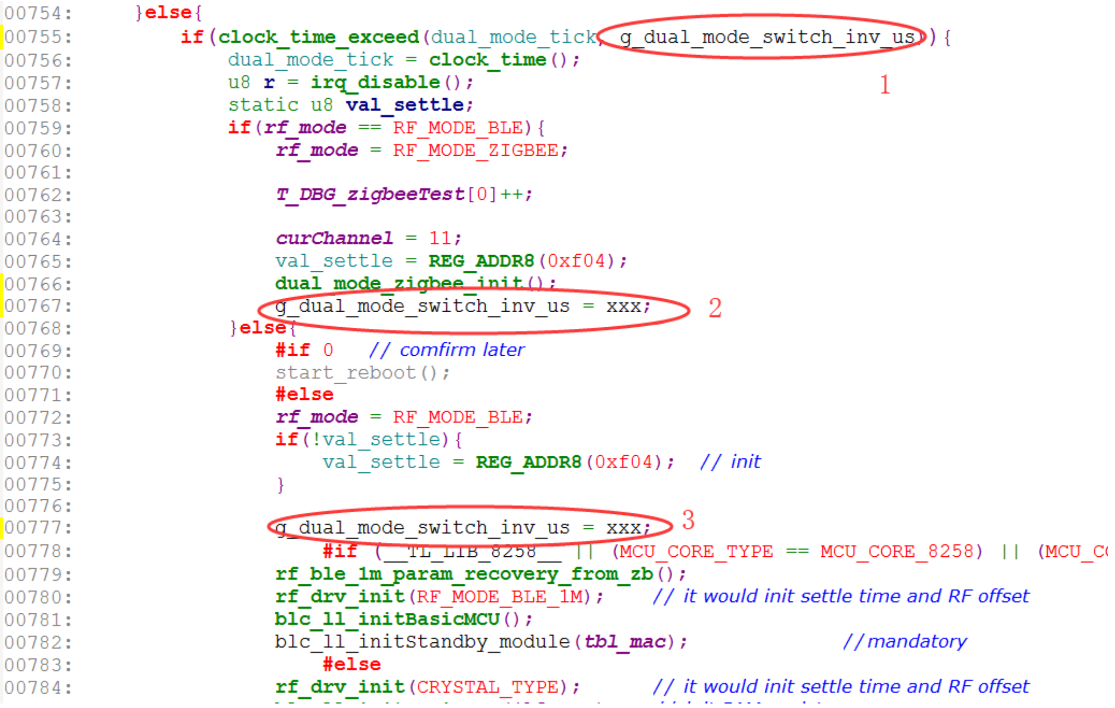
In the SIG MESH SDK, select the 8258_mesh project to build the Mesh binary.

The interval between SIG BLE MESH and Zigbee mode is 3 seconds, you can modify it from DUAL_MODE_SWITCH_INV_US from, ***vendor/common/dual_mode_adapt.c***：

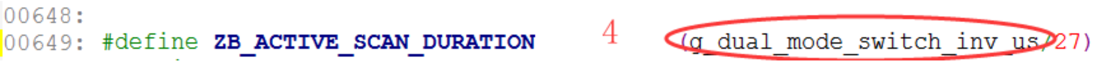


```
#define DUAL_MODE_SWITCH_INV_US     (3000*1000)
```

In current design, as you could see, the SIG BLE MESH and Zigbee has equivalent duration to stay on the mode, if user wants to change that different value, or none-equivalent duration, you can modify this macro to global variable and control it in different condition to allow different mode has different wait period configuration, like below sample：



```
}else{
    if(clock_time_exceed(dual_mode_tick, g_dual_mode_switch_inv_us)){
        dual_mode_tick = clock_time();
        u8 r = irq_disable();
        static u8 val_settle;
        if(rf_mode == RF_MODE_BLE){
            rf_mode = RF_MODE_ZIGBEE;

            T_DBG_zigbeeTest[0]++;

            curChannel = 11;
            val_settle = REG_ADDR8(0xf04);
            dual_mode_zigbee_init();
            g_dual_mode_switch_inv_us = xxx;
        }else{
            #if 0   // comfirm later
            start_reboot();
            #else
            rf_mode = RF_MODE_BLE;
            if(!val_settle){
                val_settle = REG_ADDR8(0xf04);  // init
            }

            g_dual_mode_switch_inv_us = xxx;
                #if (__TL_LIB_8258__ || (MCU_CORE_TYPE == MCU_CORE_8258) || (MCU_C
            rf_ble_1m_param_recovery_from_zb();
            rf_drv_init(RF_MODE_BLE_1M);    // it would init settle time and RF offset
            blc_ll_initBasicMCU();
            blc_ll_initStandby_module(tbl_mac);              //mandatory
                #else
            rf_drv_init(CRYSTAL_TYPE);      // it would init settle time and RF offset
```



```

#define ZB_ACTIVE_SCAN_DURATION     (g_dual_mode_switch_inv_us/27)
```

2.2 LED Indication

Dual mode

- Power cycle - Red LED blink once, and Green LED on
- Standby to be connected – Green LED on

SIG Mesh mode - Green LED on

- provisioning - Green LED on, Red LED blink
- after done provision - Green LED on (and now you can control from app)
- kick out - Green LED on, Red LED blink (~8s)

ZigBee mode - Yellow LED on

- joining network - Green LED on, Yellow LED blink (2 or 3 times)

2.3 Mesh APP Network Creation (None Remote Provision)

You can follow the instruction from our wiki page to use the APP to create the network to delete the node, and control the light.

http://wiki.telink-semi.cn/tools_and_sdk/BLE_Mesh/SIG_Mesh/sig_mesh_sdk.zip

2.4 Factory Reset

Factory reset by a sequence of power on/off operation, you could also refer to the Mesh SDK Handbook for further details - AN_17120401-CX_Telink SIG Mesh SDK Developer Handbook.pdf

SIG Mesh module (not LPN) could use the following power on/off sequence to reset the device back to factory mode:

1) Power on the module for 30 seconds, this will clean up the previous power on/off sequence information.

2) Short power cycle the module 3 times:

    - Power on the device for short period of time 0~3s and power off, then repeat this for 3 times

3) Long power cycle the module 2 times:

    - Power on the devices for longer period of time 4~30s and power off, then repeat this for 2 times

In user_init(), the factory_reset_handle() will detect the previous 5 times of the power on/off sequence and trigger the “Factory Reset” operation, and the Red LED will blink as 1Hz for 8s.

2.5 Switch to Zigbee Mode

1. Erase entire flash(1024K).
2. Program **Bootloader image** to flash offset 0x00
3. Program **BLE SIG Mesh image** to flash offset 0x80000
4. Program **Zigbee/BLE Image** to flash offset 0xC0000
5. Power up the device
6. Use the Zigbee Coordinator to enter the discovery mode, when the light sends out the beacon and receive a valid beacon from Zigbee Coordinator

7. Then the firmware will follow the steps in Ch1.4 to validate the beacon and configures the mode to Zigbee, and enters into the Zigbee firmware, following are the steps:
   1) Power on Coordinator Node, press SW2 to enable “Permit Joint”, the Green LED will be solid on for 180s. Or you could manual disable it by pressing the SW2 button again;
   2) Power on the Light node, it'll be in Switch mode by default;
   3) As mentioned in Ch1.4, the Light node will scan the valid beacon and detect the Coordinator allows to join the network, once it joins the network, the Green LED be solid ON, or otherwise if it failed to join, it'll go back to Switch mode.;
   4) If Light joins the network successfully, press SW1 on Coordinator node, it'll send out 1s toggle command. You should see the Light node start flashing every second., and you could press SW1 again to stop sending the toggle command.;
   5) At the same time, you could use any BLE app to test the BLE connection;
8. If you need to reset back to Dual Selection Mode, then you can follow the ch2.4 Factory Reset sequence.

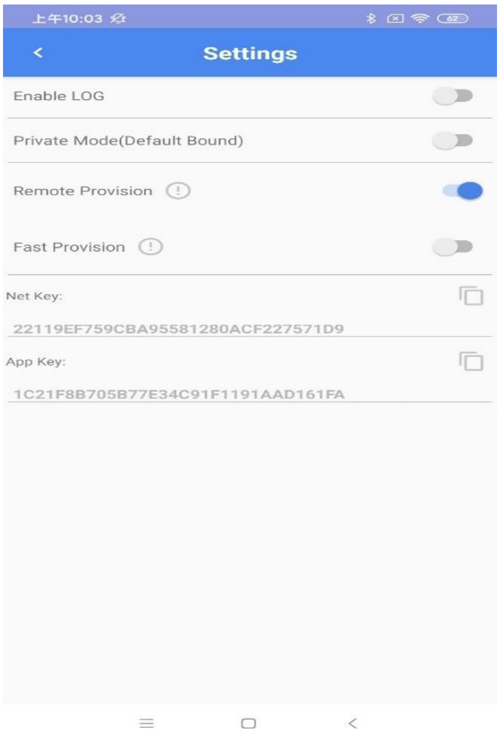
2.6 Remote Provision

In order to support the nodes far away from the provisioner/phone, SIG Mesh 1.1 defines the Remote Provision to allow the remote node to join the network through the nodes that have been provisioned and is capable of doing the provisioning.

Here is the example of how the network is provisioned and created:

- Provision the nodes which is within RF range, which are one hop away from the provisioner.
- Scan the nodes which is in the range of first provisioned nodes, which are two hops away, then the provisioner will provision these scanned nodes.
- Remote provision then provision nodes which are 3...4...5… hops away accordingly, until there is no further unprovisioned nodes. The default config for remote provision is 4 hops.

Step 1 Enable the remote provision feature in “**Settings”** page of the APP



Enable LOG ○

Private Mode(Default Bound) ○

Remote Provision ●

Fast Provision ○

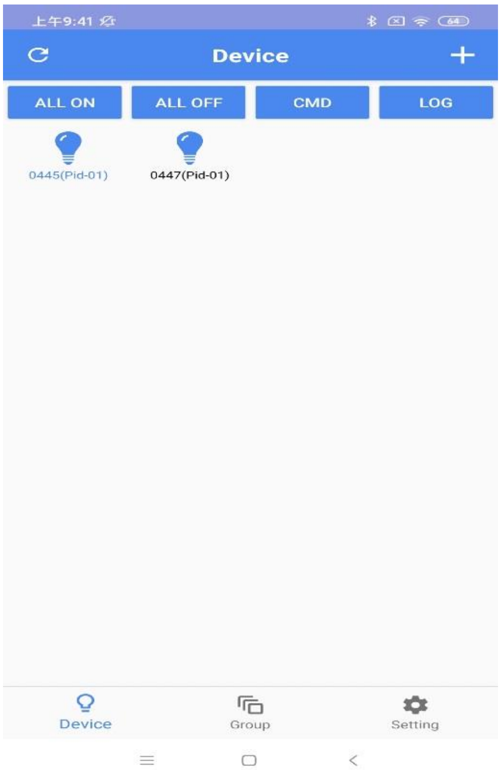
Net Key:

22119EF759CBA95581280ACF227571D9

App Key:

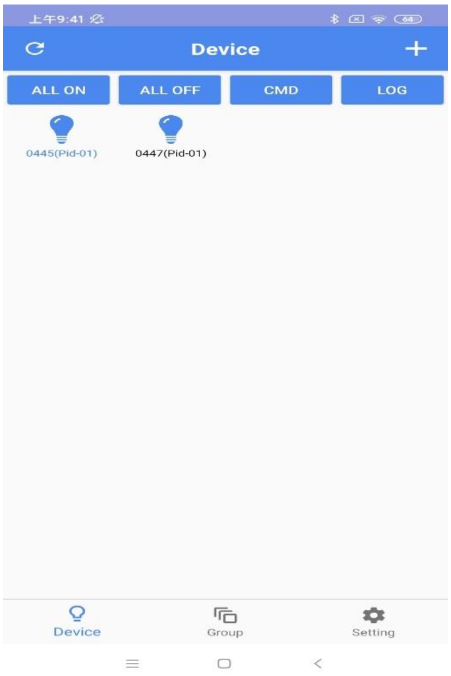
1C21F8B705B77E34C91F1191AAD161FA

Step 2 Return to **”Device”** page, click **“+”**on the top-right corner to scan for new device

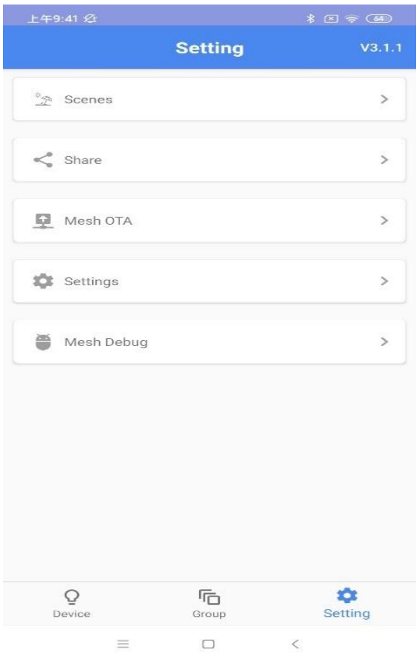


2.7 Mesh OTA

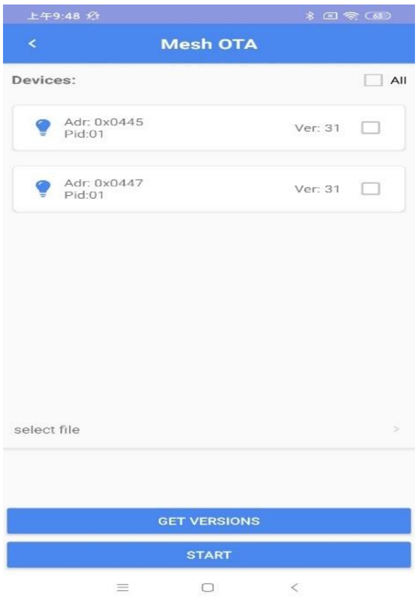
Step 1 Provision two 8258 dongles, after provisioning, click “**ALL ON”** and **”ALL OFF**” to make sure the nodes could be controlled correctly



Step 2 Click on the **‘Setting”** button on the right-bottom to enter the Setting Page, click **“Mesh OTA”**



Step 3 Click on the **“Mesh OTA”**, and select the nodes to be upgraded, and click **“Select file”** to choose the firmware to do the OTA, **“GET VERSION”** will show the current version information



Adr: 0x0445 Pid:01 Ver: 31 □

Adr: 0x0447 Pid:01 Ver: 31 □

Step 4 Click **“start”** to start the OTA process and OTA progress will be displayed in the UI. Once the OTA is finished, there will be a notification

2.8 OTA Flow

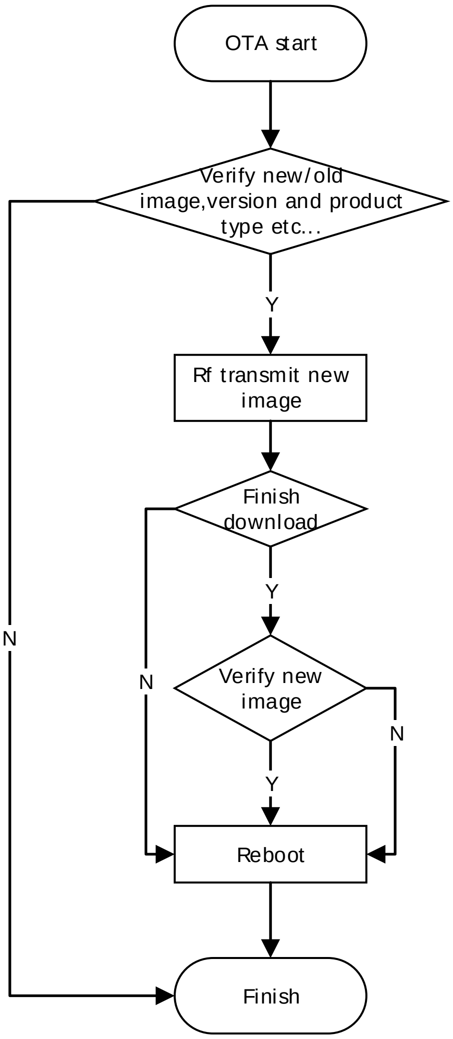
There are 3 sections for application image and OTA staging area.

- SIG MESH image - SIG Mesh image location
- Zigbee/BLE image - Zigbee/BLE image location
- OTA Staging Area - OTA staging area, will keep copy of the OTA image

Whenever the OTA successfully updated, the OTA Staging Area will save the entire new firmware, and then reboot, Bootloader will copy the new firmware from 0x40000 back to the designated area (e.g Sig Mesh: 0x80000, ZIgbee: 0xC0000) based on the Mesh Type Image Information set in 0x24000, you can refer to section 1.1 Bootloader Flow with OTA Check.

Following is the OTA flow chart.

Figure 2-1 OTA Flow Chart

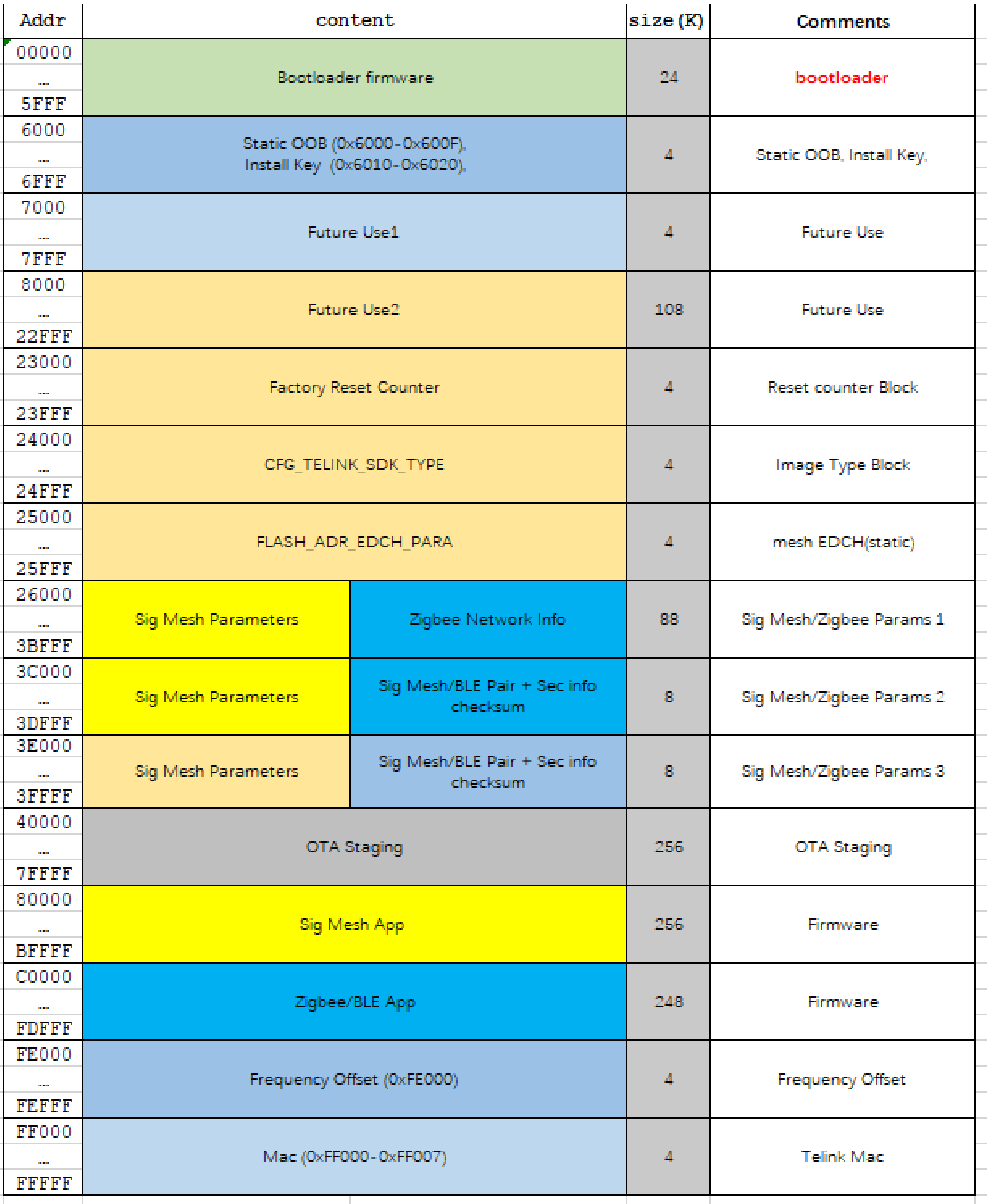


3. Flash and Memory Usage Information

3.1 Flash Map

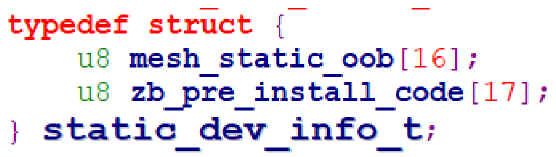
refert to \telink_sig_mesh_sdk\doc\SigmeshZigbeeFlashmapNormal.xlsx

Figure 3-1 Flash Map



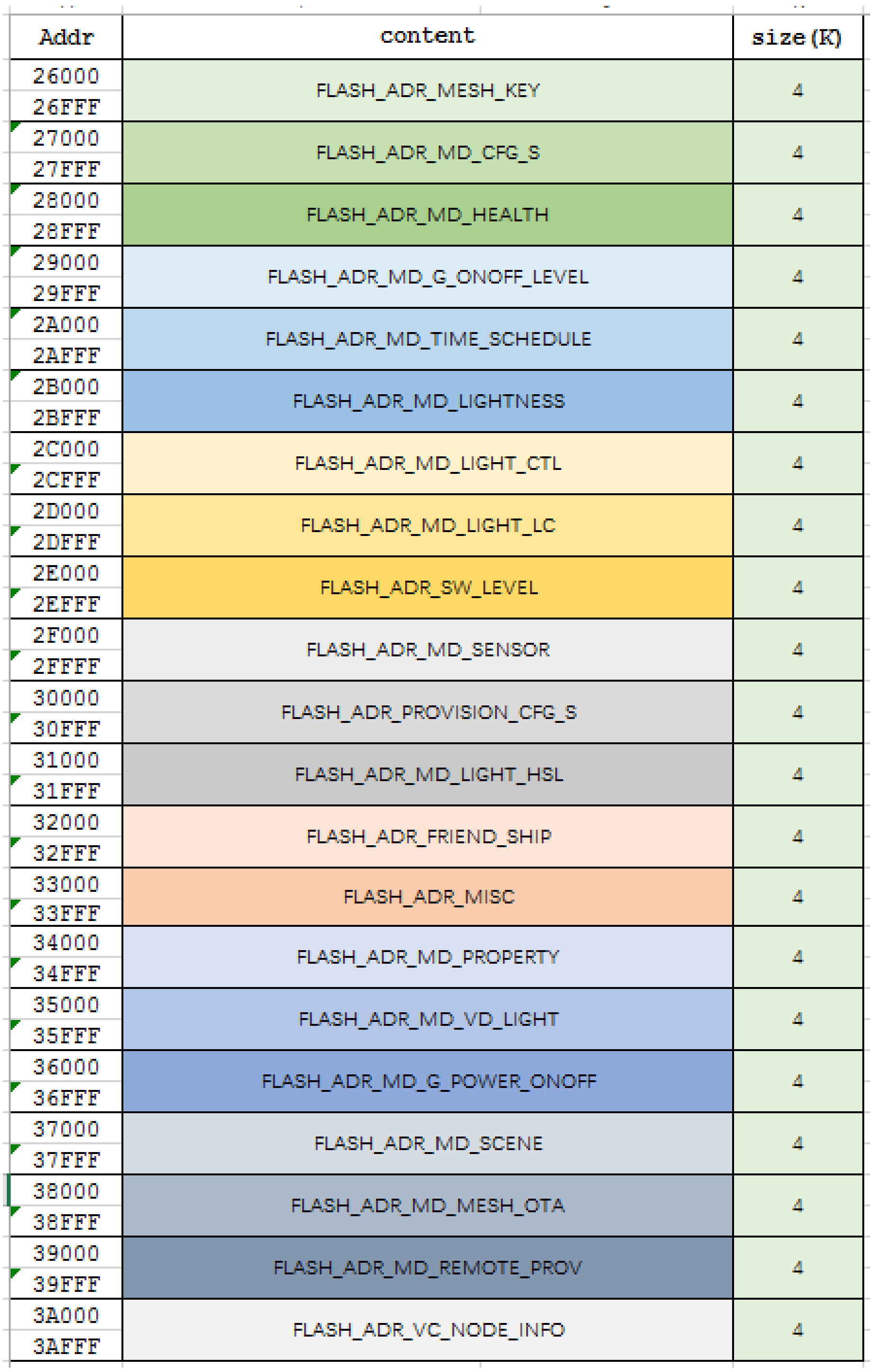
Addr	content		size (K)	Comments
00000 ... 5FFF	Bootloader firmware		24	bootloader
6000 ... 6FFF	Static OOB (0x6000-0x600F), Install Key (0x6010-0x6020),		4	Static OOB, Install Key,
7000 ... 7FFF	Future Use1		4	Future Use
8000 ... 22FFF	Future Use2		108	Future Use
23000 ... 23FFF	Factory Reset Counter		4	Reset counter Block
24000 ... 24FFF	CFG_TELINK_SDK_TYPE		4	Image Type Block
25000 ... 25FFF	FLASH_ADR_EDCH_PARA		4	mesh EDCH(static)
26000 ... 3BFFF	Sig Mesh Parameters	Zigbee Network Info	88	Sig Mesh/Zigbee Params 1
3C000 ... 3DFFF	Sig Mesh Parameters	Sig Mesh/BLE Pair + Sec info checksum	8	Sig Mesh/Zigbee Params 2
3E000 ... 3FFFF	Sig Mesh Parameters	Sig Mesh/BLE Pair + Sec info checksum	8	Sig Mesh/Zigbee Params 3
40000 ... 7FFFF	OTA Staging		256	OTA Staging
80000 ... BFFFF	Sig Mesh App		256	Firmware
C0000 ... FDFFF	Zigbee/BLE App		248	Firmware
FE000 ... FEFFF	Frequency Offset (0xFE000)		4	Frequency Offset
FF000 ... FFFFF	Mac (0xFF000-0xFF007)		4	Telink Mac

Detail of "Static OOB, Install Key"



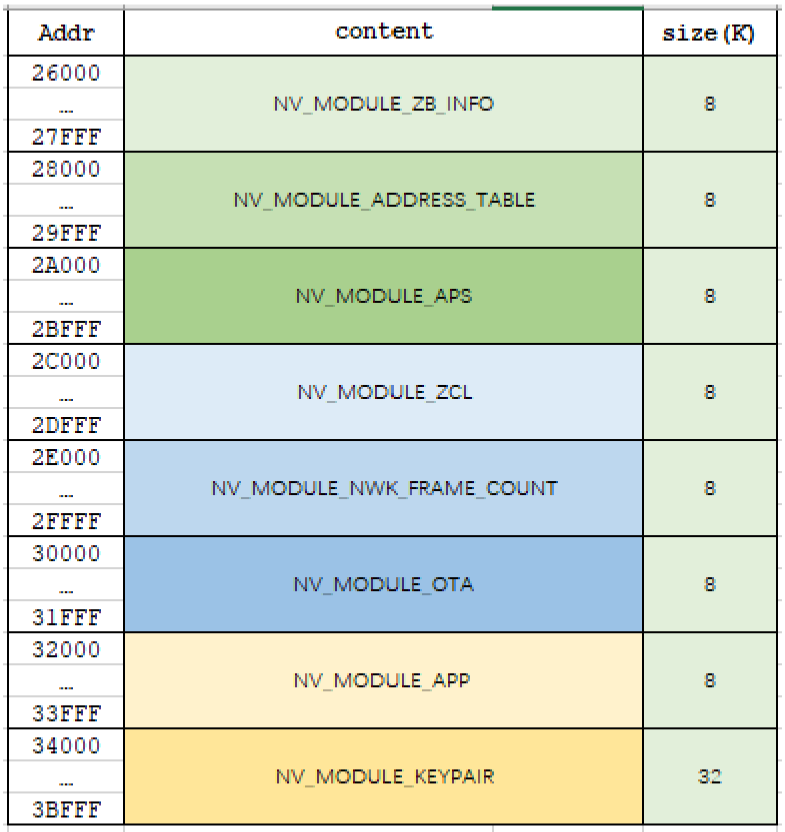
```
typedef struct {
      u8 mesh_static_oob[16];
      u8 zb_pre_install_code[17];
} static_dev_info_t;
```

Detail of "Sig Mesh Parameters"



Addr	content	size (K)
26000 26FFF	FLASH_ADR_MESH_KEY	4
27000 27FFF	FLASH_ADR_MD_CFG_S	4
28000 28FFF	FLASH_ADR_MD_HEALTH	4
29000 29FFF	FLASH_ADR_MD_G_ONOFF_LEVEL	4
2A000 2AFFF	FLASH_ADR_MD_TIME_SCHEDULE	4
2B000 2BFFF	FLASH_ADR_MD_LIGHTNESS	4
2C000 2CFFF	FLASH_ADR_MD_LIGHT_CTL	4
2D000 2DFFF	FLASH_ADR_MD_LIGHT_LC	4
2E000 2EFFF	FLASH_ADR_SW_LEVEL	4
2F000 2FFFF	FLASH_ADR_MD_SENSOR	4
30000 30FFF	FLASH_ADR_PROVISION_CFG_S	4
31000 31FFF	FLASH_ADR_MD_LIGHT_HSL	4
32000 32FFF	FLASH_ADR_FRIEND_SHIP	4
33000 33FFF	FLASH_ADR_MISC	4
34000 34FFF	FLASH_ADR_MD_PROPERTY	4
35000 35FFF	FLASH_ADR_MD_VD_LIGHT	4
36000 36FFF	FLASH_ADR_MD_G_POWER_ONOFF	4
37000 37FFF	FLASH_ADR_MD_SCENE	4
38000 38FFF	FLASH_ADR_MD_MESH_OTA	4
39000 39FFF	FLASH_ADR_MD_REMOTE_PROV	4
3A000 3AFFF	FLASH_ADR_VC_NODE_INFO	4

Detail of "Zigbee Network Info"



Addr	content	size(K)
26000 ... 27FFF	NV_MODULE_ZB_INFO	8
28000 ... 29FFF	NV_MODULE_ADDRESS_TABLE	8
2A000 ... 2BFFF	NV_MODULE_APS	8
2C000 ... 2DFFF	NV_MODULE_ZCL	8
2E000 ... 2FFFF	NV_MODULE_NWK_FRAME_COUNT	8
30000 ... 31FFF	NV_MODULE_OTA	8
32000 ... 33FFF	NV_MODULE_APP	8
34000 ... 3BFFF	NV_MODULE_KEYPAIR	32

3.2 Firmware Flash and SRAM Usage

3.2.1 SIG Mesh and Zigbee/BLE Image Size

Table 3-1 SIG Mesh and Zigbee/BLE Image Size

Type	Code Size	SRAM
SIG Mesh	129k	29k
Zigbee/BLE	208k	37k

3.2.2 SIG Mesh Flash Major Module Resource Information

Table 3-2 SIG Mesh Flash Major Module Resource Information

Function	Code Size	SRAM	Comments
Onoff Server	1.0k	0.3k	
Level Server	1.0k	0.5k	
Lighting Server	5.5k	1.0k	Include：lightness/lightness setup/light CTL/ light CTL setup/light CTL Temperature
Mesh OTA	4.5k	1.0k	
Remote Provision	4.6k	0.5k	
Friend	5.3k	0.8k	Support 2 LPN node connection and friend ship, cache 2 message for each LPN node.
Proxy	1.2k	0.1k	
Dual Model Switch	2.8k	0.4k	DUAL_MODE_ADAPT_EN

3.2.3 Zigbee_BLE Flash Resource Information

Table 3-3 Zigbee_BLE Flash Resource Information

Function	Code Size	SRAM	Comments
ZIGBEE/BLE	208k	37k	
ZIGBEE	190k	30k	Zigbee/BLE and Zigbee size difference to enable the support for BLE.
OTA	11k	200B	
WWAH	12k	200B~500B	The final number will be updated once feature is completely certified

4. SIG MESH Features and Configurations

4.1 How Many Buffers Would Proxy Role Support？

Downstream (from app to mesh node):

Proxy GATT has rx buffer is 16 (blt_rxfifo = 8) + 8 network PDU fifo (**MESH_ADV_CMD_BUF_CNT**)

Upstream (from mesh node to app):

Default has 96 GATT upstream buffer (blt_txfifo = 32) + (blt_notify_fifo = 64)

4.2 How Many Replay Protections?

CACHE_BUF_MAX (which is equivalent to **MESH_NODE_MAX_NUM**) is the configuration number for it, the default number is 105.

This number now is defined in the stack and currently not configurable for customers

4.3 Number of Lightbulbs

The current node number support is 105 nodes, and replay protection is also 105.

The number of relay protection will be limited by memory usage. With every increasing node, it takes another 6 bytes of memory usage.

4.4 Switching Time Overhead

In the 3S interval switch mode with MCU running at 16MHz, it takes about 650us to switch to ZigBee Radio, and 450us back to BLE Radio.