



GR551x Second Boot示例手册

版本： 2.0

发布日期： 2022-02-20

版权所有 © 2022 深圳市汇顶科技股份有限公司。保留一切权利。

非经本公司书面许可，任何单位和个人不得对本手册内的任何部分擅自摘抄、复制、修改、翻译、传播，或将其全部或部分用于商业用途。

商标声明

GOODIX 和其他汇顶商标均为深圳市汇顶科技股份有限公司的商标。本文档提及的其他所有商标或注册商标，由各自的所有人持有。

免责声明

本文档中所述的器件应用信息及其他类似内容仅为您提供便利，它们可能由更新之信息所替代。确保应用符合技术规范，是您自身应负的责任。

深圳市汇顶科技股份有限公司（以下简称“GOODIX”）对这些信息不作任何明示或暗示、书面或口头、法定或其他形式的声明或担保，包括但不限于针对其使用情况、质量、性能、适销性或特定用途的适用性的声明或担保。GOODIX对因这些信息及使用这些信息而引起的后果不承担任何责任。

未经GOODIX书面批准，不得将GOODIX的产品用作生命维持系统中的关键组件。在GOODIX知识产权保护下，不得暗或以其他方式转让任何许可证。

深圳市汇顶科技股份有限公司

总部地址：深圳市福田区腾飞工业大厦B座2层、13层

电话：+86-755-33338828 传真：+86-755-33338099

网址：www.goodix.com

前言

编写目的

本文档介绍了如何使用和验证GR551x SDK中的Second Boot示例，旨在帮助用户快速进行二次开发。

读者对象

本文适用于以下读者：

- GR551x用户
- GR551x开发人员
- GR551x测试人员
- 开发爱好者

版本说明

本文档为第6次发布，对应的产品系列为GR551x。

修订记录

| 版本 | 日期 | 修订内容 |
|-----|------------|---|
| 1.5 | 2020-08-30 | 首次发布 |
| 1.6 | 2020-11-25 | <ul style="list-style-type: none">• “下载固件”章节，新增下载<code>ble_tem_dfu.bin</code>固件前需进行的操作• “Second Boot OTA验证”章节，新增使用Second Boot进行OTA升级时、升级后的操作/影响说明• “应用固件校验跳转运行验证”章节，新增在Keil中使能Second Boot模式后重新编译固件的操作步骤 |
| 1.7 | 2020-12-25 | <ul style="list-style-type: none">• “下载固件”章节，新增下载<code>second_boot.bin</code>固件前需进行的操作• “应用固件校验跳转运行”章节，新增说明• 新增章节“应用固件睡眠后无法被唤醒” |
| 1.8 | 2021-04-19 | <ul style="list-style-type: none">• 更新“下载固件”章节的<code>user_config.h</code>中的参数• 新增“自定义固件拷贝升级、校验和跳转实现”章节 |
| 1.9 | 2021-12-29 | <ul style="list-style-type: none">• 删除“硬件连接”章节• 更新“下载固件”为“固件烧录” |
| 2.0 | 2022-02-20 | 基于SDK修改固件名称 |

目录

| | |
|---------------------------------|----|
| 前言..... | I |
| 1 简介..... | 1 |
| 2 Flash布局..... | 2 |
| 3 初次运行..... | 3 |
| 3.1 准备工作..... | 3 |
| 3.2 固件烧录..... | 3 |
| 3.3 测试验证..... | 5 |
| 3.3.1 Second Boot OTA验证..... | 5 |
| 3.3.2 应用固件校验跳转运行验证..... | 6 |
| 3.3.3 安全验签验证..... | 9 |
| 4 应用详解..... | 12 |
| 4.1 工程目录..... | 12 |
| 4.2 交互流程及代码介绍..... | 12 |
| 4.2.1 DFU固件拷贝升级..... | 13 |
| 4.2.2 应用固件校验跳转运行..... | 15 |
| 4.2.3 自定义固件拷贝升级、校验和跳转实现..... | 17 |
| 5 常见问题..... | 18 |
| 5.1 Second Boot安全OTA升级固件失败..... | 18 |
| 5.2 应用固件睡眠后无法被唤醒..... | 18 |

1 简介

Second Boot示例利用蓝牙无线传输、固件二次引导技术，演示了DFU（Device Firmware Update）固件拷贝升级、应用固件（Application Firmware）校验跳转运行、安全验签的功能，实现了灵活、可靠、安全的固件空中升级（OTA，Over The Air）。

- **DFU固件拷贝升级：**采用双区后台拷贝升级方式，通过蓝牙低功耗（BLE）无线传输实现固件空中升级。
- **应用固件校验跳转运行：**对APP Image Info进行匹配，校验并跳转至应用固件（本文以`ble_tem_dfu.bin`固件为例）运行。
- **安全验签：**为防止DFU固件被篡改和防否认，升级固件时可附加签名信息，Second Boot示例会对其进行验签，验签通过后即可升级。

在进行操作前，可参考以下文档。

表 1-1 文档参考

| 名称 | 描述 |
|-----------------|---|
| GR551x开发者指南 | 介绍GR551x SDK以及基于SDK的应用开发和调试 |
| GR551x固件升级指南 | 介绍GR551x的固件升级原理和应用 |
| GR551x OTA示例手册 | GR551x空中升级使用说明 |
| GProgrammer用户手册 | GProgrammer软件的使用说明，包括固件下载、加密加签等 |
| GR551x固件加密及应用介绍 | 介绍GR551x的固件加密加签模式 |
| J-Link用户指南 | J-Link使用说明： http://www.segger.com/downloads/jlink/UM08001_JLink.pdf |
| Keil用户指南 | Keil详细操作说明： http://www.keil.com/support/man/docs/uv4/ |

2 Flash布局

适用于GR551x Second Boot示例的Flash布局如下图所示。

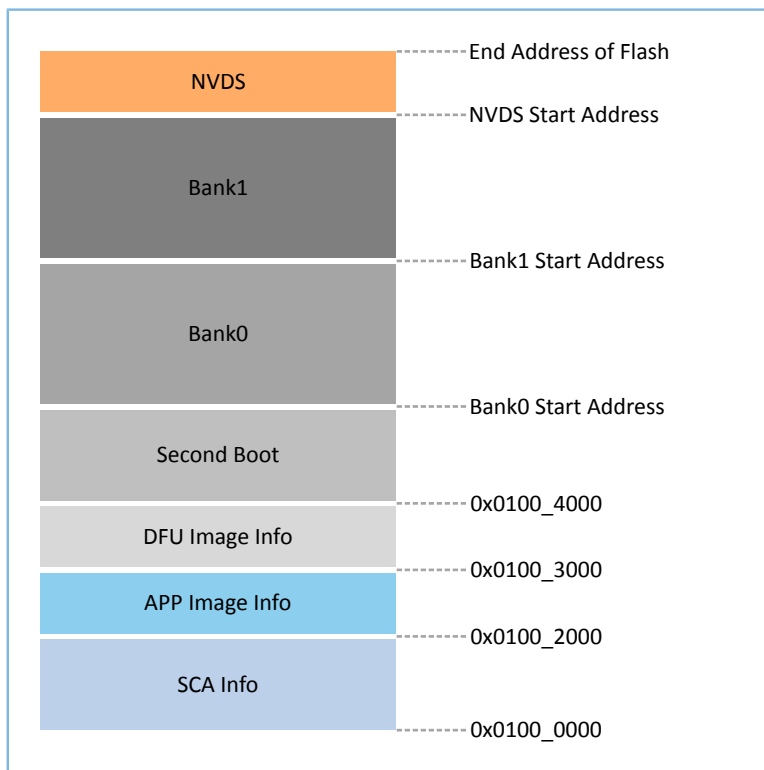



图 2-1 Second Boot的Flash布局

- SCA Info: SCA (System Configuration Area) 系统配置区，主要用于存储系统信息和Second Boot固件的Boot Info。
- APP Image Info: 应用固件信息区，用于存放应用固件运行的相关设置信息。
- DFU Image Info: DFU固件信息区，用于DFU固件拷贝之前的有效性检查。
- Second Boot: Second Boot示例存放和运行区。
- Bank0: 应用固件存放和运行区。
- Bank1: DFU固件缓存区，通过有效性检查的DFU固件将被拷贝至Bank0。
- NVDS (Non-volatile Data Storage): 非易失性数据存储区。

3 初次运行

本章主要介绍如何运行和验证GR551x Second Boot示例。

 说明:

SDK_Folder为GR551x SDK的根目录。

3.1 准备工作

运行Second Boot示例之前，请完成以下准备工作。

- 硬件准备

表 3-1 硬件准备

| 名称 | 描述 |
|-----------|--|
| J-Link工具 | SEGGER公司推出的JTAG仿真器，如需更多了解，请访问： http://www.segger.com/products/debug-probes/j-link/ |
| 开发板 | GR5515 Starter Kit开发板（以下简称“开发板”） |
| 数据线 | Micro USB 2.0数据线 |
| Android手机 | Android 5.0（KitKat）及以上版本 |

- 软件准备

表 3-2 软件准备

| 名称 | 描述 |
|----------------------|---|
| Windows | Windows 7/10 |
| J-Link Driver | J-Link驱动程序，下载网址： http://www.segger.com/downloads/jlink/ |
| Keil MDK5 | IDE工具，支持MDK-ARM 5.20 及以上版本，下载网址： www.keil.com/download/product/ |
| GProgrammer（Windows） | Programming工具，位于SDK_Folder\tools\GProgrammer |
| GRUart（Windows） | 串口调试工具，位于SDK_Folder\tools\GRUart |
| GRToolbox（Android） | BLE调试工具，位于SDK_Folder\tools\GRToolbox |

3.2 固件烧录

本文涉及second_boot.bin固件和ble_tem_dfu.bin固件的烧录，在烧录固件前需使用GProgrammer将GR551x SoC中的Flash全擦除。

使用GProgrammer烧录固件至开发板前，还需执行以下操作：

- ble_tem_dfu.bin固件：需在Keil中使能USE_SECOND_BOOT_MODE（详见[3.3.2 应用固件校验跳转运行验证](#)）后重新编译固件，再烧录到开发板。

- **second_boot.bin**固件：需配置**user_config.h**，该文件用于配置Second Boot工程的参数和公钥Hash值，文件位于：SDK_Folder\projects\ble\dfu\second_boot\Src\config。配置完成后需重新编译固件，再烧录到开发板。

表 3-3 user_config.h中的参数

| 宏 | 描述 |
|------------------------------------|---|
| BOOTLOADER_DEFAULT_STRATEGY_ENABLE | <p>是否使用默认的固件覆盖升级、校验、跳转策略。</p> <ul style="list-style-type: none"> ◦ 0: 使用自定义的固件覆盖升级、校验、跳转策略 ◦ 1: 使用默认的固件覆盖升级、校验、跳转策略 |
| BOOTLOADER_WDT_ENABLE | <p>是否开启Second Boot看门狗。</p> <ul style="list-style-type: none"> ◦ 0: 关闭Second Boot看门狗 ◦ 1: 开启Second Boot看门狗 |
| BOOTLOADER_OTA_ENABLE | <p>是否开启Second Boot空中升级（OTA）功能</p> <ul style="list-style-type: none"> ◦ 0: 关闭Second Boot空中升级（OTA）功能 ◦ 1: 开启Second Boot空中升级（OTA）功能 |
| BOOTLOADER_SIGN_ENABLE | <p>是否开启Second Boot加签验签方案，当BOOTLOADER_DEFAULT_STRATEGY_ENABLE开启时有效。</p> <ul style="list-style-type: none"> ◦ 0: 关闭Second Boot验签方案 ◦ 1: 开启Second Boot验签方案 <p>说明： 使用安全验签功能的具体操作请参考3.3.3 安全验签验证章节。</p> |
| USER_FW_COMMENTS | <p>应用固件comments定义，当BOOTLOADER_DEFAULT_STRATEGY_ENABLE开启（值为1）时有效。</p> <p>查找应用固件Image Info时是通过应用固件comments进行匹配，该定义最大长度为12 Bytes。</p> <p>当前默认值为“ble_tem_dfu”。</p> |
| APP_FW_RUN_ADDRESS | <p>应用固件运行地址，当BOOTLOADER_DEFAULT_STRATEGY_ENABLE（值为0）关闭时有效。</p> <p>说明： 请参考4.2.3 自定义固件拷贝升级、校验和跳转实现章节。</p> |

使用GProgrammer的具体操作方法请参考《GProgrammer用户手册》。

说明:

- *second_boot.bin*固件位于: SDK_Folder\projects\ble\dfu\second_boot\build\, 默认运行地址为0x01004000。
- *ble_tem_dfu.bin*固件位于: SDK_Folder\projects\ble\ble_peripheral\ble_app_template_dfu\build, 默认运行地址为0x01040000。
- 如果修改了*second_boot.bin*固件和*ble_tem_dfu.bin*固件的运行地址, 需确保两个固件存储空间不存在冲突。
- 如果使用自定义固件拷贝升级、校验和跳转策略, 需要将BOOTLOADER_DEFAULT_STRATEGY_ENABLE设置为0, 然后自行实现vendor_fw_copy_update()和vendor_fw_verify()。

3.3 测试验证

本章节将基于Second Boot OTA、应用固件校验跳转运行和安全验签功能来快速验证Second Boot示例。

3.3.1 Second Boot OTA验证

1. 在使用GProgrammer将Second Boot固件烧录至开发板前, 先将GR551x SoC中的Flash全擦除以确保Flash中不存在任何OTA拷贝任务和应用固件。
2. 将Second Boot固件烧录至开发板后, 则进入Second Boot OTA等待固件升级(原理参考[4.2 交互流程及代码介绍](#)中的步骤3), 此时GRUart串口界面如下图所示。

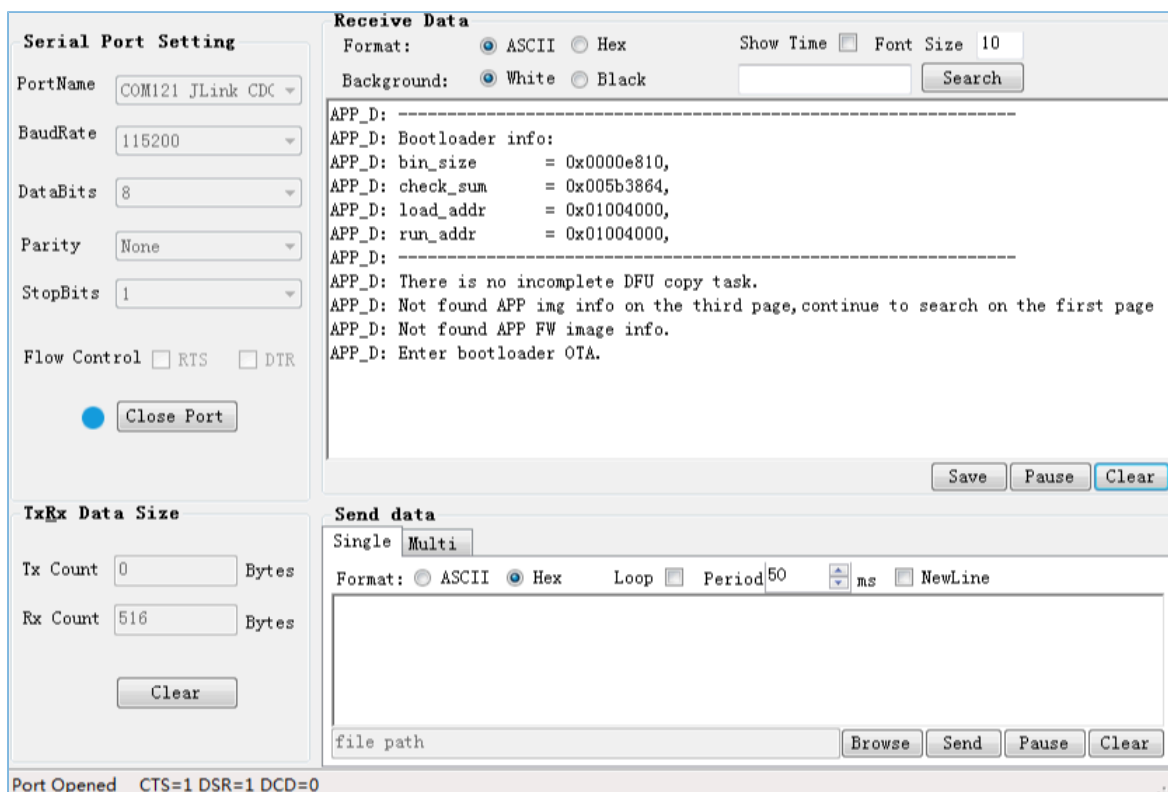


图 3-1 擦除Flash后进入Second Boot OTA

3. 开启手机端的蓝牙功能，打开GRToolbox，在扫描列表中发现“Goodix_Boot”，表明Second Boot固件运行正常。
4. Second Boot固件中集成了BLE OTA功能，可参考《GR551x OTA示例手册》“在ble_app_template_dfu中进行固件升级”章节升级固件，升级完成之后会自动跳转至新的应用固件运行。此时GRUart串口界面如下图所示。

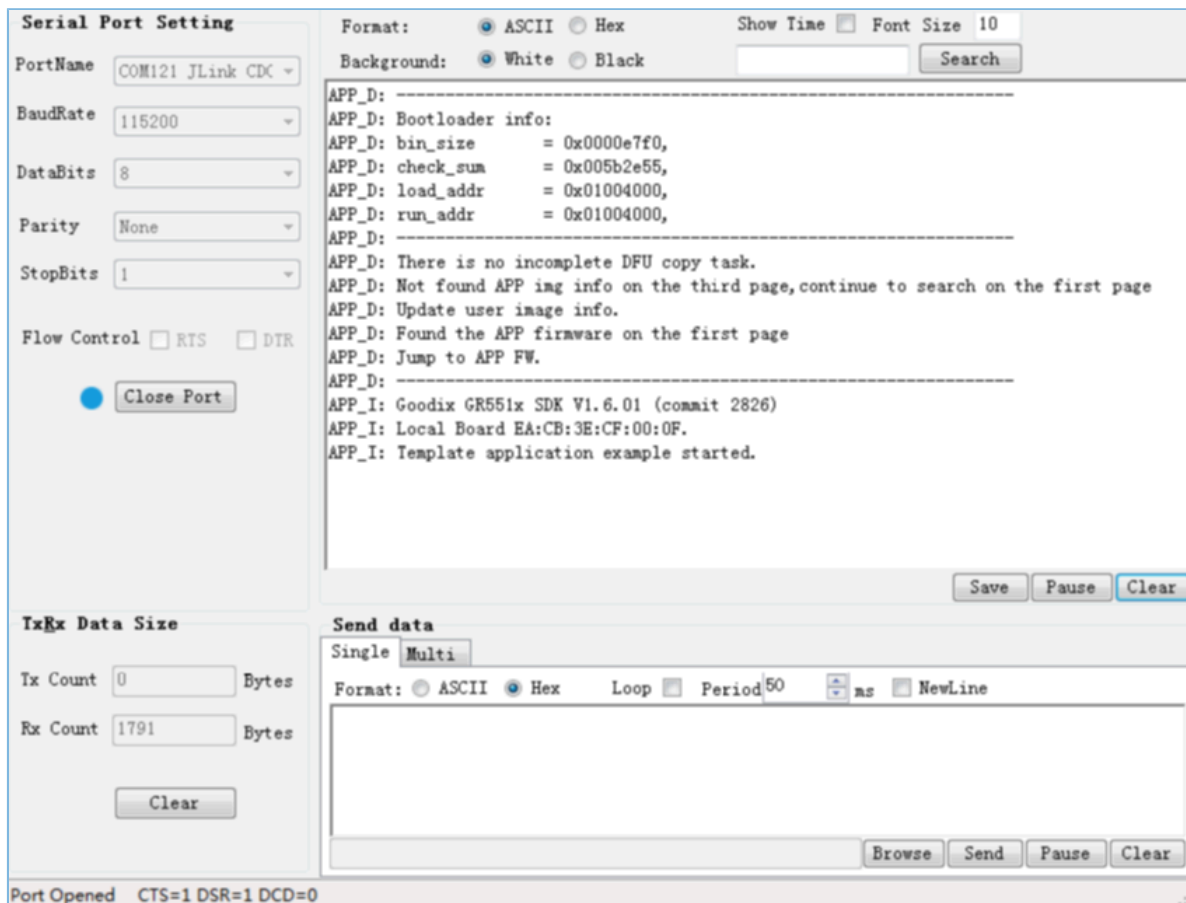



图 3-2 升级完成后应用固件成功运行

说明:

- 使用Second Boot模式进行OTA升级时，在GRToolbox的“固件升级”界面中需勾选“拷贝升级模式”。勾选后，“Copy Address”指向区域的内容将被覆盖，若配置不当会造成该区域原有信息丢失。
- 使用Second Boot模式进行OTA升级后，在GProgrammer中将无法显示升级后的固件信息。

3.3.2 应用固件校验跳转运行验证

1. 使用GProgrammer工具先将GR551x SoC中的Flash全擦除以确保Flash中不存在任何OTA拷贝任务和应用固件。
2. 在Keil中修改ble_app_template_dfu示例工程配置后重新编译固件。具体操作如下：

- (1) 进入示例工程目录SDK_Folder\projects\ble\ble_peripheral\ble_app_template_dfu\Keil_5，双击`ble_app_template_dfu.uvprojx`，在Keil中打开示例工程。
- (2) 点击Keil工具栏中的“Options for Target”  图标，在弹出的“Options for Target ‘GR551x_SK’”窗口中选择“C/C++”标签页。
- (3) 在“Preprocessor Symbols”面板的“Define”参数值中添加“USE_SECOND_BOOT_MODE”，如下图所示。

说明:

添加的“USE_SECOND_BOOT_MODE”需要与前面的宏以“,”分隔。

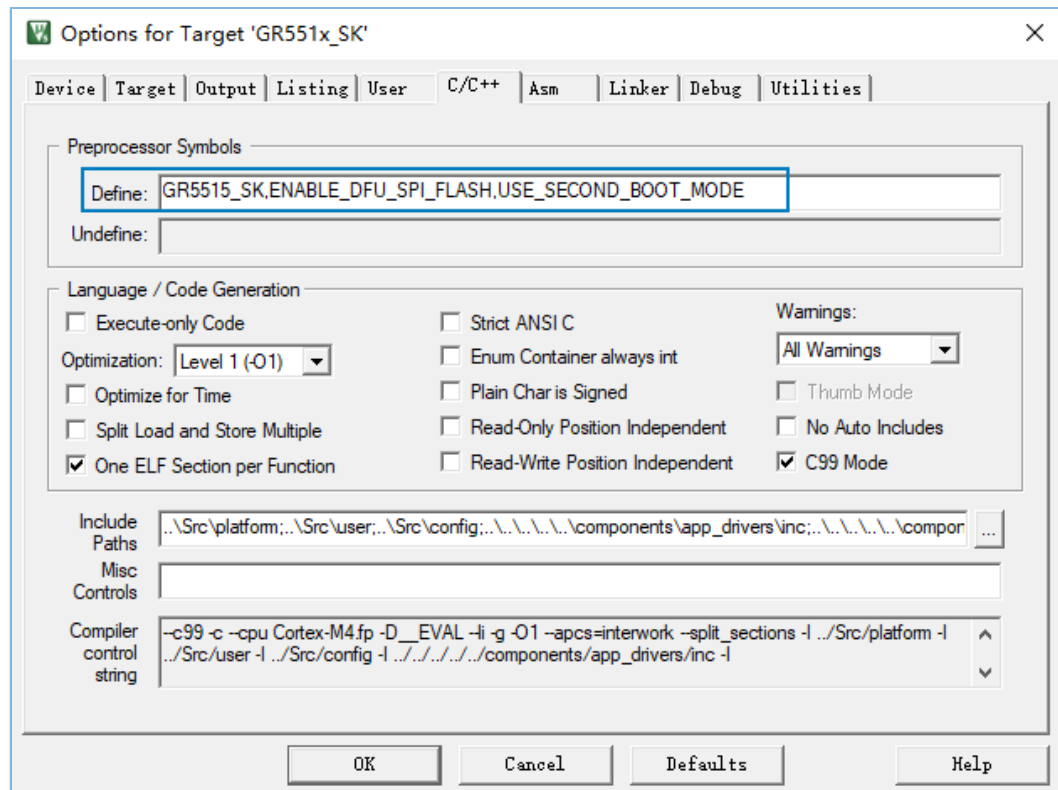



图 3-3 使能Second Boot模式

- (4) 保存设置后，点击Keil工具栏中的  图标，编译示例工程，生成固件bin文件。
3. 使用GProgrammer将Second Boot固件和应用固件`ble_tem_dfu.bin`烧录至开发板，并设置Second Boot固件作为Startup固件。

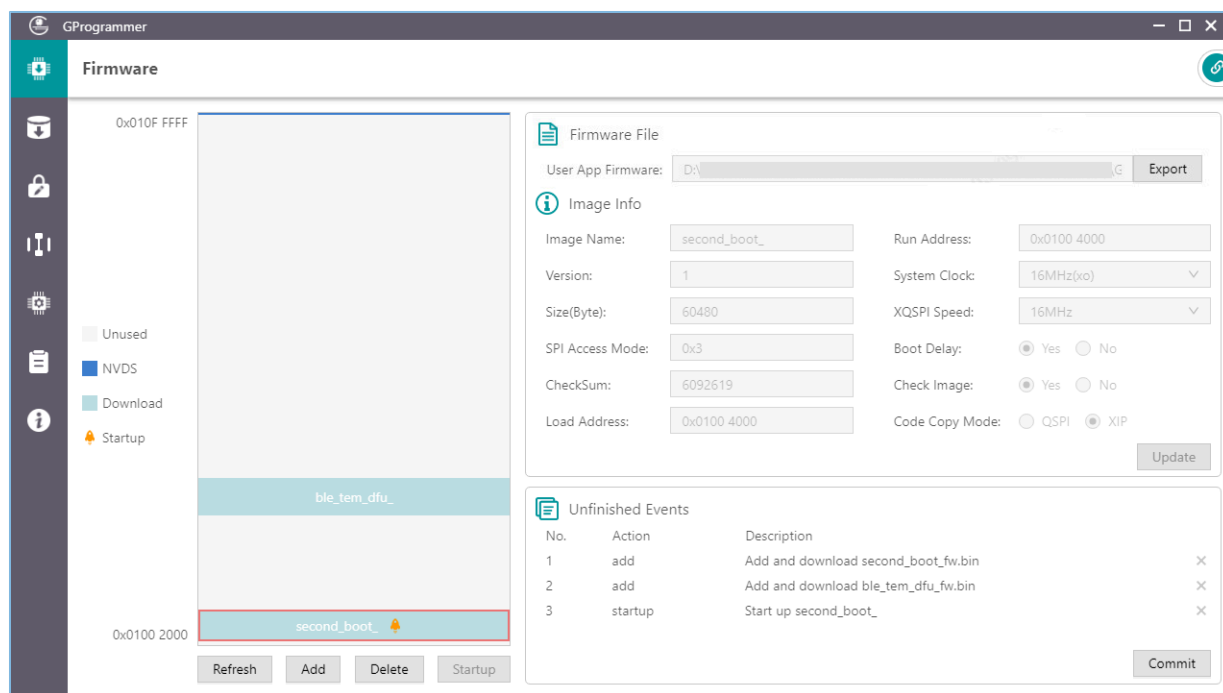


图 3-4 选中Second Boot固件作为Startup固件

4. GR551x SoC系统启动时检测到存在该应用固件`ble_tem_dfu.bin`，且通过有效性检查。会跳转至应用固件存储的起始地址，并开始运行程序，此时GRUart串口界面如下图所示。

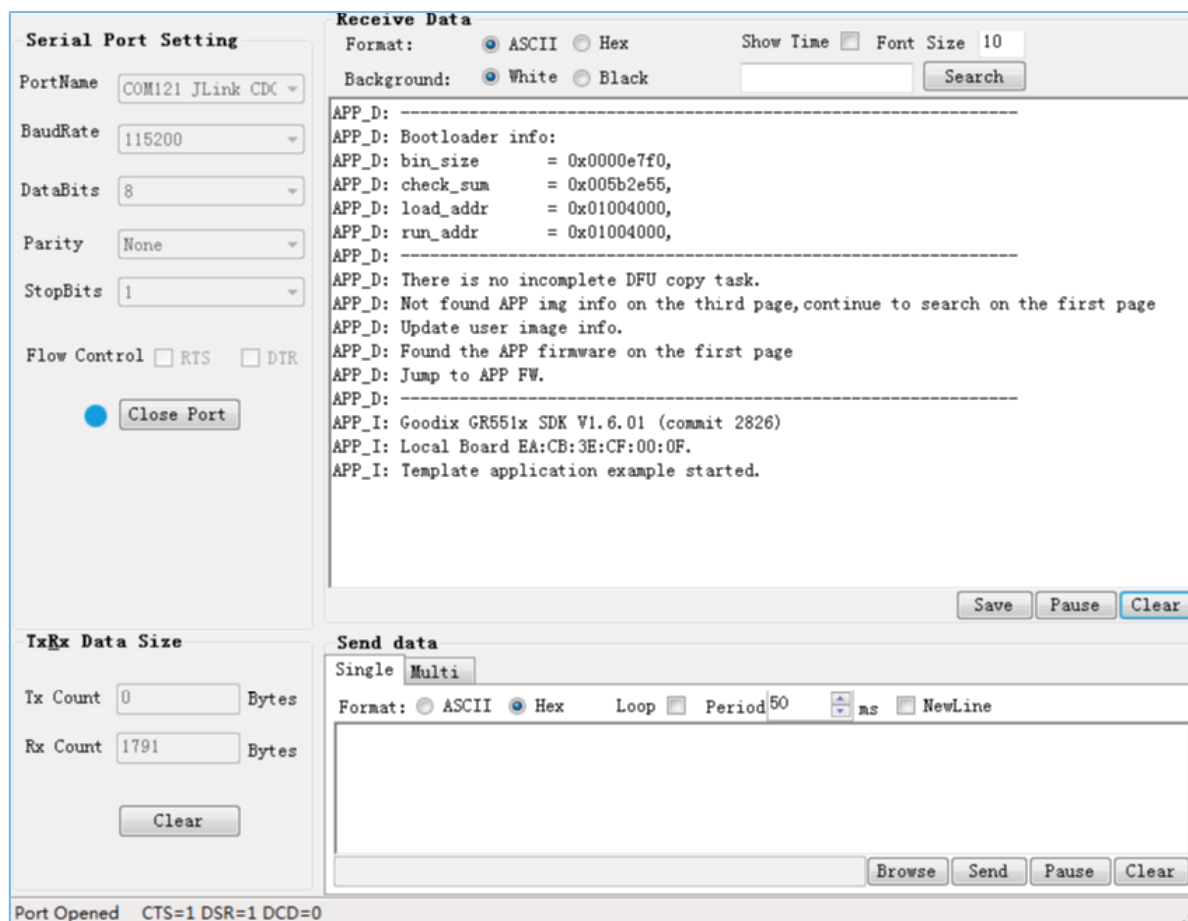


图 3-5 升级完成后应用固件成功运行

3.3.3 安全验签验证

Second Boot示例支持OTA固件安全验签功能，用户可根据自身使用场景，选择是否开启该验签功能。如开启该验签功能，可在Second Boot示例工程目录下的`user_config.h`中，设置“`BOOTLOADER_SIGN_ENABLE = 1`”。

在验签前，用户可先使用GProgrammer对应用固件进行加签，完整的加签和验签过程如下：

1. 生成私钥和公钥Hash值

生成签名信息的具体操作请参考《GProgrammer用户手册》“加密加签”章节，原理可参考《GR551x固件加密及应用介绍》“数字签名技术”章节。

通过GProgrammer生成的用于加密加签的文件如下：

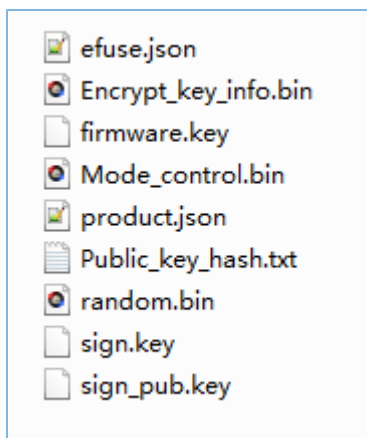


图 3-6 生成的用于加密加签的文件

2. 加签固件

导入`product.json`文件和应用固件`ble_tem_dfu.bin`，再点击“Sign”按钮，如图 3-7：

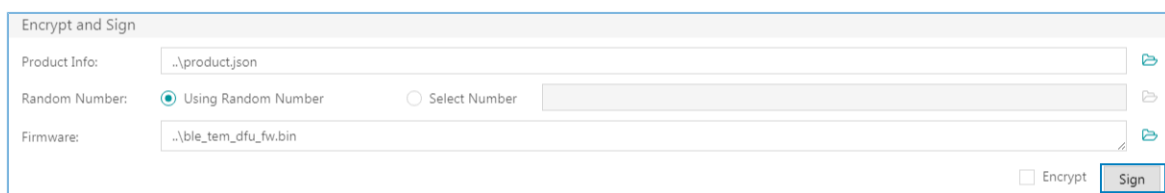


图 3-7 应用固件加签

选择加签文件路径后，即可生成加签应用固件（文件名带有`_sign`，`ble_tem_dfu_fw_sign.bin`），如图 3-8所示：

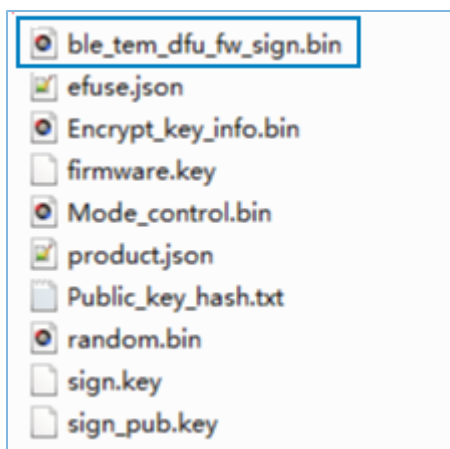


图 3-8 生成的加签固件

3. 将`Public_key_hash.txt`中的公钥Hash值复制到Second Boot工程`user_config.h`的`public_key_hash`数组后，重新编译Second Boot固件。

```
//Hash value of the signed public key
static const uint8_t public_key_hash[] =
{
    0x08, 0x57, 0x41, 0xDD, 0x34, 0x17, 0x0C, 0x01, 0x43, 0xFB, 0xCA, 0xA5, 0x5C, 0x51, 0x81, 0xF5
};
```

4. 对加签固件进行验证

利用GProgrammer将重新编译生成的Second Boot固件`second_boot.bin`和加签的`ble_tem_dfu_fw_sign.bin`下载至开发板，并设置Second Boot固件作为Startup固件并运行。Second Boot固件会对`ble_tem_dfu_sign.bin`固件进行校验和验签，若验证成功，则跳转到该应用固件中运行，如下图所示：

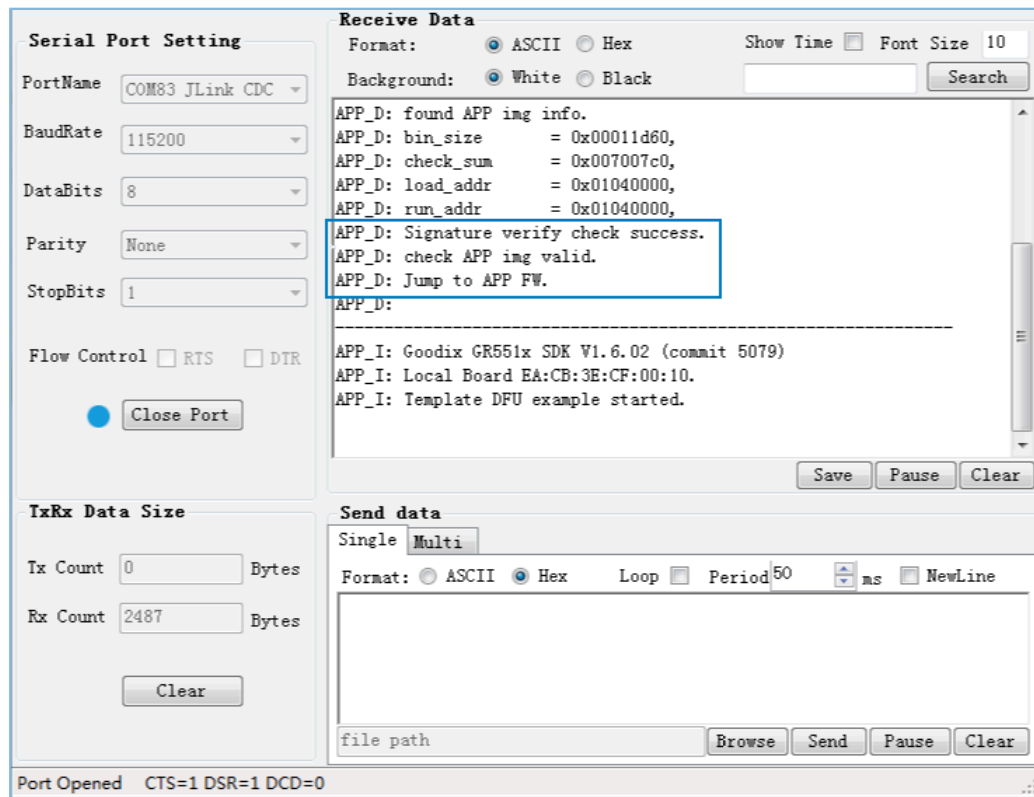


图 3-9 对加签固件进行验证

4 应用详解

本章主要介绍Second Boot示例的交互流程及相关代码。

4.1 工程目录

Second Boot示例的源代码和工程文件位于：SDK_Folder\projects\ble\dfu\second_boot\Keil_5。

双击打开second_boot.uvprojx工程文件，在Keil中查看Second Boot示例工程目录结构，相关文件介绍如表 4-1 示。

表 4-1 Second Boot工程文件说明

| Group | 文件 | 描述 |
|---------------|-----------------------------|--------------------------------|
| gr_profiles | otas.c | OTA Service实现 |
| user_callback | user_gap_callback.c | GAP Callback实现，如连接、断连、GAP参数更新等 |
| | user_gatt_common_callback.c | GATT Common Callback实现，如MTU更新 |
| user_platform | user_periph_setup.c | APP Log和看门狗的配置 |
| user_app | main.c | main()入口函数 |
| | user_app.c | 实现OTA Service初始化与BLE事件处理 |
| | user_dfu.c | 实现DFU初始化 |
| | oc_board.h | 实现固件有效性检查和固件跳转 |
| | sign_verify.lib | 实现固件签名验证的静态库 |
| | user_config.h | 看门狗、固件验签相关配置 |

4.2 交互流程及代码介绍

本节介绍Second Boot示例DFU固件拷贝升级、应用固件校验跳转运行的主要代码，便于用户深入了解Second Boot示例的运行机制。

Second Boot示例运行流程如图 4-1所示。

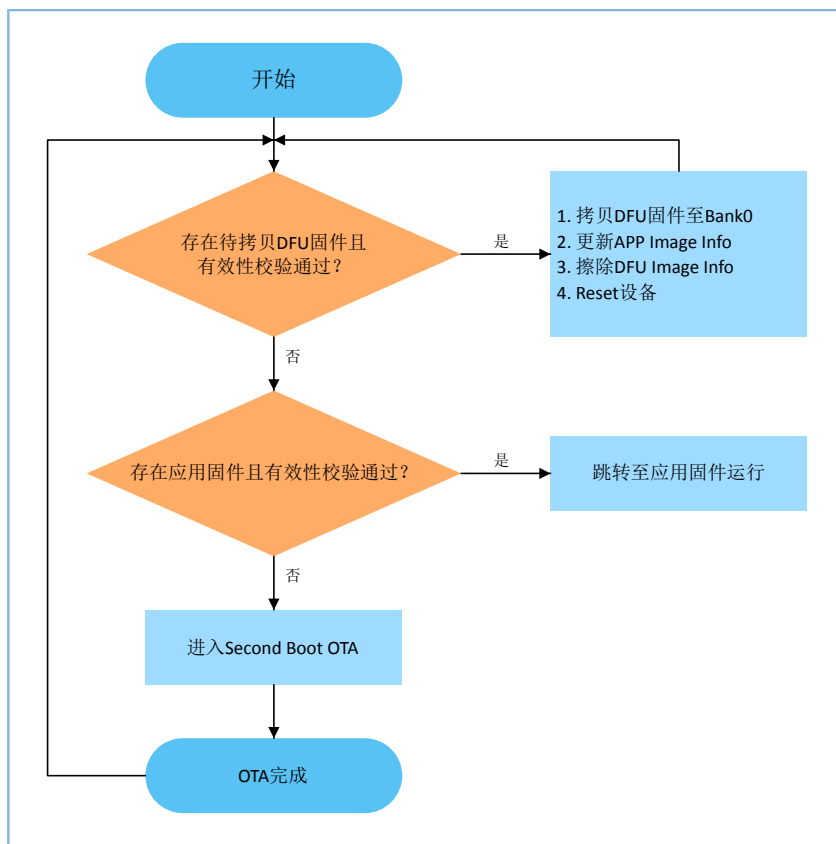


图 4-1 Second Boot示例运行流程

1. 读取DFU Image Info。当Bank1中存在需要拷贝至Bank0的DFU固件，且其有效性校验通过，则执行步骤2；如未通过，则执行步骤3。
2. 拷贝Bank1中的DFU固件至Bank0，更新APP Image Info和擦除DFU Image Info后，复位GR551x设备。
3. 读取APP Image Info。当Bank0中存在应用固件，且其有效性校验通过，则跳转至应用固件运行；如未通过，则执行步骤4。
4. 进入Second Boot OTA。OTA结束后，更新APP Image Info，复位GR551x设备。

4.2.1 DFU固件拷贝升级

GR551x的应用固件进行OTA时采用双区后台升级方式。

1. 将DFU固件存放至Bank1区域，并更新DFU Image Info区域信息。
2. 复位GR551x设备后运行Second Boot固件，将Bank1区域中的DFU固件拷贝至Bank0区域运行。

DFU固件拷贝升级主要代码介绍如下。

路径：工程目录下的user_app\user_boot.c

名称：is_fw_need_copy();

is_fw_need_copy()函数用于读取DFU Image Info，判断是否存在DFU固件拷贝任务。

```
static bool is_fw_need_copy(void)
```

```

{
    copy_load_addr = 0;
    hal_flash_read_judge_security(IMG_INFO_DFU_ADDR, (uint8_t*)&copy_load_addr, 4);
    memset((uint8_t*)&dfu_img_info, 0, sizeof(img_info_t));
    hal_flash_read_judge_security(IMG_INFO_DFU_ADDR+4, (uint8_t*)&dfu_img_info,
                                sizeof(img_info_t));

    if (dfu_img_info.pattern != 0x4744 ||\
        (memcmp(dfu_img_info.comments, USER_FW_COMMENTS, strlen(USER_FW_COMMENTS)) != 0))
    {
        APP_LOG_DEBUG("There is no incomplete DFU copy task.");
        return false;
    }

    APP_LOG_DEBUG("-----");
    APP_LOG_DEBUG("copy addr      = 0x%08x", copy_load_addr);
    APP_LOG_DEBUG("DFU fw boot info:");
    log_boot_info(&dfu_img_info.boot_info);
    APP_LOG_DEBUG("-----");

    APP_LOG_DEBUG("There is incomplete DFU copy task.");
    return true;
}

```

路径：工程目录下的user_app\user_boot.c

名称：incplt_dfu_task_continue();

incplt_dfu_task_continue()函数用于检查DFU固件有效性，有效性通过之后将DFU固件从Bank1区域拷贝至Bank0区域，并更新APP Image Info，擦除DFU Image Info，然后复位设备，程序将跳转到新的应用固件并运行，示例代码如下。

```

static void incplt_dfu_task_continue(void)
{
    if (!boot_fw_valid_check(copy_load_addr, &dfu_img_info.boot_info))
    {
        APP_LOG_DEBUG("DFU FW image valid check fail.");
        return;
    }
    if(copy_load_addr != dfu_img_info.boot_info.load_addr)
    {
        uint32_t copy_size = dfu_img_info.boot_info.bin_size + 48;
        APP_LOG_DEBUG("DFU FW image start copy.");

        if(sys_security_enable_status_check())
        {
            copy_size += 856;
        }
        else
        {
            #if BOOTLOADER_SIGN_ENABLE

```

```

        copy_size += 856;
    #endif
}
dfu_fw_copy(dfu_img_info.boot_info.load_addr, copy_load_addr, copy_size);
}
user_img_info_update(&dfu_img_info);
hal_flash_erase(IMG_INFO_DFU_ADDR, CODE_PAGE_SIZE); //clear copy info
hal_nvic_system_reset();
}

```

4.2.2 应用固件校验跳转运行

当不存在应用固件OTA拷贝升级任务时，**Second Boot**会对该应用固件进行有效性检查并跳转至该应用固件中运行。

路径：工程目录下的user_app\user_boot.c

名称：is_jump_user_fw();

is_jump_user_fw()函数用于跳转到应用固件运行前的检查。

在is_jump_user_fw()函数中，通过读取APP Image Info的comments和实际应用固件的comments（USER_FW_COMMENTS）进行对比，判断两者comments是否相同，相同说明Bank0中存在应用固件，进一步校验APP Image Info有效性，有效性通过则可跳转到该应用固件运行。

如comments不相同，说明Bank0中不存在应用固件，此时可通过读取SCA区域索引到的应用固件Image Info的comments，和USER_FW_COMMENTS对比，如两者comments相同，再校验该SCA中应用固件Image Info有效性，有效性通过，会将SCA中的应用固件Image Info更新至APP Image Info；如两者comments不同，或者有效性不通过，则不能跳转。

```

static bool is_jump_user_fw(void)
{
    memset((uint8_t*)&app_img_info, 0, sizeof(img_info_t));
    hal_flash_read_judge_security(IMG_INFO_APP_ADDR, (uint8_t*)&app_img_info,
                                  sizeof(img_info_t));

    if ((app_img_info.pattern == 0x4744) &&\
        (0 == memcmp(app_img_info.comments, USER_FW_COMMENTS, strlen(USER_FW_COMMENTS))))
    {
        APP_LOG_DEBUG("found APP img info.");
        log_boot_info(&app_img_info.boot_info);
        if (boot_fw_valid_check(app_img_info.boot_info.load_addr, &app_img_info.boot_info))
        {
            APP_LOG_DEBUG("check APP img valid.");
            return true;
        }
    }
    APP_LOG_DEBUG("Not found APP img info on the third page,continue to search on the first page");

    img_info_t img_info_main;

```

```
for (uint8_t i = 0; i < IMG_INFO_SAVE_NUM_MAX; i++)
{
    fw_img_info_get(BOOT_INFO_ADDR + 0x40, i, &img_info_main);

    if (0 == memcmp(img_info_main.comments, USER_FW_COMMENTS, strlen(USER_FW_COMMENTS)))
    {
        if (boot_fw_valid_check(img_info_main.boot_info.load_addr,
                                &img_info_main.boot_info))
        {
            user_img_info_update(&img_info_main);
            memcpy(&app_img_info, &img_info_main, sizeof(img_info_t));
            APP_LOG_DEBUG("Found the APP firmware on the first page");
            return true;
        }
    }
}

APP_LOG_DEBUG("Not found APP FW image info.");
return false;
}
```

路径：工程目录下的user_app\user_boot.c

名称1：jump_user_fw();

名称2：sec_boot_jump();

跳转前需更新热启动Boot Info、设置MSP、重定位向量表。

```
static void jump_user_fw(void)
{
    APP_LOG_DEBUG("Jump to APP FW.");
    APP_LOG_DEBUG("-----");
    sec_boot_jump(&app_img_info.boot_info);
}

static void sec_boot_jump(boot_info_t *p_boot_info)
{
    extern void rom_init(void);
    extern void jump_app(uint32_t addr);
    extern boot_info_t bl1_boot_info;
    extern void bl_xip_dis(void);
    uint16_t enc_mode = *(uint16_t*)0x30000020;
    bool mirror_mode = false;

    if(p_boot_info->run_addr != p_boot_info->load_addr)//mirror mode
    {
        mirror_mode = true;
        if(!enc_mode)
            SET_CODE_LOAD_FLAG();
        memcpy((uint8_t*)p_boot_info->run_addr, (uint8_t*)p_boot_info->load_addr,
               p_boot_info->bin_size);
    }
}
```

```
    }
    if(enc_mode)
    {
        REG(0xA000C578UL) &= ~0xFFFFFC00;
        REG(0xA000C578UL) |= (p_boot_info->run_addr & 0xFFFFFC00);
    }

    rom_init();
    __disable_irq();
    SysTick->CTRL = 0;
    __enable_irq();
    memcpy(&bl1_boot_info, p_boot_info, sizeof(boot_info_t));
    if(mirror_mode)
    {
        if(enc_mode)
        {
            REG(0xa000d470) = ENCRY_CTRL_DISABLE;
        }
    }
    jump_app(p_boot_info->run_addr);
}
```

说明:

为了在睡眠唤醒后热启动时，可直接跳转至应用固件，需先将应用固件boot info赋值给全局变量bl1_boot_info，即memcpy(&bl1_boot_info, p_boot_info, sizeof(boot_info_t));，此处切勿改动。

4.2.3 自定义固件拷贝升级、校验和跳转实现

当不采用默认的固件拷贝升级、校验和跳转策略，则可以将BOOTLOADER_DEFAULT_STRATEGY_ENABLE设置为0，然后自行实现vendor_fw_copy_update()和vendor_fw_verify()以完成固件拷贝升级和校验。另外vendor_fw_jump()已实现跳转功能，也可自行扩展功能。

以上三个函数位于工程目录下的user_app\user_boot.c。

5 常见问题

本章描述了在使用及验证Second Boot示例时，可能出现的问题、原因及处理方法。

5.1 Second Boot安全OTA升级固件失败

- 问题描述

使用Second Boot OTA升级时，验签失败。

- 问题分析

升级固件验签公钥获取失败。

- 处理方法

确保加签私钥和验签公钥为一对，并将Public_key_hash.txt中的公钥Hash值复制到Second Boot工程user_config.h的public_key_hash数组。

5.2 应用固件睡眠后无法被唤醒

- 问题描述

应用固件配合Second Boot使用时，当应用固件睡眠后，无法被唤醒。

- 问题分析

Second Boot固件中校验和跳转流程代码被修改，当前应用固件的boot_info未赋值给全局变量bl1_boot_info，导致睡眠唤醒后热启动时失败。

- 处理方法

确保在sec_boot_jump()中将应用固件的boot_info赋值给全局变量bl1_boot_info。