AN13469

在 i.MX RT 上进行 AWS OTA 更新

版本 0 — 2021 年 12 月 6 日

应用笔记

目录

1 引言

本应用笔记将介绍在 i.MX RT1xxx 系列平台上利用 AWS 进行远程 OTA 更新的过程。

AWS OTA 更新基于 Secure Bootloader（SBL）和 Secure Firmware（SFW）两个工程实现。SBL 是 OTA 的安全引导加载程序，它可以通过恩智浦 Soc 安全引擎确保设备进行安全的 OTA。SFW 基于 FreeRTOS、恩智浦 SDK 等功能模块设计，与 SBL 协同工作以提供一套完整且安全的 OTA 解决方案。SFW 支持基于 SD 卡或 U 盘的本地 OTA 以及基于 AWS 或阿里云的远程 OTA。

本应用笔记将介绍基于 AWS 的 OTA 更新过程。

2 AWS OTA 概述

本章主要介绍 AWS OTA 更新的整体流程，如 图 1 所示。当需要更新固件时，首先将新版本的固件上传到 AWS 的 S3 bucket。然后创建 OTA 更新任务。 此任务将通知 RT1170 设备有可用的固件更新。最后，设备下载新的固件，验证固件后更新自身的应用程序代码。届时，设备更新完毕并运行新的应用程序代码，直到有新的更新为止。同时，设备也会向 AWS 上报任务的状态。

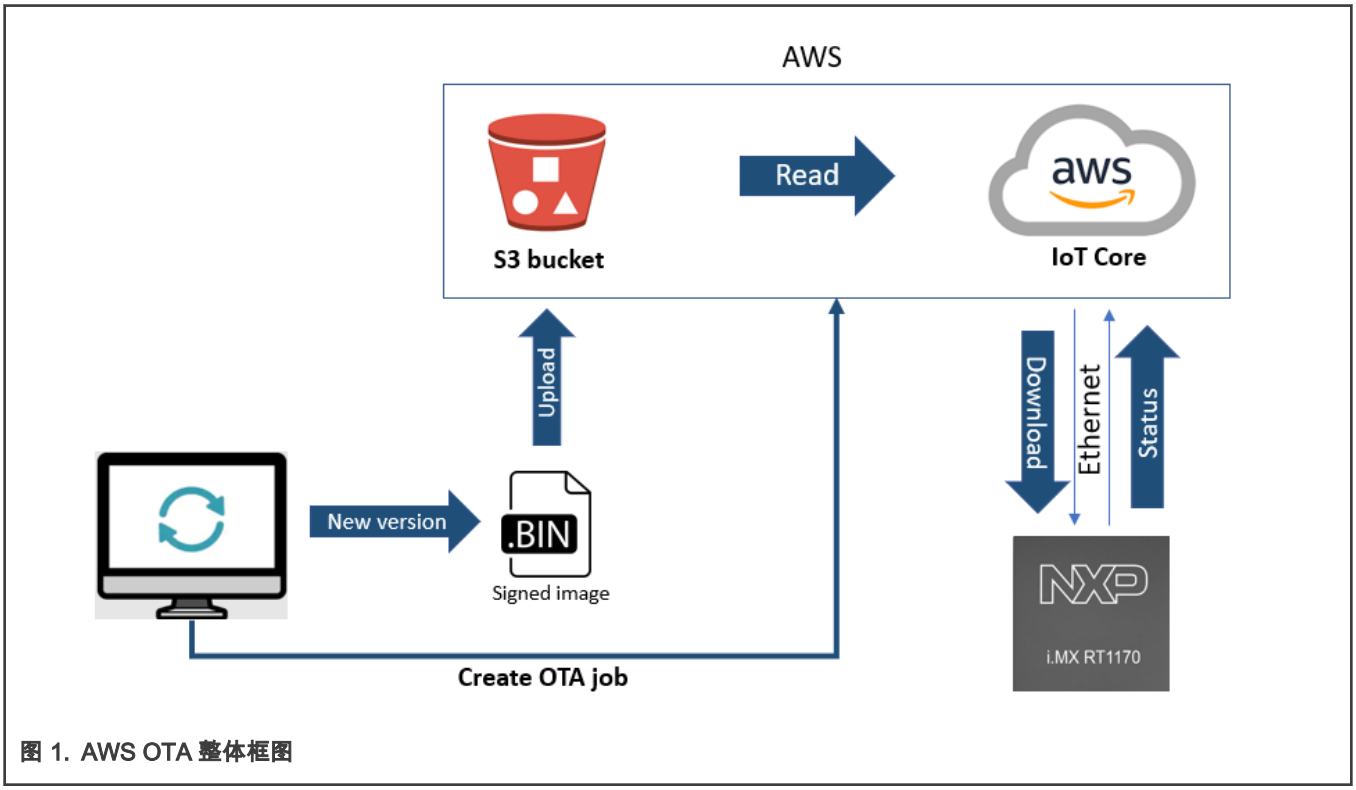

图 1. AWS OTA 整体框图

3 AWS OTA 的基础配置

本章将简单介绍一下进行 AWS OTA 更新所必须的一些基础配置，主要有如下配置流程：

1. 创建 OTA 更新服务角色，S3 bucket，OTA 用户策略等。
2. 利用 OpenSSL 和 AWS CLI 创建代码签名证书。
3. 创建 AWS IoT thing。

由于具体配置过程篇幅较长，在这里不做赘述。详细配置过程，请参考文档 *MCU-OTA SBL and SFW User Guide* (document MCUOTASBLSFWUG)的 **7.3.1.1 AWS OTA Prerequisites**。

4 AWS OTA 更新

SFW 工程支持 i.MX RT1020, RT1050, RT1060, RT1064 和 RT1170 平台上的 AWS OTA 更新。SFW 与 SBL 协同工作，提供一套完整安全的 OTA 解决方案。SFW 通过以太网连接到 AWS。本章将演示一下如何使用 SBL 和 SFW 在 RT1170-EVK 板上实现 AWS OTA 更新。

SBL 和 SFW 工程可以从以下链接下载：

- SBL: https://github.com/NXPmicro/sbl
- SFW: https://github.com/NXPmicro/sfw

4.1 SBL 准备

要实现 AWS OTA，首先要将 SBL 工程下载到目标板上，步骤如下：

1. 进入 `sbl/target/evkmimxrt1170` 目录，双击 `env.bat`。
2. 在 `env.bat` 中，运行 `scons --menuconfig` 命令，然后会出现 图 2 显示的 SBL 配置界面。

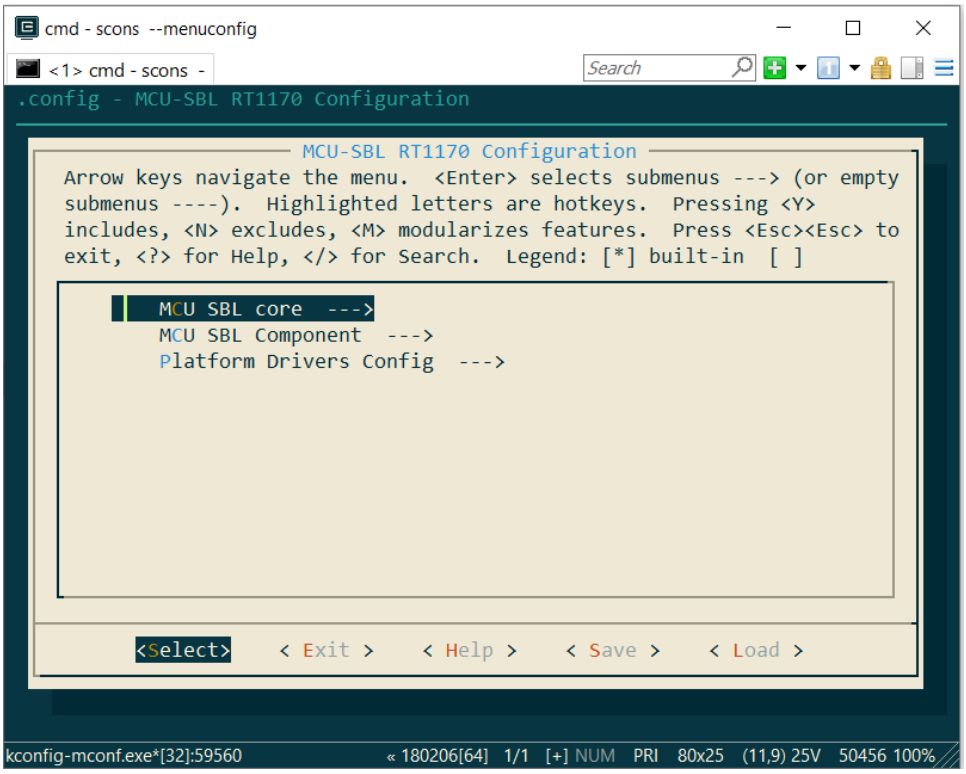

图 2. SBL configuration menu

3. 在配置界面中，取消勾选 **Enable single image function** 和 **Enable mcu isp support** 两个选项。保存并退出。
4. 然后运行 `scons --ide=iar` 命令生成 IAR 工程 `sbl.eww`，该工程位于 `sbl/target/evkmimxrt1170/iar` 目录下。

> **注**
>
> 生成 `Keil` 或者 `gcc` 工程，请参照 *MCU-OTA SBL and SFW User Guide*（文档 MCUOTASBLSFWUG）中的 **Chapter 2 Quick start**。

5. 编译 `sbl.eww` 工程并下载到 RT1170-EVK 板卡中。

4.2 配置 SFW

下载完 SBL 工程后，第二步要将 SFW 工程下载到目标板卡。在此之前，首先需要对 SFW 工程进行相关配置以支持 AWS OTA，步骤如下：

1. 生成 `aws_clientcredential_keys.h` 文件。
   a. 进入 `sfw/firmware/aws_ota/tool` 目录。
   b. 使用浏览器打开 `CertificateConfigurator.html`。
   c. 在页面中导入证书和密钥文件（该文件在 *MCU-OTA SBL and SFW User Guide*（文档 MCUOTASBLSFWUG 中的 **Chapter 7.3.1.1 AWS OTA Prerequisites** 部分里生成）。然后点击 **Generate and save aws_clientcredential_keys.h** 按钮生成并下载 `aws_clientcredential_keys.h`。

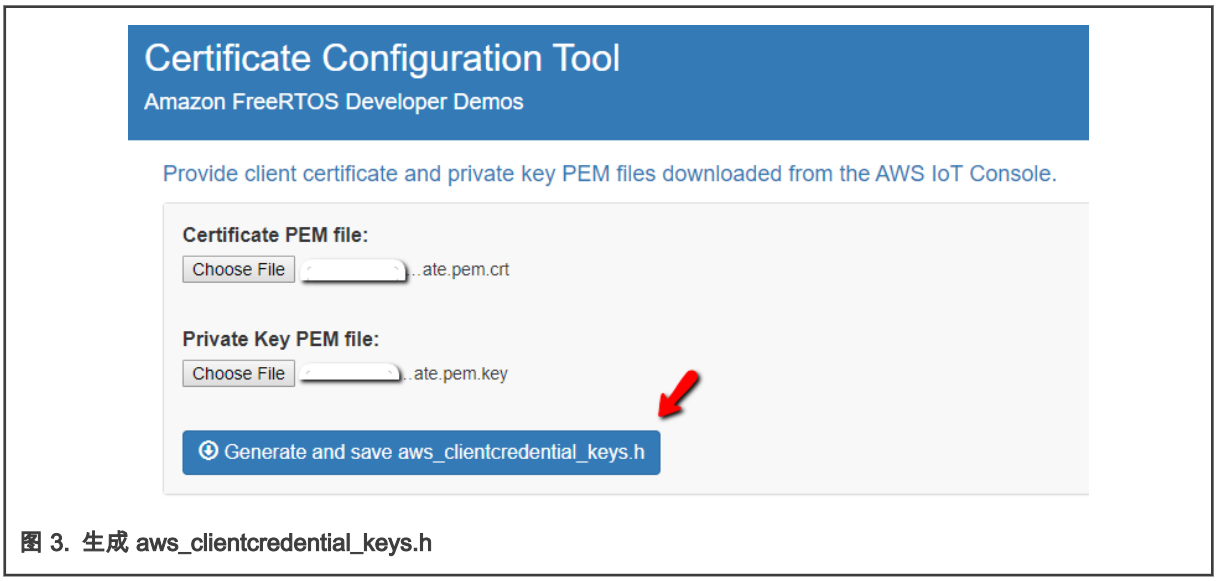

图 3. 生成 aws_clientcredential_keys.h

   d. 用 Step c 生成的文件替换掉 `sfw/firmware/aws_ota/demos/include` 目录下的 `aws_clientcredential_keys.h` 文件。

2. 修改 `aws_ota_codesigner_certificate.h` 文件。
   a. 利用文本编辑器打开 `ecdsaigner.crt` 文件（该文件在 *MCU-OTA SBL and SFW User Guide*（文档 MCUOTASBLSFWUG 中的 **Chapter 7.3.1.1 AWS OTA Prerequisites** 部分里生成）。
   b. 打开 `sfw/firmware/aws_ota/demos/include/aws_ota_codesigner_certificate.h` 文件。
   c. 复制 `ecdsaigner.crt` 文件中的所有内容并粘贴到 `signingcredentialSIGNING_CERTIFICATE_PEM` 文件的 `aws_ota_codesigner_certificate.h`。

> **注**
>
> 确保在每行开头添加 **"**并在每行末尾添加 **\n"**，如 图 4 所示。

```
/*
 * PEM-encoded code signer certificate
 *
 * Must include the PEM header and footer:
 * "-----BEGIN CERTIFICATE-----\n"
 * "...base64 data...\n"
 * "-----END CERTIFICATE-----\n";
 */
static const char signingcredentialSIGNING_CERTIFICATE_PEM[] =
"-----BEGIN CERTIFICATE-----\n"
"MIIBYTCCAQegAwIBAgIJAKCX9bIhkilFMAoGCCqGSM49BAMCMCMxITAfBgNVBAMM\n"
"GEFsZWphbmRyYS5HdXptYW5AbnhwLmNvbTAeFw0xOTEwMjMxNjMzNDJaFw0yMDEw\n"
"MjIxNjMzNDJaMCMxITAfBgNVBAMMGEFsZWphbmRyYS5HdXptYW5AbnhwLmNvbTBZ\n"
"MBMGByqGSM49AgEGCCqGSM49AwEHA0IABGUghBD51mF1J3wf4LYsQ2VgOaDpg98G\n"
"dNC38FWGS7owT4NC5848JumrD8SonnnXpu77Pt7ShuW39hC3Vdi7zlGjJDAiMAsG\n"
"A1UdDwQEAwIHgDATBgNVHSUEDDAKBggrBgEFBQcDAzAKBggqhkjOPQQDAgNIADBF\n"
"AiEAr0pNzlaMax4arCPNiW9HYFdQTvUGyZdRLcDrUol/LQoCIH2U2REoZ59V7r6z\n"
"CMLfHA+kWq84IjxDUE2OgV60RVvC\n"
"-----END CERTIFICATE-----\n";
```

图 4. 证书格式

3. 进入 `sfw/target/evkmimxrt1170` 目录。双击 `env.bat` 文件。
4. 在 `env.bat` 中，运行 `scons --menuconfig` 命令来配置 `evkmimxrt1170` 工程。
5. 在配置界面中，选择 **MCU SFW core**，然后禁用 `Enable sfw standalone xip` 选项，勾选 `enable OTA` 选项并选择 **AWS OTA**。

```
[ ] Enable sfw standalone xip
[*] Enable OTA
[*]   OTA from cloud
        OTA Cloud Select (AWS)  --->
        AWS Config  --->
[*]   OTA from sdcard
[*]   OTA from u-disk
    MCU SFW Flash Map  --->
    MCU SFW metadata header  --->
```

图 5. SFW 配置

6. 然后选择 **AWS Config** 进入 图 6 所示界面。

```
()  Set MQTT broker DNS name (NEW)
()  Set IoT Thing name (NEW)
```

图 6. AWS 配置

7. 需要设置两个 AWS 相关的选项，一是 MQTT DNS name。
   a. 打开 AWS IoT 网站。
   b. 在左边的导航窗口，选择 **Manage** → **Things**。然后选择之前创建的 Thing（创建过程详见 *MCU-OTA SBL and SFW User Guide*（文档 MCUOTASBLSFWUG 文档 **Chapter 7.3.1.1 AWS OTA Prerequisites**）。

c. 选择 **Interact**。

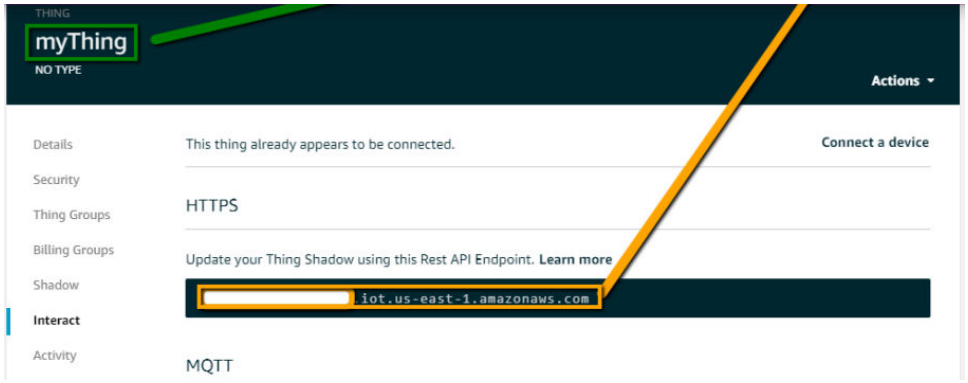

图 7. Interact of Thing

d. 在 **AWS Configuration** 界面，选择 **Set MQTT broker DNS name** 选项，将 图 7 黄色框里的内容输入到下面的界面中，然后选择 **<ok>**。

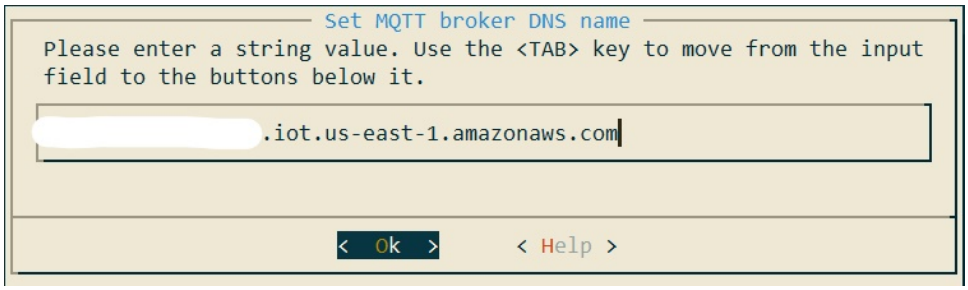

图 8. MQTT broker DNS name set

8. 二是设置 IoT Thing name。

   在 **AWS Config** 配置界面，选择 **Set IoT Thing name**，将 图 7 绿色框里的内容输入到 图 9 的界面中，然后选择**<ok>**。

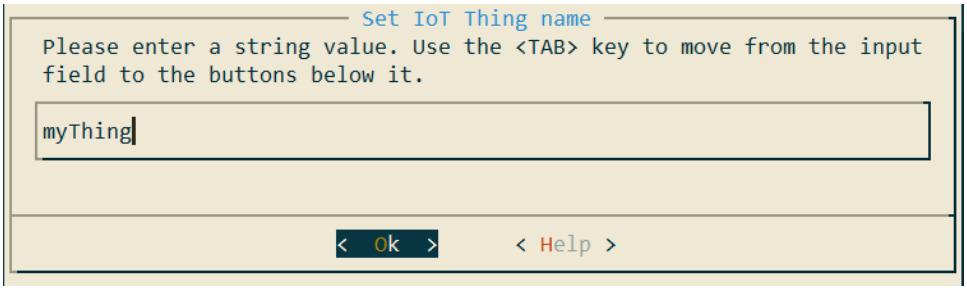

图 9. 设置 Thing 名称

9. 在配置界面中，选择 **MCU SFW Component -> secure** 选项，然后勾选 **enable mbedtls** 选项并将 `mbedtls` 配置文件设置成 `aws_mbedtls_config.h`。

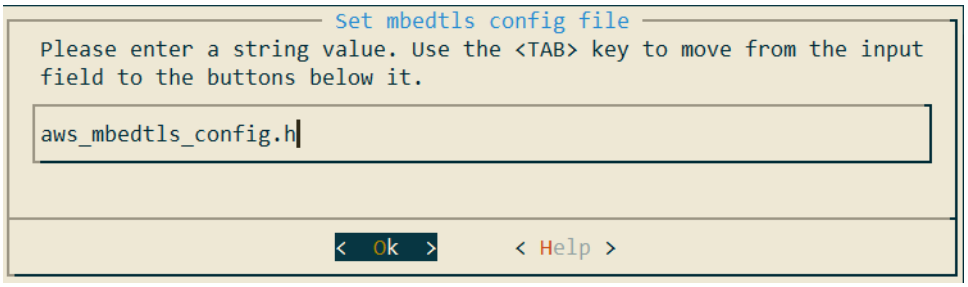

图 10. 修改 `mbedtls` 配置文件

10. 退出并保存上述的所有设置。

4.3 image 准备

为了演示 AWS OTA，需要准备两个 SFW image，一个用于运行在目标版卡上，另一个上传到 AWS 的 S3 bucket 用于后续的 OTA 更新。根据配置 SFW 配置完 SFW 工程后即可生成 SFW image，步骤如下：

1. 进入 `sfw/target/evkmimxrt1170` 目录，双击 *env.bat*。
2. 输入 `scons --ide=iar` 命令来生成 IAR 工程。

> **注**
>
> 生成 `keil` 或者 `gcc` 工程，参见 *MCU-OTA SBL and SFW User Guide*（手册 MCUOTASBLSFWUG）中 **Chapter 2 Quick start** 内容。

3. 进入 `sfw/target/evkmimxrt1170/iar` 目录，打开 `sfw.eww` 工程。
4. 进入 **Options** 界面，按 图 11 进行选择以输出 `bin` 文件。

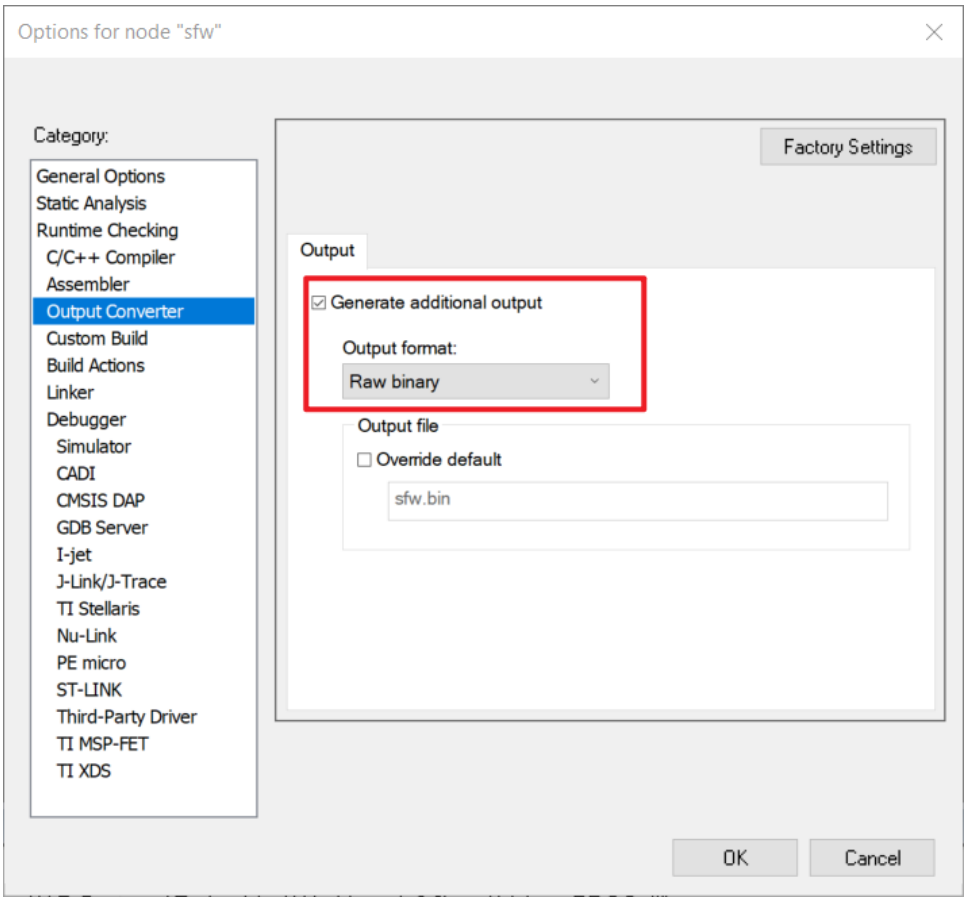

图 11. 配置生成 .bin 文件

5. 打开 `sfw/firmware/aws_ota/main_enet.c` 文件，确认当前应用程序的版本号。

```
119  #define APP_VERSION_MAJOR 0
120  #define APP_VERSION_MINOR 9
121  #define APP_VERSION_BUILD 2
```

图 12. 应用版本

6. 编译程序，编译成功后，会在 `sfw/target/evkmimxrt1170/iar/build/iar/Exe` 文件夹下生成 `sfw.bin`。为方便区别，根据上面的版本号将其重新命名为 `sfw_092.bin`，然后把 `sfw_092.bin` 拷贝到 `sbl/component/secure/mcuboot/scripts` 文件夹下。
7. 将版本号中的 `APP_VERSION_BUILD` 改为 3 以生成一个版本号较新的 image 用于后续 OTA 更新。将编译生成的新 image 新命名为 `sfw_093.bin` 并拷贝到 `sbl/component/secure/mcuboot/scripts` 文件夹下。

注

新固件的版本号要大于当前运行固件的版本号，否则会造成 OTA 失败。

```
119  #define APP_VERSION_MAJOR 0
120  #define APP_VERSION_MINOR 9
121  #define APP_VERSION_BUILD 3
```

图 13. 新的版本

8. 利用下面的两条命令给 `sfw_092.bin` 和 `sfw_093.bin` 两个 image 加上 RSA 签名，运行完毕后在相同文件下会生成 `sfw092.bin` 和 `sfw093.bin` 两个加签的 image。

```
python imgtool.py sign --key sign-rsa2048-priv.pem --align 4 --version "0.9.2" --header-size
0x400 --pad-header --slot-size 0x100000 --max-sectors 32 sfw_092.bin sfw092.bin
```

```
python imgtool.py sign --key sign-rsa2048-priv.pem --align 4 --version "0.9.3" --header-size
0x400 --pad-header --slot-size 0x100000 --max-sectors 32 sfw_093.bin sfw093.bin
```

4.4 上传新 image 到 S3 bucket

准备好 SFW image 后，需要将较新版本的一个 image 上传到 AWS 的 S3 bucket 中，步骤如下：

1. 打开 AWS S3 服务网站。
2. 选择之前创建的 S3 bucket（详情见 AWS OTA 的基础配置）。
3. 点击 **Upload** 按钮，然后将 `sfw093.bin` 文件拖拽进来，最后点击 **Upload**。

4.5 运行应用程序

在进行 OTA 更新之前，首先要在目标板卡上运行 SFW 应用程序，步骤如下：

1. 利用 MCUBootUtility 工具将 image 准备 中生成的 `sfw092.bin` 下载到目标板卡的 Slot1 中，Slot1 的默认地址是 `flash_offset+0x100000` 到 `flash_offset+0x200000`。整个 Slot1 大小为 1 MB。

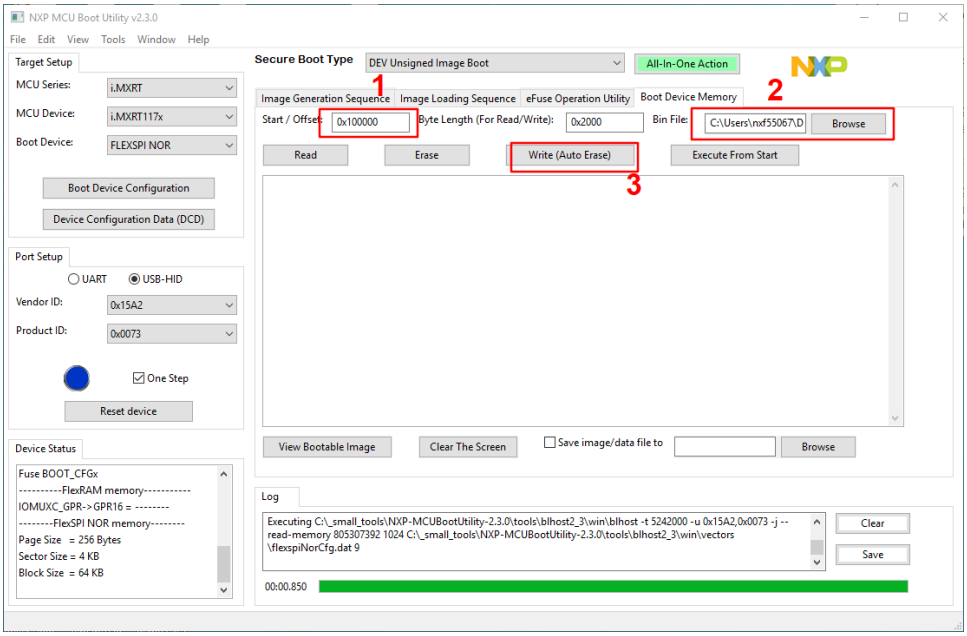

图 14. 下载 `sfw092.bin` 到 Slot1

2. 下载成功后，在 RT1170-EVK 板卡的 1 G（**J4**）口上插上以太网网线，按下 **Reset** 键开始运行程序。此时串口会打印出一些 log 信息，如 图 15 所示。

   串口打印出 **The image now in PRIMARY_SLOT slot** 和 **Getting IP address from DHCP…**，表明 Slot1 中的程序已成功运行。打印出 **IPv4 Address: 192.168.8.106** 和 **OTA demo version 0.9.2**，表明网络连接成功以及目前运行的程序的版本号。

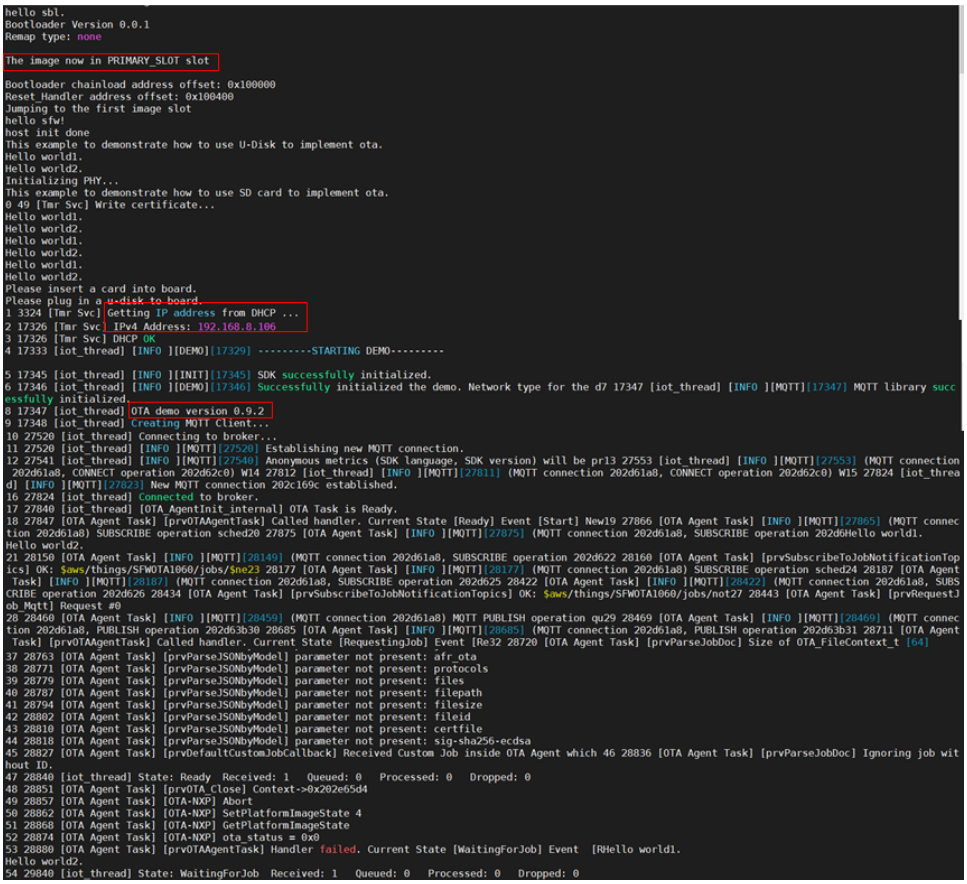

图 15. 程序 log

3. 当程序成功运行且成功连接到 AWS 后，等待串口打印出 图 16 的 log 信息，这表示 OTA 准备完毕并且在等待 OTA job。

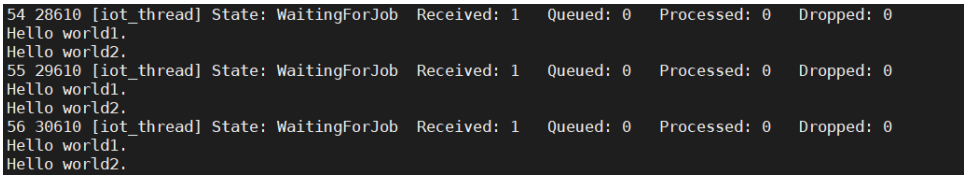

图 16. OTA 准备完毕 log

此时，一切都准备就绪，接下来就可以创建一个 OTA 更新任务来演示 OTA 更新过程了。

4.6 OTA 更新过程

最后，通过创建 OTA 更新任务来进行 OTA 更新，步骤如下：

1. 打开 AWS IoT 网站。
2. 在导航栏中选择 **Manage** -> **Jobs**。
3. 选择 **Create job**。

4. 选择 **Create FreeRTOS OTA update job**，然后点击 **Next**。
5. 输入 **Job name**，然后点击 **Next**。

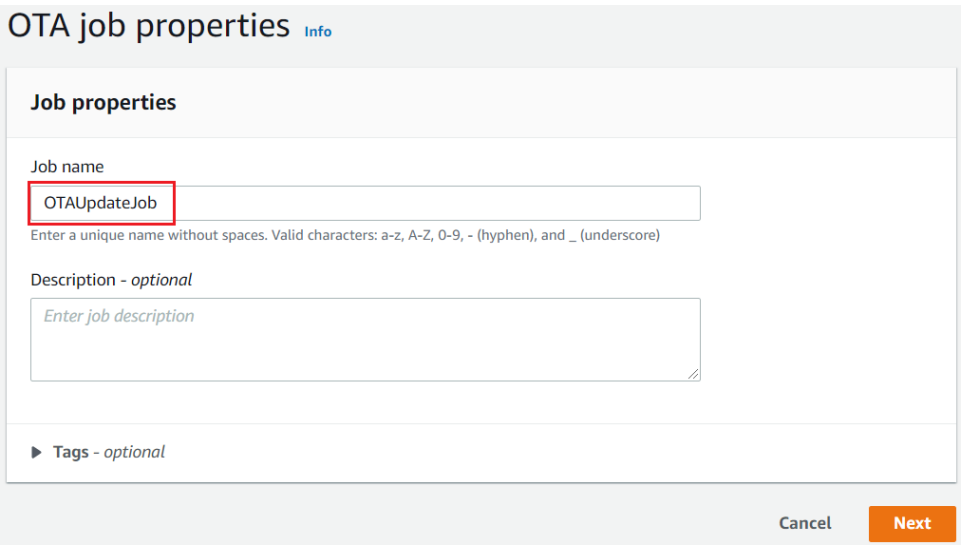

图 17. 输入 job 名称

6. 选择创建的 **Thing**（详见 AWS OTA 的基础配置），勾选 **Sign a new file for me**。

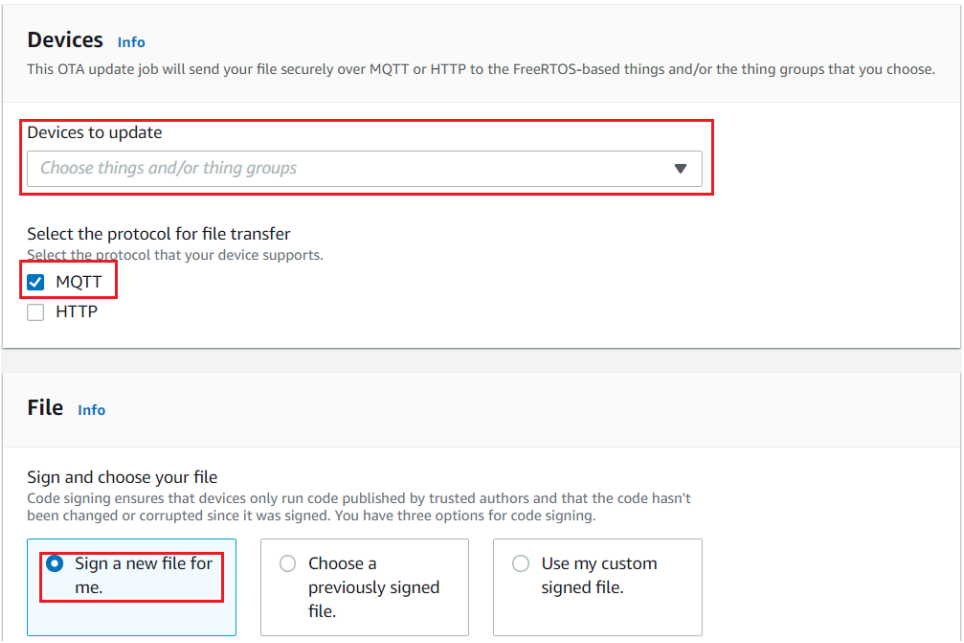

图 18. 设备信息

7. 对于 **Code signing profile**，若未曾创建过 profile，则选择 **Create new profile**，创建完毕后，后续创建其他 OTA 任务时，这里可以直接选择已创建的 profile。
8. 若选择 **Create new profile**，则按以下步骤创建一个 profile：
    a. 输入 **Profile name**。
    b. 对于 **Device hardware platform**，勾选 **Windows Simulator**。

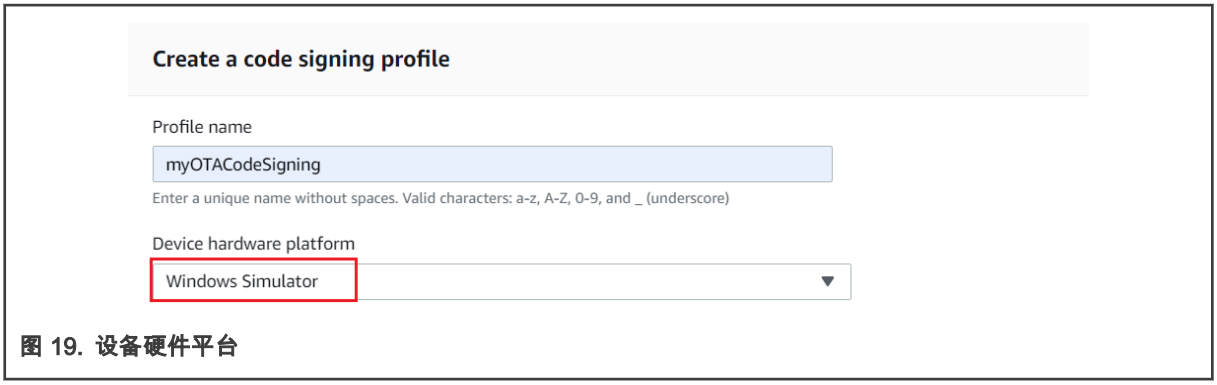

图 19. 设备硬件平台

c. 对于 **Code signing certificate**，勾选 **Import new code signing certificate**，并上传由 AWS CLI 生成的相关验证文件（详见 AWS OTA 的基础配置），然后点击 **Import**。

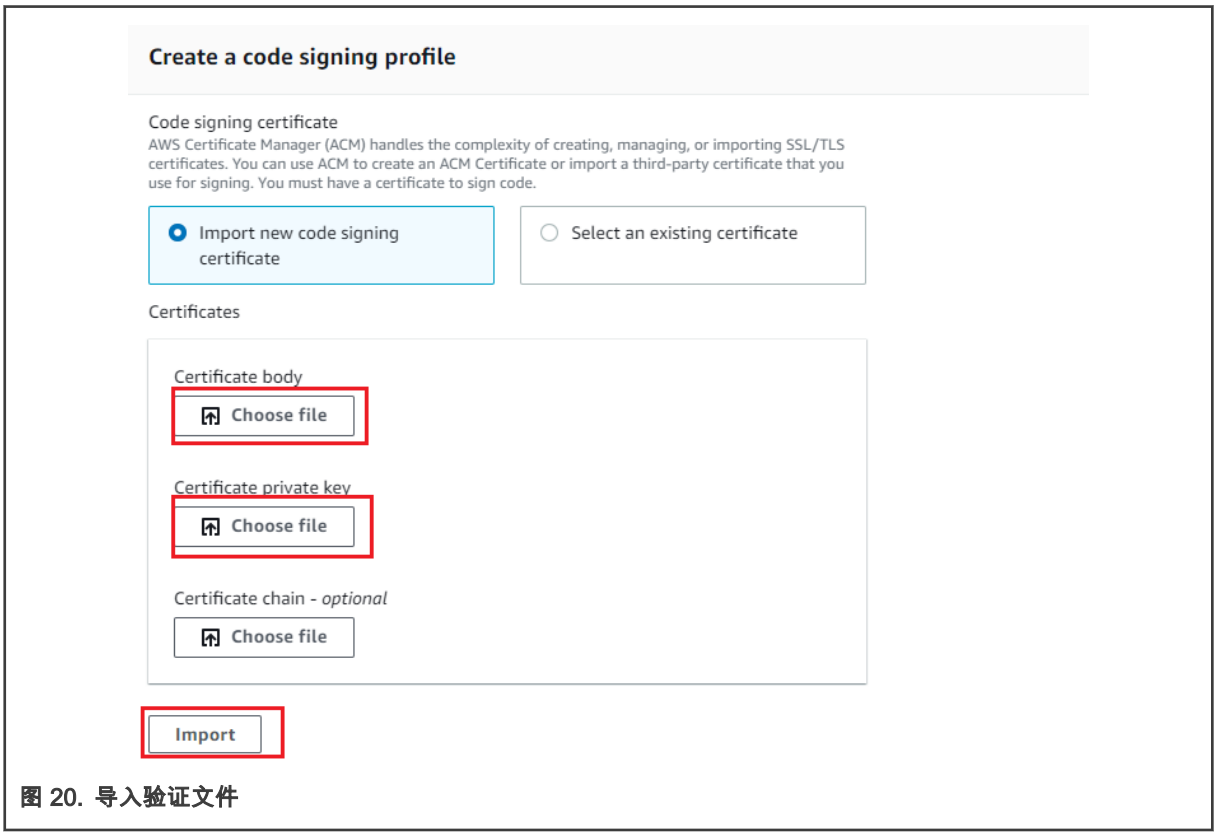

图 20. 导入验证文件

d. 对于 **Pathname of code signing certificate on device**，输入默认路径为 `path/certificates/authcert.pem`，最后点击 **Create**。

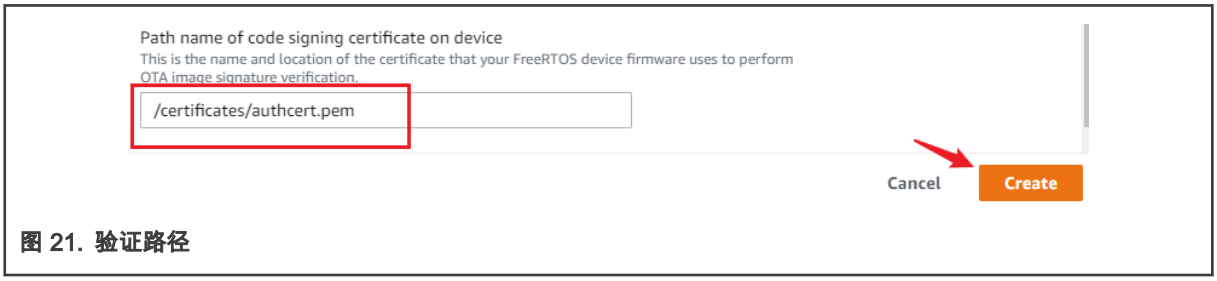

图 21. 验证路径

9. 对于 **File** 选项，勾选 **Select an existing file**。点击 **Browse S3** 从创建的 S3 bucket 中选择之前上传的 `sfw093.bin` 文件。

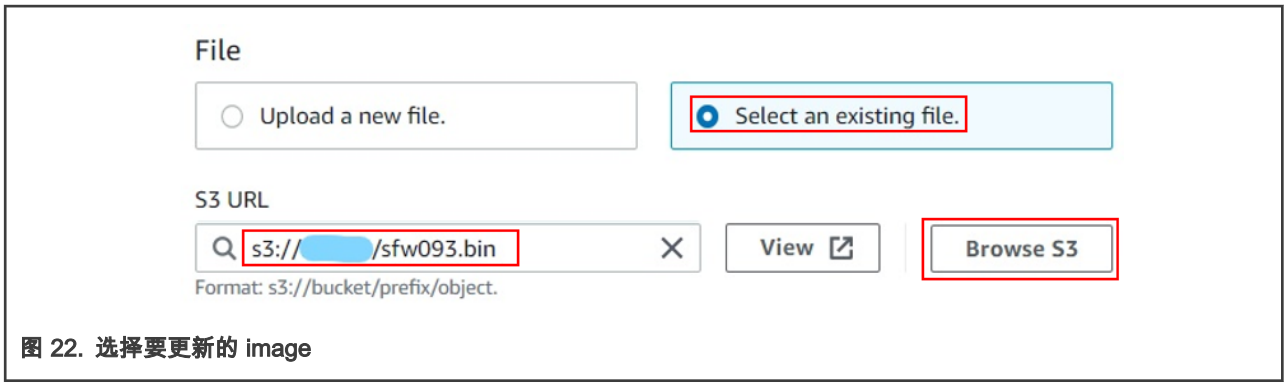

图 22. 选择要更新的 image

10. 对于 **Path name of file on device**，输入默认路径为 **path /device/updates**。

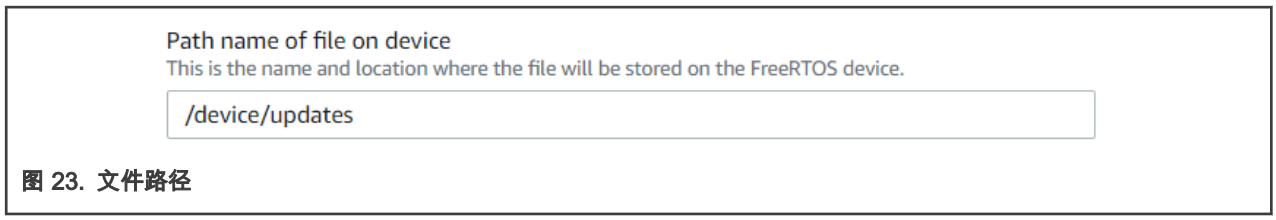

图 23. 文件路径

11. 对于 **IAM role**，选择 AWS OTA 的基础配置创建的 role。

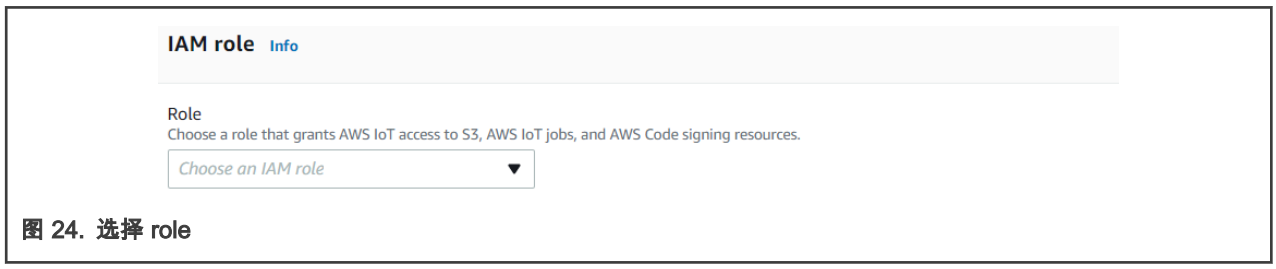

图 24. 选择 role

12. 点击 **Next**。
13. 对于 **Job run type**，勾选第一个选项，最后点击 **Create job**。

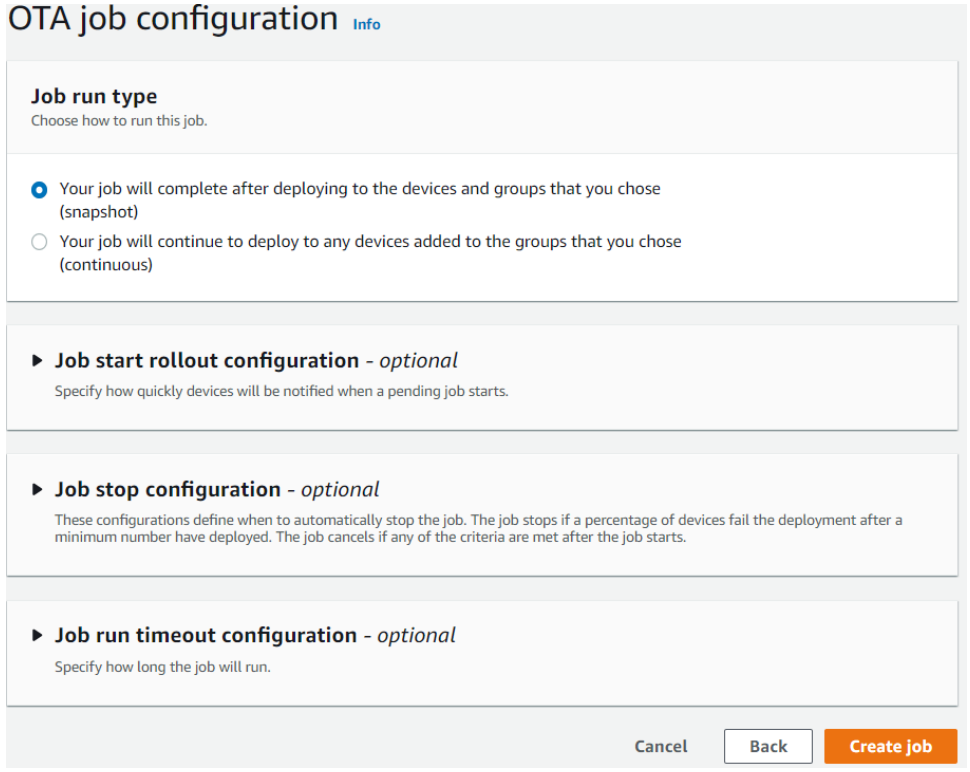

图 25. 创建 Job

14. 创建 OTA 任务后，RT1170-EVK 板卡就会收到更新任务，然后开始进行 OTA，整个 OTA 更新过程中，串口会打印出如下的 log 信息：

    a. 开始从 AWS 接收文件。

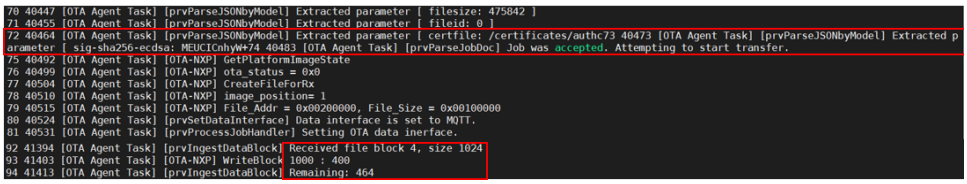

图 26. 文件传输开始 log

    b. 接收完整的文件。

图 27. 接收到完整文件 log

    c. 检查文件签名。

图 28. 签名检查 log

    d. 检查文件版本。

```
2266 156992 [OTA Agent Task] [OTA-NXP] cmp result=1
2267 156997 [OTA Agent Task] [OTA-NXP] new image version: 0.9.3
2268 157004 [OTA Agent Task] [prvIngestDataBlock] File receive complete and signature is valid.
2269 157013 [OTA Agent Task] [prvStopRequestTimer] Stopping request timer.
2270 157021 [OTA Agent Task] [prvUpdateJobStatus_Mqtt] Msg: {"status":"IN_PROGRESS","statusDetails"2271 157045 [OTA Agent Task] [INFO ][MQTT][157045] (MQTT co
nnection 202d61a8) MQTT PUBLISH operatio2272 157055 [OTA Agent Task] [INFO ][MQTT][157055] (MQTT connection 202d61a8, PUBLISH operation 202Hello world1.
```

图 29. 文件版本检查 log

e. 写更新类型。

```
2282 158200 [OTA Agent Task] [OTA-NXP] Write update type
write update type = 0x3
```

图 30. 写更新类型 log

f. 写 image trailer.

```
2283 158208 [OTA Agent Task] [OTA-NXP] Write image trailer
write magic number offset = 0xffff0
```

图 31. 写 image 尾部 log

g. 激活新的 image，设备重启。

```
2284 158218 [OTA Agent Task] [OTA-NXP] ActivateNewImage
2285 158224 [OTA Agent Task] [OTA-NXP] ResetDevice
```

图 32. 激活新文件 log

h. 运行新的 image。

```
hello sbl.
Bootloader Version 0.0.1
Remap type: test

The image now in SECONDARY_SLOT slot

Bootloader chainload address offset: 0x100000
Reset_Handler address offset: 0x100400
Jumping to the first image slot
hello sfw!
host init done
This example to demonstrate how to use U-Disk to implement ota.
Hello world1.
Hello world2.
Initializing PHY...
This example to demonstrate how to use SD card to implement ota.
0 49 [Tmr Svc] Write certificate...
2 17172 [Tmr Svc] IPv4 Address: 192.168.8.106
3 17172 [Tmr Svc] DHCP OK
4 17175 [iot_thread] [INFO ][DEMO][17175] ---------STARTING DEMO---------

5 17191 [iot_thread] [INFO ][INIT][17191] SDK successfully initialized.
6 17192 [iot_thread] [INFO ][DEMO][17192] Successfully initialized the demo. Network type for the d7 17193 [iot_thread] [INFO ][MQTT][17193] MQTT library succ
essfully initialized.
8 17193 [iot_thread] OTA demo version 0.9.3
9 17194 [iot_thread] Creating MQTT Client...
```

图 33. 新 image 运行 log

i. 实施自检测。

```
56 27756 [OTA Agent Task] [prvOTAAgentTask] Called handler. Current State [WaitingForJob] Event [Re57 27765 [OTA Agent Task] [prvInSelfTestHandler] prvInSelfT
estHandler, platform in self-test.
58 27774 [OTA Agent Task] [OTA-NXP] GetPlatformImageState
59 27781 [OTA Agent Task] [OTA-NXP] ota_status = 0x1
```

图 34. 自检测 log

j. 写 OK 标志位。

```
64 27814 [OTA Agent Task] [OTA-NXP] ota_status = 0x1
Write OK flag: off = 0xfffe0
```

图 35. 写 OK 标志位 log

k. OTA 更新成功。

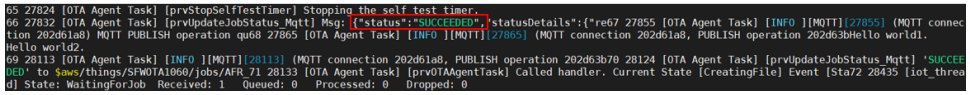

图 36. OTA 成功 log

15. 在 OTA 更新成功后，按下 Reset 键，可以看到应用程序已经成功更新，新的程序运行在 Slot2 中且版本为 0.9.3。

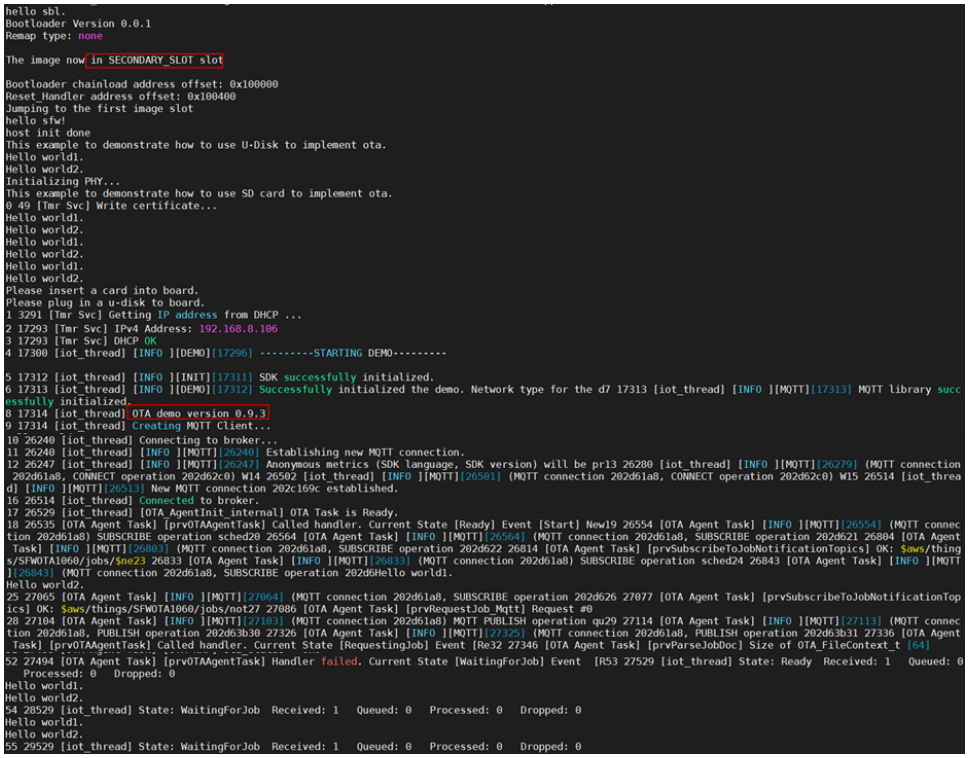

图 37. 新 image 运行 log

5 参考资料

1. SBL project
2. SFW project

6 修订记录

版本号	日期	说明
0	2021 年 12 月 6 日	初次发布

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Limited warranty and liability — Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

Right to make changes - NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Security — Customer understands that all NXP products may be subject to unidentified or documented vulnerabilities. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer’s applications and products. Customer’s responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer’s applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 2021 年 12 月 6 日

Document identifier: AN13469

arm