实验手册： Matter 设备 OTA

Silicon Labs EFR32MG24 Breakout Board 评估板是一使用方便和低成本套件，可以用于 Matter Thread SoC 应用评估以及固件开发。

EFR32MG24 评估板上引出了 ***Mini-Simplicity*** 调试编程接口。Simplicity Studio IDE 可通过此接口进行固件烧写和在线代码级调试；同时也可以通过引脚 ***SWCLK\SWDIO\3V3\GND*** 接口进行烧录。

实验目的

- Matter 设备 OTA 功能需要用到哪些工具
- Matter 设备的固件在 Flash 中的分布
- 掌握芯科科技 Matter 设备 OTA 升级流程

EFR32MG24 评估板结合基于 EFR32MG21 的 USB Stick、J-Link 工具以及 Simplicity Studio 可以构成一套低成本的 Matter SoC 固件应用调试与开发工具。

Table of Contents

1 预备知识

本实验中涉及的专有名词及相关知识：

- 对 Ubuntu 开发环境有一定的使用基础
- 对 OpenThread 协议有一定的了解
- 对 C++有一定的基础
- 对 Matter 协议规范有个初步认识
- 名词解释
  - OT-RCP：Open Thread Radio Co-Processor。Thread 无线协处理器
  - OTBR：Open Thread Board Router。Thread 边界路由器
  - chip-tool：Linux 应用程序。用于 Matter 协议控制
  - ot-ctl：Thread 网络控制的应用程序

2 实验目的

通过本次实验，开发者可以理解并掌握以下内容：

- Matter 设备的 OTA 功能需要用到哪些工具
- Matter 设备的固件在 Flash 中的分布
- 掌握芯科科技 Matter 设备 OTA 升级流程

3 实验内容

1. 使用芯片内部 Flash 实现 OTA
2. 使用外部 Flash 实现 OTA

4 实验准备

4.1 硬件准备

- 安装 Ubuntu 虚拟机的电脑一台
- MG24 Breakout 开发板两块
- USB Stick (用作 OTBR)
- USB 线一条
- J-Link 烧录器一台
- BLE Dongle 一块
- 树莓派一台（可选）

4.2 软件准备

芯科科技将提供一个 ubuntu 镜像，已完全搭建好 Matter 开发环境，开发者可直接使用。

若需自行搭建环境的，可参考Matter 开发环境和 Matter SDK 环境两个章节。

4.2.1 Matter 开发环境

Matter 设备开发需要用到 Windows 环境和 Linux 环境，可根据实际情况进行选择。

- Windows 环境
  - Simplicity Commander
    - 烧录设备固件
    - 转换固件格式（比如：将 s37 文件转换成 gbl 文件）
  - Simplicity Studio
    - 创建设备工程、编译固件

- Linux 环境：Ubuntu
  - OTBR 控制环境
  - 如果是桌面环境，也可以安装 Simplicity Studio 和 Simplicity Commander
- Linux 环境：Raspberry Pi 4B (可选)
  - OTBR 控制环境
  - 芯科科技提供的系统镜像：SilabsMatterPi
    - 集成了 Matter SDK 和开发环境
    - 默认用户名和密码都为：ubuntu
- Linux 环境依赖安装

```
$ sudo apt-get install git gcc g++ pkg-config libssl-dev libdbus-1-dev libglib2.0-dev libavahi-client-dev
ninja-build python3-venv python3-dev python3-pip unzip libgirepository1.0-dev libcairo2-dev libreadline-dev
# 安装实用工具
$ sudo apt-get install net-tools openssh-server openssh-client
# 启用双向复制粘贴功能
$ sudo apt-get install virtualbox-guest-x11
$ sudo VBoxClient --clipboard
```

4.2.2 Matter SDK 环境

从 silicon labs 官方 github 获取 Matter 代码：

```
$ git clone https://github.com/SiliconLabs/matter.git
$ cd matter
# 同步子模块（受网速影响，可能需要比较长的时间）
$ ./scripts/checkout_submodules.py --shallow --recursive --platform efr32
# 检测并完善编译环境（对网络有要求，需要访问国外网站。受网速影响，可能需要约 1 小时左右）
$ source scripts/activate.sh
```

注：这个过程需要同步较多子模块，并且有些需要访问到国外网站，整个过程可能耗时可能在数小时或更长时间。这里推荐使用芯科科技提供的 ubuntu 镜像文件。

5 实验步骤

5.1 使用芯片内部 Flash 实现 OTA

由于芯片内部 Flash 空间只有 1536KB，在实现 OTA 功能时，需要关掉一些 log 来减小 code size。

1. 在`config/sl_matter_config.h`中修改日志配置

```
// <o SL_MATTER_STACK_LOCK_TRACKING_MODE> Stack Lock Tracking Mode
// <SL_MATTER_STACK_LOCK_TRACKING_NONE=> None
// <SL_MATTER_STACK_LOCK_TRACKING_LOG=> Log
// <SL_MATTER_STACK_LOCK_TRACKING_FATAL=> Fatal
// <d> SL_MATTER_STACK_LOCK_TRACKING_FATAL
#define SL_MATTER_STACK_LOCK_TRACKING_MODE  SL_MATTER_STACK_LOCK_TRACKING_NONE

// <o SL_MATTER_LOG_LEVEL> Log Level
// <SL_MATTER_LOG_NONE=> None
// <SL_MATTER_LOG_ERROR=> Error
// <SL_MATTER_LOG_PROGRESS=> Progress
// <SL_MATTER_LOG_DETAIL=> Detailed log (debug)
// <SL_MATTER_LOG_AUTOMATION=> Automation
// <d> SL_MATTER_LOG_ERROR
#define SL_MATTER_LOG_LEVEL SL_MATTER_LOG_NONE

// <q EFR32_LOG_ENABLED> Enable EFR32 specific log used in matter
#define EFR32_LOG_ENABLED 0
```

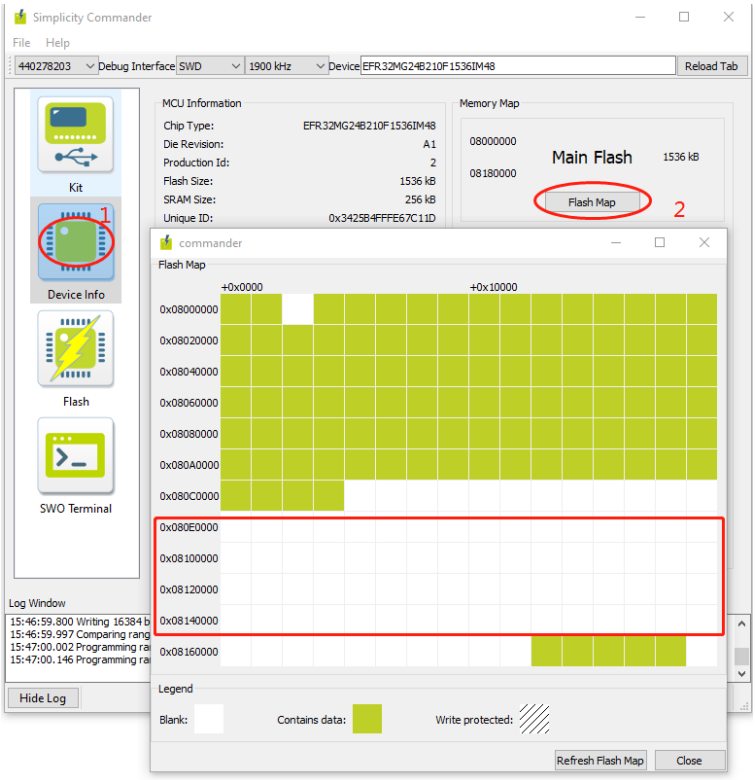
2. 在`include/CHIPProjectConfig.h`中修改软件版本号为 2

```
/**
 * CHIP_DEVICE_CONFIG_DEVICE_SOFTWARE_VERSION_STRING
 *
 * A string identifying the software version running on the device.
 * CHIP service currently expects the software version to be in the format
 * {MAJOR_VERSION}.0d{MINOR_VERSION}
 */
#ifndef CHIP_DEVICE_CONFIG_DEVICE_SOFTWARE_VERSION_STRING
#define CHIP_DEVICE_CONFIG_DEVICE_SOFTWARE_VERSION_STRING "0.2"  //modify "0.1ALPHA" to "0.2"
#endif

/**
 * CHIP_DEVICE_CONFIG_DEVICE_SOFTWARE_VERSION
 *
 * A uint32_t identifying the software version running on the device.
 */
/* The SoftwareVersion attribute of the Basic cluster. */
#ifndef CHIP_DEVICE_CONFIG_DEVICE_SOFTWARE_VERSION
#define CHIP_DEVICE_CONFIG_DEVICE_SOFTWARE_VERSION 0x0002   //modify 0x0001 to 0x0002
#endif
```

3. 重新编译并烧录固件

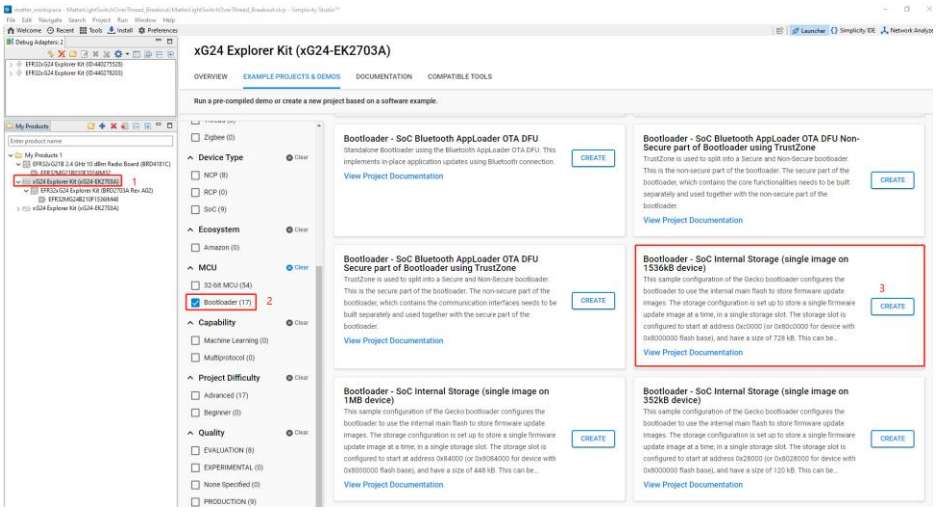
4. 查看 Flash Map



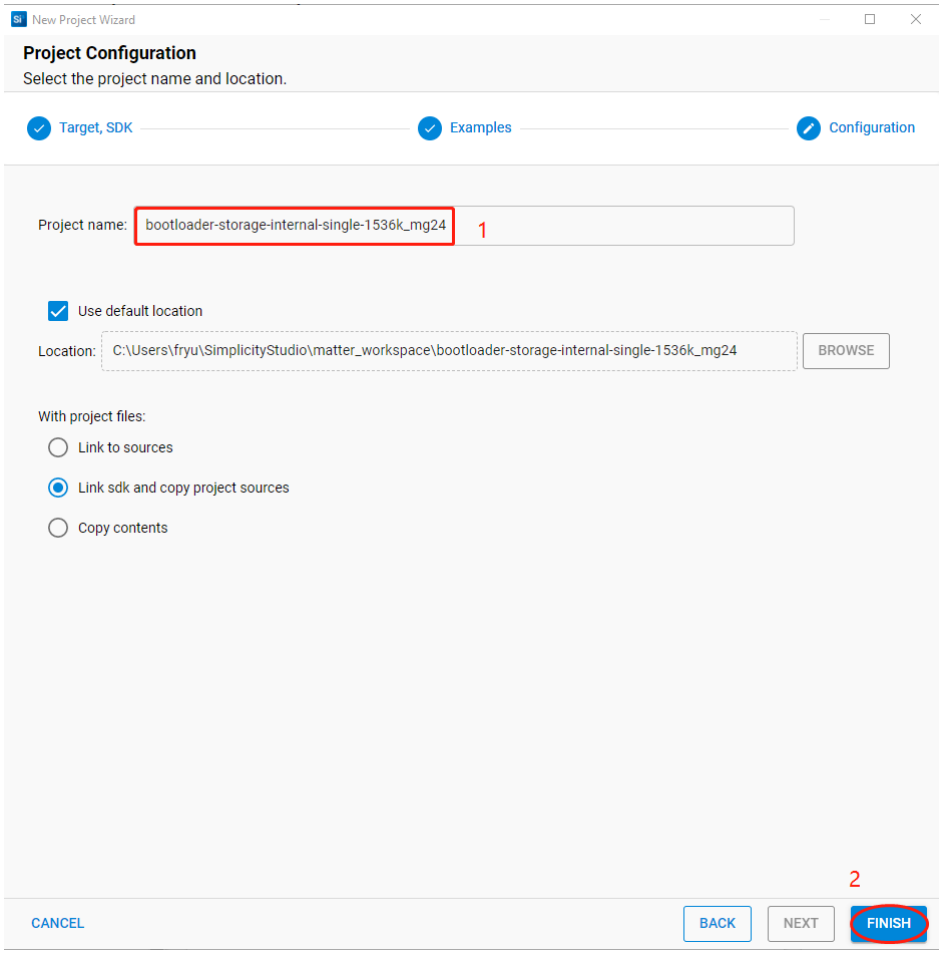
为存储 OTA 文件分配空间。这里按红框部分分配从 0x80e0000 开始的 512KB 大小的空间。后面在 bootloader 中添加 OTA 功能时会用到。

5.1.1 创建 Bootloader 工程：使用内部 Flash 实现 OTA

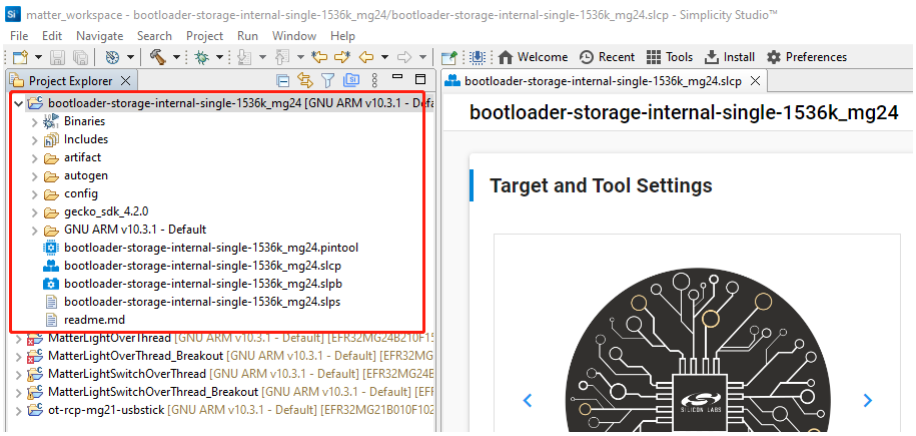
1. 选择`Bootloader-SoC Internal Storage(single image on 1536kB device)`示例创建工程。



2. 修改工程名为`bootloader-storage-internal-singal-1536k_mg24`。（这里也可以默认不修改）



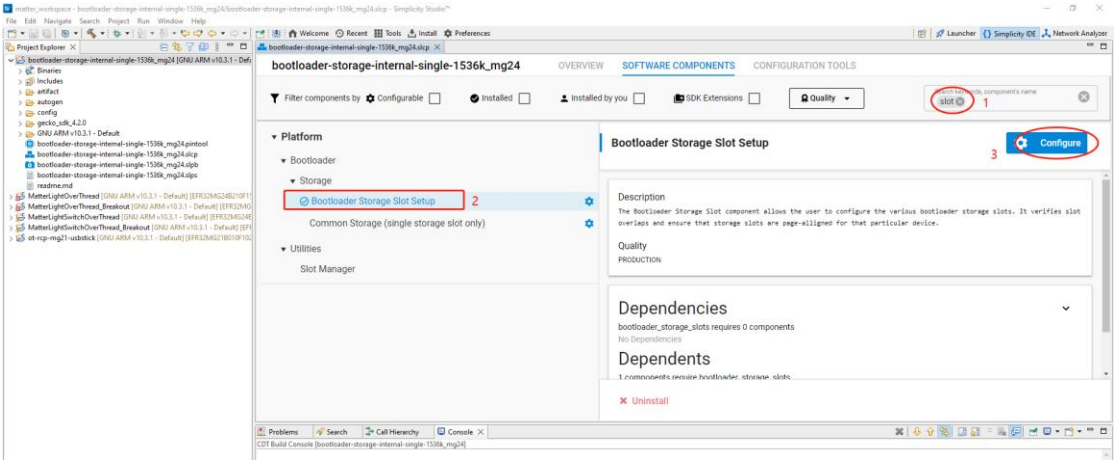
3. 工程创建完成后，如下：



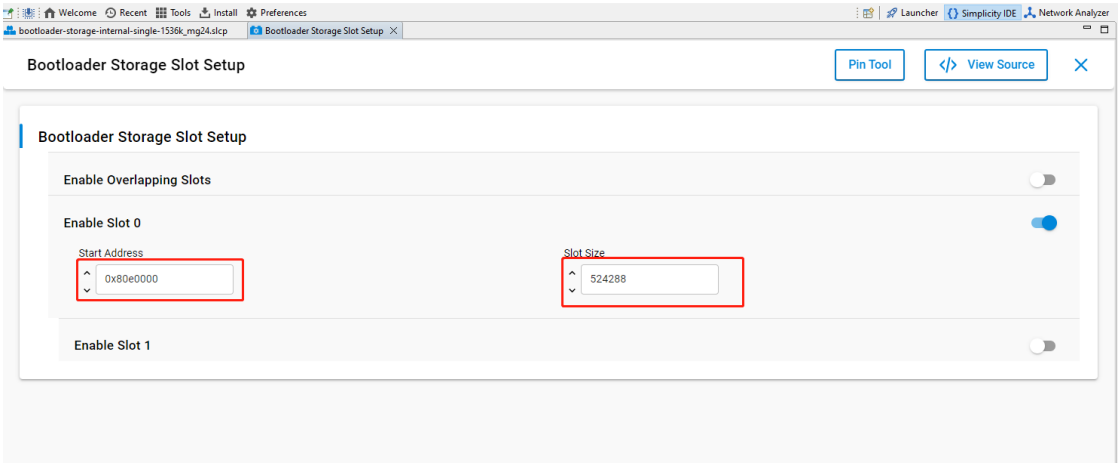
5.1.2 添加 OTA 功能

若需要支持 OTA 功能，需要参考此章节做一些修改。

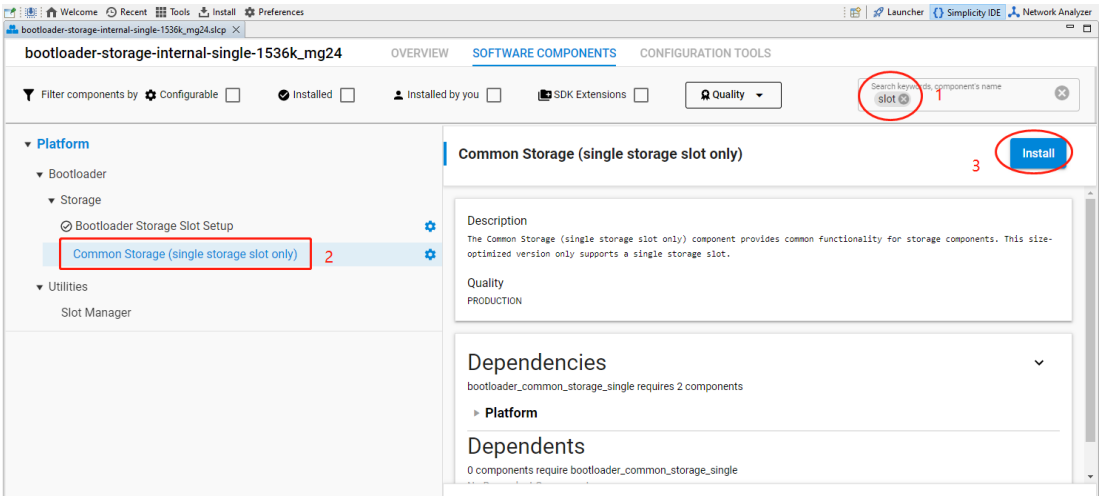
1. 进入`Bootloader Storage Slot Setup`组件，配置 Slot 的起始地址和大小，Slot 用来存放 OTA 文件。



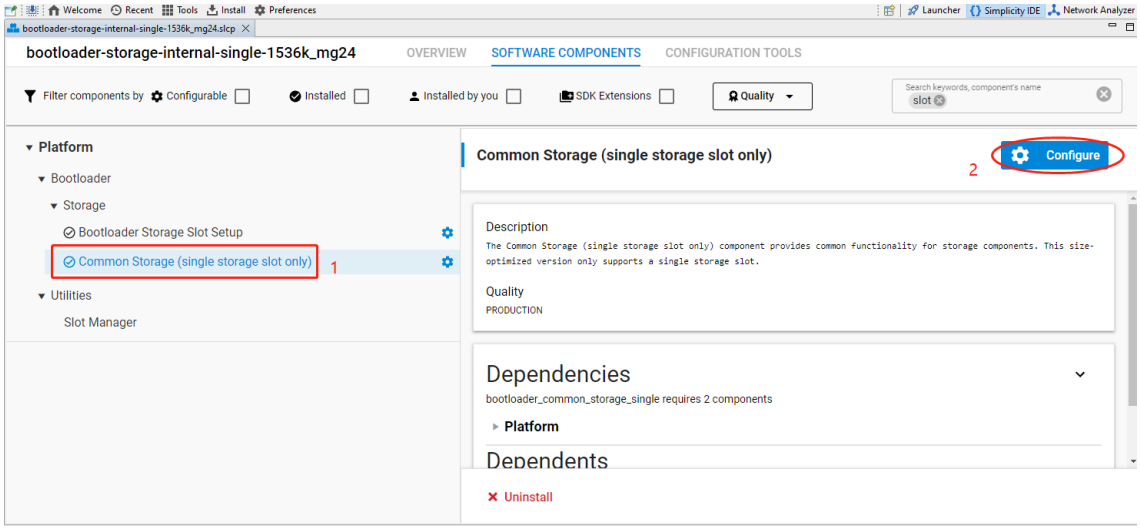
- 根据 Flash Map，我们可以将使用 LZMA 算法压缩后的 OTA 文件存储在 0x080E0000 到 0x0815ffff，大小为 512K。



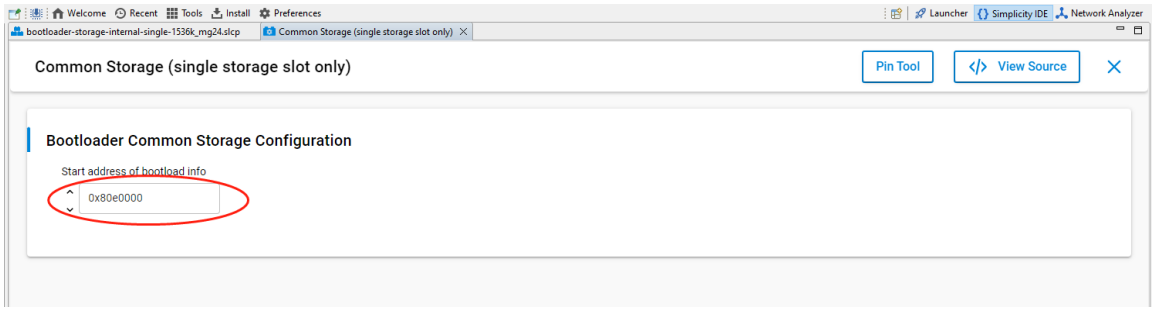
2. 安装`Common Storage`组件，并配置参数。



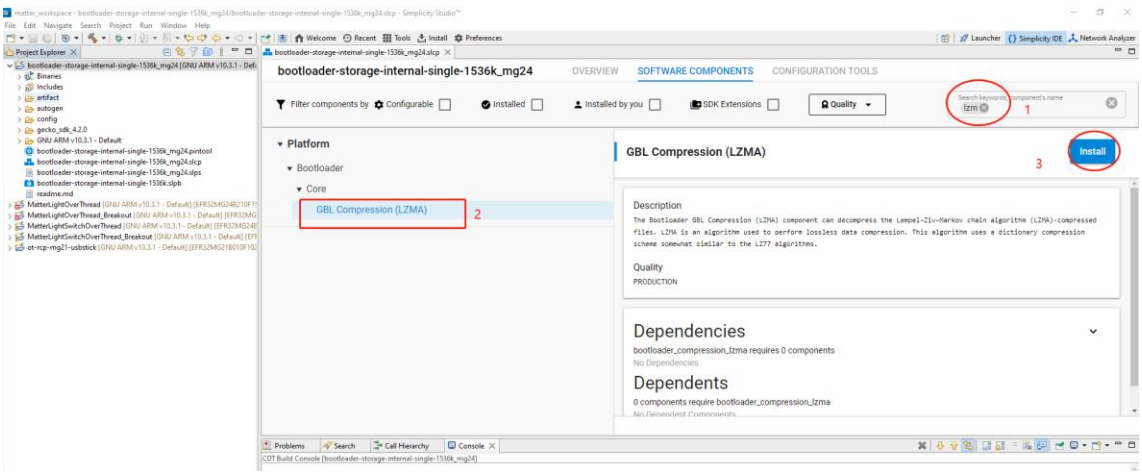
- 进入 Common Storage 配置页面：



- 参数配置注意事项：`Common Storage`配置页面中的 start address 需要与`Bootloader Storage Slot Setup`中设置的保持一致。

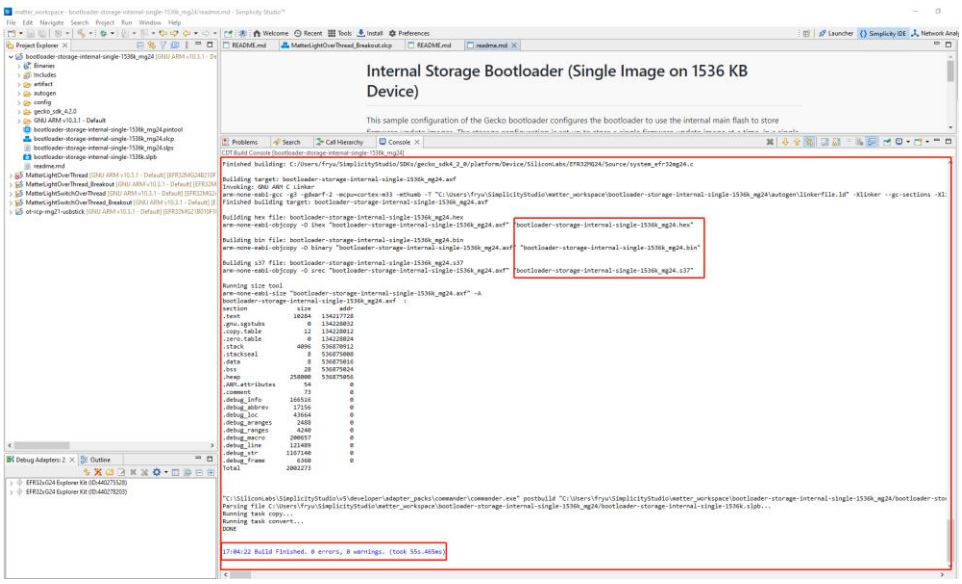


3. 安装 LZMA，以支持 OTA 压缩固件。



5.1.3 编译 Bootloader 固件

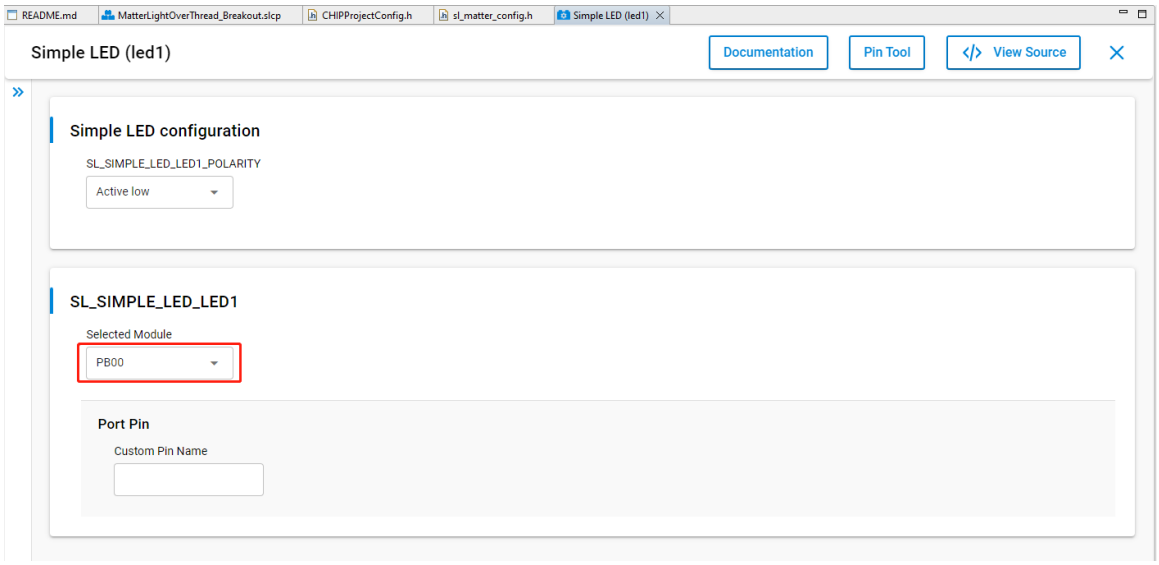
编译 Bootloader 固件：



编译完成后将生成固件 bootloader-storage-internal-single-1536k_mg24.s37。

5.1.4 修改设备固件

1. 修改灯的颜色：将 LED1 改为 PB00 控制(蓝色)。



2. 修改软件版本号。在`CHIPProjectConfig.h`中修改软件版本号，需要大于设备上的固件版本号。这里将版本号改为 3

```
/**
 * CHIP_DEVICE_CONFIG_DEVICE_SOFTWARE_VERSION_STRING
 *
 * A string identifying the software version running on the device.
 * CHIP service currently expects the software version to be in the format
 * {MAJOR_VERSION}.0d{MINOR_VERSION}
 */
#ifndef CHIP_DEVICE_CONFIG_DEVICE_SOFTWARE_VERSION_STRING
#define CHIP_DEVICE_CONFIG_DEVICE_SOFTWARE_VERSION_STRING "0.3"  //modify "0.2" to "0.3"
#endif

/**
 * CHIP_DEVICE_CONFIG_DEVICE_SOFTWARE_VERSION
 *
 * A uint32_t identifying the software version running on the device.
 */
/* The SoftwareVersion attribute of the Basic cluster. */
#ifndef CHIP_DEVICE_CONFIG_DEVICE_SOFTWARE_VERSION
#define CHIP_DEVICE_CONFIG_DEVICE_SOFTWARE_VERSION 0x0003  //modify 0x0002 to 0x0003
#endif
```

如果需要配网成功后关闭红灯：

```
            else if (sIsProvisioned && sIsEnabled)
            {
                if (sIsAttached)
                {
                    #ifdef ENABLE_WSTK_LEDS
                    sStatusLED.Set(false);// modify to false sStatusLED.Set(true);
                    #endif // ENABLE_WSTK_LEDS
                }
                else
                {
                    #ifdef ENABLE_WSTK_LEDS
                    sStatusLED.Blink(950, 50);
                    #endif
                }
            }
```

3. 重新编译固件。
4. 使用 Commander 工具生成 lzma 压缩的 GBL 文件

```
# 创建经 lzma 压缩压缩的 gbl 文件
$ commander gbl create ./MatterLightOverThread_Breakout_v3.gbl --app ./MatterLightOver-
Thread_Breakout_v3.s37 --compress lzma
```

5. 生成 OTA 固件，`-vn -vs`参数要与软件版本号一致

```
$ cd matter
# 通过 gbl 文件生成 ota 文件
$ ./src/app/ota_image_tool.py create -v 0xFFF1 -p 0x8005 -vn 3 -vs "0.3" -da sha256 ../firmware/Matter-
LightOverThread_Breakout_v3_lzma.gbl ../firmware/MatterLightOverThread_Breakout_v3_lzma.ota
```

完成以上步骤将获得升级用到的 OTA 固件`MatterLightOverThread_Breakout_green_red_v3-lzma.ota`。

5.1.5 OTA 升级操作

通过前面的步骤，我们已经获得了以下固件：

固件	说明
MatterLightOverThread_Breakout_v2.s37	待升级设备当前正在运行固件 v2
MatterLightOverThread_Breakout_v3-lzma.ota	通过 LZMA 压缩的 OTA 文件 v3
bootloader-storage-internal-single-1536k_mg24.s37	使用芯片内部 Flash 实现 OTA 的 bootloader
bootloader-storage-spiflash-single-1024k_24.s37	使用芯片外部 Flash 实现 OTA 的 bootloader

1. 烧录设备固件

将`MatterLightOverThread_Breakout_green_blue_v2.s37`和`bootloader-storage-internal-single-1536k_mg24.s37`烧录到测试设备中。**注：当使用外部 flash 升级时，只需要更换 bootloader 即可。**

2. 配置设备入网

```
# 清除 chip tool 的缓存。清除缓存后，OTA Provider 和 Matter 设备需要重新入网。
$ sudo rm -r /tmp/chip_*

# 创建 Thread 网络
$ sudo ot-ctl dataset init new
$ sudo ot-ctl dataset networkkey 00112233445566778899aabbccddeeff
$ sudo ot-ctl dataset extpanid 1111111122222222
$ sudo ot-ctl dataset panid 0x1234
$ sudo ot-ctl dataset channel 15
# 将以上配置提交为活动配置
$ sudo ot-ctl dataset commit active
# 打开 ipv6 接口
$ sudo ot-ctl ifconfig up
# 启动 Thread 协议
$ sudo ot-ctl thread start
# 查看 Thread 网络配置
$ sudo ot-ctl dataset active -x

# 为设备分配节点 ID 为 1001
$ sudo ./chip-tool pairing ble-thread 1001
hex:0e080000000000010000000300000f35060004001fffe00208111111112222222220708fdb0ab694c9b3a17051000112233445
566778899aabbccddeeff030f4f70656e5468726561642d663663610102123404 10d237761823728dd2cbfe64f477b38b4c0c0402
a0f7f8 20202021 3840
```

3. 配置 OTA-Provider

开启一个新的终端窗口：

```
$ cd matter
# 在一个新的终端窗口启动 OTA 服务。指向 ota 文件
$ sudo ./out/debug/ota-provider/chip-ota-provider-app -f ../firmware/MatterLightOver-
Thread_Breakout_v3_lzma.ota
```

开启一个新的终端窗口：

```
$ cd matter/out/debug/standalone/
# 为 OTA-Provider 分配一个节点 ID：5678
$ sudo ./chip-tool pairing onnetwork 5678 20202021

# 授予网络中所有设备操作 OTA Provider cluster (0x0029)的权限。5678 为 OTA Provider 的 node id，后面的 0 为 end-point。
$ sudo ./chip-tool accesscontrol write acl '[{"fabricIndex": 1, "privilege": 5, "authMode": 2, "subjects": [112233], "targets": null}, {"fabricIndex": 1, "privilege": 3, "authMode": 2, "subjects": null, "targets": null}]' 5678 0
```

4. 控制设备升级

```
# 通知 Matter 设备升级。5678 为 OTA Provider 的 node id，1001 为 Matter Lighting 的 node id。
$ sudo ./chip-tool otasoftwareupdaterequestor announce-ota-provider 5678 0 0 0 1001 0
```

Matter 设备收到 announce-ota-provider 命令后向 OTA Provider 请求并更新固件。

固件升级过程大约要几分钟时间……

升级完成后，可以看到 LED 变为蓝色了。

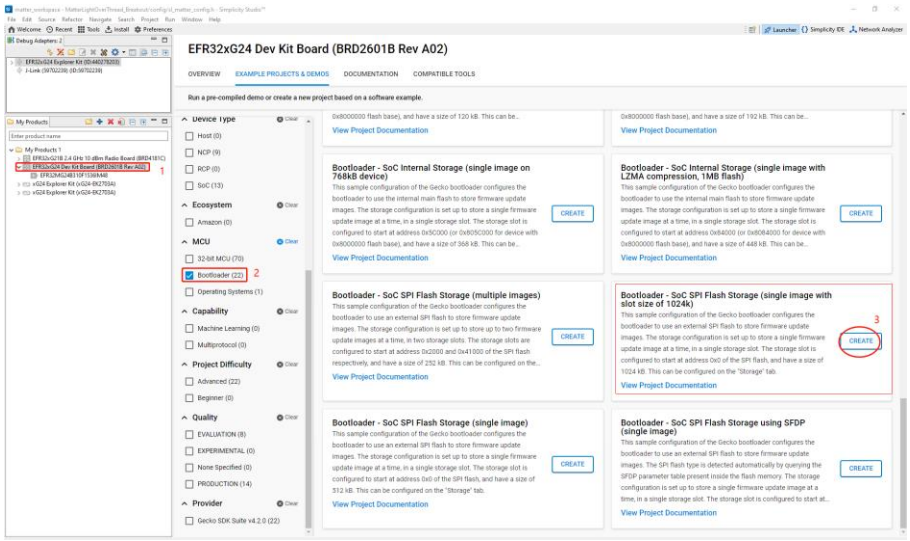
5.2 使用芯片外部 Flash 实现 OTA

5.2.1 创建 Bootloader 工程：使用外部 Flash 实现 OTA

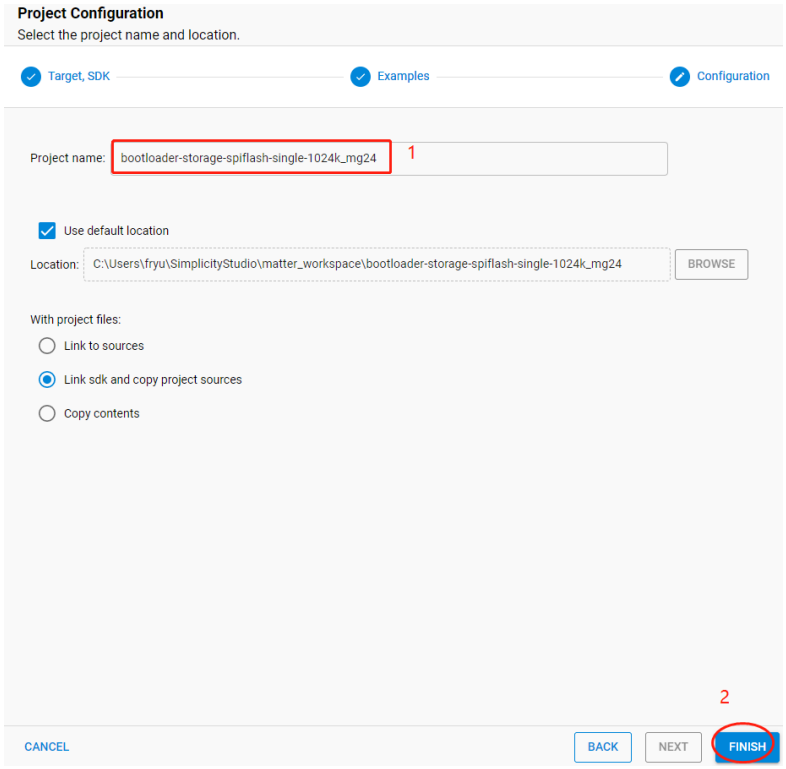
当使用外部 Flash 时注意事项：

- 使用的 flash 型号是否支持？
- 升级文件是否使用了 LZMA 压缩技术

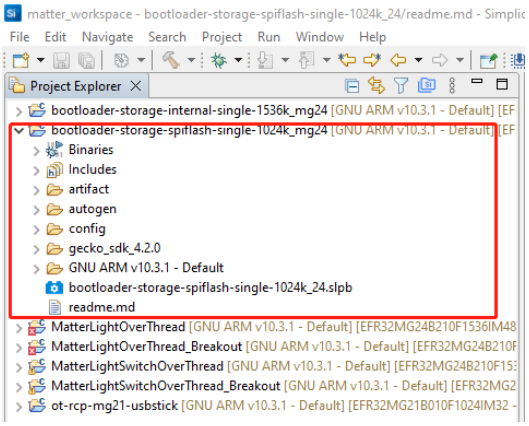
1. 添加设备选择模版 Bootloader-SoC SPI Flash Storage (single Image with slot size of 1024k)创建工程



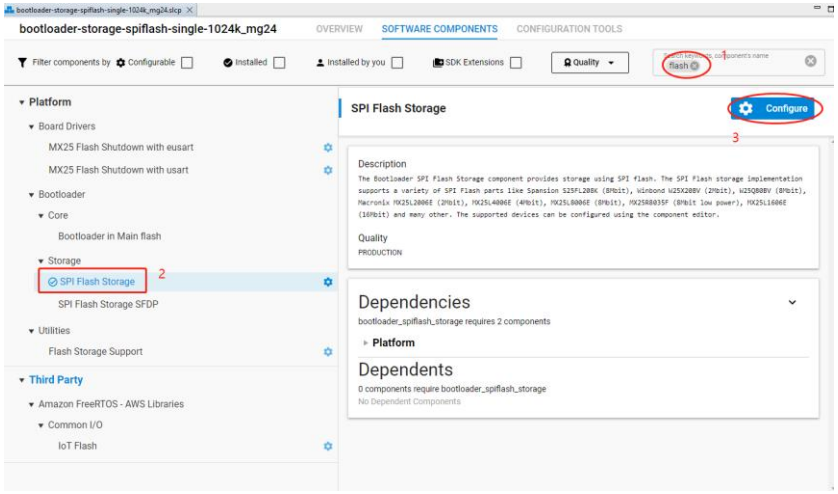
2. 修改工程名为`bootloader-storage-spiflash-single-1024k_mg24`并创建。（修改工程名不是必须的）

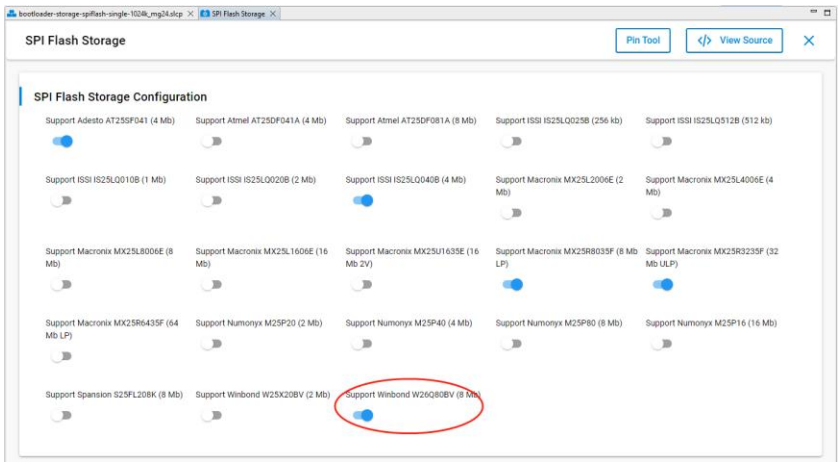


3. 创建后工程如下



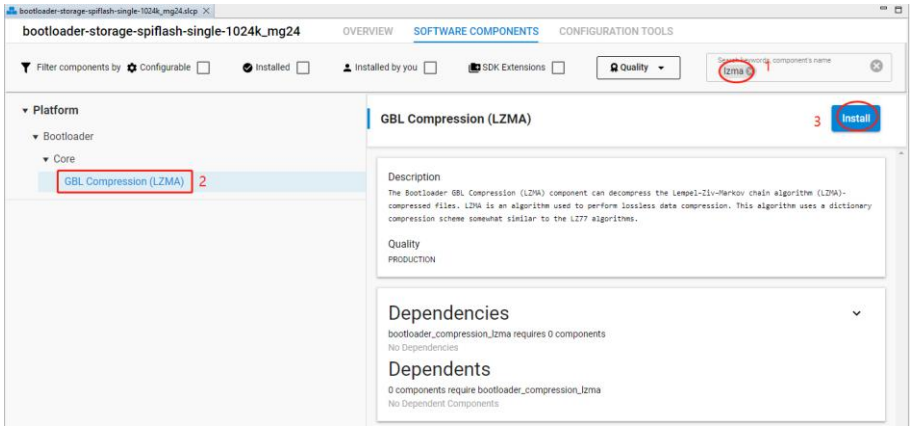
4. 打开 slpb 文件，修改 flash 配置





查看板上 flash 型号为 Winbond 25Q80，所以按上图选择对应型号即可。选择后工程会自动保存。

5. 如果升级包使用了 LZMA 压缩技术，还需要安装 LZMA 组件。



6. 编译工程可得到 bootloader 固件：`bootloader-storage-spiflash-single-1024k_mg24.s37`。
7. 升级操作参考 OTA 升级操作。

6 常见问题

- 使用 Flash 型号是否已勾选支持？
- ota 升级包是否使用了 LZMA 压缩技术
  若使用了 LZMA 技术，bootloader 中需要安装对应组件。
- OTA Provider 和 Matter 设备多次入网时，需要分配不同的 node id， 否则会出现入网失败的问题。
- 可以通过以下命令清除 chip tool 的缓存。

```
$ rm -r /tmp/chip_*
```

  清除缓存后，OTA Provider 和 Matter 设备需要重新入网。

7 参考资料

- 芯科科技 Simplicity-studio 集成开发环境：https://www.silabs.com/developers/simplicity-studio
- 芯科科技开发者文档：https://docs.silabs.com/
- 芯科科技 Matter 方案介绍：https://www.silabs.com/wireless/matter
- 芯科科技 Matter 开发文档：https://docs.silabs.com/matter/1.0.1/matter-start/
- Matter 协议规格书：https://csa-iot.org/developer-resource/specifications-download-request/
- OpenThread 参考资料：https://openthread.google.cn/

8 文档修订历史

Revision 1.0.0

Jan 31, 2023

- 初始版本

- 文档结束 -