文档ID：MCUOTASBLSFWUG
第1.1.0版，2021年11月8日

MCU-OTA SBL和SFW用户指南

目录

第一章
简介

本文档完整地讲述了安全引导加载程序（Secure Bootloader，SBL）的功能、工程框架、快速上手和各种软件设置。它还详细讲述了FOTA，包括映像烧写、切换、恢复、签名、加密等。安全是非常重要的，文中对恩智浦MCU SoC安全引擎进行了阐述。本文也包含通过MCU ISP（UART/USB）烧写映像的详细步骤。其他必要的信息也可以在文中找到，以方便理解和开发。

安全固件（SFW）是基于 FreeRTOS 创建的，与SBL一起来实现完整的FOTA过程。SFW支持通过本地U盘、SD卡，或远程的AWS云、阿里云，来获取OTA固件映像。然后SBL检查、验证OTA固件映像，并正常启动。

1.1 缩略语和释义

下表列出了文中使用的缩略语和释义

表1. 缩略语和释义

术语	释义
AES	Advanced Encryption Standard 高级加密标准
Aliyun	Alibaba cloud 阿里云
AWS	Amazon Web Services 亚马逊网络服务
BEE	Bus Encryption Engine 总线加密引擎
CAAM	Cryptographic Accelerator and Assurance Module 加密算法加速器和可靠性模块
CRC	Cyclic Redundancy Check 循环冗余校验
DCP	Data Co-Processor 数据协处理器
FOTA	Firmware Over-The-Air 固件升级
HAB	High Assurance Boot 高可靠性启动
LWIP	Lightweight TCP/IP stack 轻量级TCP/IP协议栈
MCU ISP	MCU In-System programming MCU系统内烧录
MQTT	Message Queuing Telemetry Transport 消息队列分发传输
OCOTP	On-Chip One Time Programmable 片上一次性可编程
OTA	Over-The-Air 远程升级
OTFAD	On-The-Fly AES Decryption 即时AES解密
OTPMK	One-Time Programmable Master Key 一次性可编程主密钥

表格在下一页继续.....

表1. 缩略语和释义（续）

术语	释义
RTOS	Real-time operating system 实时操作系统
SB	NXP MCU Secure Binary 恩智浦MCU安全映像
SBL	Secure Bootloader 安全引导加载程序
SFW	Secure Firmware 安全固件
SHA	Secure Hash Algorithms 安全哈希算法
SKB	Secure Kinetis bootloader 安全Kinetis引导加载程序
SNVS	Secure Non-Volatile Storage 安全非易失性存储
TRNG	True Random Number Generator 真随机数发生器
XIP	eXecute In Place eXecute就地执行

1.2 关于MCU SBL和SFW

MCU SBL和SFW是用于安全OTA的C代码工程，它们通过UART、USB支持本地OTA，或通过以太网、WIFI等进行远程OTA，并能提供完整的安全信任链。主机工具（Host Tool）可以方便地通过UART/USB接口，对映像进行烧写、签名和加密，以及管理eFuse并创建一个SB/SB2二进制映像。

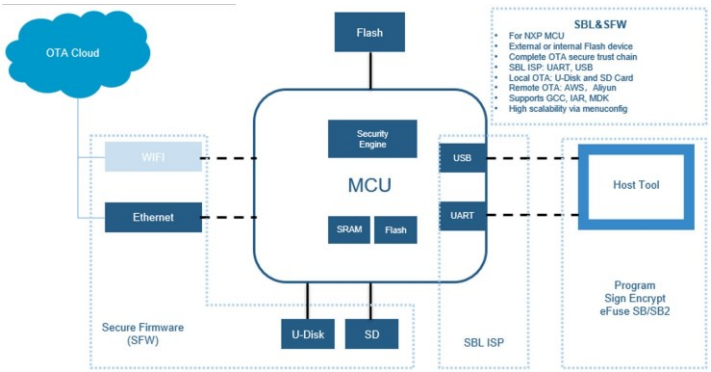


图1. SBL和SFW示意图

图2展示了SBL架构的详细信息，以及与“固件”和“主机工具”的关系，它包括项目中所有可能的模块。当为一个MCU平台构建特定的SBL映像时，该工程能够（应该）根据SoC的特点轻松配置。

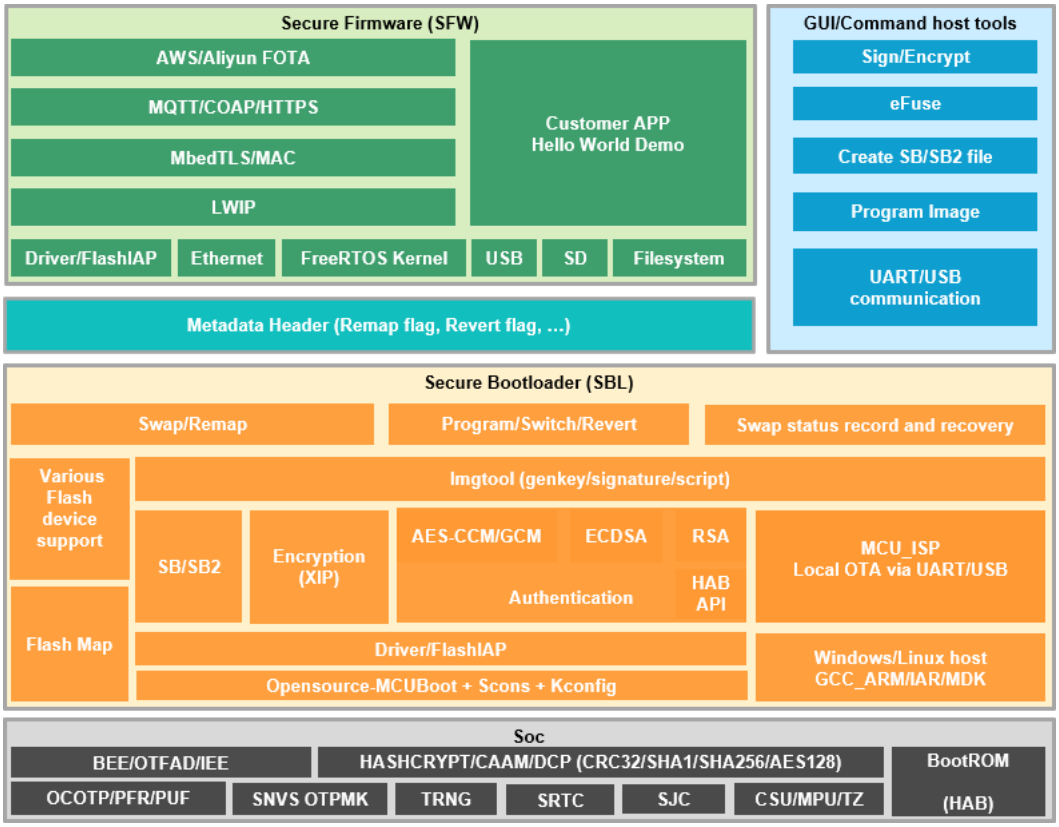


图2. MCU-OTA 项目架构

1.3 功能

- 支持恩智浦MCU平台（表2列出了目前支持的平台），一致性的代码和架构。
- 完整的OTA安全信任链，支持SoC安全引擎进行签名和加密
- 单一映像或交换映像OTA功能
- 支持SoC重映射功能以减少对闪存的擦除/烧写
- 本地OTA：UART，USB通信（SBL）
- 本地OTA：SD卡，U-盘（SFW）
- 远程OTA：AWS，阿里云（SFW）
- 最小的MCU系统资源需求
- 支持外部或内部闪存设备
- 支持多种工具链和开发环境：
  - Linux主机：GCC_ARM
  - Windows主机：GCC_ARM、IAR、MDK
  - 通过SCons扩展命令方便地创建IAR、MDK项目
- 通过 Kconfig 机制实现高度且简单的可扩展性
- 遵循开源 MCUboot 项目的OTA流程

1.4 支持的MCU平台

表2. 支持的恩智浦MCU平台

平台	架构	启动设备	安全		SBL			SFW OTA			
			签名	加密	系统内烧写	交换	重映射	U盘	SD卡	AWS	阿里云
evkmimxrt1010	CM7	QSPI Flash	•	•	•		•	•			
evkmimxrt1020	CM7	QSPI Flash	•	•	•	•		•	•	•	•
evkbmimxrt1050	CM7	Hyper Flash	•	•	•	•		•	•	•	•
evkmimxrt1060	CM7	QSPI Flash	•	•	•		•	•	•	•	•
evkmimxrt1064	CM7	QSPI Flash	•	•	•		•	•	•	•	•
evkmimxrt1170	CM7+CM4	QSPI Flash	•	•	•		•	•	•	•	•
evkmimxrt500	CM33+F1	Octal Flash	•	•	•		•	•	•		
evkmimxrt600	CM33+HiFi4	Octal Flash	•	•	•		•	•	•		
lpc55s69	CM33+CM33	Internal Flash	•	•	•	•		•	•		

关于详细的平台信息，请参考以下文件：

- MIMXRT1010 EVK 板卡硬件用户指南
- MIMXRT1020 EVK 板卡硬件用户指南
- MIMXRT1050 EVK 板卡硬件用户指南
- MIMXRT1060 EVK 板卡硬件用户指南
- MIMXRT1064 EVK 板卡硬件用户指南
- MIMXRT1170 EVK 板卡硬件用户指南
- MIMXRT500 EVK 板卡硬件用户指南
- MIMXRT600 EVK 板卡硬件用户指南
- LPC55S69 EVK 板卡硬件用户指南

1.5 SBL和SFW组织架构

SBL和SFW项目是由源代码、SCons工具、Kconfig脚本、Python脚本、Windows可执行文件和文档构成。SBL的组织层图和描述见图3和表3。SFW的组织层图和描述见图4和表4。

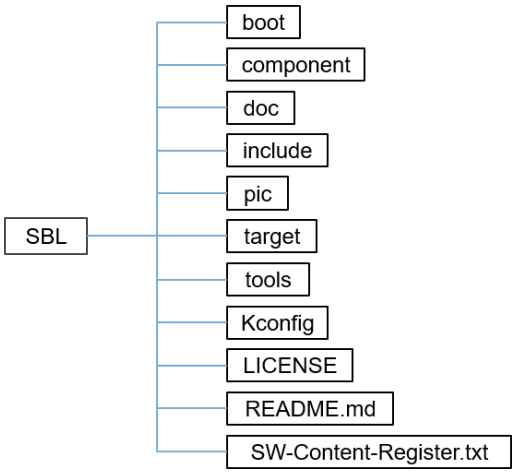


图3. SBL目录组织

表3. SBL源代码目录

目录	说明
Boot	MCUboot的源代码，部分来自开放源码
component	包括SDK组件和电路板外围驱动程序，例如，闪存IAP和UART驱动程序。
doc	SBL项目的文档
include	SBL项目的头文件
pic	README.md使用的图片
target	所有支持的平台：RT1010，RT1020，RT1050，RT1060，RT1064，RT1170，RT500，RT600，LPC55S69
tools	用于构建和配置项目的工具
Kconfig	menuconfig工具的脚本文件
LICENSE	Apache许可
README.md	SBL项目介绍
SW-Content-Register.txt	用于SBL项目的许可检查

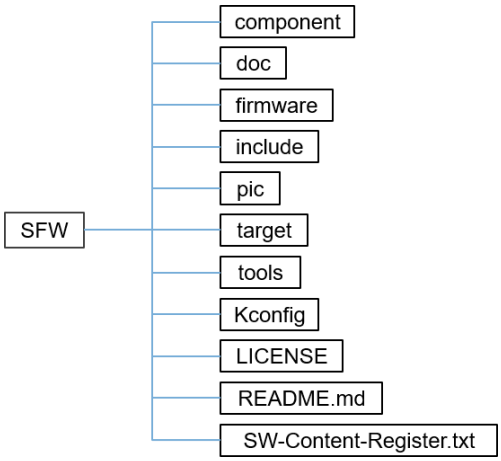


图4. SFW目录组织

表4. SFW源代码目录

目录	说明
component	包括SDK组件和电路板外围驱动程序，例如，闪存IAP和UART驱动程序。
doc	SFW项目的文档
firmware	OTA相关的源代码，例如，FreeRTOS、AWS、阿里云
include	SFW项目的头文件
pic	README.md 使用的图片
target	所有支持的平台：RT1010，RT1020，RT1050，RT1060，RT1064，RT1170，RT500，RT600，LPC55S69
tools	用于构建和配置项目的工具
Kconfig	menuconfig工具的脚本文件
LICENSE	Apache许可
README.md	SFW项目介绍
SW-Content-Register.txt	用于SFW项目的许可检查

1.6 主机系统要求

SBL和SFW项目可以在Linux主机和Windows主机上开发。系统要求如下：

- Linux主机
  - Git
  - Python 3.6
  - SCons

  - GCC_ARM toolchain
  - Library：ncurses5-dev
- Windows主机
  - Git bash
  - GCC_ARM toolchain
  - or IAR IDE v8.40
  - or MDK IDE v5.30

第二章
快速起步

本章介绍了SBL和SFW项目的快速起步。第2.1和2.2节介绍了SBL项目的快速起步，第2.3节介绍了SFW项目的快速起步。以EVKMIMXRT1170平台为例。

2.1 Windows主机

在Windows主机上，可以选择3种工具链来构建SBL：GCC_ARM、IAR和MDK。

2.1.1 GCC_ARM

首先，从Arm或MinGW网站获得GCC_ARM工具链，并将其安装到Windows主机上。

1. 克隆SBL项目并checkout到1.1.0版本，或下载发布包。

   ```
   git clone https://github.com/NXPmicro/sbl.git.
   ```

2. 进入目录 `sbl/target/evkmimxrt1170/`。
3. 双击批处理文件 `env.bat`。
4. 配置 `evkmimxrt1170` 项目。

   在 `env.bat` 中，运行 `scons --menuconfig` 命令。然后就会生成SBL配置菜单。

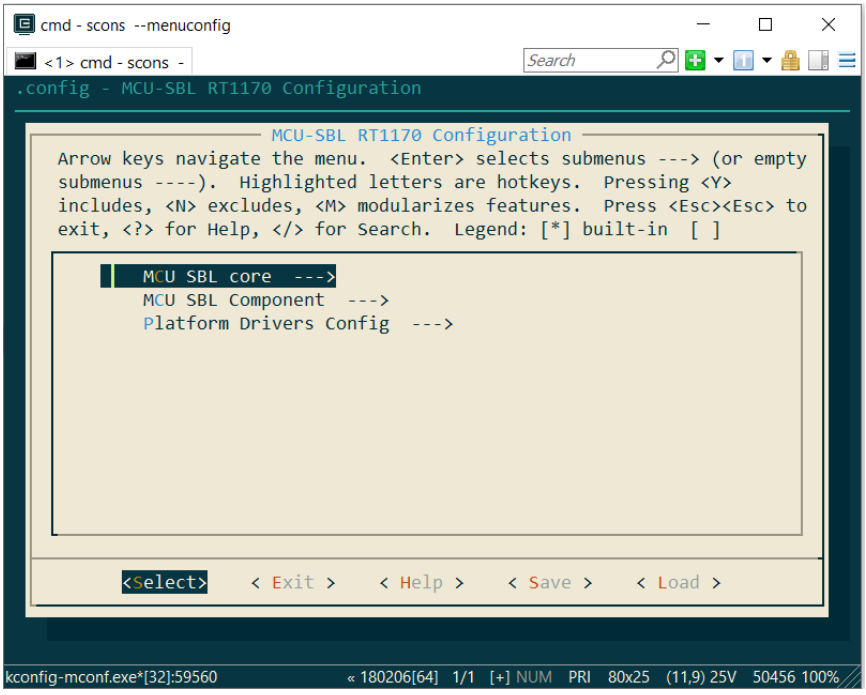


图5. SBL配置菜单

   根据具体的平台和具体的应用来配置SBL项目。配置完成后，保存配置并退出菜单。

5. 构建和下载SBL项目。

   a. 在 `sblprofile.py` 文件中为 `gcc CROSS_TOOL` 设置参数，其中 `EXEC_PATH` 设置为gcc工具链的安装路径：

```
if  CROSS_TOOL == 'gcc':
    PLATFORM    = 'gcc'
    EXEC_PATH   = r'/opt/share/toolchain/gcc-arm-none-eabi-9-2019-q4-major/bin'
```

图6. sblprofile.py中的EXEC_PATH

例如，在我的windows中，gcc工具链的安装路径是 `C:\Program Files (x86)\GNU Tools Arm Embedded\9 2019-q4-major\bin`。EXEC_PATH设置如下：

```
EXEC_PATH = r'C:\Program Files (x86)\GNU Arm Embedded Toolchain\9 2020-q2-update\bin'
```

或者，可以将 `SBL_EXEC_PATH` 添加到Windows环境变量中，以覆盖 `EXEC_PATH`。对于SFW，环境变量为 `SFW_EXEC_PATH`。

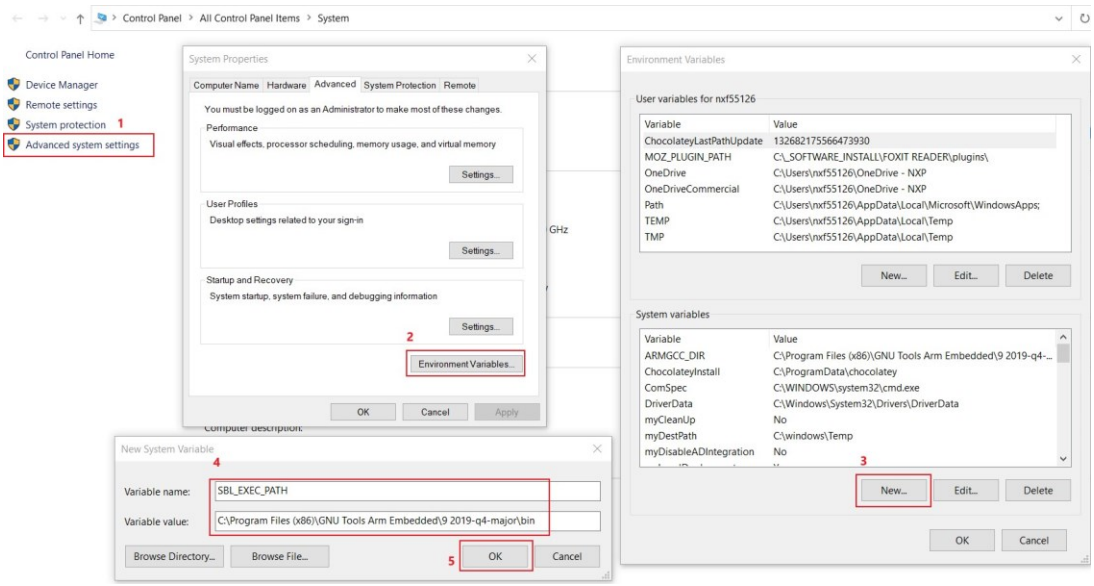


图7. Windows环境变量 SBL_EXEC_PATH

b. 构建该项目

在 `env.bat` 中，使用 `scons` 命令来构建项目。

如果成功构建，`sbl.bin` 映像将被构建在 `sbl/target/evkmimxrt1170/build` 目录中。

c. 下载该项目

i. 使用micro USB数据线将 EVKMIMXRT1170 电路板连接到计算机。

ii. 将该电路板设置为串行下载模式。

iii. 使用DapLink的拖放功能或其他工具将 `sbl.bin` 映像下载到电路板上。

iv. 将电路板设置为XIP模式。

v. 复位电路板。

2.1.2 IAR IDE

对于IAR工具链，快速开始步骤如下：

步骤1~步骤4与 2.1.1 节相同

步骤5：构建并下载SBL项目。

1. 为 EVKMIMXRT1170 平台创建IAR项目。

在 `env.bat` 中，使用 `scons --ide=iar` 命令来生成IAR项目。

2. 进入目录：`sbl/target/evkmimxrt1170/iar`
3. 双击IAR项目文件：`sbl.eww`
4. 点击“Make”（制作）按钮以构建项目。
5. 使用 micro USB 连接线将 EVKMIMXRT1170 电路板连接到电脑上，将板子设置为串行下载模式。要将项目下载到电路板上，点击“Download”（下载）按钮。映像下载成功后，将电路板设置为XIP模式，然后复位电路板。

2.1.3 MDK IDE

对于MDK工具链，快速开始的步骤如下：

步骤1至步骤4与 2.1.1 节相同。

步骤5：构建并下载SBL项目。

1. 为 EVKMIMXRT1170 平台创建MDK项目。

   在 `env.bat` 中，使用 `scons --ide=mdk5` 命令来生成MDK项目。

2. 进入目录：`sbl/target/evkmimxrt1170/mdk`。
3. 双击MDK项目文件：`sbl.uvprojx`。
4. 点击“Build”（创建）按钮以构建项目。
5. 使用micro USB数据线将 EVKMIMXRT1170 电路板连接到计算机上，将板子设置为串行下载模式。要将项目下载到电路板上，点击“Download”（下载）按钮。映像下载成功后，将电路板设置为XIP模式，然后复位电路板。

注意

在使用MDK构建SBL项目时，可能会遇到以下错误。

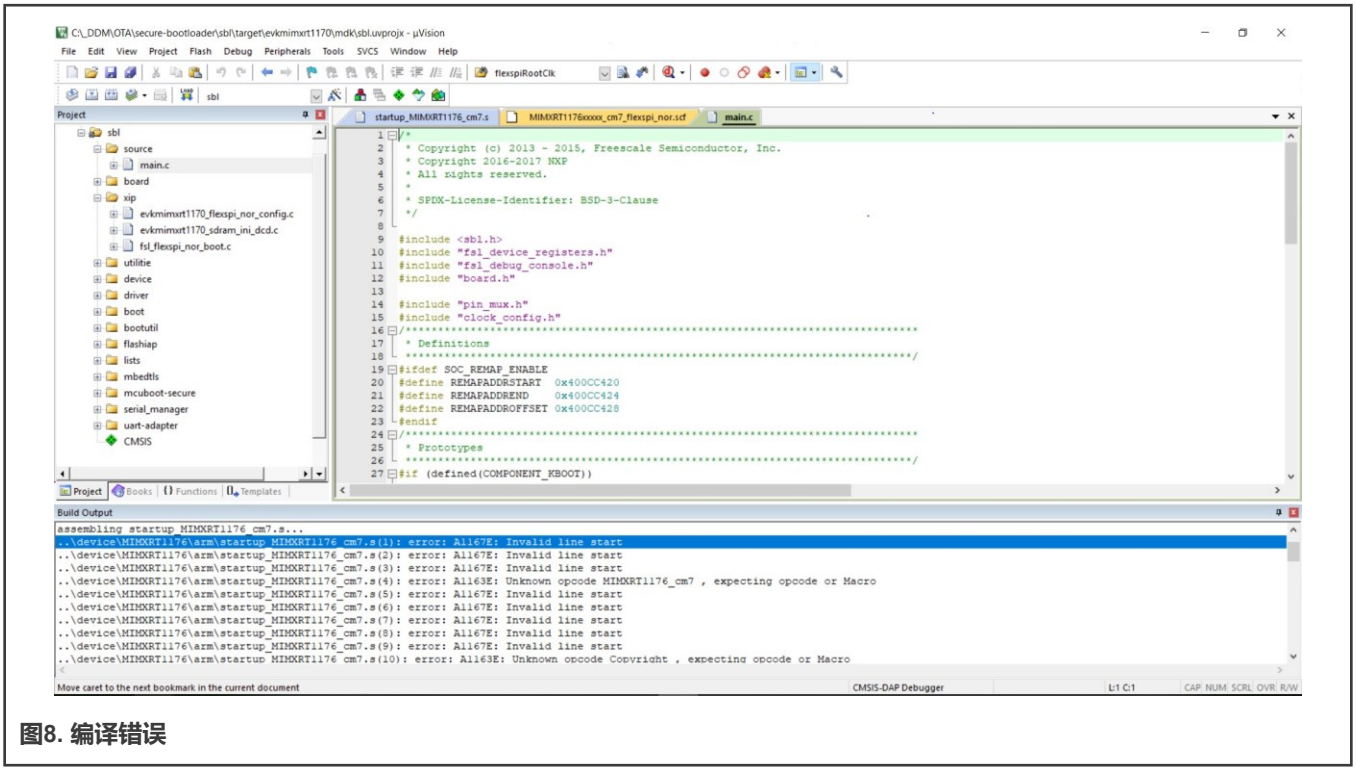


图8. 编译错误

可以通过以下方式解决它：

- 使用MDK 5.30版（或更新版本）

  为项目配置汇编器选项（Assembler Option）：armclang（GUN Syntax）

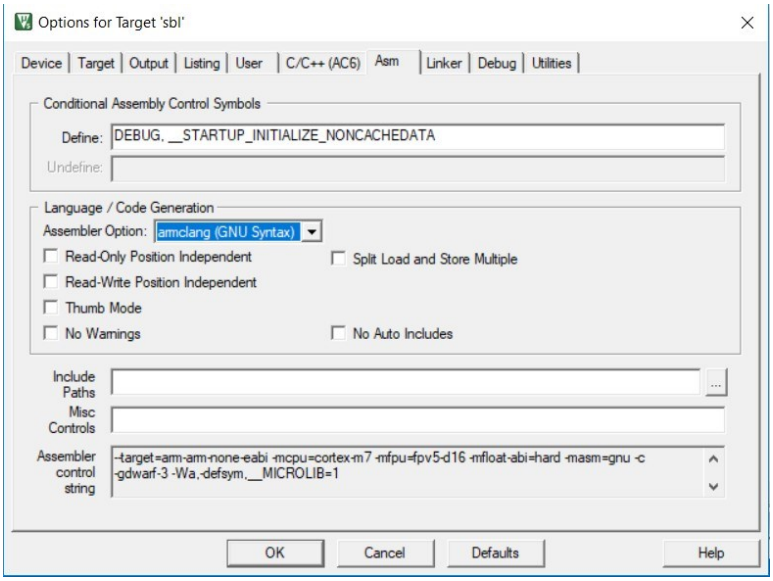


图9. 汇编器选项

- 使用比MDK 5.30更早的版本

  1. 使用ArmClang V6选择选项“Assemble”（汇编）
  2. 将“Misc Controls”配置为 `-masm=auto`

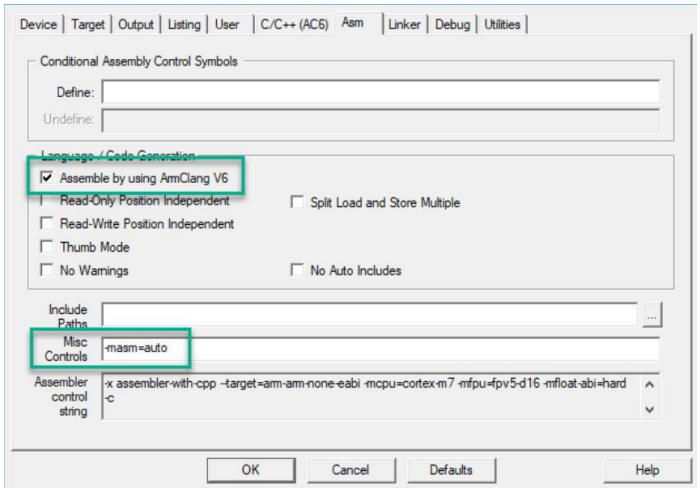


图10. Asm 选项

2.2 Linux主机

在Linux主机上，快速开始的步骤如下：

1. 安装 SCons

   对于 Ubuntu 或 Debian，使用以下命令：

   ```
   $ sudo apt-get install scons
   ```

对于基于RPM的（Red Hat, SUSE, Fedora ...），使用命令：

```
$ sudo yum install scons
```

2. 安装 GCC_ARM 工具链，如 `gcc-arm-none-eabi-9-2019-q4-major`。
3. 克隆SBL项目并checkout到1.1.0版本，或下载发布包

```
$ git clone https://github.com/NXPmicro/sbl.git
```

4. 切换到 `evkmimxrt1170` 目录

```
$ cd target/evkmimxrt1170
```

5. 在 `sblprofile.py` 文件中为 `gcc CROSS_TOOL` 设置参数，其中 EXEC_PATH 设置为gcc工具链安装路径：

```
if   CROSS_TOOL == 'gcc':
     PLATFORM    = 'gcc'
     EXEC_PATH   = r'/opt/share/toolchain/gcc-arm-none-eabi-9-2019-q4-major/bin'
```

图11. `sblprofile.py` 中的 `EXEC_PATH`

或者，可以将 `SBL_EXEC_PATH` 添加到Linux环境变量中，以覆盖 `EXEC_PATH`。对于SFW，环境变量为`SFW_EXEC_PATH`。

6. 配置 `evkmimxrt1170` 项目

```
$ scons --menuconfig
```

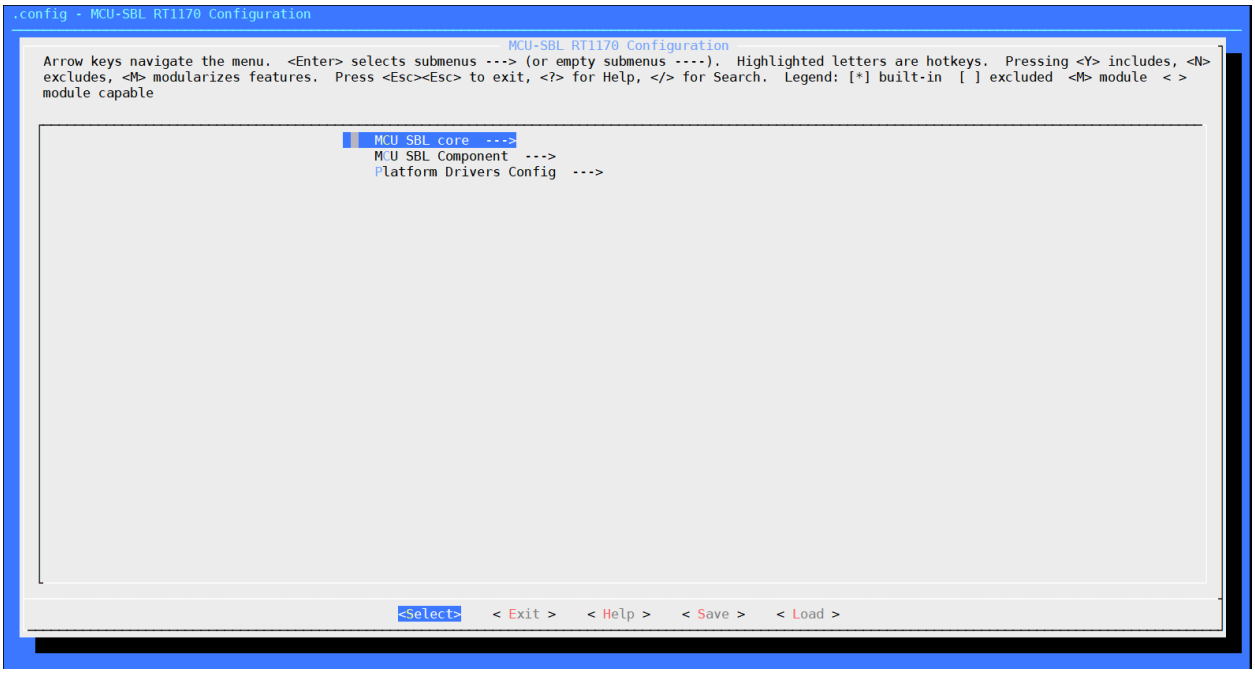


图12. SBL配置菜单

在这个菜单中，根据平台和bootloader具体的情况来配置SBL项目，比如是否启用单一映像功能。配置完成后，保存配置并退出菜单。

7. 使用GCC_ARM工具链构建映像

```
$ scons
```

`sbl.bin` 映像被构建在 `sbl/target/evkmimxrt1170/build` 目录中。

8. 烧写映像。

使用micro USB数据线将 EVKMIMXRT1170 电路板连接到计算机上，通过 DapLink 拖放或其他工具对 sbl.bin 进行烧写。将电路板设置为XIP启动模式并复位，以启动SBL。

2.3 安全固件

SFW项目的架构与SBL项目相似，因此快速开始的步骤与2.1节和2.2节中介绍的SBL相同，除了以下步骤：

1. 克隆SFW项目并checkout到1.1.0版，或下载发布包。

   ```
   git clone https://github.com/NXPmicro/sfw.git
   ```

2. SFW支持两种调试模式：单独的SFW项目XIP或SFW生成与SBL一起使用的bin文件。SFW通过 `scons –menuconfig` 来配置使用哪种模式。

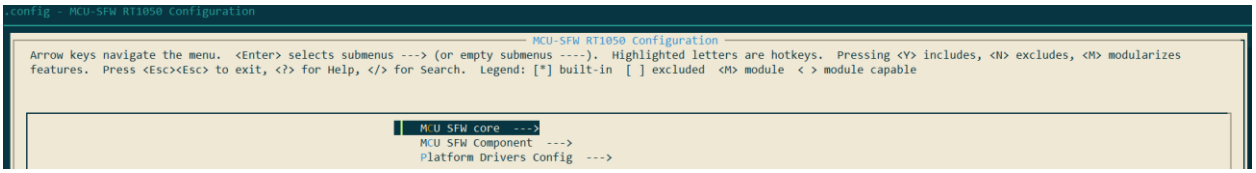


图13. SFW配置菜单

然后选择 `MCU SFW core`，在 MCU SFW core 菜单中，如果选择 `Enable sfw standalone xip` 选项，SFW项目将单独XIP。如果没有选择 `Enable sfw standalone xip` 选项，SFW会生成与SBL一起使用的bin文件。

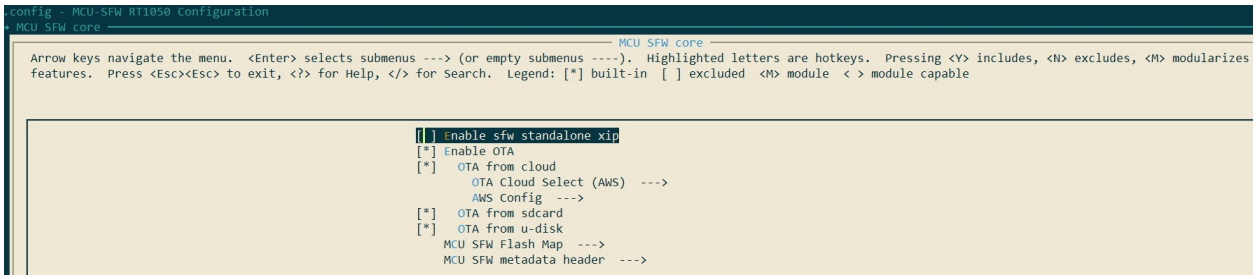


图14. SFW调试模式配置

第三章
框架

本章介绍SBL和SFW的构建框架。SBL和SFW项目是由 SCons 软件构建工具构建的，并由 Kconfig 文件进行配置。

3.1 SCons

本节介绍 SCons 软件构建工具。

3.1.1 概述

SCons是一个用Python编写的开源构建系统，类似于GNU Make。然而，它使用SConstruct和SConscript文件而不是通常的Makefile文件。这些文件也是Python脚本，可以用标准的Python语法来编写。因此，在SConstruct 和SConscript文件中，可以调用Python标准库来进行各种复杂的处理，而不局限于Makefile所设定的规则。

在使用SCons和Python工具之前，应该先安装它们。在Windows主机上，不需要安装这些SCons和Python，因为SBL中的Env配置工具已经包含了它们。在Linux主机上，Python应该默认被安装，SCons可以按照2.1节的命令安装。

3.1.2 SConscript 和 SConstruct

SCons使用SConscript和SConstruct文件来组织源代码结构。

在每个SBL和SFW平台目录中都存在以下三个文件：`sblconfig.py`（用于SBL）或 `sfwconfig.py`（用于SFW）、SConstruct和SConscript（控制平台的编译）。一般来说，一个平台中只有一个SConstruct文件，但有多个SConscript文件。

SConscript文件可以控制源代码文件的添加，并可以指定源代码文件的“组”（类似于IDE中“组”的概念，如MDK/IAR）。

SConscript文件也存在于SBL和SFW项目的大多数源代码文件夹中。这些文件被特定平台目录下中的SConscript文件所“找到”，以便将与某些宏相对应的源代码添加到编译中，这些宏是在 `sblconfig.h` 或 `sfwconfig.h` 中定义的。

3.1.3 基本命令

本节介绍了一些基本的SCons命令。在Windows主机上，这些命令在目标目录中的特定平台的 `env.bat` 文件中使用。在Linux主机上，这些命令直接在特定的平台目录中使用。

1. `scons`

   为特定平台构建项目。

   如果某些源文件在执行命令后被修改，当再次执行 `scons` 命令时，SCons会执行增量编译，只编译和链接修改后的源文件。

   调用Kconfig文件来配置项目并生成 `sblconfig.h` 文件。

2. `scons --ide=xxx`

   为特定平台生成IAR或MDK项目。

   使用 `scons --ide=iar` 命令来生成一个IAR项目。

   使用 `scons --ide=mdk5` 命令来生成一个MDK项目。

3.2 Kconfig

SBL项目使用由Kconfig文件生成的配置文件 `sblconfig.h` 来配置系统，SFW项目使用 `sfwconfig.h`。Kconfig文件是各种配置接口的源文件。

所有配置工具都是通过读取当前平台目录中的Kconfig文件来生成配置界面。这个文件是所有配置的总条目，它包含了其他目录中的Kconfig文件。配置工具读取每个Kconfig文件，生成配置界面供开发人员配置系统，最后生成SBL系统的配置文件 `sblconfig.h` 和SFW的 `sfwconfig.h`。

当使用env工具或Linux主机在特定平台（target/xxx/）目录下执行 `scons --menuconfig` 命令时，将出现SBL和SFW系统的配置界面，如图15和图16所示。

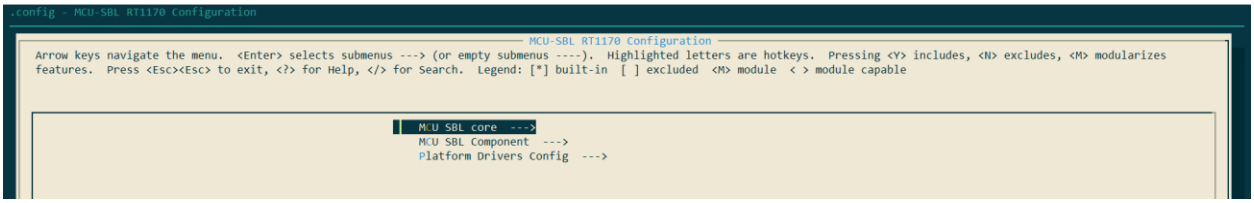


图15. SBL menuconfig 菜单

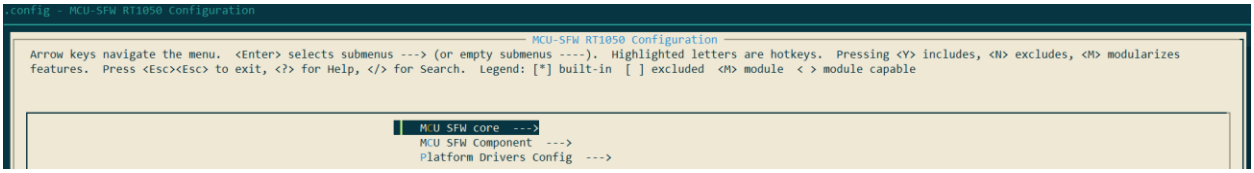


图16. SFW的 menuconfig 菜单

在这个菜单中，有3个子菜单可以选择。例如，要选择MCU SBL core，请使用下面的子菜单：

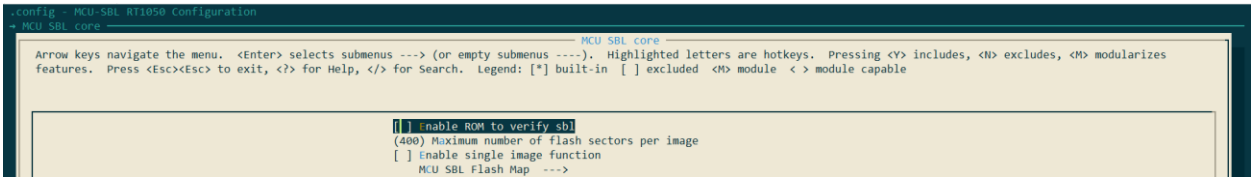


图17. MCU SBL子菜单

在这个菜单中，有一些可配置的选项，按 'y' 是包括该项，按 'n' 是排除该项。

配置完所有选项后，保存配置并退出该菜单。然后就可以编译项目了。

3.3 Host tool 主机工具

恩智浦提供各种主机工具来帮助SBL和SFW的开发和测试。以下是3个基本工具，更多其他工具请访问恩智浦官方网站或联系FAE。

1. MCUXpresso配置工具

   MCUXpresso配置工具是一套集成的配置工具，有助于指导用户从首次评估到软件开发生产。这些配置工具允许开发人员快速构建定制的SDK，并利用引脚、时钟和外设工具来生成初始化C代码，以支持定制电路板。在SBL目标平台中，有一个 `MCUX_Config.mex` 文件，可由MCUXpresso配置工具打开，并帮助生成特定的C代码，用于时钟、引脚等。例如：`target/evkbmimxrt1050/board/MCUX_Config/MCUX_Config.mex`。更多信息请参考网站。

2. Bootloader Host Application (blhost)

   blhost 应用程序是一个命令行程序，用于在主机上发起通信，并通过UART或USB连接向MCU ISP模块发出命令。该应用程序每次调用只发送一条命令。blhost 应用程序支持多平台，包括Windows、Linux（基于X86）、MACOSX和Linux（基于Arm）。欲了解更多信息，请访问网站。

3. MCU Boot Utility

   恩智浦MCU Boot Utility是一个专门为恩智浦MCU安全启动设计的GUI工具。它的功能与恩智浦MCU中的BootROM功能相对应。目前，它主要支持i.MXRT系列MCU芯片。与恩智浦官方安全工具集（OpenSSL、CST、sdphost、blhost、elftosb、BD、MfgTool2）相比，恩智浦MCU启动实用程序是一个真正的一站式工具，这个工具包含了恩智浦官方安全工具集的所有功能，更重要的是，它支持全图形用户界面操作。有了恩智浦MCU Boot Utility，就可以很容易地开始使用恩智浦MCU安全启动。恩智浦MCU Boot Utility的主要功能包括：

   - 支持UART和USB-HID两种串行下载模式
   - 支持各种用户应用程序映像文件格式（elf/axf/rec/hex/bin）
   - 能够验证用户应用程序映像的范围和适用性
   - 支持将裸映像转换为可启动映像
   - 支持将可启动映像加载到外部启动设备中
   - 支持普通启动设备烧录操作（闪存烧录器）

   有关MCU Boot Utility的更多信息，请参考网站。

第四章
MCU ISP

本节介绍MCU ISP的具体情况。

4.1 关于ISP

MCU ISP提供闪存烧录程序，通过MCU上的串行连接进行操作。它使MCU的映像烧录快速而简单。主机端命令行和GUI工具可用于与SBL设备通信。用户可以利用主机工具来上传/下载应用程序代码，并通过MCU ISP进行大量生产。

4.2 功能

- 支持UART和USB外设接口。
- 支持恩智浦blhost工具和恩智浦MCUBootUtility GUI工具。
- 自动检测可用的外设。
- 用户定义的检测可用外设超时时间。
- UART外设上的自动波特率。
- 保护SBL在运行时使用的RAM。
- Nor Flash烧写。

4.3 设置ISP超时时间

在SBL menuconfig中，当MCU ISP特性被使能时，可以设置ISP超时时间，默认超时时间为5秒。如果SBL在超时时间内没有收到来自主机的ISP命令，那么ISP流程将被绕过。

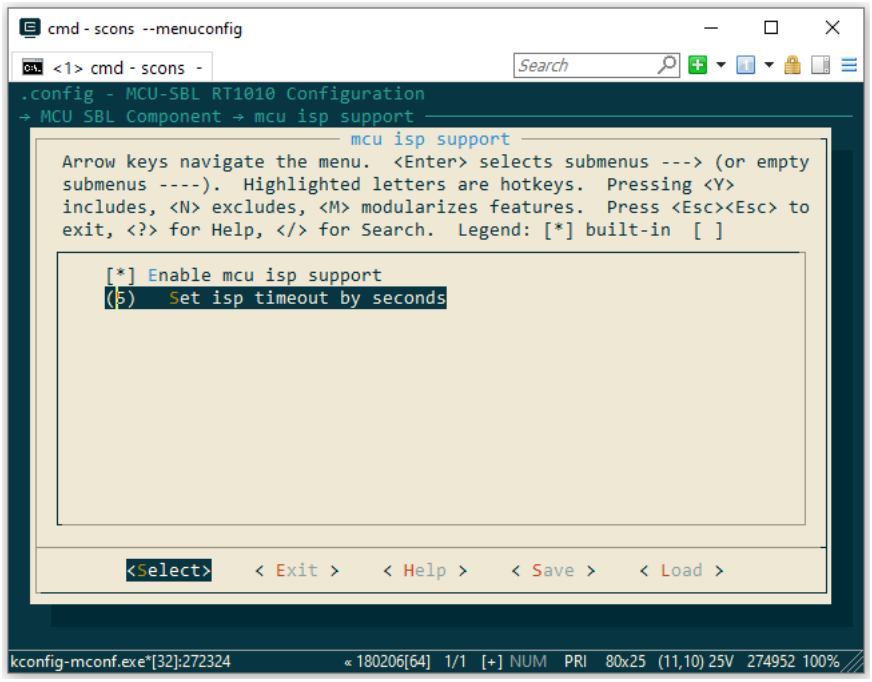


图18. SBL menuconfig 设置超时时间

如果默认的5秒超时值不够，请选择此选项并编辑超时值。

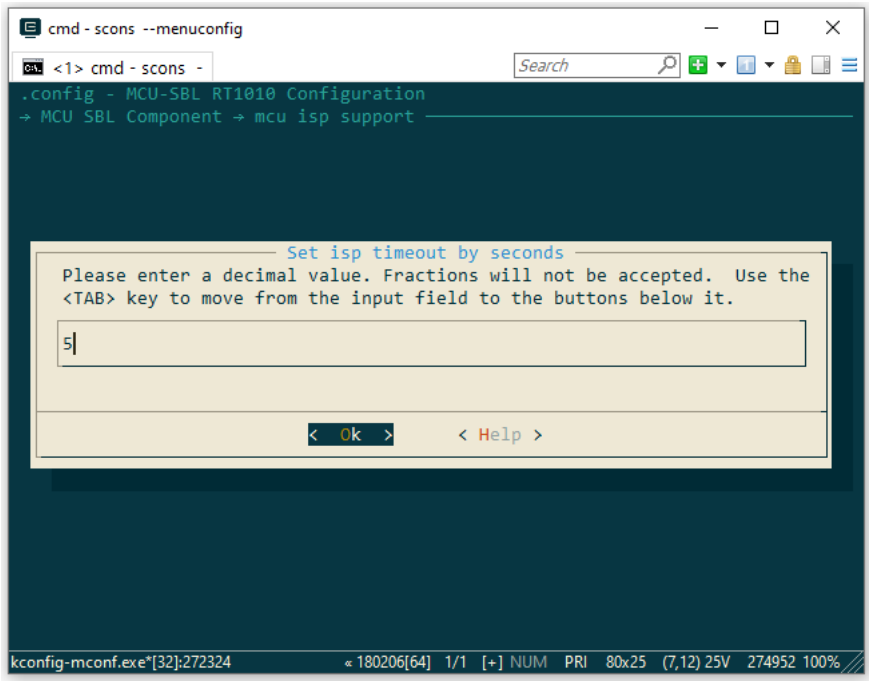


图19. SBL menuconfig 设置超时

4.4 使用MCU Boot Utility

建议使用恩智浦MCU Boot Utility GUI工具（v3.3或更新版本）作为ISP下载的首选主机工具。想使用blhost命令行工具的客户，请与恩智浦联系。有关详细信息，请参见下面的步骤：

1. 打开MCUBootUtility，在菜单的工具/运行模式中设置模式为 `SBL OTA`。

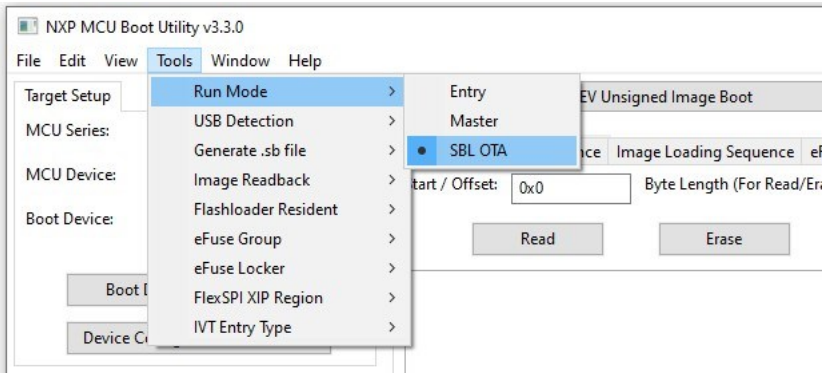


图20. 设置工具运行模式

2. 打开SBL目标电路板的电源（以EVKMIMXRT1010为例），然后将USB数据线连接到J9。如果一切正常，USB vid/pid会被检测到。点击“连接到SBL ISP”按钮。

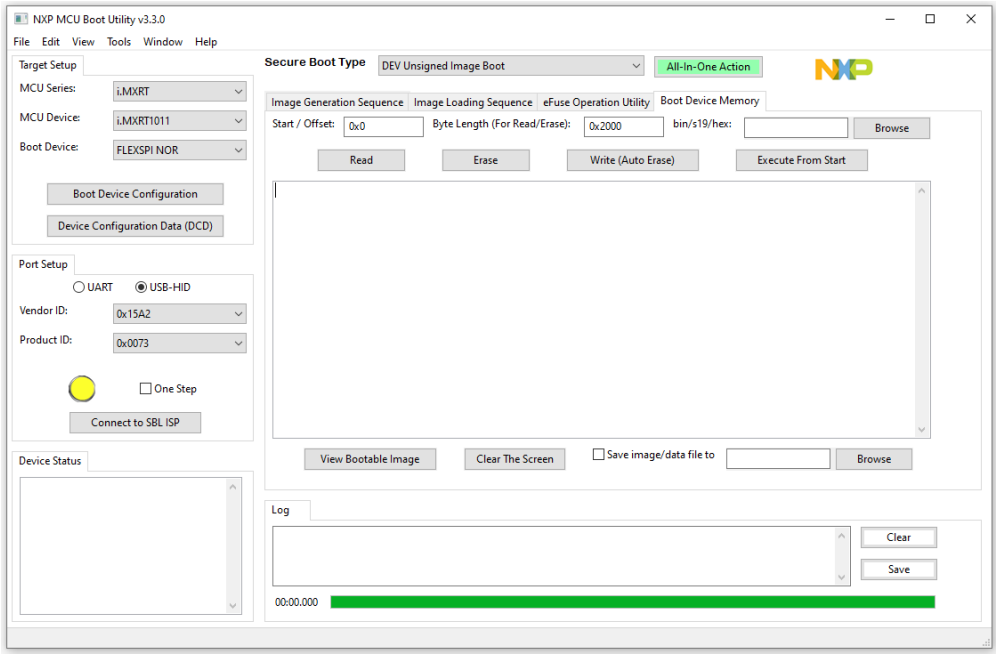


图21. 检测到USB IDs

3. 现在可以做读/擦除/写ISP操作。映像格式可以是bin/hex/s19

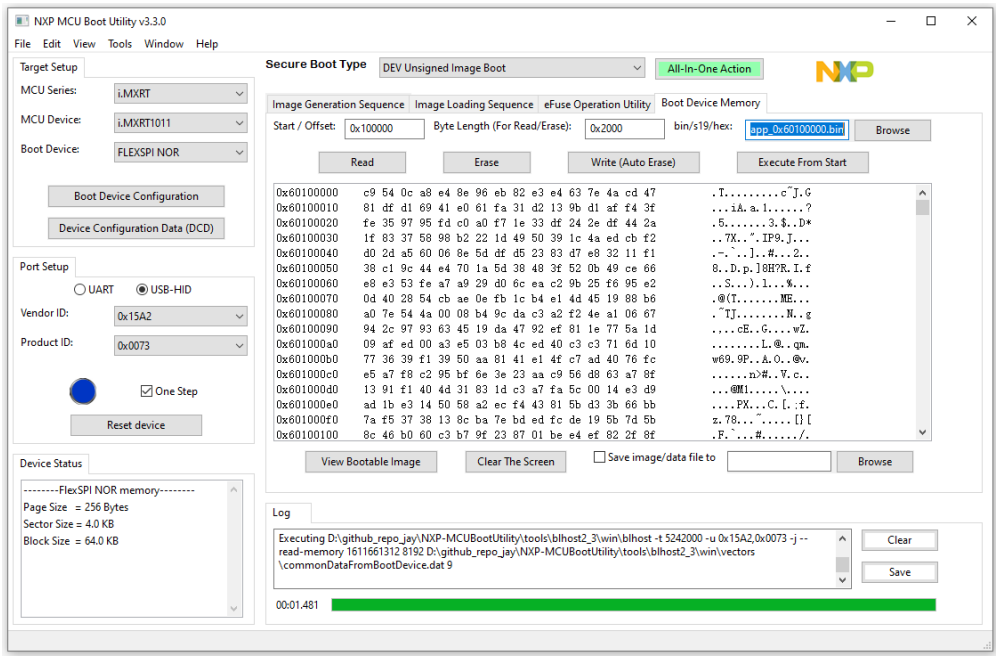


图22. 下载应用程序

第五章
安全

本节介绍已经实现的安全功能。安全引导装载程序（SBL）是基于MCUboot项目的。SBL保留了MCUboot传统的RSA和ECDSA签名。它还提供基于ROM引导加载程序的安全启动和基于硬件引擎的加密启动（XIP）。因此，image可以是签名的，加密的，或签名+加密的。

MCUboot传统的签名方法RSA和ECDSA通过计算image的哈希值，然后签署该哈希值来对image进行签名。详细情况请参考MCUboot设计文档。SBL默认使用RSA-2048和ECDSA-P256。

SBL能够支持MIMXRT 4位平台（MIMXRTxxxx）、MIMXRT 3位平台（MIMXRTxxx）和LPC55S69的ROM安全启动和加密启动（XIP）。

5.1 BootROM安全启动

安全启动保证未经授权的代码不能在给定产品上执行。因为设备复位后始终从ROM开始执行。ROM检查常位于闪存中的第一个用户可执行镜像，以确定该代码的可信性。如果该代码是可信的，控制权就会被转移给它。这会建立一个从ROM到用户启动代码的信任链。在这种情况下，BootROM验证SBL，SBL验证应用程序镜像。

5.1.1 高可靠性启动（HAB）

恩智浦MIMXRT 4位平台提供高可靠性启动（HAB）。它是系统启动ROM中的高可靠性启动功能，可在启动过程中检测并防止执行未经授权的软件（恶意软件）。

HAB使用非对称加密技术对image进行签名。启动image可以由CST工具签名。该工具生成CSF（command sequence file）的数据以二进制形式存储在文件中，它由命令序列和基于给定输入命令序列文件（CSF文件）的签名组成。

原始设备制造商（OEM）使用恩智浦提供的实用程序来生成一对公私钥对。然后私钥被用来加密OEM要发布的镜像摘要。这种加密为镜像生成了一个唯一的标识符，被称为签名。带有公钥的证书也被附加在镜像中。在执行该应用程序之前，公钥被用来验证签名。OEM将公钥的摘要（哈希值）刻录到MIMXRT芯片的eFuses中。一旦被烧写，它就不能被修改。BootRom可以通过这个值来验证公钥。

下面是HAB的可启动镜像格式。

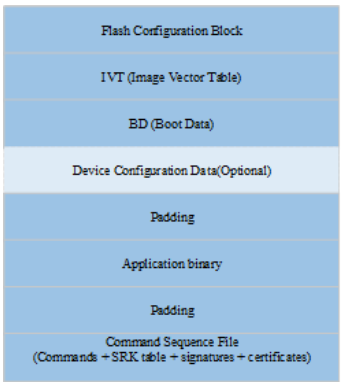


图23. 无MCUboot标头的HAB签名的镜像格式

它不包含可启动应用程序中内嵌的闪存配置块（FCB），因为BootROM已经通过读取SBL的这个字段配置了闪存。所有MIMXRT平台都支持RSA（1024、2048、3072或4096）。MIMXRT1170还支持使用ECC（P256、P384、P521）进行ECDSA签名验证。

5.1.2 LPC55S69安全启动

设备LPC55S69使用RSASSA-PKCS1-v1_5签名方案，并支持从RSA签名的镜像启动。启动代码是用RSA私钥签名的。已签名的镜像中包含用于签名验证的相应RSA公钥。

LPC55S69设备支持2048位或4096位的RSA密钥和X.509 V3证书。

镜像验证是一个两步的过程。

1. 验证并从嵌入在镜像中的x509证书中提取公钥。

2. 使用Image_key（Public）来验证镜像签名。

BootROM API skboot_authenticate 是用来验证镜像的可信性。在运行要使用这个IAP API的应用程序之前，应该先配置PFR区域（CFPA和CMPA）。

PFR位于flash的末端，可以在ISP模式下通过ROM进行烧写。

LPC55S69在受保护的flash区域（PFR）中存储ROM启动时需要的配置信息。

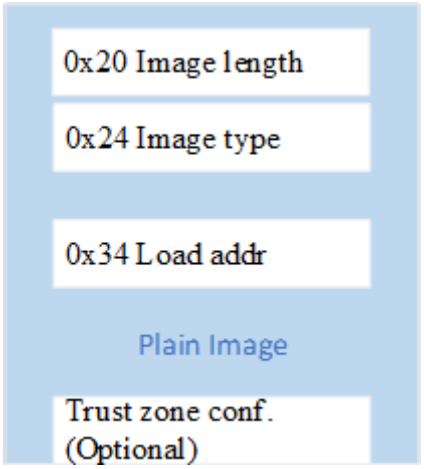


图24. 无MCUboot标头的签名的映像格式

5.1.3 加密的XIP启动

MIMXRT 4位系列BootROM具有由BEE/OTFAD控制器提供的即时解密功能（使用AES），直接支持“串行NOR闪存设备”上的片上执行（XIP）。

PRINCE用于对LPC55S69片上flash内容进行实时加密/解密操作。

5.1.3.1 基于BEE的加密XIP启动

EVKMIMXRT1060/1064/1050/1020支持XIP，这是通过“总线加密引擎”（BEE）和FlexSPI（QSPI）flash即时解密而实现的。BootROM支持使用两个独立的AES密钥和两个独立的加密区域。一个加密区域可用于SBL，另一个可用于应用程序。image可以由AES-CTR-128或AES-ECB-128进行加密。

在进行加密XIP之前，BootROM必须正确设置BEE控制器，可配置的参数被组织为保护区域描述符块（PRDB）。整个PRDB使用AES-CBC-128模式加密且加密PRDB的AES KEY和IV在密钥信息块（KIB）中。通过使用eFUSE（SW_GP2）中提供的AES密钥或从OTPMK（一次性可编程主密钥）中衍生的AES密钥，KIB被加密为加密KIB（EKIB）。BootROM使用AES ECB-128模式对KIB进行解密，最多支持2个EKIB，EKIB0位于偏移0x400，KIB1位于偏移0x800。

在此解决方案中，SW_GP2被用作KEK来加密密钥信息块（KIB）。

可以使用`image_enc.exe`工具对主机上的image进行加密。它是一个命令行主机程序，客户可以使用它来验证加密流程。

5.1.3.2 基于OTFAD的加密XIP启动

EVKMIMXRT1170/1010 和 EVKMIMXRTxxx 支持XIP，这是通过“即时AES解密模块”（OTFAD）对FlexSPI(QSPI) flash进行即时解密而实现的。OTFAD支持多达4个使用独立AES密钥的独立加密区域。

在启动加密XIP之前，BootROM必须正确设置OTFAD模块，可配置的参数被组织为KeyBlob。KeyBlob包含OTFAD的加密密钥，并且总是用KEK进行加密。KEK可以对每个加密区域进行加扰处理。整个KeyBlob使用AES-CTR-128模式进行加密。KeyBlob在flash的偏移为0x0。

KEK被存储在OTP/EFUSE块中。对于EVKMIMXRT1170，KEK也可以由PUF存储恢复，恢复密钥时使用PUF的key code和加密的image一样存储在flash上。

在这个解决方案中，使用了两个KeyBlob。一个KeyBlob用于SBL，另一个用于应用程序。

5.1.3.3 基于PRINCE的加密XIP启动

LPC55S69 支持通过PRINCE对内部flash进行即时加密/解密。存储在片上内部flash中的数据可以被实时加密。

LPC55S69支持3个区域，允许来自不同加密密钥的多个代码映像共存。每个PRINCE区域都有一个密钥，由片上SRAM PUF通过秘密总线接口提供（SW不可访问）。PRINCE加密算法不增加延迟。

PRINCE密钥是128位对称密钥，通过内部硬件接口从片上SRAM PUF获取，而不在系统总线上暴露该密钥。

PUF控制器提供安全的密钥存储。这是通过使用从SRAM衍生的设备的数字指纹来完成的。不是存储密钥，而是生成一个密钥码（key code），该码与数字指纹一起用于重构PRINCE密钥，然后被路由到AES引擎或供软件使用的。这些密钥码存储在flash的PFR区。

在启动过程中，ROM检查PFR中是否存在有效的密钥存储数据结构。如果是的话，整个密钥存储的数据结构被加载到RAM中，ROM发起PUF的启动过程。它初始化PUF并重构原始密钥，以便在需要时每个密钥都可使用。

5.1.4 Image格式

下面是最终的文件格式。应用程序映像可以是签名的、加密的或签名的+加密的。如果image是加密的，则应将密钥上下文插入MIMXRT 4位平台的image标头部分。它在 MCUboot 标头的偏移为0x100。

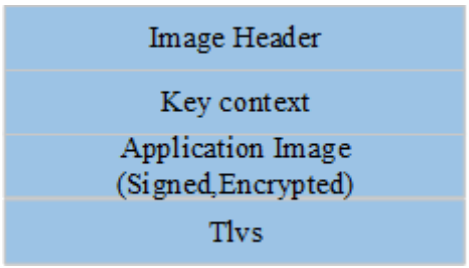


图25. 最终image格式

5.1.5 工具

要使用安全功能，请准备以下工具：

- CST工具（可选）– 代码签名工具，这是一个在构建主机上运行的应用程序，允许制造商对其包含恩智浦处理器的产品中的软件进行签名或加密。
- `elftosb.exe v4.0.0` – 结合CST用于生成无签名/有签名的可启动映像。
- `image_enc.exe` – 它是一个用于加密image的命令行主机程序。

- MCUXpresso Secure Provisioning Tool (SPT) – 它是一个GUI程序，用于简化恩智浦MCU平台上可启动执行文件的生成和烧录。

MCUX Secure Provisioning Tool 包括 `cst.exe`, `elftosb.exe`和 `image_enc.exe`。可从官网上下载它们。

在 `sbl\target\evkmimxrtxxxx\secure` 文件夹中，有一站式的脚本，可以用这些工具生成签名加密的image。更多细节，请参见第 7.4 节。

第六章
固件

本节介绍安全固件（SFW）的操作细节。

6.1 安全固件

安全固件（SFW）是一个基于FreeRTOS创建和开发的应用实例，它与SBL一起实现完整的FOTA过程。SFW支持通过本地的U盘、SD卡或远程的AWS云、阿里云获取OTA固件映像并写入闪存。然后，SBL对固件镜像进行检查、验证，并正常启动。

SFW采用与SBL相同的框架，它们具有相同的构建环境、配置流程和编译命令。一旦熟悉了SBL，使用SFW就不难了。

不管是交换还是重映射模式，SFW提供了一个统一的函数 `enable_image()`，让用户在向闪存写入新的映像后调用。由于这两种模式的标志机制不同，SFW使用宏来区分它们。

6.2 设置OTA标志的操作

本节给出设置OTA标志的详细信息。

6.2.1 交换模式OTA的操作

对于交换模式的OTA，两个slots的 image_trailer（trailer在两个slots的最后32字节中）被用来判断交换类型和控制回滚。图26显示了该标志的状态。0xFF表示未设置，0x01表示已置位。

图26. 交换标志状态

为了初始化OTA进程，在将新固件写入slot2后，接收新固件的旧固件程序必须将magic字段（固定值，16字节）写入slot2的末尾，以通知引导加载程序新固件已写入slot2。在写入magic值后，复位电路板。

引导加载程序开始检测OTA类型，即test类型。然后引导加载程序执行交换，在交换过程中，slot1中的trailer被slot2中的trailer所覆盖，而原本slot2中trailer的位置会被清除。在交换完成后，引导加载程序将会跳转到slot1以执行新固件。如果新固件运行正常，它将 `image_ok` 标志写入slot1 的trailer，以禁用回滚。否则，一旦新固件出现错误，`image_ok` 标志没有被设置，并且看门狗复位电路板，引导加载程序再次判断OTA类型，现在类型为Revert，引导加载程序将会交换两个slots的内容，并且清空slot2的trailer，现在两个slots的trailer都为未设置状态。

注意：对于使用交换模式OTA的电路板，固件必须包含两个写标志位的操作。首先，写入标志位的magic部分，这个操作必须在新固件写入后进行，magic字段的地址是0x2FFFF0。第二个操作是写 `image_ok` 标志。在固件本身运行了整个任务周期且这期间一切正常后，新固件必须设置该标志。`image_ok`字段 的地址是0x2FFFE8。Magic的值如图27所示。

```
const uint32_t boot_remap_magic[] = {
    0xf395c277,
    0x7fefd260,
    0x0f505235,
    0x8079b62c,
};
```

图27. Magic值

6.2.2 重映射模式OTA的操作

对于重映射模式的OTA，使用重映射更新标志位来判断重映射类型和控制回滚。该标志位于闪存的固定偏移地址，偏移为0xFFFE0，标志结构占用32字节空间。图28 显示了该标志位的状态。未设置为0xFF，设置为0x01，回滚为0x04。

图28. 重映射标志状态

为了初始化OTA过程，在将新固件写入slot2或slot1后，接收新固件的旧固件必须将magic值（固定值，16字节，与交换模式的值相同）写入magic标志的位置，以告知引导加载程序新固件已经写入slot1或slot2。写完magic值后，复位该电路板。

引导加载程序读取重映射更新标志，来获得当前固件位置并判断OTA类型，现在类型是test类型。如果当前位置是0x01 (slot1)，将标志的 `image_ok` 部分设置为0x04（表示回滚），启用重映射功能并运行物理上存储slot2的固件。如果新固件运行正常，它会清除 `image_ok` 和magic字段，以禁用回滚。否则，如果新固件发生错误，未清除标志，并且看门狗复位电路板，引导加载程序再次判断OTA类型，现在类型是回滚，翻转重映射功能的状态，并清除 标志位的 `image_ok` 和magic字段。

注意：对于使用重映射模式OTA的板子，固件必须包含两个写标志的操作。首先，写入标志的魔法部分，这个操作必须在新固件写入后进行，魔法地址是0xFFFF0。第二个操作是写 `image_ok` 标志。在固件本身运行整个任务周期并且在此期间一切正常之后，固件必须清除这两部分标志。

第七章
FOTA

SBL是为在线固件升级应用设计的二级引导加载程序。它通过读、验证和把固件映像写到内部/外部闪存设备，来存储和管理OTA固件升级。

它提供了以下OTA功能：

- 映像交换和回滚
- 映像重映射和回滚
- FlashIAP
- 安全
- ISP

7.1 设计

本节专门介绍在线固件升级应用的设计。

7.1.1 OTA单一映像模式

单一映像升级模式的闪存布局。

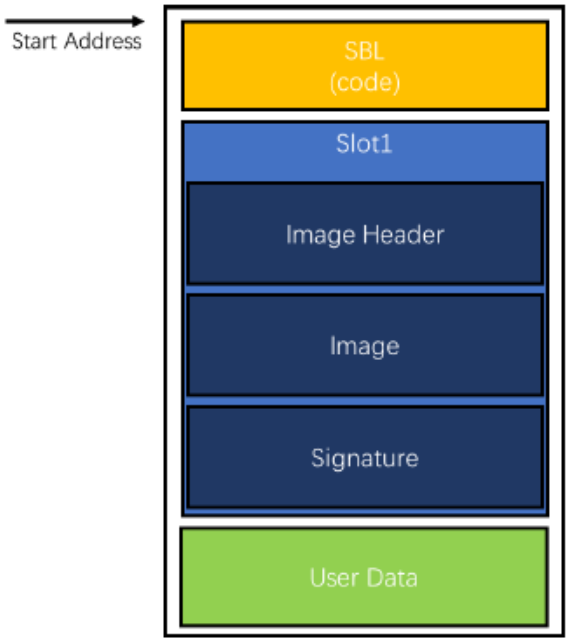


图29. 单一映像模式的闪存布局

单一映像升级模式的工作流程如下：

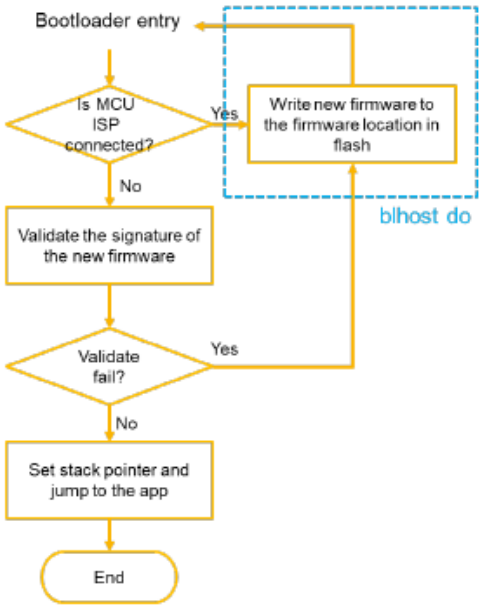


图30. 单一映像模式的引导加载程序工作流程

单一映像升级模式的周期如下：

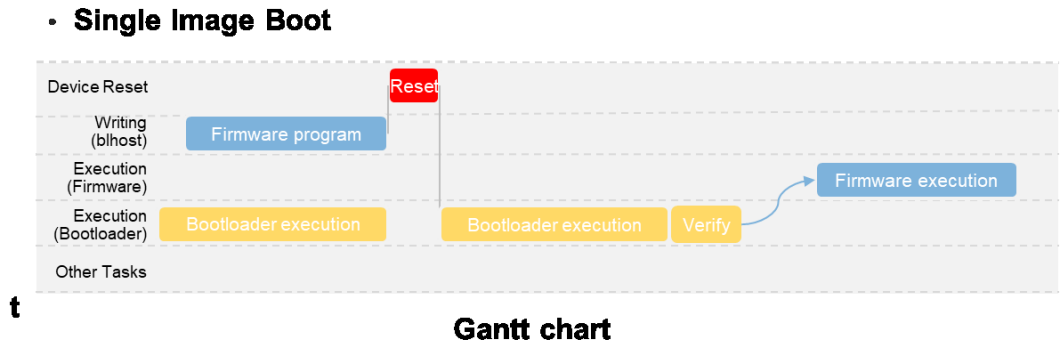


图31. 单一映像模式引导加载程序的更新周期

7.1.2 OTA的互换模式

升级映像本身由映像标头、映像数据和映像trailer组成。映像标头信息如下表所示。

表5. 映像标头格式

偏移	宽度（字节）	字段	说明
0x00	4	Magic	映像标头标签 Fixed value固定值
0x04	4	load_addr	指向应用程序的加载地址

表格在下一页继续.....

表5. 映像标头格式（续）

偏移	宽度（字节）	字段	说明
0x08	2	header_size	映像标头的大小
0x0a	2	reserved	保留给未来使用
0x0c	4	image_size	映像的大小（不包括映像标头的大小）
0x10	4	flags	现在不使用
0x14	8	image_version	映像版本
0x1c	4	reserved	保留给未来使用

映像数据是实际的映像内容，它支持原始二进制映像格式。

映像trailer的信息如下表所示。

表6. 映像trailer的格式

偏移	宽度（字节）	字段	说明
0x00	1	copy_done	映像交换完成的标志
0x01	7	Pad	保留
0x08	1	image_ok	控制OTA状态的标志
0x09	7	pad	保留
0x10	16	magic	映像trailer标签 固定值

OTA交换模式的闪存布局如下：

图32. 交换模式的闪存布局

- SBL驻留在闪存的起始位置。
- 现在的交换区域被省去，因为在两个slot中各自都预留了一个扇区的空间用作交换空间。

OTA交换模式的工作流程如下：

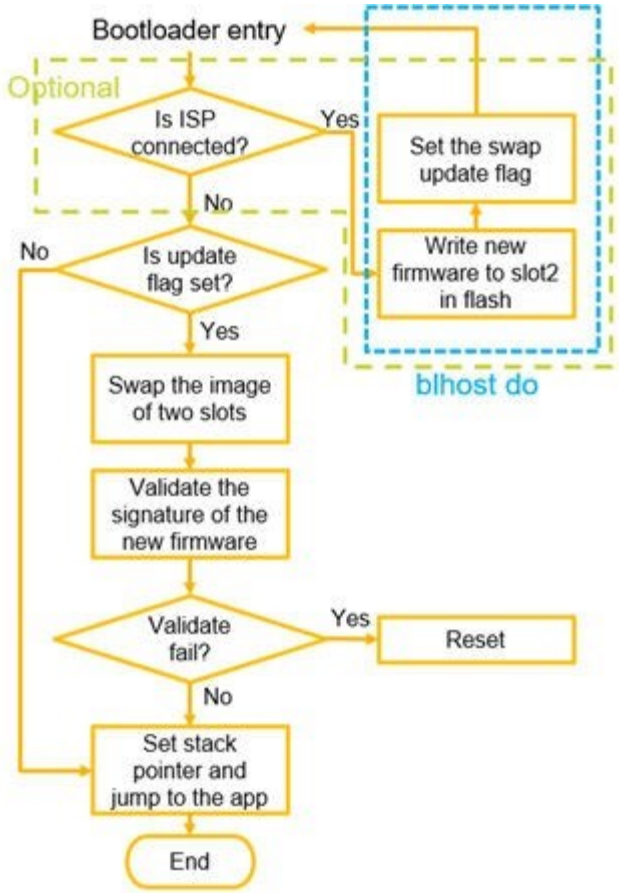


图33. 交换模式的引导加载程序工作流程

OTA互换模式的周期如下：

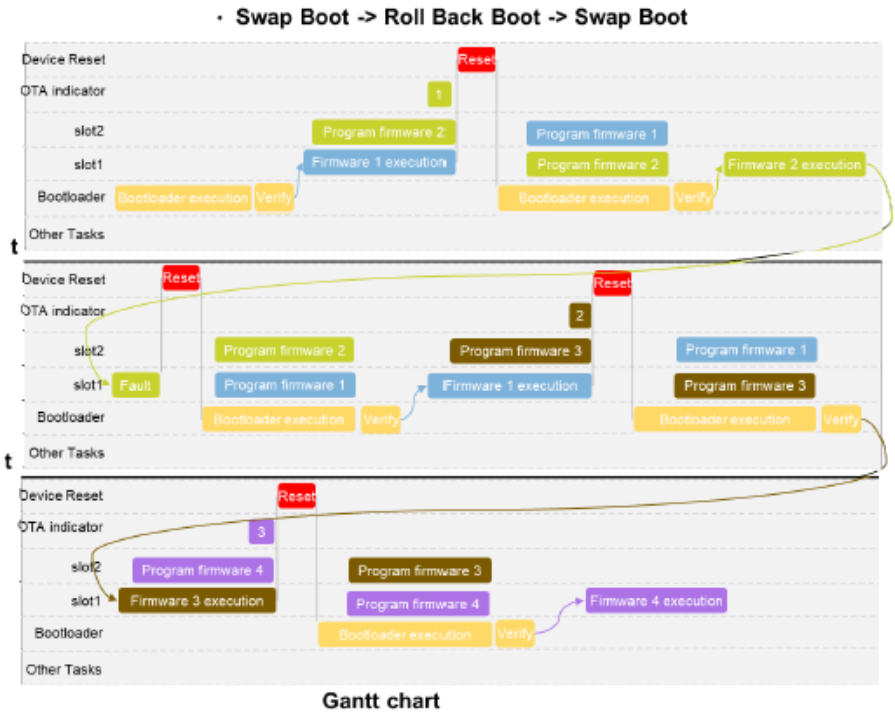


图34. 交换模式的引导加载程序的更新周期

7.1.3 OTA重映射模式

重映射模式的OTA映像包含映像标头和映像数据。映像标头的信息与交换模式相同，参考表5。

为了控制重映射状态，在slot1之前，设置重映射更新标志结构来控制更新过程。其格式如下表所示。

表7. 重映射标志格式

偏移	宽度（字节）	字段	说明
0x00	1	Image_position	当前固件位置
0x01	7	Pad	保留
0x08	1	image_ok	控制OTA状态的标志
0x09	7	pad	保留
0x10	16	magic	映像trailer标志 固定值

OTA重映射模式的闪存布局如下：

图35. 重映射模式的闪存布局

OTA重映射模式的工作流程如下：

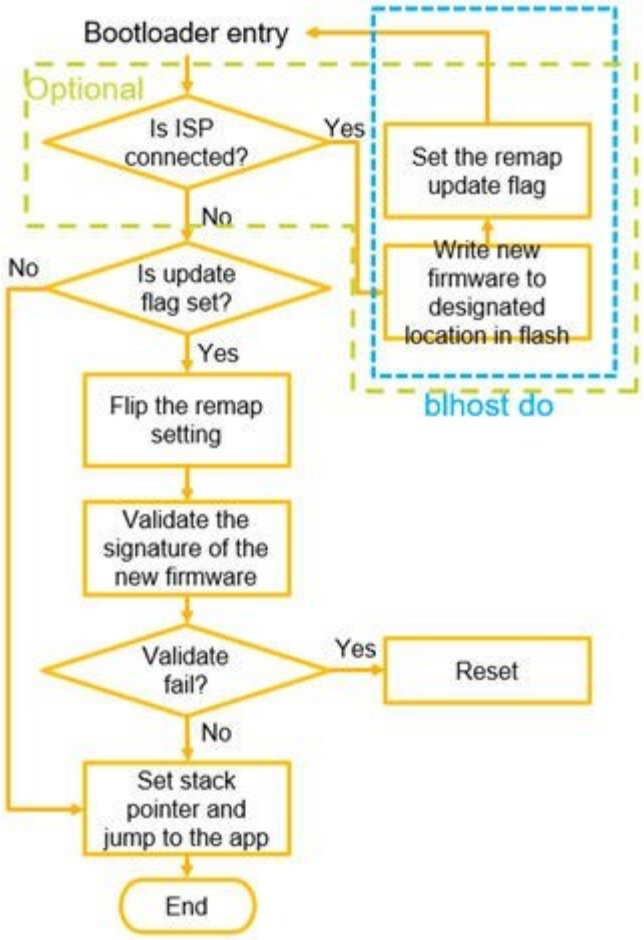


图36. 重映射模式的引导加载程序工作流程

OTA重映射模式的周期如下：

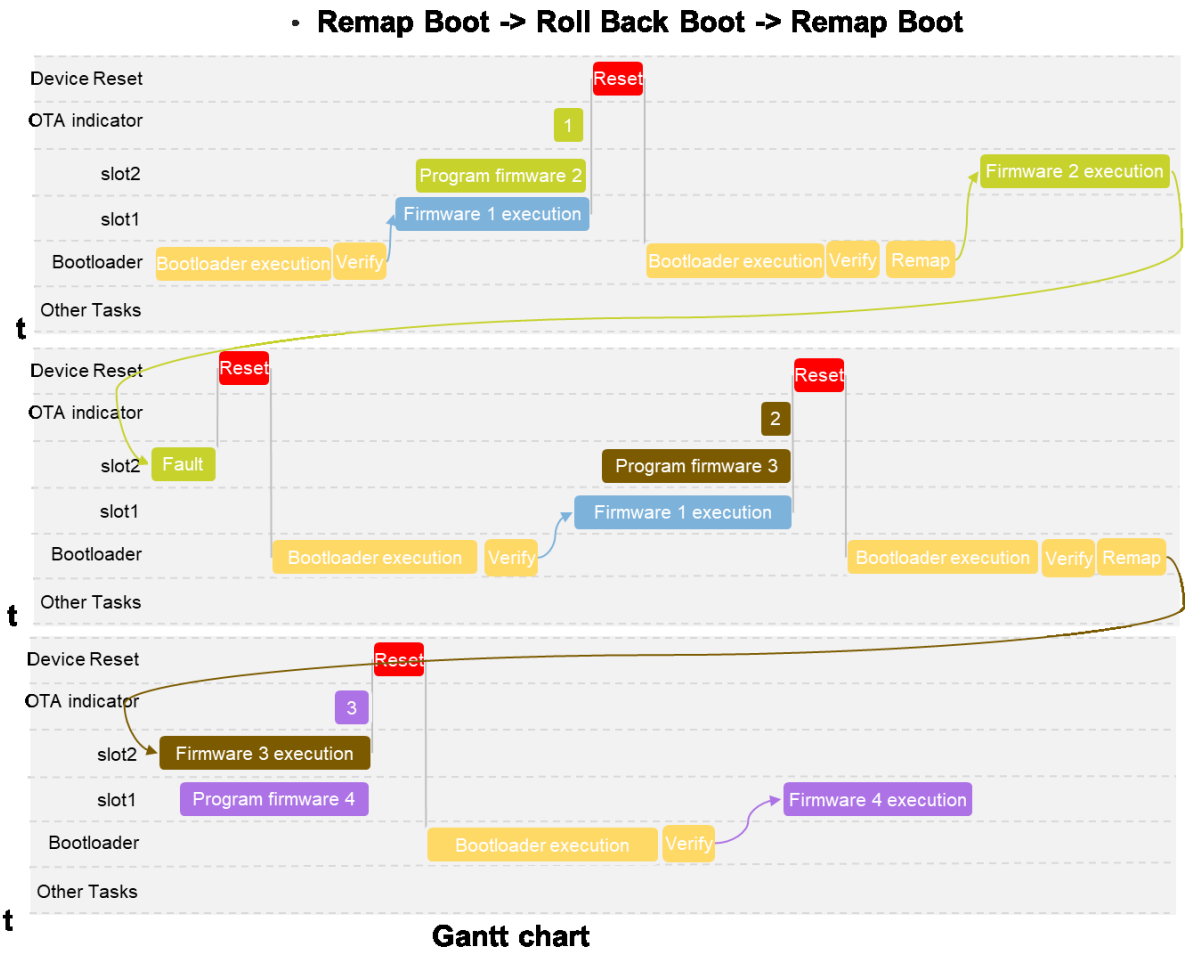


图37. 重映射模式引导加载程序的更新周期

7.2 本地FOTA

对于所有3种OTA模式（单一、交换、重映射），SBL的默认配置必须要验证映像的签名，因此在用IAR/MDK/GCC生成映像文件后，需要在映像文件中添加标头和签名。下面的步骤介绍如何制作一个可用的应用程序映像。

所有的SBL支持的板卡都可以支持U盘来更新映像；除 EVKMIMXRT1010 外，所有这些板卡都可以支持SD卡更新。

1. 准备好映像

   对于单一映像模式：

   选择SDK中的“hello world”演示作为示例。首先，为了给以后添加头文件腾出空间，需要修改链接文件。默认的应用程序偏移地址是0x100000，修改链接文件以适应这个地址，下图中 EVKMIMXRT1060 的IAR链接文件可以作为参考。

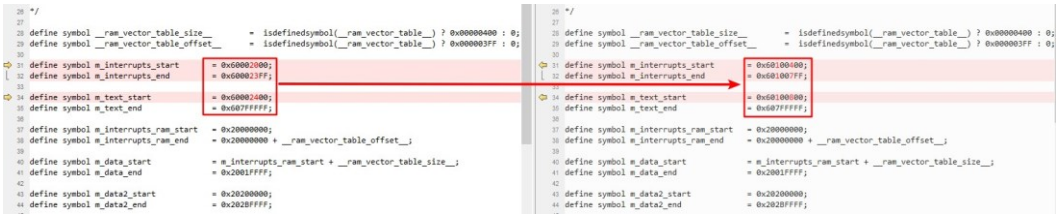


图38. 链接器文件修改示例

通过在iar项目选项中设置 `XIP_BOOT_HEADER_ENABLE = 0` 来删除XIP标头信息，然后编译项目并生成一个名为 `hello_world.bin` 的二进制文件。

SFW项目已经包括了上述更改，因此在单一映像模式下可以直接构建和使用SFW映像。

对于交换模式和重映射模式：

双击SFW相应目标目录下的 `env.bat`，然后使用命令进入配置菜单，取消勾选 `Enable sfw standalone xip` 选项，并勾选 `OTA from sdcard` 和 `OTA from u-disk` 选项。

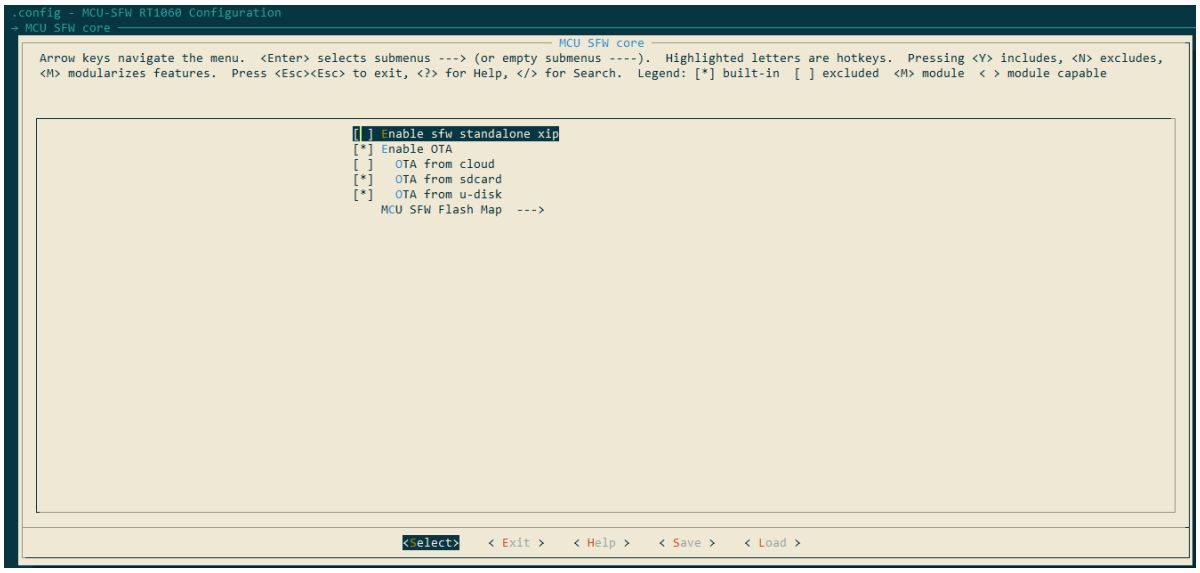


图39. 配置SFW

按照第 2 章中描述的命令来生成项目，构建项目并生成映像。

2. 生成签名密钥对并准备引导加载程序

   双击SBL相应目标目录下的 `env.bat`。在弹出的shell环境中使用如下命令切换目录 `cd ..\..\component\secure\mcuboot\scripts`，然后使用该脚本命令来生成签名密钥：

   ```
   python imgtool.py keygen -k xxxx_priv.pem -t rsa-2048-sign
   ```

   `xxxx_priv.pem` 文件会用于对应用程序映像进行签名。

   生成公钥：`python imgtool.py getpub -k xxxx_priv.pem -o xxxx_pub.pem -t sign`

   上述命令生成的 `xxxx_pub.c` 文件包含了公钥的数据结构，它是一个数组。它应该和引导加载程序一起编译以验证签名。用它的内容来替换 `sbl\component\secure\mcuboot\` 目录下 `sign-rsa2048-pub.c` 中的内容。

3. 为映像添加签名和标头

使用 `imgtool` 命令生成有用的应用程序映像，在shell中输入命令完成操作。对于单一映像模式，使用步骤1中生成的 `hello_world.bin`。对于交换和重映射模式，要运行测试，使用SFW项目生成的映像。

并使用下面的命令来生成第一个已签名的映像：

```
python imgtool.py sign --key xxxx_priv.pem --align 4 --version "1.1" --header-size 0x400 --
pad-header --slot-size 0x100000 --max-sectors 32 hello_world1.bin app1.bin
```

使用与第一个映像不同的另一个映像图，并重复该命令：

```
python imgtool.py sign --key xxxx_priv.pem --align 4 --version "1.2" --header-size 0x400 --
pad-header --slot-size 0x100000 --max-sectors 32 hello_world2.bin app2.bin
```

现在生成了两个已签名的映像。关于上述签名命令的更多信息将在下面描述。

- `xxxx_priv.pem`：在步骤2中生成的私钥证书
- `--version`：版本格式可以是 `main.minor.rev`
- `hello_world.bin`, `app.bin`：还可以为 `hello_world1.bin`, `app1.bin`, `hello_world2.bin` 和 `app2.bin` 添加文件路径

注意

对于 EVKMIMXRT685，要启用重映射功能和单一映像功能（有复位操作），要使用J-link提前写入影子寄存器，具体步骤如下：

1. 取下JP2跳线帽。
2. 确保用J-link连接到 EVKMIMXRT685，然后用J-link提前写入影子寄存器，指令如下：`w4 0x40130184 0x314000`。
3. 影子寄存器写入后，要确保在整个操作过程中MCU不能断电，否则之前的操作无效。这个限制只存在于EVKMIMXRT685 FlexSPI端口b。为了保证整个操作过程中不掉电，在映像下载到闪存后，用SW3复位MCU，而不是掉电复位。此外，如果使用DAPLink下载映像，使JP2(PIN1-2)短接，然后去掉J-Link探针。

注意

对于 EVKMIMXRT595/EVKMIMXRT685 的U盘更新功能，请将EXT PWR端口接通电源。否则，电源电流不够，会导致U盘更新失败。

注意

最新的 EVKMIMXRT595 电路板的默认SDIO接口是eMMC，而不是SD卡。当使用U盘更新功能和最新电路板时，请先重写这些电路板。

7.2.1 单一映像OTA

1. 配置SBL

   单一映像功能只涉及到映像的签名验证，在SBL的阶段，没有对闪存进行擦除和写入操作。这种模式下的OTA必须通过MCU ISP进行。

   勾选Scons的menuconfig界面中的 `Enable single image` 功能选项，启用单一映像模式。同时，在Scons的menuconfig界面勾选 `Enable mcu isp support` 选项，以启用MCU ISP功能。

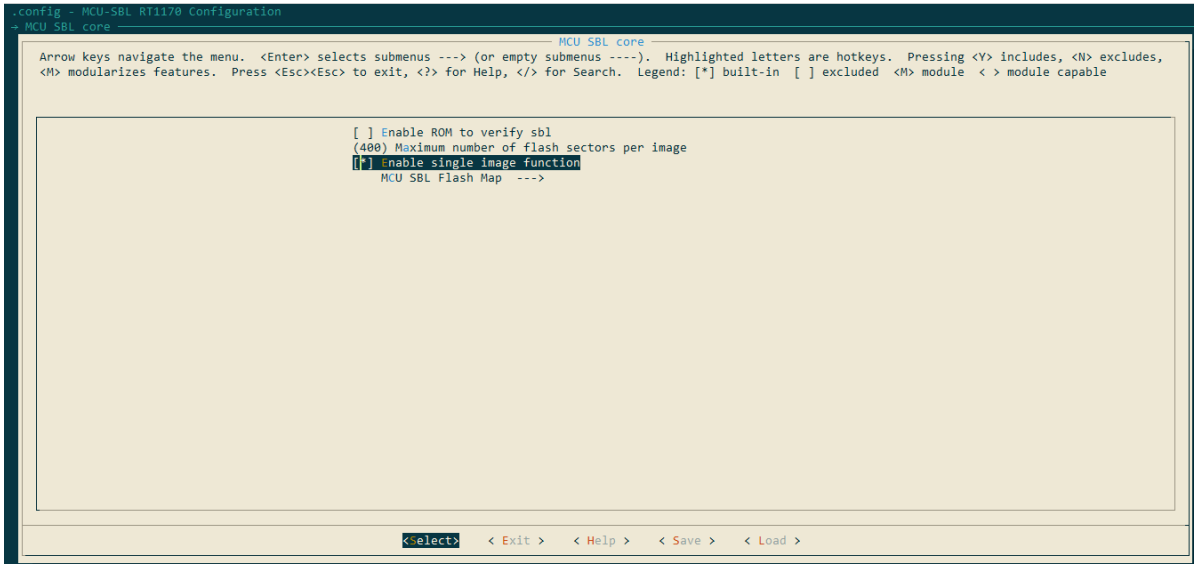


图40. 启用单一映像

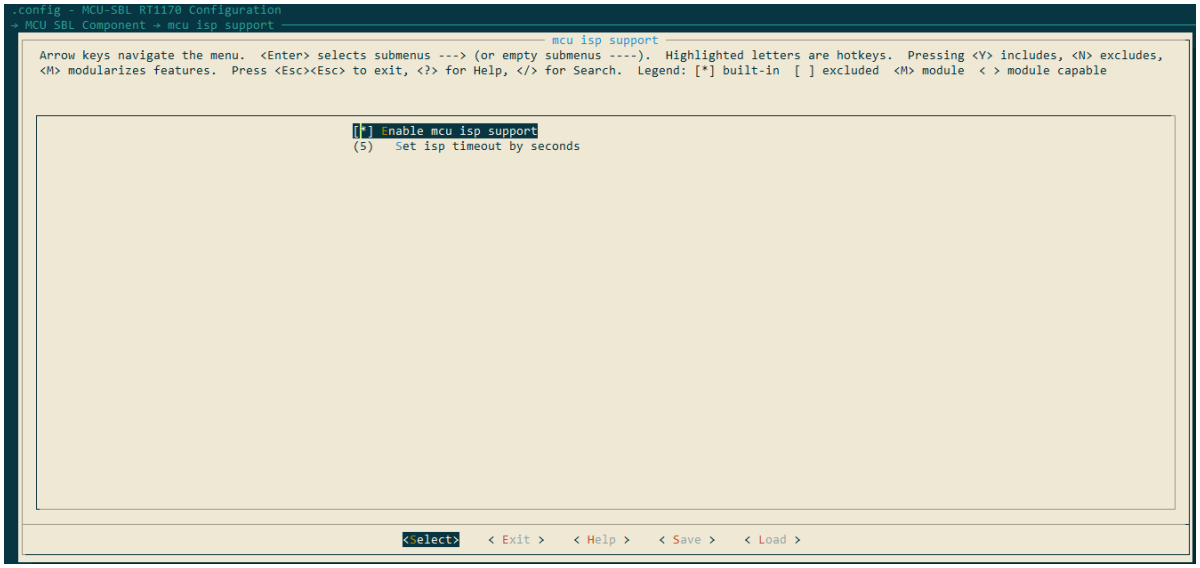


图41. 启用mcu ISP

生成项目、进行编译，并将SBL下载到目标电路板上。然后，将UART或USB端口连接到PC上，并复位电路板。

2. 运行测试

UART连接命令类型：`blhost -p COMx,115200 – command`

USB连接命令类型：`blhost -u <vid>,<pid> -- command`

注意

当使用UART类型运行测试时，必须使用TTL2USB模块将UART端口直接连接到PC，请参考EVK电路板原理图。

复位后，该电路板等待5秒钟。在这5秒钟内，如果PC发送 `connect` 命令（ blhost发送 `get-property` 命令并在终端内获得成功的回复），电路板就会进入MCU ISP模式。如果没有，电路板就会直接进入启动模式。

在PC端，将要下载的映像和 `blhost.exe` 工具放到一个文件夹中，并在该文件夹下打开终端，在电路板通电后5秒内输入命令连接blhost和电路板。`blhost -u -- get-property 1 0`

连接成功后，键入以下命令以进行与闪存有关的操作。

读取：`blhost -u -- read-memory [address] [size]`

擦除：`blhost -u -t [ms] -- flash-eras-region [address] [size]`

写入：`blhost -u -- write-memory 0x60100000 xxx.bin`

注意

如果大小非常大，在使用擦除操作擦除闪存时，在命令中加入 -t [ms] 来增加超时时间。如果超时时间太短，MCU ISP可能会返回擦写失败。

下图显示了整个PC端Blhost更新映像的操作过程。

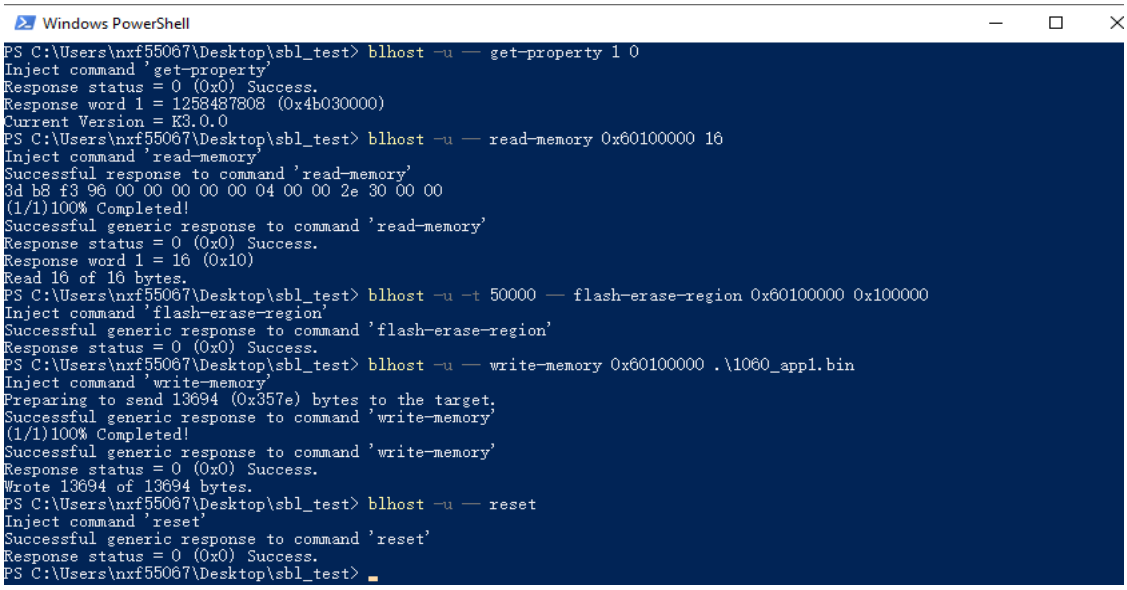


图42. PC上的Blhost操作

要复位电路板，请将新的映像写入闪存slot，然后输入复位命令。5秒后进入启动程序。

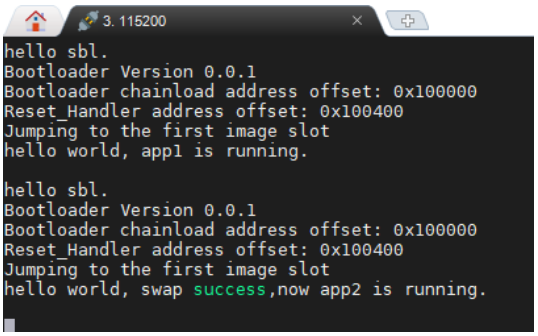


图43. 单一映像的更新日志

7.2.2 SD卡OTA

在这种模式下，使用SD卡来更新映像作为参考。

注意

对于 EVKMIMXRT1170，为了启用SD卡，需要将R136连接到REV C EVK电路板。

1. 准备好SBL

   要禁用单一映像模式和禁用MCU ISP支持，请在Scons的menuconfig界面禁用 `Enable single image function` 选项和 `Enable mcu isp support` 选项。

   编译SBL项目并将其下载到目标电路板。

   注意

   对于 EVKMIMXRT595、EVKMIMXRT685、EVKMIMXRT1170和EVKMIMXRT1010，当下载 sbl.bin 文件时，从闪存的偏移0x400开始。

2. 下载第一个映像

   第一次测试SBL时，请将第一个固件映像下载到电路板上运行测试。使用MCUBootUtility工具将 `app1.bin` 下载到电路板的slot1，slot1的默认位置是从 `flash_offset+0x100000` 到 `flash_offset+0x200000`，整个slot大小为1MB。

   注意

   如果没有启用MCU ISP功能，则设置MCUBootUtility为主模式。

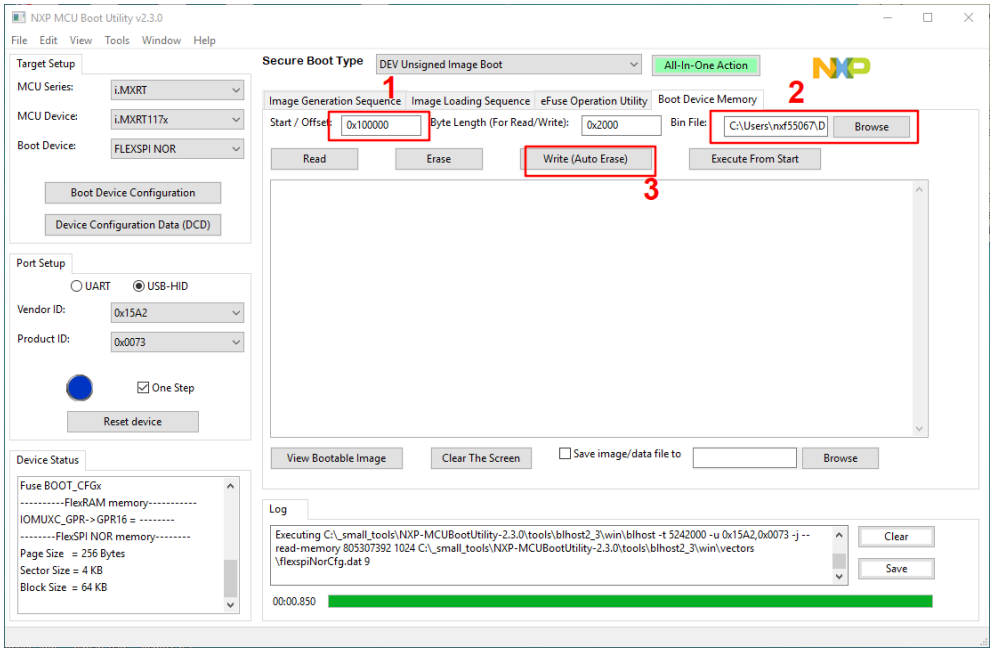


图44. 下载映像1到slot1

3. 运行测试

   - 将SD卡插入PC，并将 `app2.bin` 复制到SD卡上。
   - 将文件名重命名为 `newapp.bin`。
   - 取出SD卡并将其插入目标电路板。
   - 调试控制台随后显示出更新日志。
   - 在SD卡下载完成后，当日志 ‘`sys rst…`’ 出现时，将SD卡从电路板上移走，否则，它将开始新的更新。

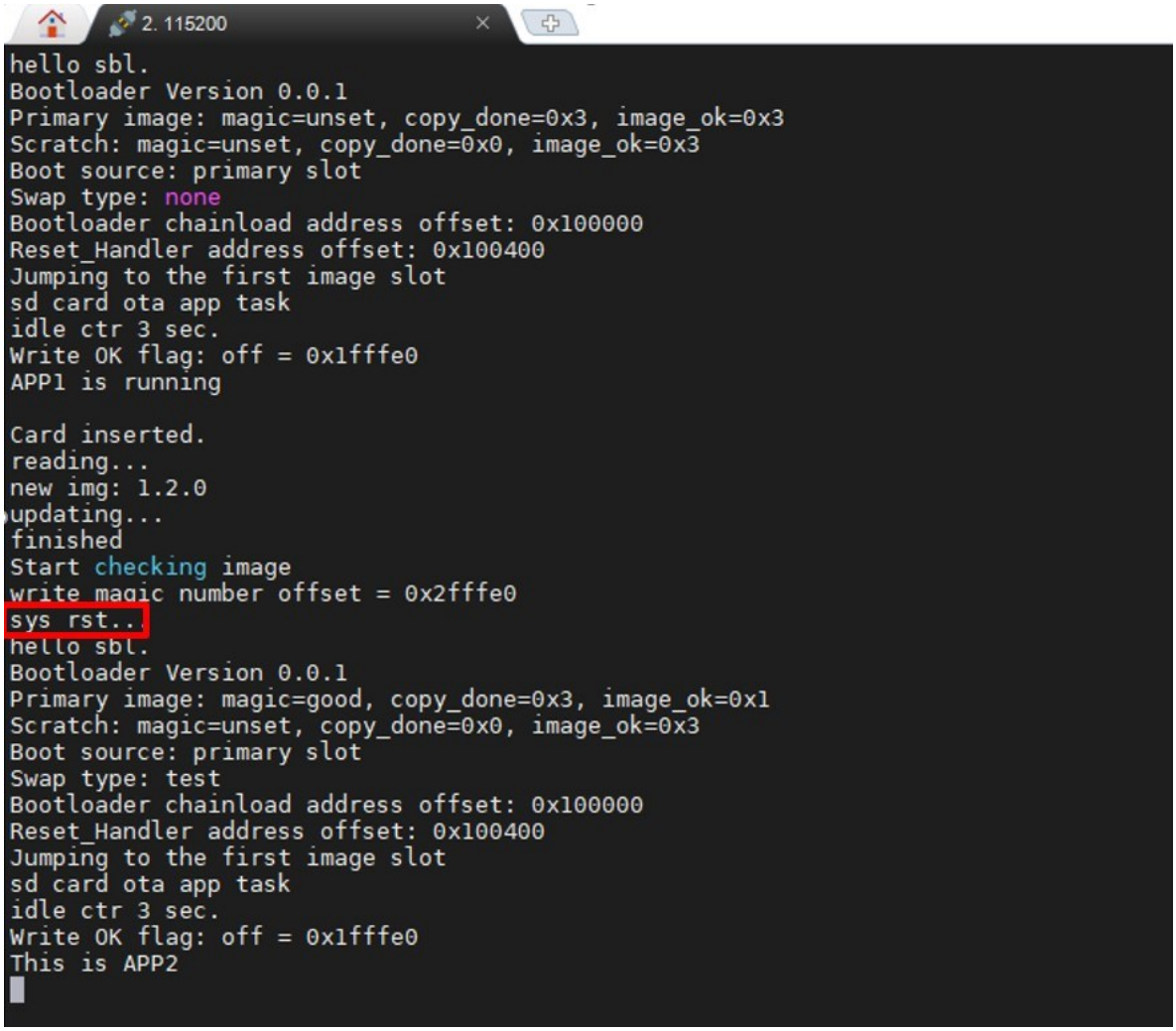


图45. 交换更新日志

在app2日志显示后，按下电路板上的复位按钮。为了确认更新成功，app2日志应该再次显示出来。

7.2.3 U盘OTA

在这种模式下，使用U盘来更新映像作为参考。

1. 准备好SBL

   要禁用单一映像模式和禁用MCU ISP支持，请在Scons的menuconfig界面禁用 `Enable single image function` 选项和 `Enable mcu isp support` 选项。

   编译SBL项目并将其下载到目标电路板。

2. 下载第一个映像和重映射的映像标志

   为了第一次测试SBL，请将第一个映像下载到电路板上运行测试。使用MCUBootUtility工具将 `app1.bin` 下载到电路板的slot1，slot1的默认位置是从 `flash_offset+0x100000` 到 `flash_offset+0x200000`，整个slot大小为1MB。

3. 运行测试

   - 将U盘连接到PC上。

- 将 `app2.bin` 复制到U盘。
- 将bin文件重命名为 `newapp.bin`。
- 将U盘与PC上断开，并将其连接到目标电路板上（使用otg数据线，如果该板有两个USB端口，则将其连接到USB1）。
- 调试控制台显示出更新日志。
- 映像下载完成后，当日志显示 `sys rst…` 时，将U盘从电路板上取出，否则，它将开始新的更新。

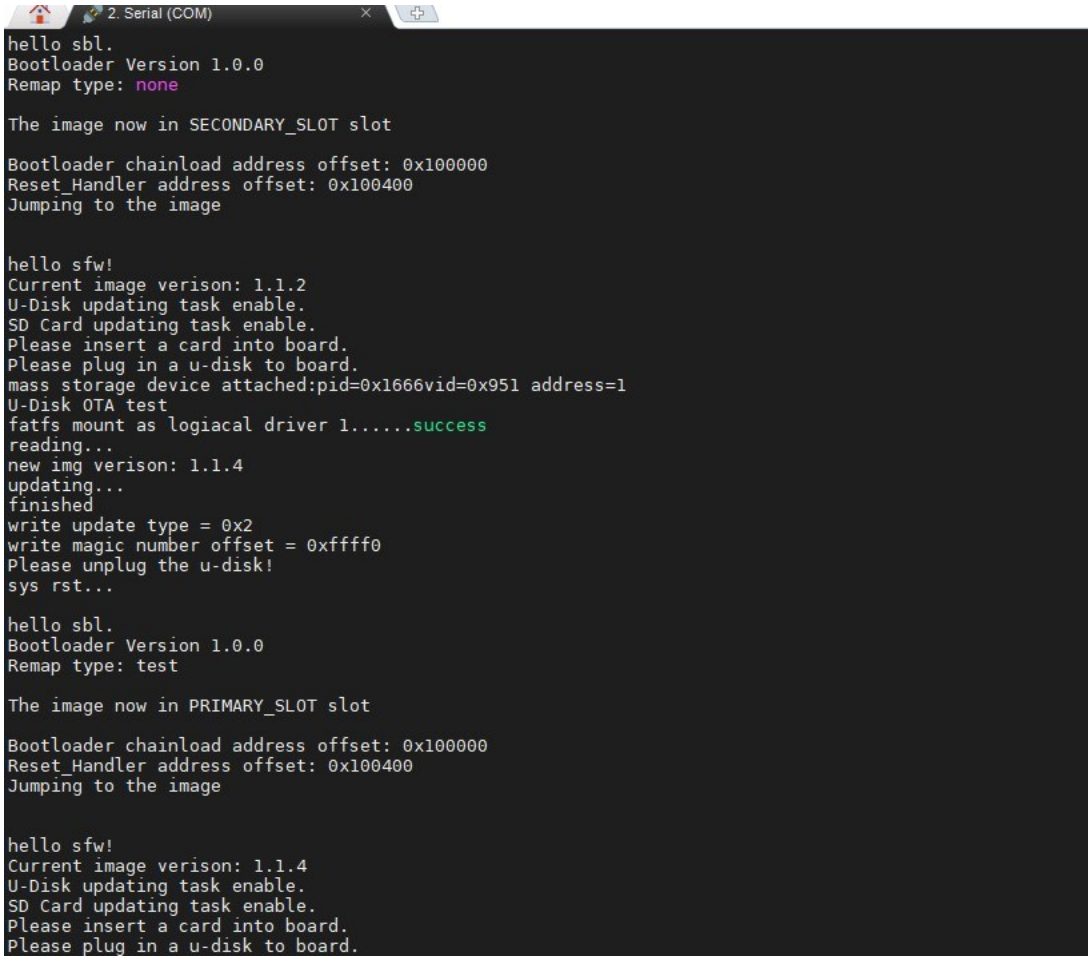


图46. 重映射更新日志

在app2日志显示后，按下电路板上的复位按钮。为了确认更新成功，app2日志应该再次显示出来。

7.3 远程FOTA

本节专门介绍远程FOTA。

7.3.1 AWS OTA

本节以EVKMIMXRT1170 平台为例，讲述了如何利用AWS IoT对板卡进行远程固件更新的操作步骤。目的是演示测试过程。

7.3.1.1 AWS OTA的前提条件

- 创建一个AWS账户

  创建一个AWS账户：https://console.aws.amazon.com/console/home

- 创建一个亚马逊S3桶来存储更新内容

1. 前往 https://console.aws.amazon.com/s3/
2. 选择“Create bucket”。

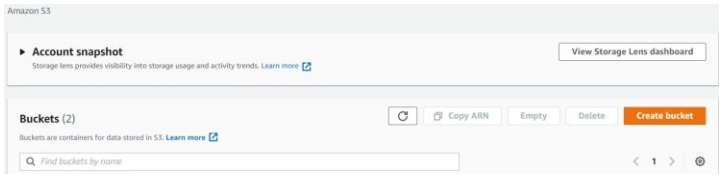


图47. 创建桶

3. 键入一个桶的名称。
4. 选择“Bucket Versioning”的“Enable”。

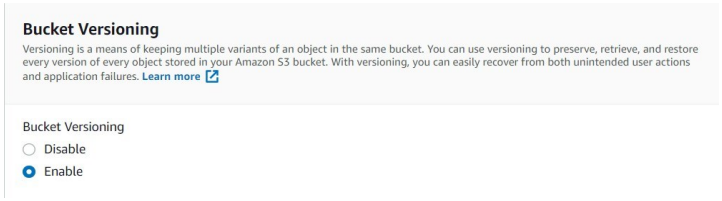


图48. 桶版本控制

5. 其他选项保持默认配置，然后选择创建桶。

- 创建OTA服务角色

1. 登录到 https://console.aws.amazon.com/iam/
2. 在导航窗格中，选择“Roles”。

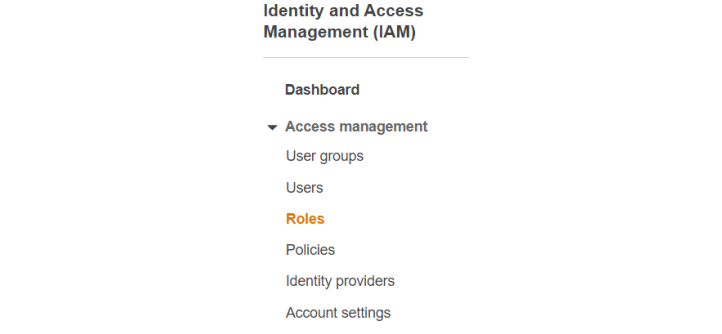


图49. 角色

3. 选择“Create role”。
4. 在“Select type of trusted entity”下面，选择“AWS Service”。

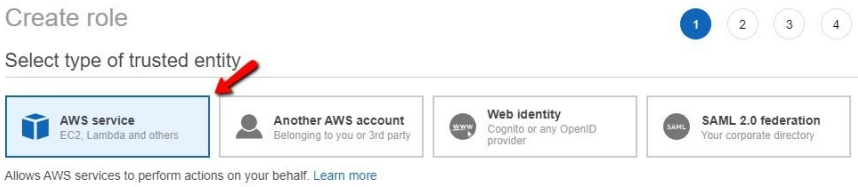


图50. AWS服务

5. 从AWS服务列表中选择IoT。

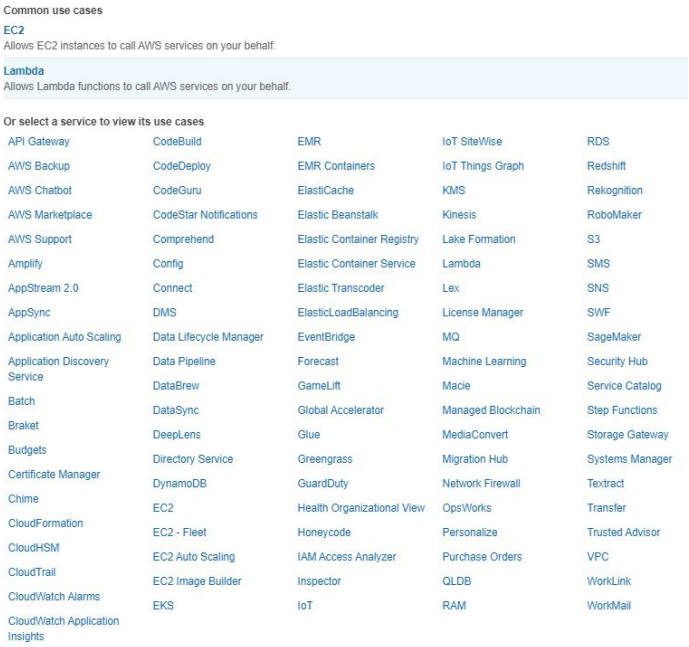


图51. IoT服务

6. 在“Select the use case”下，选择IoT。

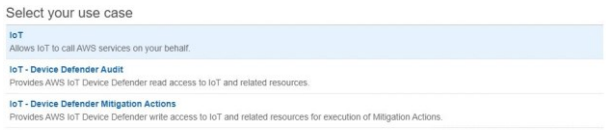


图52. IoT（物联网）用例

7. 选择“Next”：“Permissions”。
8. 选择“Next”：“Tags”。
9. 选择“Next”：“Review”。
10. 输入一个角色名称和描述，然后选择“Create role”。

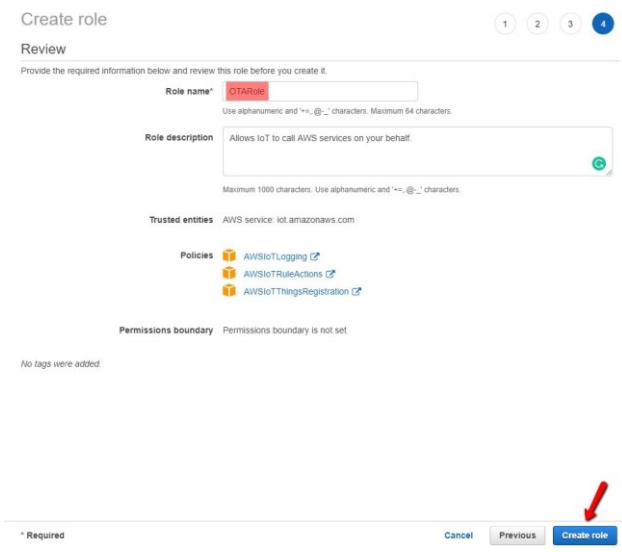


图53. 创建角色

- 为OTA服务角色添加OTA更新权限

1. 在IAM控制台页面的搜索框中，输入角色的名称，然后从列表中选择它。

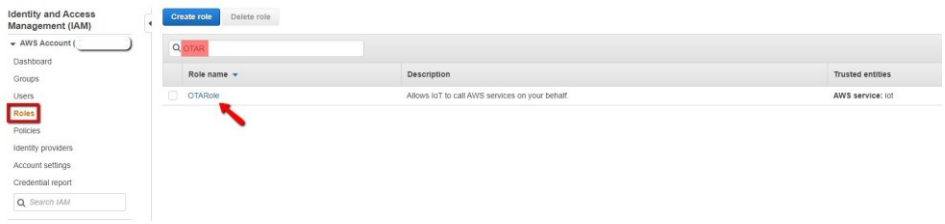


图54. 搜索角色

2. 选择“Attach policies”。

图55. 附加策略

3. 在搜索框中，输入 AmazonFreeRTOSOTAUpdate，从过滤的策略列表中选择 AmazonFreeRTOSOTAUpdate，并选择“Attach policy”，将该策略附加到服务角色。

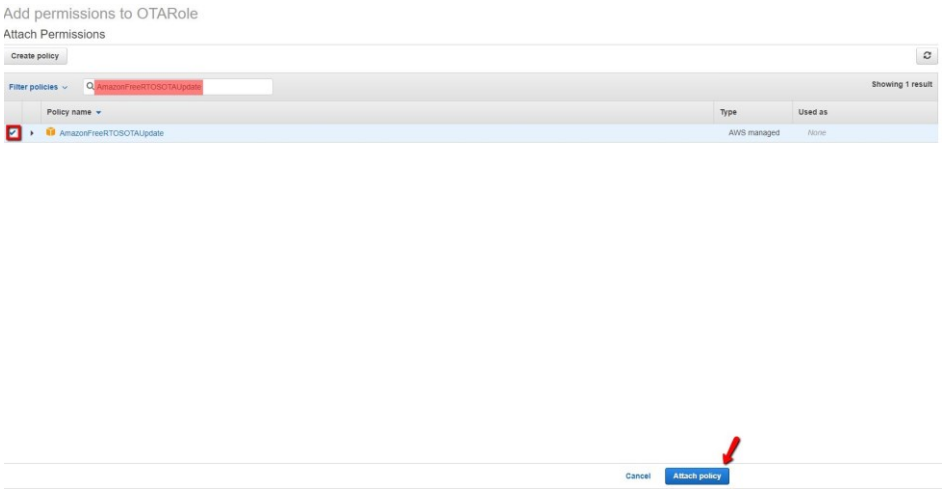


图56. 附加 AmazonFreeRTOSOTAUpdate 策略

- 为OTA服务角色添加所需的IAM权限

1. 选择“Add inline policy”。

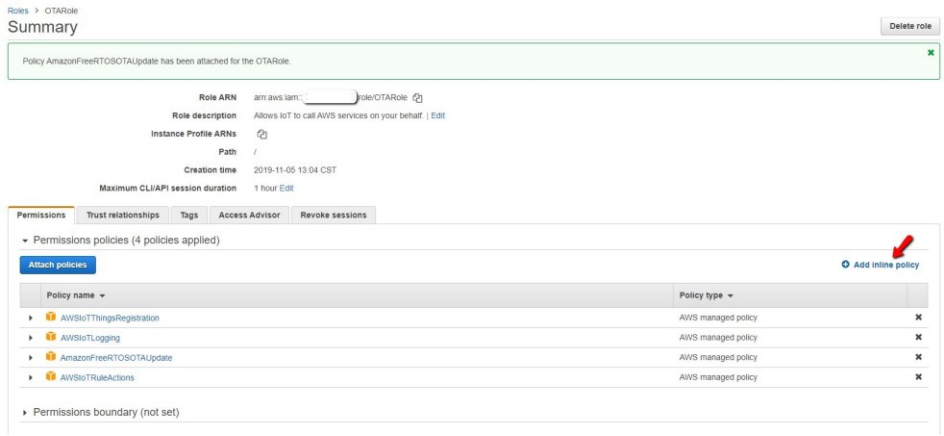


图57. 添加内联政策

2. 选择“JSON”标签。
3. 将以下策略内容复制并粘贴到文本框中。

```
{
       "Version": "2012-10-17",
       "Statement": [
          {
            "Effect": "Allow",
            "Action": [
              "iam:GetRole",
              "iam:PassRole"
            ],
            "Resource": "arn:aws:iam::<your_account_id>:role/<your_role_name>"
          }
        ]
      }
```

确保用AWS账户ID替换 <your_account_id>，用OTA服务角色的名称替换 <your_role_name>。

4. 选择“Review policy”。

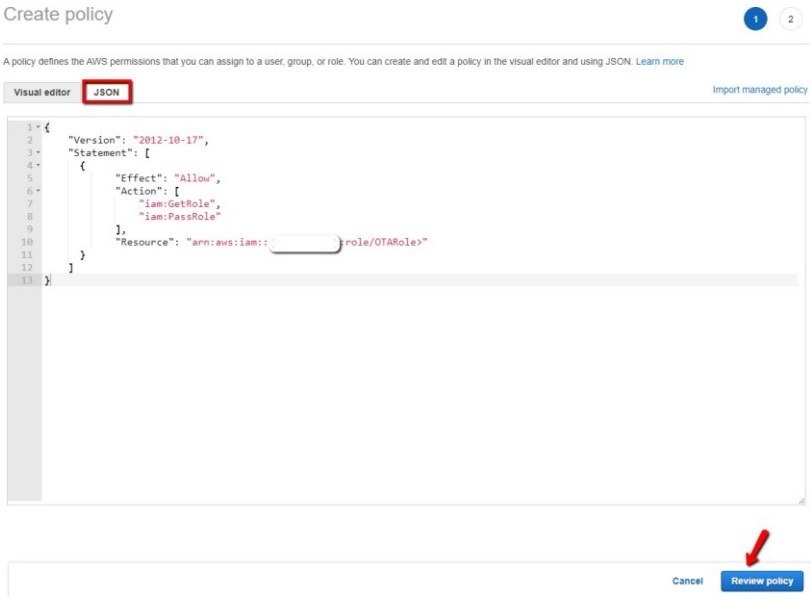


图58. 审核策略

5. 输入策略名称，然后选择“Create policy”。

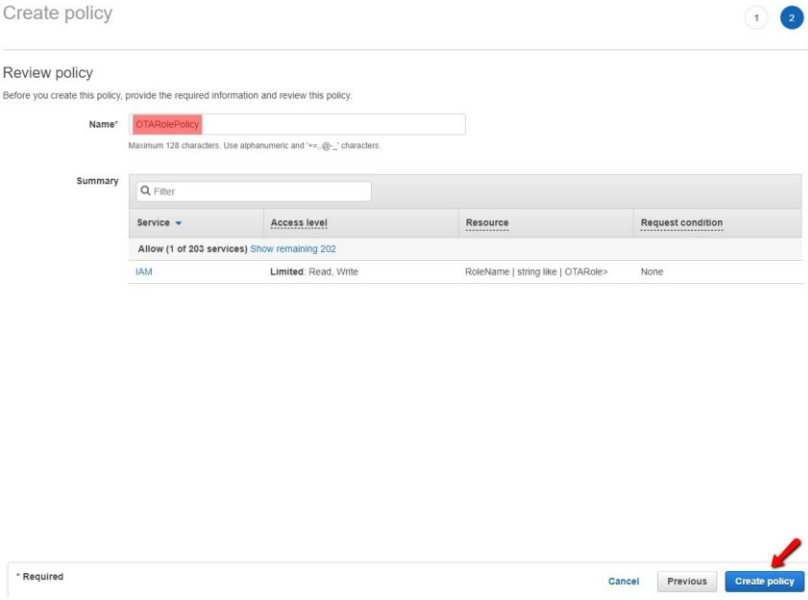


图59. 创建策略

- 为OTA服务角色添加所需的Amazon S3权限

1. 选择“Add inline policy”。

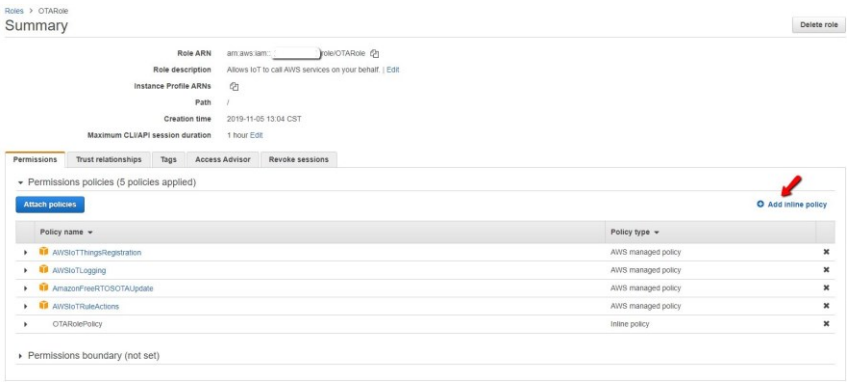


图60. 添加内联策略

2. 选择“JSON”标签。
3. 将以下策略内容复制并粘贴到方框中：

```
{
  "Version": "2012-10-17",
  "Statement": [
     {
        "Effect": "Allow",
        "Action": [
           "s3:ListBucketVersions",
           "s3:GetObjectVersion",
        "s3:GetObject",
        "s3:PutObject"
        ],
        "Resource": [
           "arn:aws:s3:::<example-bucket>/*",
           "arn:aws:s3:::<example-bucket>"
       ]
     }
   ]
}
```

此策略授予OTA服务角色读取Amazon S3对象的权限。请确保将 <example-bucket> 替换为桶的名称。

4. 选择“Review policy”。
5. 输入策略名称，然后选择“Create policy”。

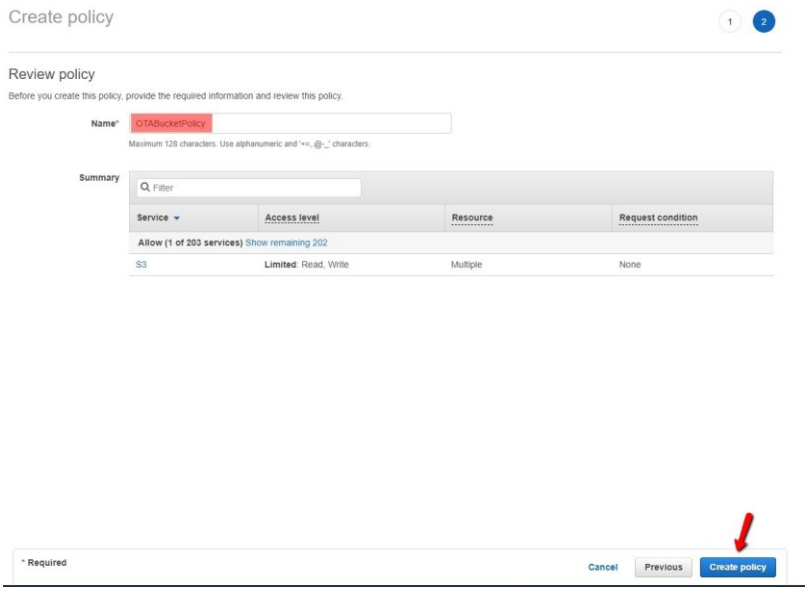


图61. OTA桶政策

- 创建一个OTA用户策略

1. 打开 https://console.aws.amazon.com/iam/https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择“Users”。
3. 选择“Add user”。
4. 输入用户名，选择“Programmatic access type”，然后选择“Next”：“Permissions”。

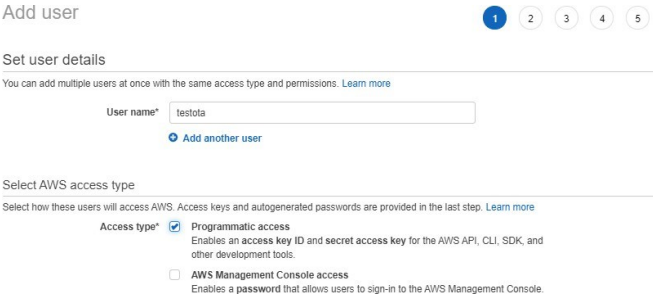


图62. 添加用户

5. 选择“Next”：“Tags”。
6. 选择“Next”：“Review”。
7. 选择“Create user”。
8. 添加用户后，从列表中选择IAM用户。
9. 选择“Add permissions”。

图63. 添加权限

10. 选择“Attach existing policies directly”，然后选择“Create policy”。

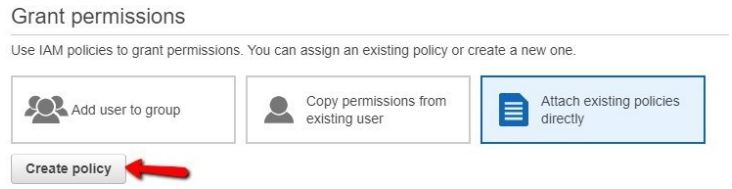


图64. 创建策略

11. 选择“JSON”标签，将以下策略内容复制并粘贴到策略编辑器中。

```
{
  "Version": "2012-10-17",
  "Statement": [
     {
        "Effect": "Allow",
        "Action": [
           "s3:ListBucket",
           "s3:ListAllMyBuckets",
           "s3:CreateBucket",
           "s3:PutBucketVersioning",
           "s3:GetBucketLocation",
           "s3:GetObjectVersion",
           "acm:ImportCertificate",
           "acm:ListCertificates",
           "iot:*",
           "iam:ListRoles",
           "freertos:ListHardwarePlatforms",
           "freertos:DescribeHardwarePlatform"
        ],
        "Resource": "*"
     },
     {
        "Effect": "Allow",
        "Action": [
           "s3:GetObject",
           "s3:PutObject"
        ],
        "Resource": "arn:aws:s3:::<example-bucket>/*"
     },
     {
        "Effect": "Allow",
        "Action": "iam:PassRole",
        "Resource": "arn:aws:iam::<your-account-id>:role/<role-name>"
     }
  ]
}
```

将 <example-bucket> 替换为存储OTA更新固件映像的Amazon S3桶的名称。

用AWS账户ID替换 <your-account-id>。AWS账户ID可以在控制台的右上方找到。当输入账户ID时，删除任何破折号（-）。

用创建的IAM服务角色的名称替换 <role-name>。

12. 选择“Review policy”。
13. 输入新的OTA用户策略名称，然后选择“Create policy”。

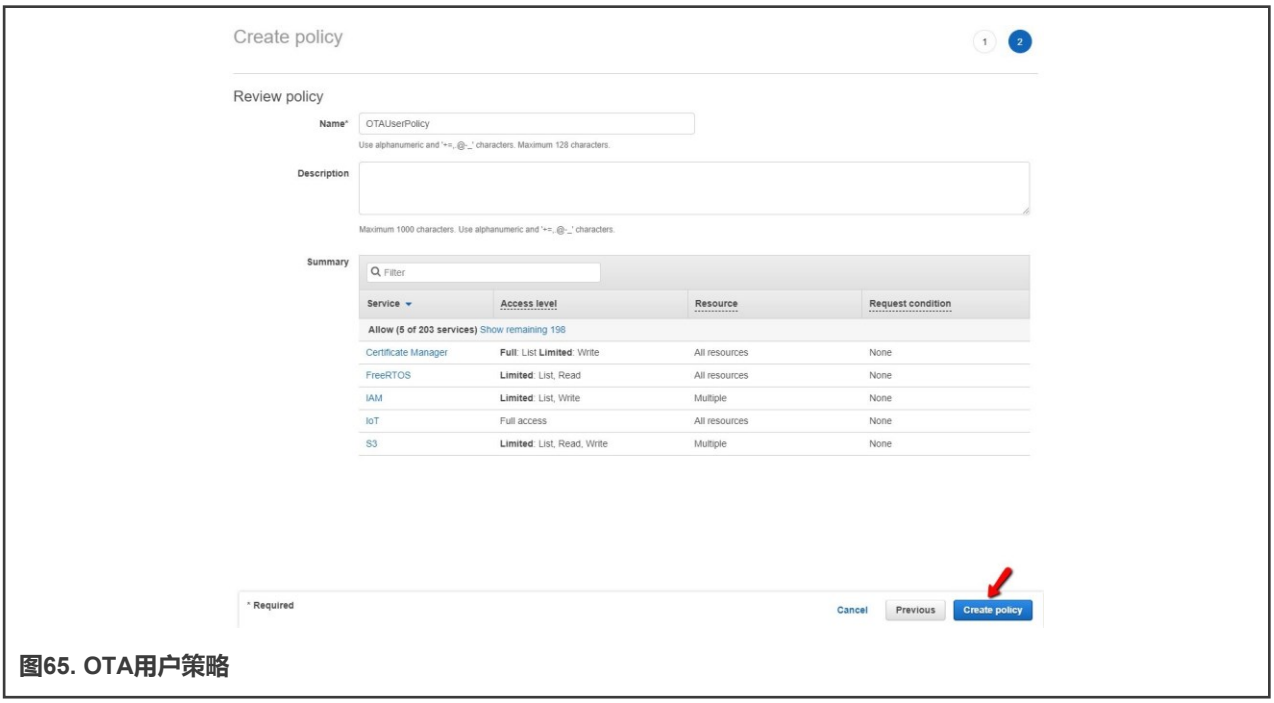


图65. OTA用户策略

- Windows的先决条件

1. OpenSSL
   a. 安装 OpenSSL：https://slproweb.com/products/Win32OpenSSL.html
   b. 修改系统环境变量路径，添加 OpenSSL bin 目录

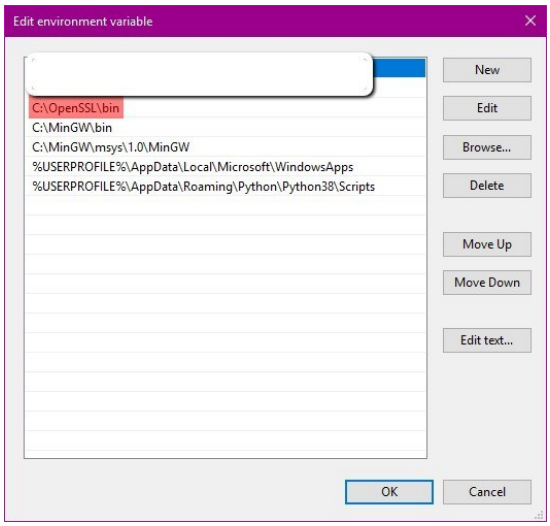


图66. OpenSSL配置

确保OpenSSL被分配给命令提示符或终端环境中的OpenSSL可执行文件。

2. 安装AWS CLI

   a. 按照AWS CLI第1版捆绑包安装程序的说明进行安装 https://docs.aws.amazon.com/cli/latest/userguide/install-cliv1.html

   b. 进入到IAM控制台 https://console.aws.amazon.com/iam/

   c. 在导航窗格中，选择“Users”。

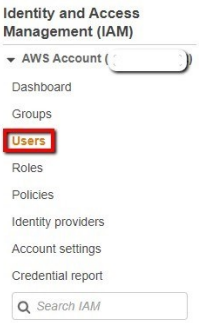


图67. 用户

   d. 选择IAM用户账户。

   e. 选择安全证书。

   f. 在访问密钥部分，选择“创建访问密钥”。

   g. 要查看新的访问密钥对，选择“显示”。关闭这个对话框后，就不能再访问秘密访问密钥了。安全证书看起来像这样：

   访问密钥ID：`AKIAIOSFODNN7EXAMPLE`

   秘密访问密钥：`wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`

   h. 要下载密钥对，点击`Download .csv`文件。将密钥储存在一个安全的地方。关闭此对话框后，秘密访问密钥不能再被访问。对密钥要保密以保护AWS账户，切勿通过电子邮件发送密钥。不要在用户的组织之外分享它们，即使查询看起来是来自AWS或Amazon.com。任何合法代表亚马逊的人都不会向别人索要密钥。

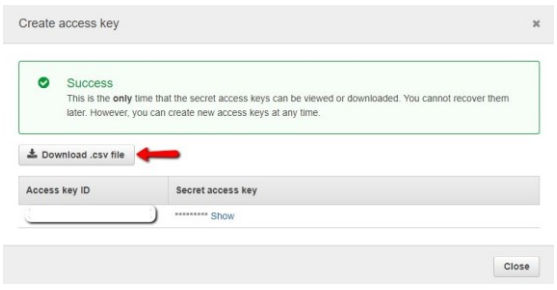


图68. 下载 .csv 文件

   i. 下载完 `.csv` 文件后，点击“关闭”。一旦生成访问密钥，该密钥对默认是激活的，可以立即使用该密钥对。

   j. 对于一般使用，AWS配置命令是设置 AWS CLI 安装的最快方式。

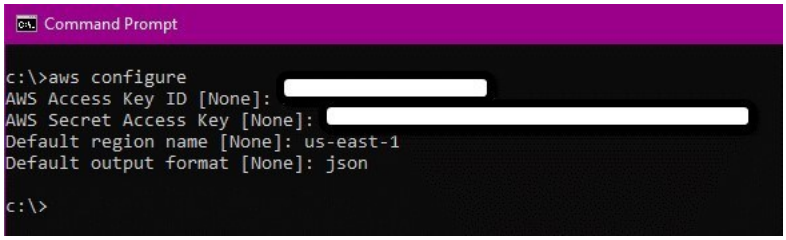


图69. AWS CLI 安装

3. 创建代码签名证书

   a. 在工作目录中，使用以下文本创建一个名为 `cert_config.txt` 的文件。将 *test_signer@amazon.com* 改为用户的电子邮件地址：

```
[ req ]
prompt = no
distinguished_name = my_dn
[ my_dn ]
commonName = test_signer@amazon.com
[ my_exts ]
keyUsage = digitalSignature
extendedKeyUsage = codeSigning
```

   b. 使用 openSSL 命令行，创建一个ECDSA代码签名的私钥：

```
openssl genpkey -algorithm EC -pkeyopt ec_paramgen_curve:P-256 -pkeyopt
ec_param_enc:named_curve -outform PEM -out ecdsasigner.key
```

   c. 创建ECDSA代码签名证书：

```
openssl req -new -x509 -config cert_config.txt -extensions my_exts
 -nodes -days 365 -key ecdsasigner.key -out ecdsasigner.crt
```

   将代码签名证书、私钥和证书链导入“AWS证书管理器”：

```
aws acm import-certificate --certificate file://ecdsasigner.crt
--private-key file://ecdsasigner.key
```

注意

该命令显示证书的ARN。将其保存在本地，以便在创建 OTA 更新作业时使用。

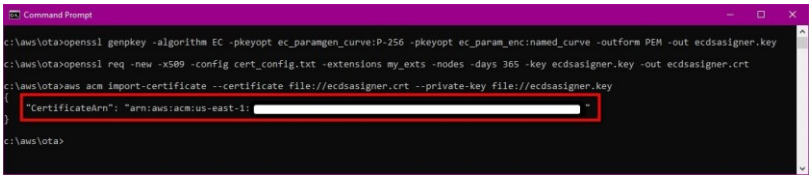


图70. 创建代码签名证书

- 授权访问AWS IoT（AWS物联网）的代码签名

1. 登录到 https://console.aws.amazon.com/iam/
2. 在导航窗格中，选择“Policies”。

图71. 政策

3. 选择“Create Policy”。
4. 在“JSON”标签上，将以下JSON内容复制并粘贴到策略编辑器中。该策略允许IAM用户访问所有代码签名操作。

```
{
  "Version": "2012-10-17",
  "Statement": [
     {
        "Effect": "Allow",
        "Action": [
           "signer:*"
        ],
        "Resource": "*"
     }
   ]
}
```

5. 选择“Review policy”。
6. 输入策略名称和描述，然后选择“Create policy”。

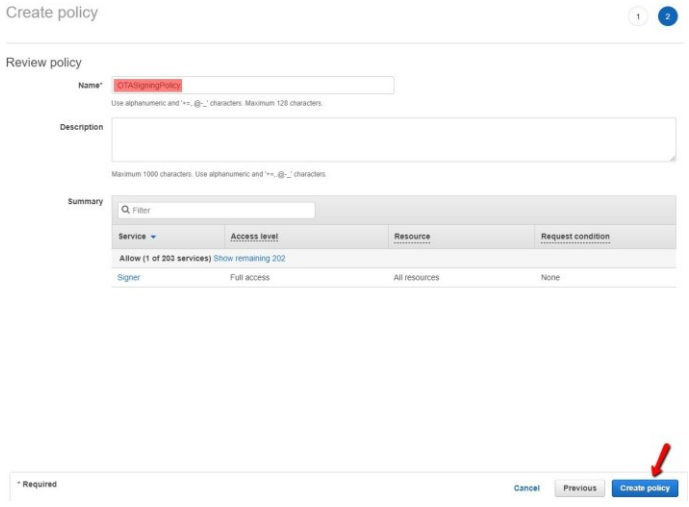


图72. OTA签名政策

7. 在导航窗格中，选择“Users”。

8. 选择IAM用户账户。
9. 在“Permissions”标签上，选择“Add permissions”。
10. 选择“Attach existing policies directly”，并选择之前创建的“代码签名策略”旁边的复选框。

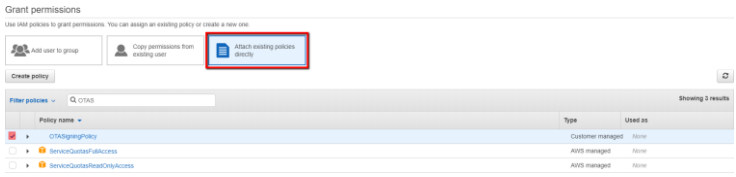


图73. 附加策略

11. 选择“Next”：“Review”。
12. 选择“Add permissions”。

- 创建一个“AWS IoT Thing”

1. 打开 AWS IoT（AWS物联网）控制台网站 https://console.aws.amazon.com/iot/

注意

当打开网站时，首先检查该网站的地区是否是S3桶所在的地区。

2. 在导航窗格中，选择“Manage”-> “Things”。
3. 选择“Create”。
4. 在“Creating AWS IoT things”页面，选择“Create a single thing”。

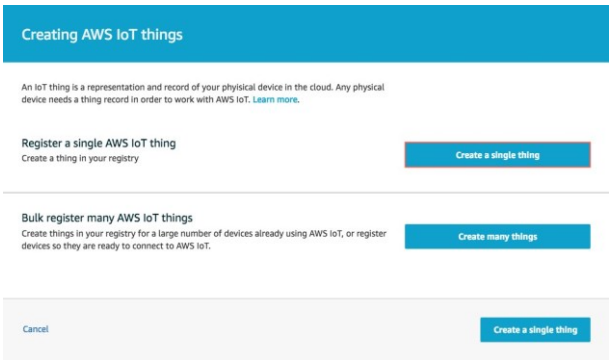


图74. 创建一个单一的事物

5. 在“Create a thing page”页面，在“Name”字段中，为该事物输入名称，如MyThing。然后选择“Next”。

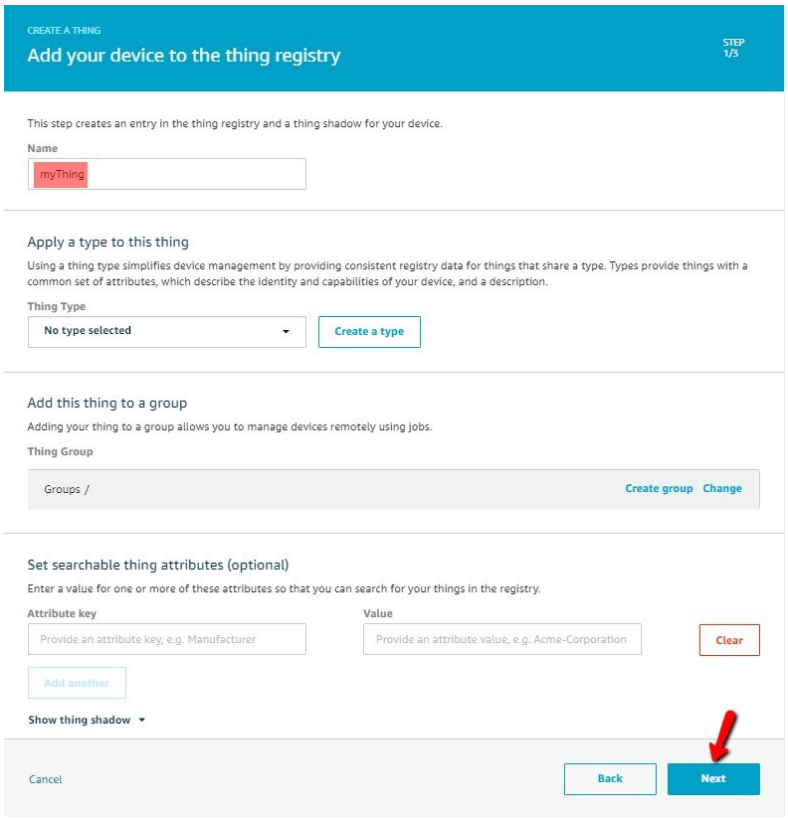


图75. 事物名称

6. 在“Add a certificate for the thing”页面，选择“Create certificate”。它将生成一个X.509证书和密钥对。

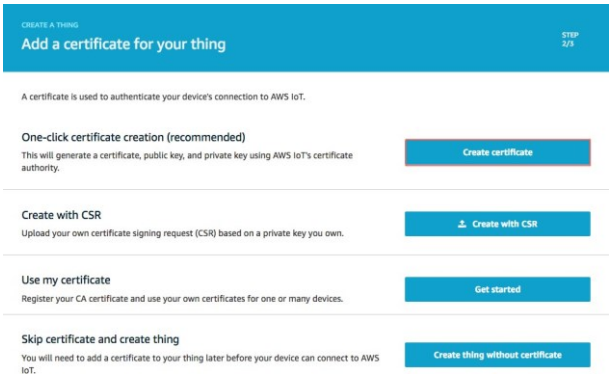


图76. 创建证书

7. 在“Certificate created!”页面，下载公钥，私钥、证书和根证书颁发机构（CA）。
8. 选择“Download”证书。
9. 选择“Download”以下载私钥。
10. 选择“Download”后会显示一个新的网页。选择RSA 2048位密钥：亚马逊根CA1。它会打开另一个网页，上面有根CA证书的文本。复制这个文本并将其粘贴到一个名为 `Amazon_Root_CA_1.pem` 的文件。

大多数网络浏览器将下载的文件保存到下载目录中。运行样本应用程序时，将这些文件复制到不同的目录。选择“Activate”以激活X.509证书，然后选择“Attach a policy”。

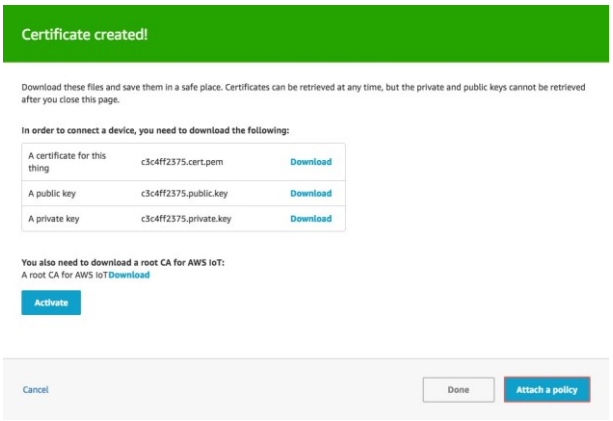


图77. 文件下载

11. 在 `Add a policy for your thing`页面，选择“Register Thing”。在注册该事物后，创建并附加一个新的策略到证书上。

- 创建AWS IoT（AWS物联网）策略

1. 打开AWS IoT（AWS物联网）控制台网站： https://console.aws.amazon.com/iot/。
2. 在左边的导航窗格中，选择“Secure”，选择“Policies”，然后选择“Create”。
3. 在 `Create a policy` 页面，在“Name”字段中，输入策略名称（例如，`MyIotPolicy`）。在“Action”字段中，输入 `iot:*` 。在“Resource ARN”字段中，输入*。选择“Allow”复选框。它允许所有客户端连接到AWS物联网。输入策略信息后，选择“Create”。

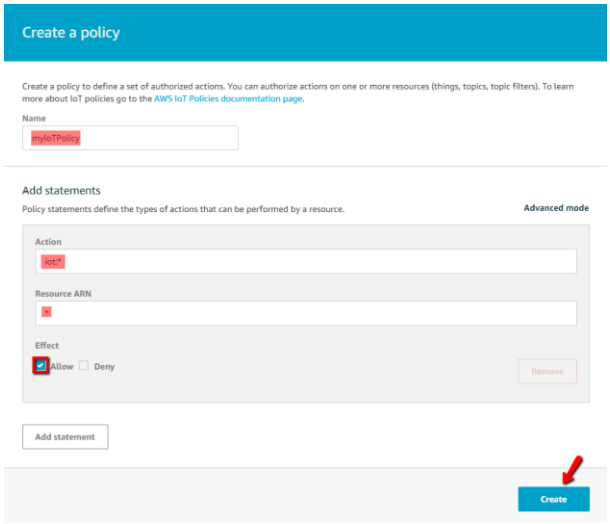


图78. 创建物联网策略

- 将一个AWS IoT策略附加到 一个“Device Certificate”上

1. 打开AWS IoT（AWS物联网）控制台网站：https://console.aws.amazon.com/iot/。
2. 在左边的导航窗格中，选择“Secure”，然后选择“Certificates”。
3. 在创建的证书框中，选择.....打开下拉菜单，然后选择“Attach policy”。

4. 在“Attach policies to certificate(s)”上，选择上一步已创建的政策旁边的复选框，然后选择“Attach”。

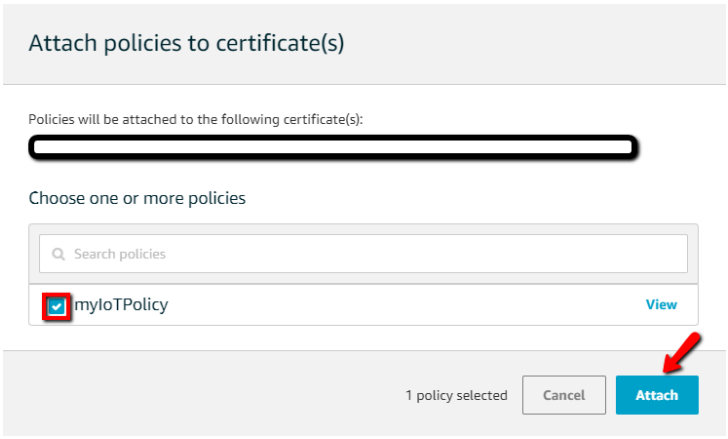


图79. 附加IoT（物联网）政策

- 将证书附加到事物上

1. 打开AWS IoT（AWS物联网）控制台网站：https://console.aws.amazon.com/iot/。
2. 在左边的导航窗格中，选择“Secure”，然后选择“Certificates”。
3. 在创建的证书框中，选择.....打开一个下拉菜单，然后选择“Attach thing”。
4. 在“Attach things to certificate(s)”中，选择已注册的事物旁边的复选框，然后选择“Attach”。

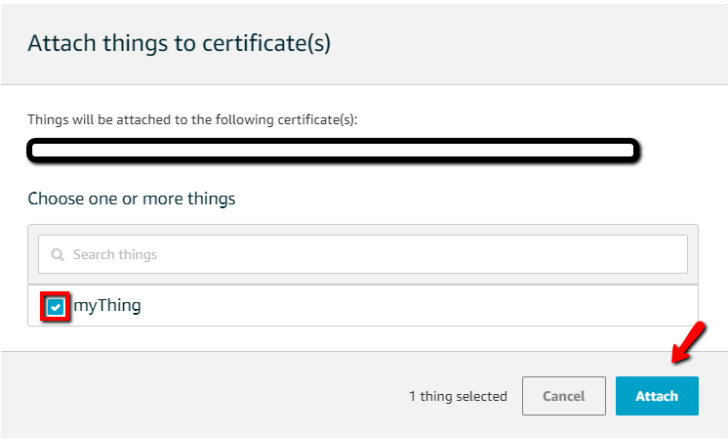


图80. 附加事物

5. 为了验证该事物是否被附上，选择证书的方框。

图81. 选择要验证的证书

6. 在证书的详细信息页面，在左边的导航窗格中，选择“Things”。

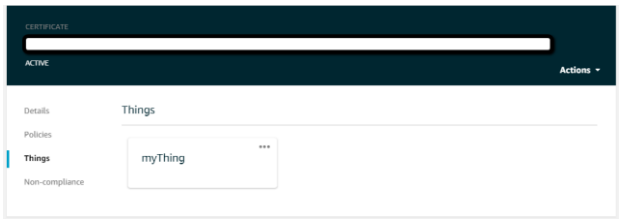


图82. 验证附加的事物

7. 要验证政策是否被附加，在证书的“Details”页面，在左边的导航窗格中，选择“Policies”。

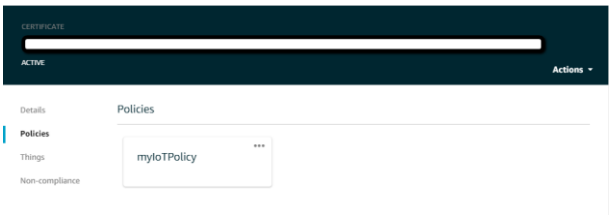


图83. 验证附加的策略

7.3.1.2 准备SBL

在SBL项目中，进入 `sbl/target/evkmimxrt1170` 目录。

禁用Scons的menuconfig界面中的 `Enable single image function` 选项和 `Enable mcu isp support` 选项，以禁用单一映像模式和禁用MCU ISP支持。

编译SBL项目并将其下载到目标板卡。

注意

1. 如果使用新的签名密钥，请修改 `sign-rsa2048-pub.c`
2. 通过DAPLink的拖放方式来下载SBL映像文件可能会擦除整个flash。

7.3.1.3 准备SFW

按以下步骤对SFW工程进行配置：

1. 生成 `aws_clientcredential_keys.h` 文件。
   a. 进入 `sfw/firmware/aws_ota/tool` 目录。

b. 使用网络浏览器，打开 `CertificateConfigurator.html`。

c. 导入从 7.3.1.1 节中“Create an AWS IoT Thing”时下载的“Certificate and Key”文件。点击Generate and save `aws_clientcredential_keys.h` 按钮。

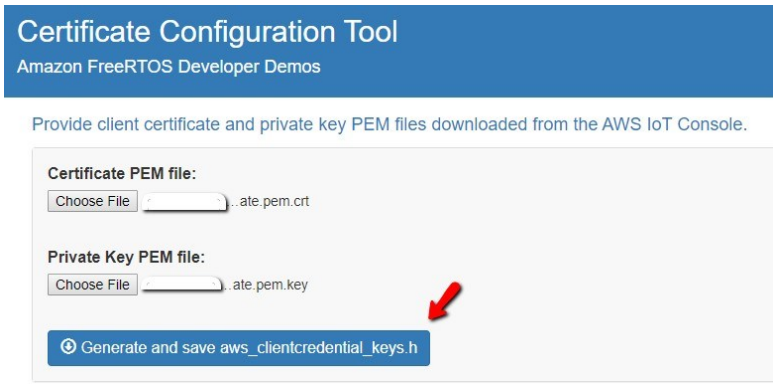


图84. 生成 `aws_clientcredential_keys.h`

d. 将 `sfw/firmware/aws_ota/demos/include/aws_clientcredential_keys.h` 替换为c) 步骤中生成的文件。

2. 修改 `aws_ota_codesigner_certificate.h` 文件。

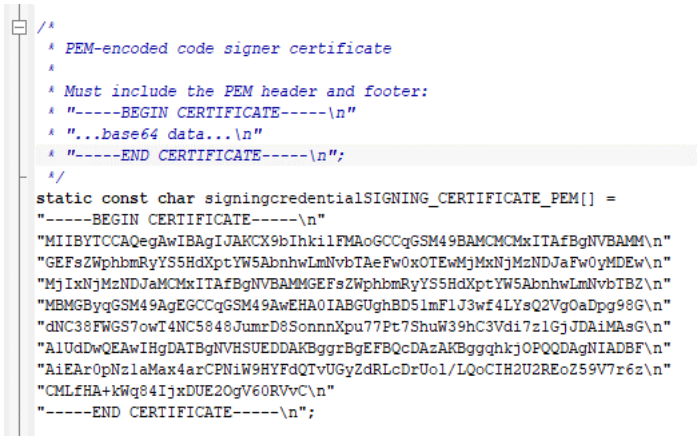
a. 使用文本编辑器打开在第 7.3.1.1 节中“Windows Pre-Requisites”部分所生成的 `ecdsaigner.crt` 文件。

b. 打开文件：`sfw/firmware/aws_ota/demos/include/aws_ota_codesigner_certificate.h`。

c. 复制 `ecdsaigner.crt` 中的所有内容，并粘贴到 `aws_ota_codesigner_certificate.h` 文件中的 `signingcredentialSIGNING_CERTIFICATE_PEM` 数组中。

注意

如下图所示，请确保在每行行首添加‘ “ ’，并在每行行末添加‘ \n” ’。



```
/*
 * 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"
"MBMGByqGSM49AgEGCCqGSM49AwEHA0IABGUghBD5lmFlJ3wf4LYsQ2VgOaDpg98G\n"
"dNC38FWGS7owT4NC5848JumrD8SonnnXpu77Pt7ShuW39hC3Vdi7z1GjJDAiMAsG\n"
"A1UdDwQEAwIHgDATBgNVHSUEDDAKBggrBgEFBQcDAzAKBggqhkjOPQQDAgNIADBF\n"
"AiEAr0pNzlaMax4arCPNiW9HYFdQTvUGyZdRLcDrUol/LQoCIH2U2REoZ59V7r6z\n"
"CMLfHA+kWq84IjxDUE2OgV60RVvC\n"
"-----END CERTIFICATE-----\n";
```

图85. 证书格式

3. 进入 `sfw/target/evkmimxrt1170` 目录。

4. 双击 `env.bat`文件。

5. 输入 `scons --menuconfig` 命令来配置 `evkmimxrt1170` 工程。

6. 选择“MCU SFW core”。

7. 禁用 `Enable sfw standalone xip` 选项，启用OTA，并选择“AWS OTA cloud（云）”。

```
[ ] 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86. SFW配置

8. 选择“AWS Config”（配置）进入下图所示的配置界面。

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

图87. AWS配置

9. 输入MQTT DNS名称。
   a. 打开AWS IoT（AWS物联网）控制台网站：https://console.aws.amazon.com/iot/
   b. 在导航窗格中，选择“Manage – Things”，并选择先前创建的事物。
   c. 在导航窗格中，选择“Interact”。

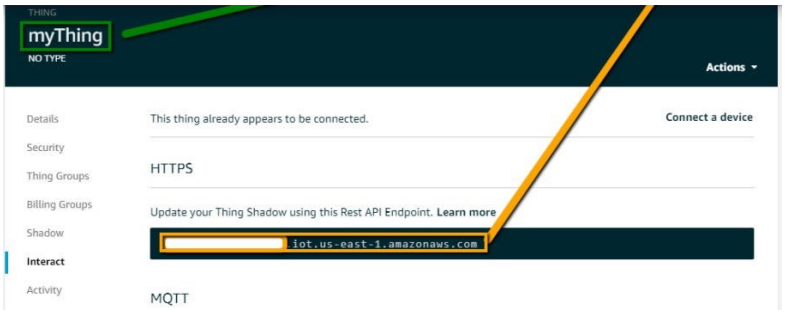


图88. 事物的交互

   d. 选择“Set MQTT broker DNS name”，复制“Rest API Endpoint”并粘贴。

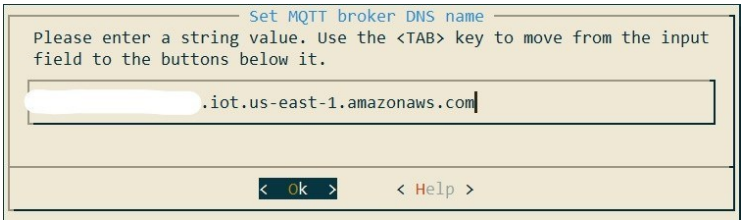


图89. 设置MQTT代理DNS名称

   e. 按OK键

10. 输入物联网事物的名称。
    a. 选择“Set IoT Thing name”，复制物联网事物的名称，然后粘贴。

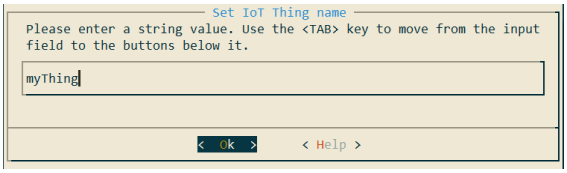


图90. 设置事物名称

b. 按OK键

11. 选择 MCU SFW Component -> secure
12. 启用 `mbedtls`，并将 `mbedtls` 配置文件修改为 `aws_mbedtls_config.h`，然后按“OK”。

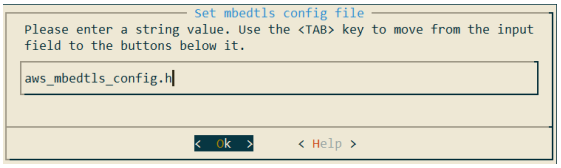


图91. 修改 `mbedtls` 配置文件

13. 退出并保存配置。

图92. 保存配置

7.3.1.4 准备映像文件

1. 进入 `sfw/target/evkmimxrt1170` 目录，双击 `env.bat` 文件。
2. 输入命令，生成iar工程。

注意

要生成keil或gcc工程，请参考第2节。

3. 进入 `sfw/target/evkmimxrt1170/iar` 目录，打开 `sfw.eww` 工程。
4. 在IAR工程中，打开Options，选择生成额外输出，并选择原始二进制格式。

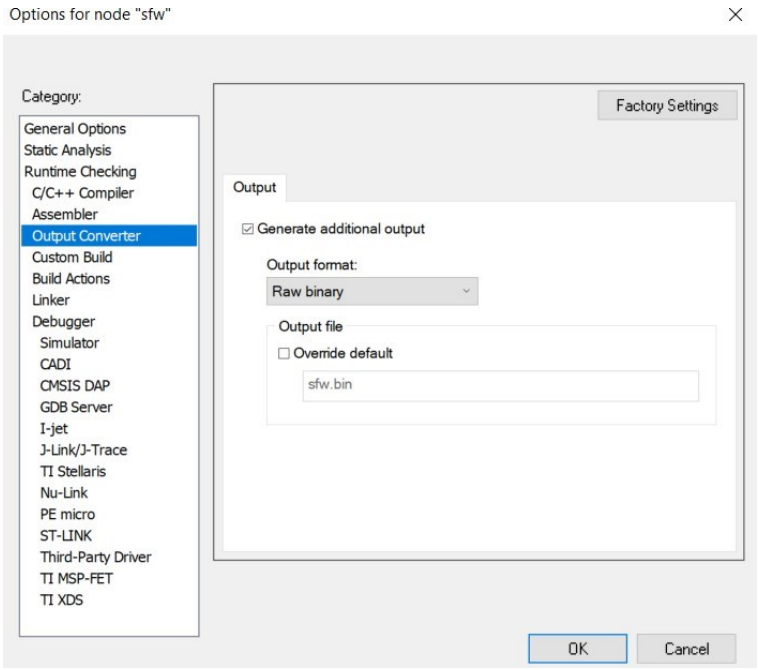


图93. 修改以生成 .bin 文件

5. 在 sfw/firmware/aws_ota/main_enet.c 文件中查看应用程序版本。

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

图94. 应用程序版本

6. 点击“make”（编译）按钮，开始编译应用程序。
7. 如果编译成功，在 sfw/target/evkmimxrt1170/iar/build/iar/Exe 文件夹中会生成sfw.bin文件。根据应用程序的版本更改其名称为sfw_092.bin。将 sfw_092.bin 移到 sbl/component/secure/mcuboot/scripts 文件夹下。
8. 将 APP_VERSION_BUILD 改为3以构编译出一个更新的映像文件。将新的bin文件重命名为 sfw_093.bi，并将其移至 sbl/component/secure/mcuboot/scripts 文件夹下。

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

图95. 新版本

9. 使用下面的命令，用RSA sfw_092.bin 和 sfw_093.bin 映像文件进行签名，生成签名后的 sfw092.bin 和 sfw093.bin 文件。

```
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
```

7.3.1.5 将新映像文件上传到到S3桶

1. 使用AWS控制台，打开S3服务 https://console.aws.amazon.com/s3
2. 选择之前创建的桶。
3. 点击上传。
4. 拖放 `sfw093.bin`。
5. 点击上传。

7.3.1.6 创建OTA作业

1. 打开AWS IoT（AWS物联网）控制台网站：https://console.aws.amazon.com/iot/
2. 在导航窗格中，选择“Manage – Jobs”
3. 选择“Create job”
4. 选择“Create FreeRTOS OTA update job”，然后选择“Next”。
5. 在步骤1中，输入作业名称，然后选择“Next”。

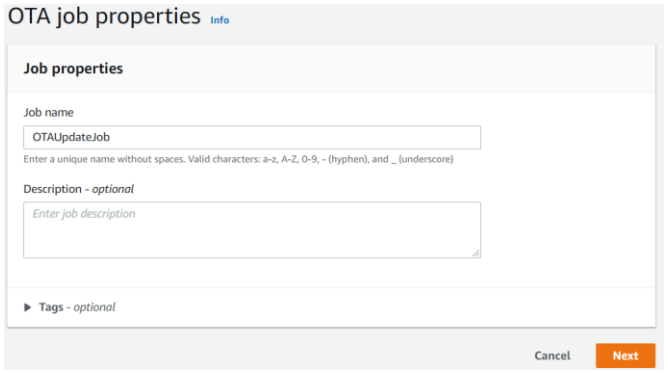


图96. 输入作业名称

6. 在第2步中，选择之前章节中创建的 Thing，选择“MQTT”，以及“Sign a new file for me”。

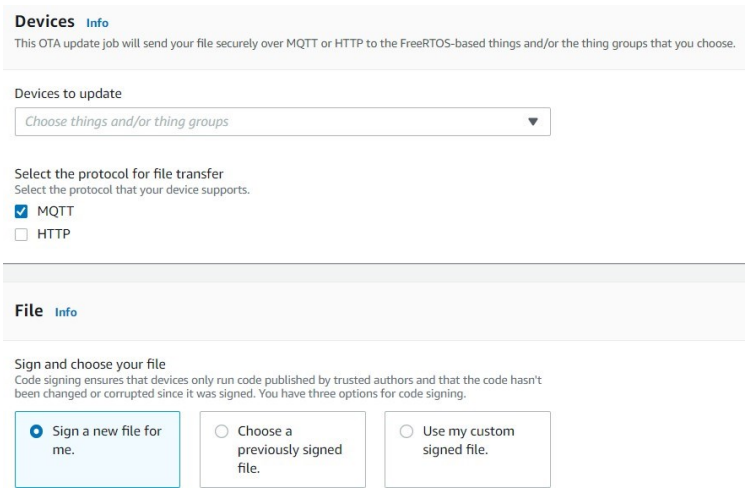


图97. 选择事物、协议和文件

7. 在“Code signing profile”下，选择“Create new profile”。

8. 输入代码签名配置的名称。

   a. 在“Device hardware platform”下，选择“Windows Simulator”。

图98. 设备硬件平台

   b. 在“Code signing certificate”下，选择“Import new code signing certificate”，选择用AWS CLI创建的证书文件，然后选择“Import”。

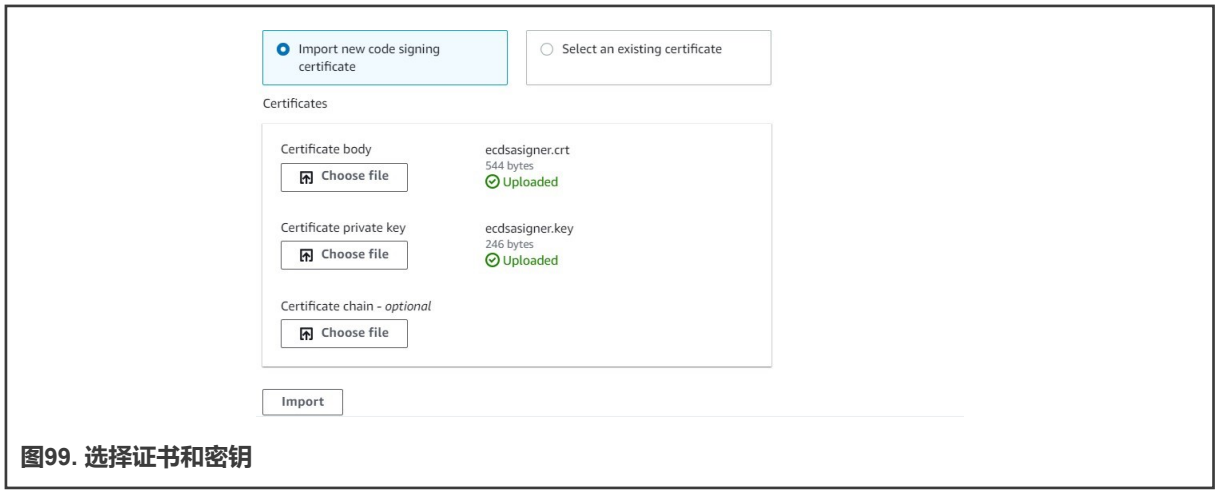


图99. 选择证书和密钥

   c. 在“Path name of code signing certificate on device”下，输入默认的路径`path /certificates/authcert.pem`，然后点击“Create”。

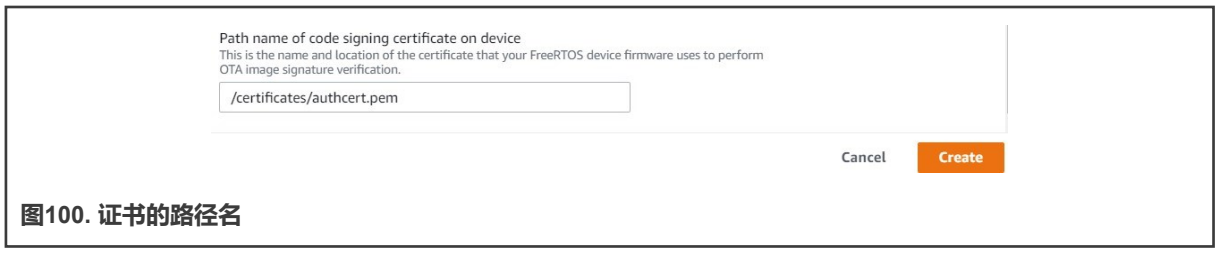


图100. 证书的路径名

9. 在“File”下，选择“Select an existing file”，然后选择“Browse S3”，选择在S3中已上传的 `sfw093.bin` 文件。

注意

请确保目前的区域与桶所在的区域一致。否则将无法找到已上传的二进制文件。

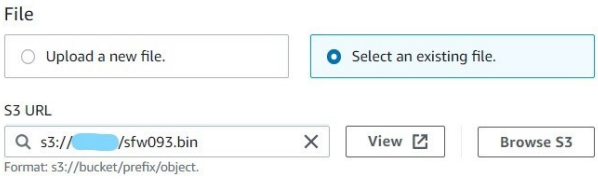


图101. 映像文件选择

图102. 文件的路径名

10. 在“Path name of file on device”下，输入默认值：path/device/updates。
11. 在“IAM role”下，选择前面步骤中创建的角色。

图103. 选择角色

12. 选择“Next”。
13. 在“Job run type”下，选择第一项。

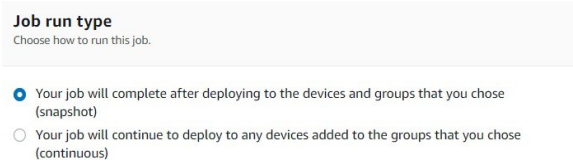


图104. 作业运行类型

14. 在点击“Create job”之前，运行应用程序。

7.3.1.7 运行应用程序

1. 使用 `MCUBootUtility` 工具将之前生成的 `sfw092.bin`下载到板卡的slot1。slot1的默认位置是 flash_offset+0x100000 到 flash_offset+0x200000，整个slot大小为1MB。

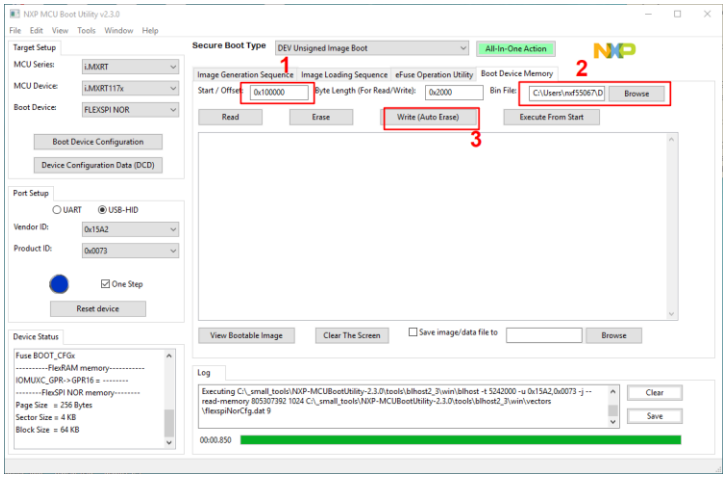


图105. 下载 `sfw092.bin` 到slot1

2. 成功下载映像文件后，复位板卡。串口将打印出应用程序的log信息，如图所示：

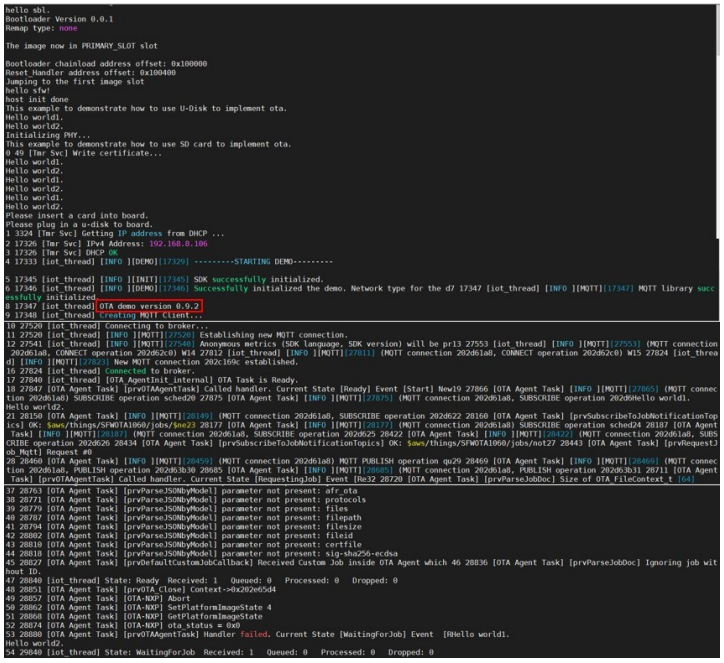


图106. 应用程序log

3. 运行应用程序时，等待串行终端出现表示OTA准备就绪的信息，如图所示：

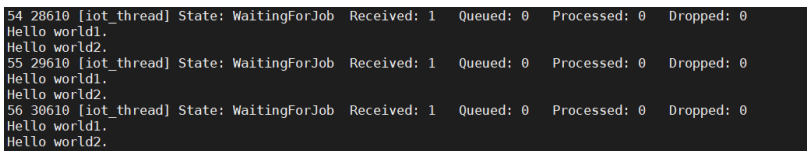


图107. OTA准备就绪log

4. 此时，OTA代理正在等待OTA作业。返回“Create OTA job”窗口，点击“Create OTA job”。

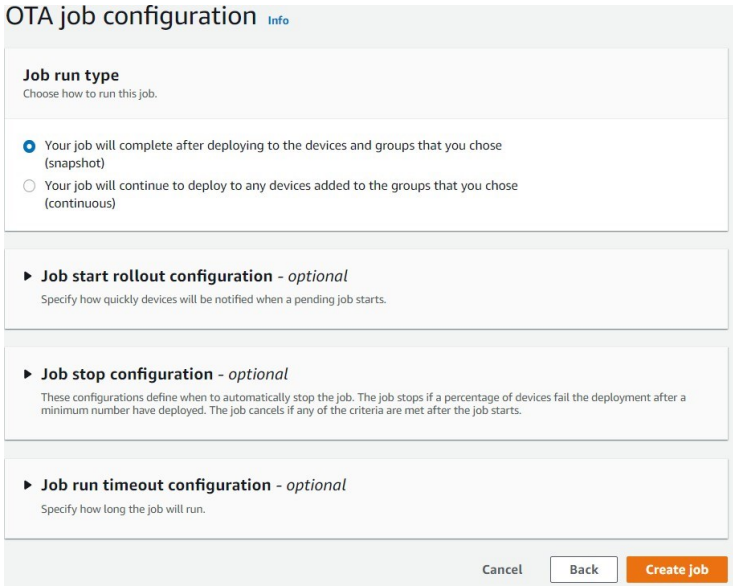


图108. 创建作业

5. 此时OTA开始，相关输出如下所示。
   a. 文件传输开始

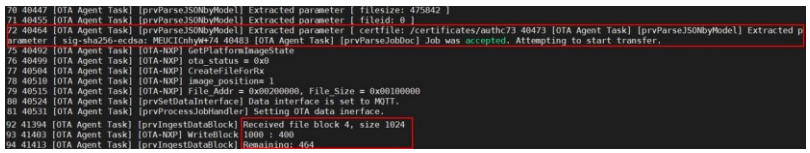


图109. 开始文件传输log日志

   b. 收到整个文件

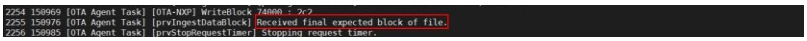


图110. 收到整个文件log

   c. 文件签名检查

图111. 检查文件签名log

   d. 映像文件版本检查

图112. 映像版本检查log

   e. 写入更新类型

图113. 写入更新类型log

f. 写映像文件的尾部

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

图114. 写映像文件尾部log

g. 激活新的映像文件，重启板卡

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

图115. 激活新映像文件log

h. 运行新映像文件

```
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116. 运行新映像文件log

i. 自检

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

图117.自检log

j. 写入OK标志位

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

图118. 写入OK标志位log

k. OTA成功

```
65 27824 [OTA Agent Task] [prvStopSelfTestTimer] Stopping the self test timer.
66 27832 [OTA Agent Task] [prvUpdateJobStatus_Mqtt] Msg: {"status":"SUCCEEDED","statusDetails":{"re67 27855 [OTA Agent Task] [INFO ][MQTT][27855] (MQTT connec
tion 202d61a8) MQTT PUBLISH operation qos0 27865 [OTA Agent Task] [INFO ][MQTT][27865] (MQTT connection 202d61a8, PUBLISH operation 202d63bHello world1.
Hello world2.
69 28113 [OTA Agent Task] [INFO ][MQTT][28113] (MQTT connection 202d61a8, PUBLISH operation 202d63b70 28124 [OTA Agent Task] [prvUpdateJobStatus_Mqtt] 'SUCCEE
DED' to $aws/things/SFWOTA1060/jobs/AFR_71 28133 [OTA Agent Task] [prvOTAAgentTask] Called handler. Current State [CreatingFile] Event [Sta72 28435 [iot_threa
d] State: WaitingForJob  Received: 1   Queued: 0   Processed: 0   Dropped: 0
```

图119. OTA成功log

6. OTA成功后，按下板卡上的复位按钮以确认更新是否成功，更新成功则串口会答应出 `sfw093.bin` 文件的相关log信息。

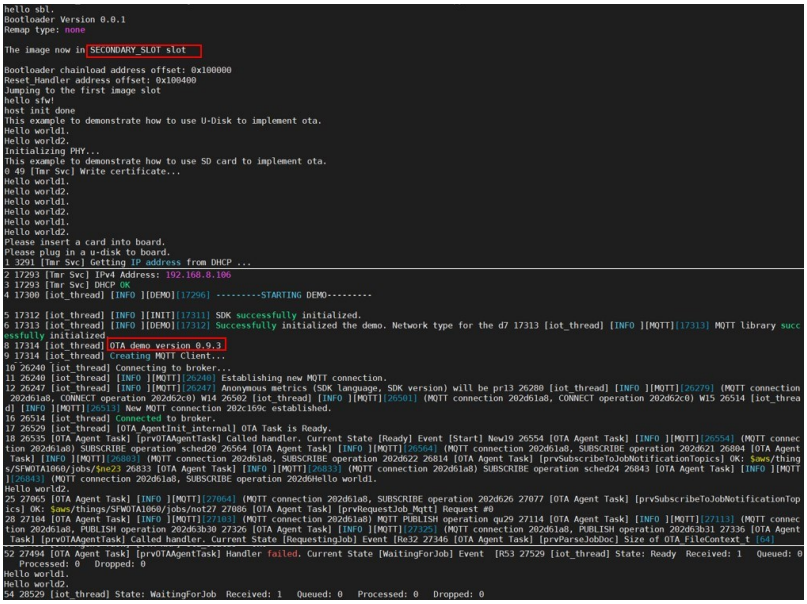


图120. 新映像文件的log

7.3.2 阿里云OTA

本节介绍如何使用“Aliyun IoT”（阿里云物联网）对板卡进行阿里云OTA固件更新的步骤，并以EVKMIMXRT1064平台为例，演示测试过程。

7.3.2.1 创建一个测试设备

1. 打开链接：https://iot.console.aliyun.com/。注册一个账户，登录平台，创建阿里云账户。

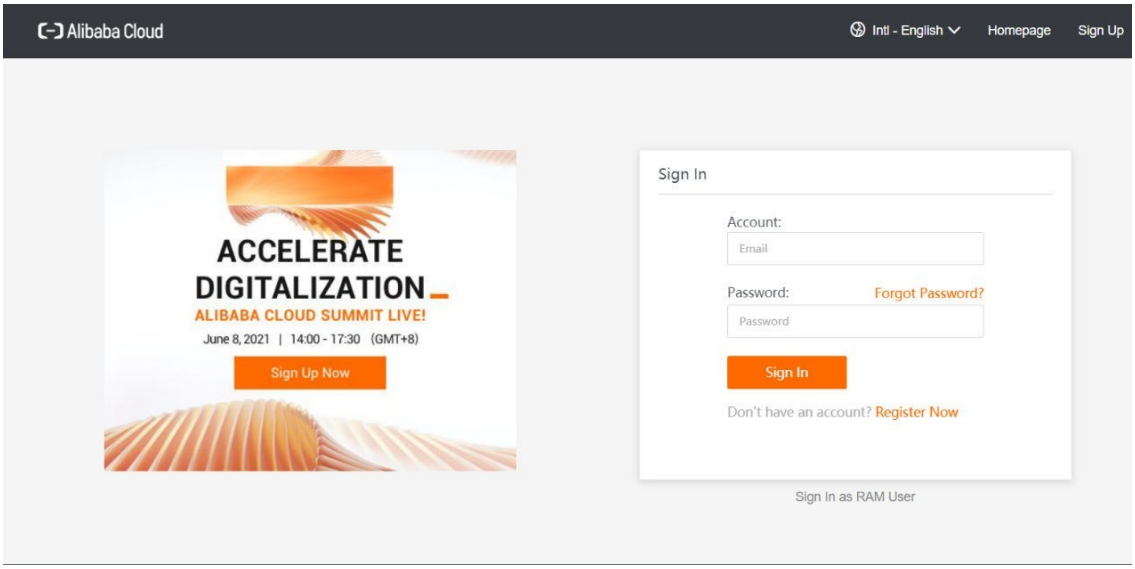


图121. 账户创建界面

2. 登录成功后，显示阿里云物联网平台的页面，点击进入“Public Instance”（公共实例）界面。

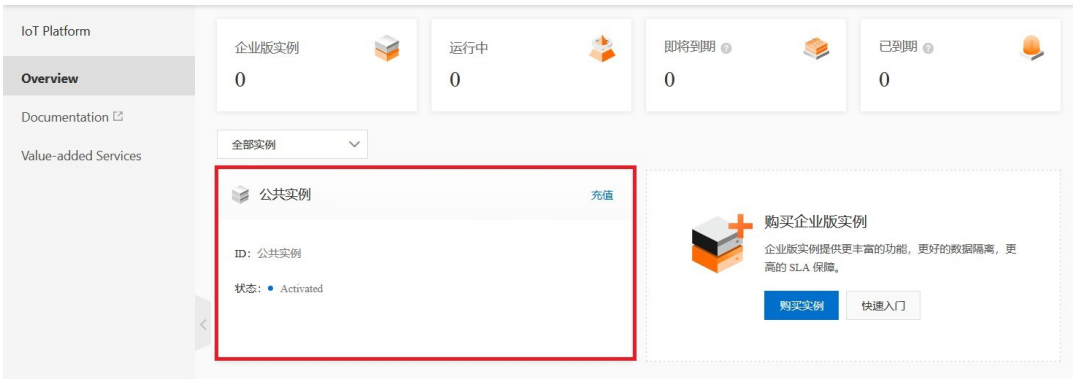


图122. 阿里云物联网平台

3. 进入“Public Instance”（公共实例）后，点击“Create Product”（创建产品），在“New Product”（新产品）页面设置如图所示的参数，自定义一个产品名称，并默认选择类别中的第一个（测试OTA功能）。

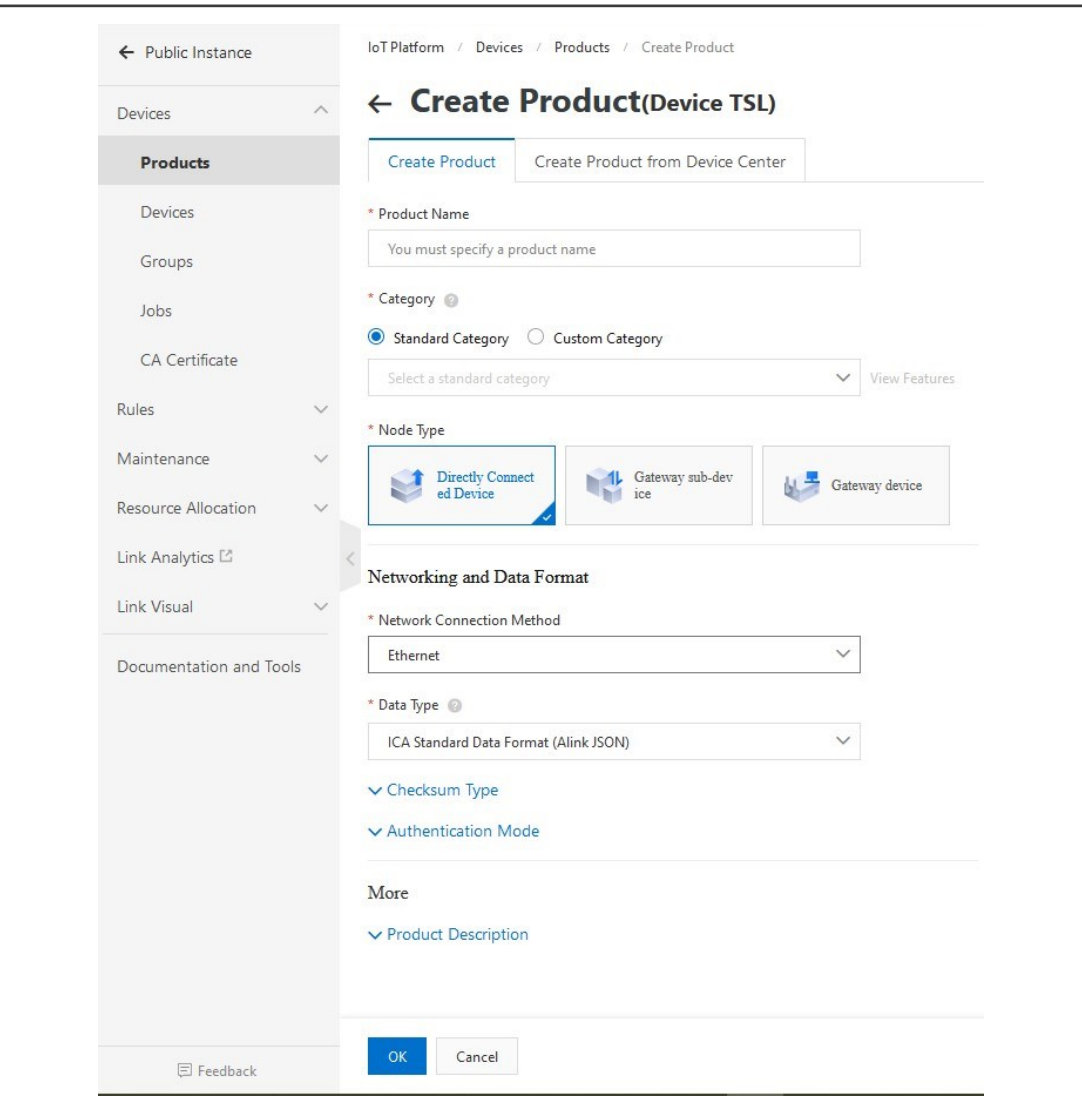


图123. 创建产品

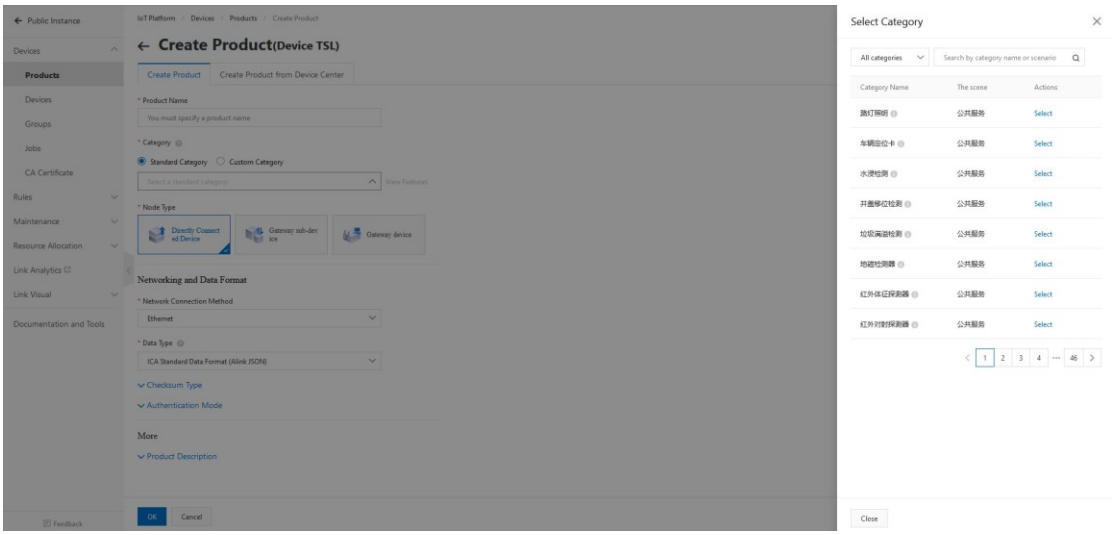


图124. 产品类型

在产品成功创建后，将设备添加到产品中。

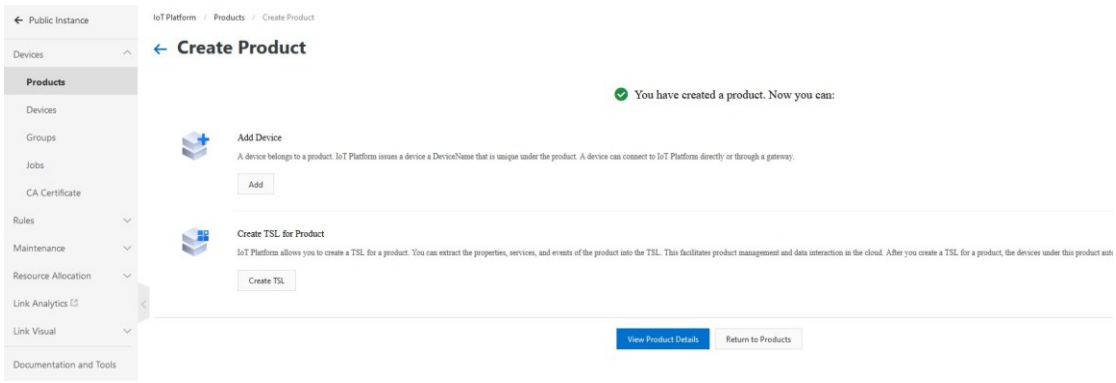


图125. 成功创建产品

4. 选择需要添加设备进行测试的产品，并点击“Add Device”（添加设备）的蓝色按钮以设置设备名称。

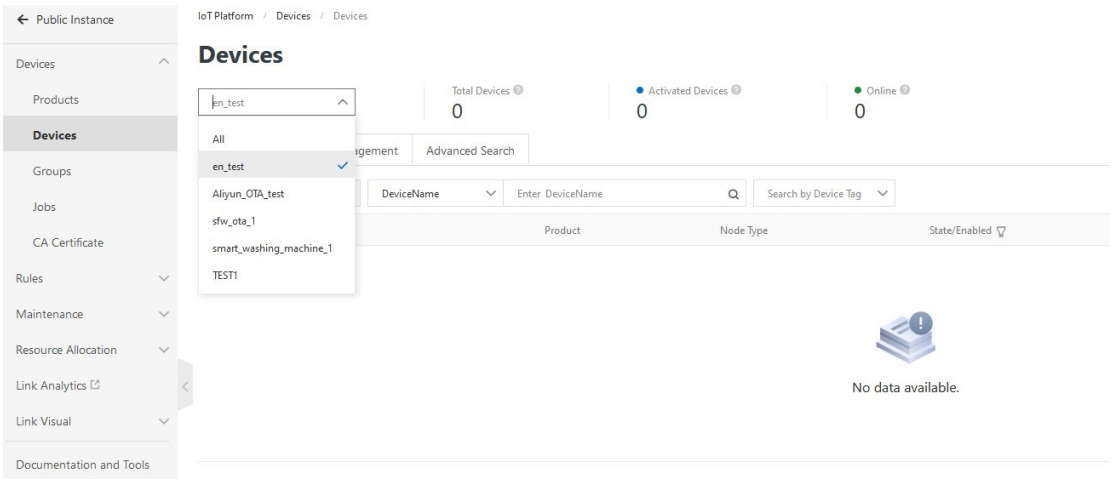


图126. 添加设备

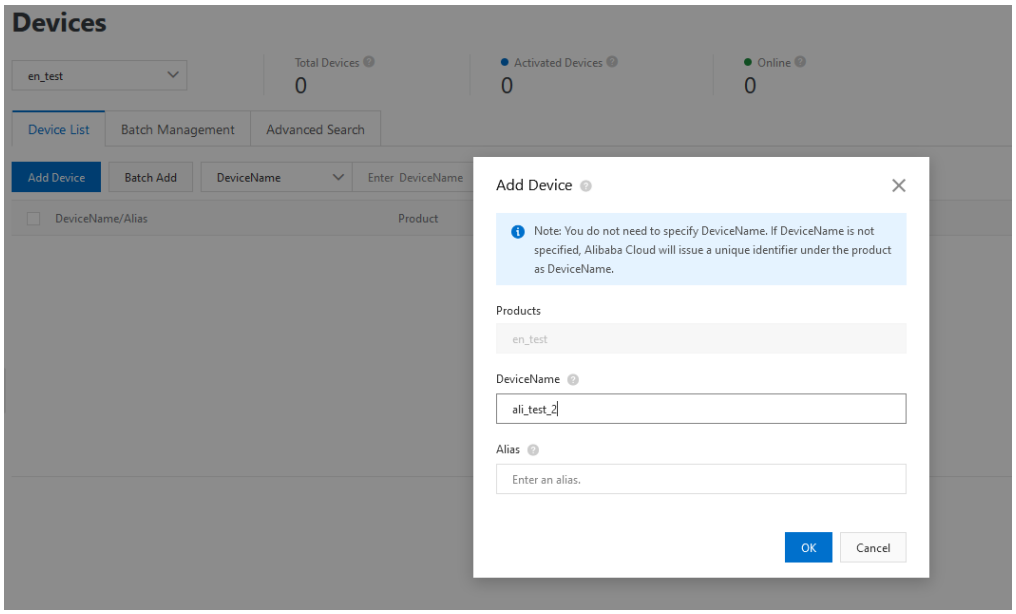


图127. 设备信息

设备添加完毕后，如下图所示：

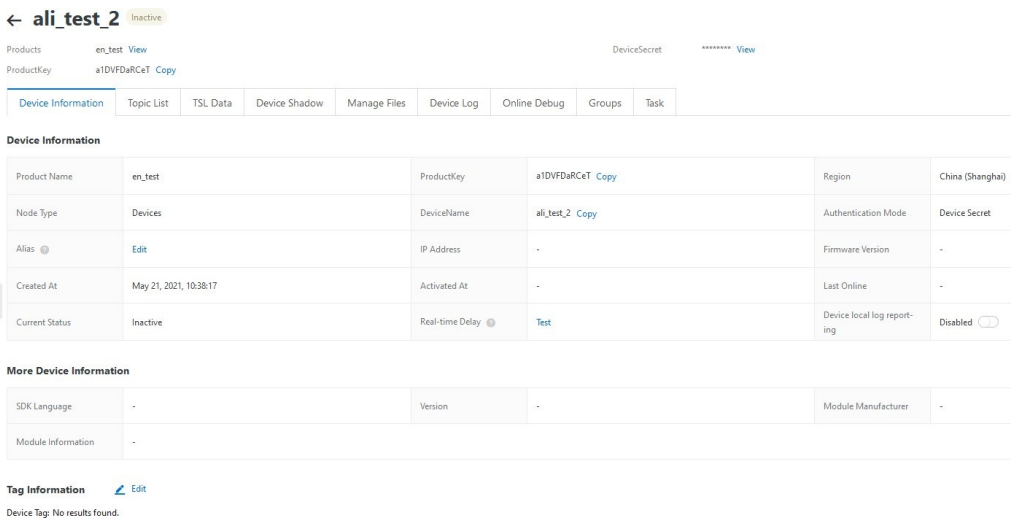


图128. 设备详情

7.3.2.2 定制设备端SDK

1. 在左边的菜单栏中选择“Documents and Tools”文档与工具。要显示如下图所示的界面，就要在“Link SDK”（链接SDK）中选择“SDK custom”（SDK定制）"，设置如下图所示的SDK版本信息，然后点击“Start Generation”（开始生成），生成设备端SDK。

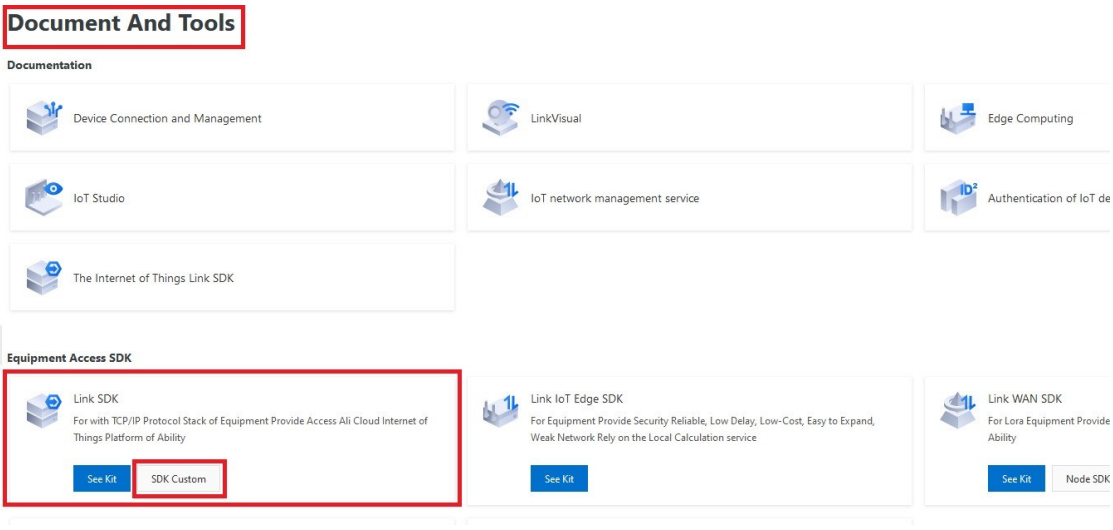


图129. 下载SDK

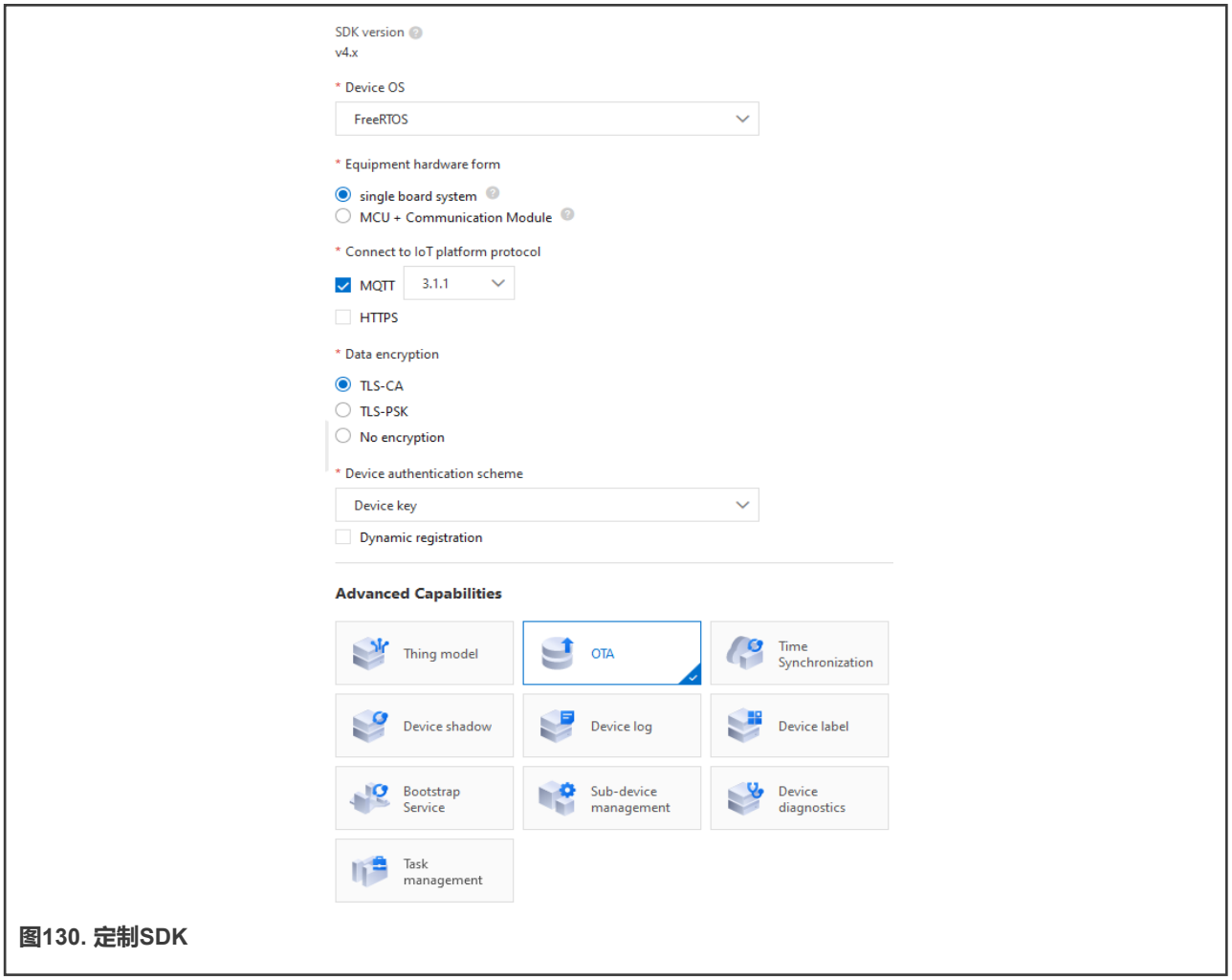


图130. 定制SDK

2. 解压下载的SDK包，将 `\LinkSDK\external`中的 `ali_ca_cert.c` 替换为测试项目的相应文件。它与当前项目中的证书是一致的，在本次测试中不需要替换。

7.3.2.3 设置测试设备信息

1. 在阿里云物联网平台的设备选项下，保存DeviceName、ProductKey和DeviceSecret信息。

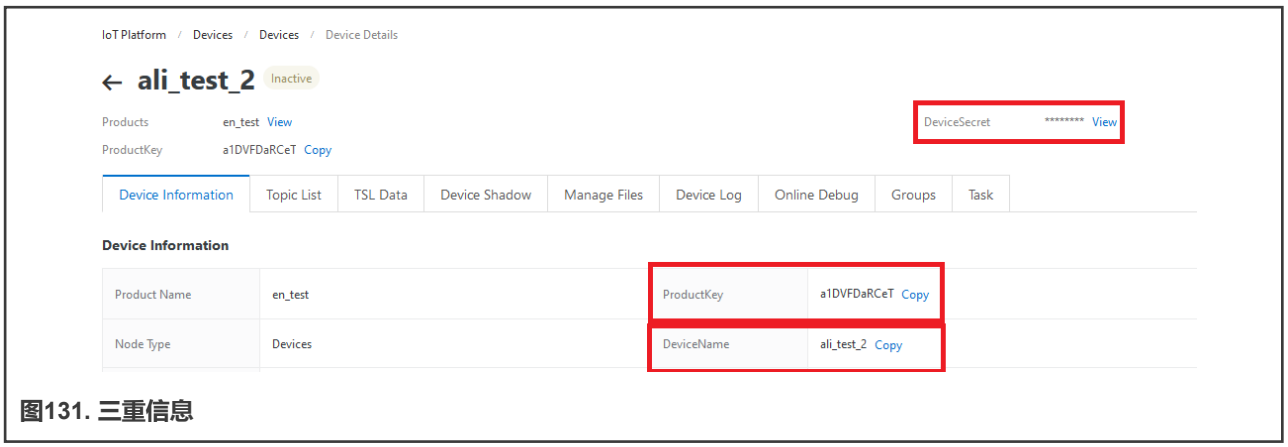


图131. 三重信息

2. 打开测试项目\ sfw \target \evkbmimxrt10XX，双击脚本 env.bat，使用命令 scons--menuconfig 选择阿里云项目，设置三要素，如下图。将product key/device name/product secret复制到其中。

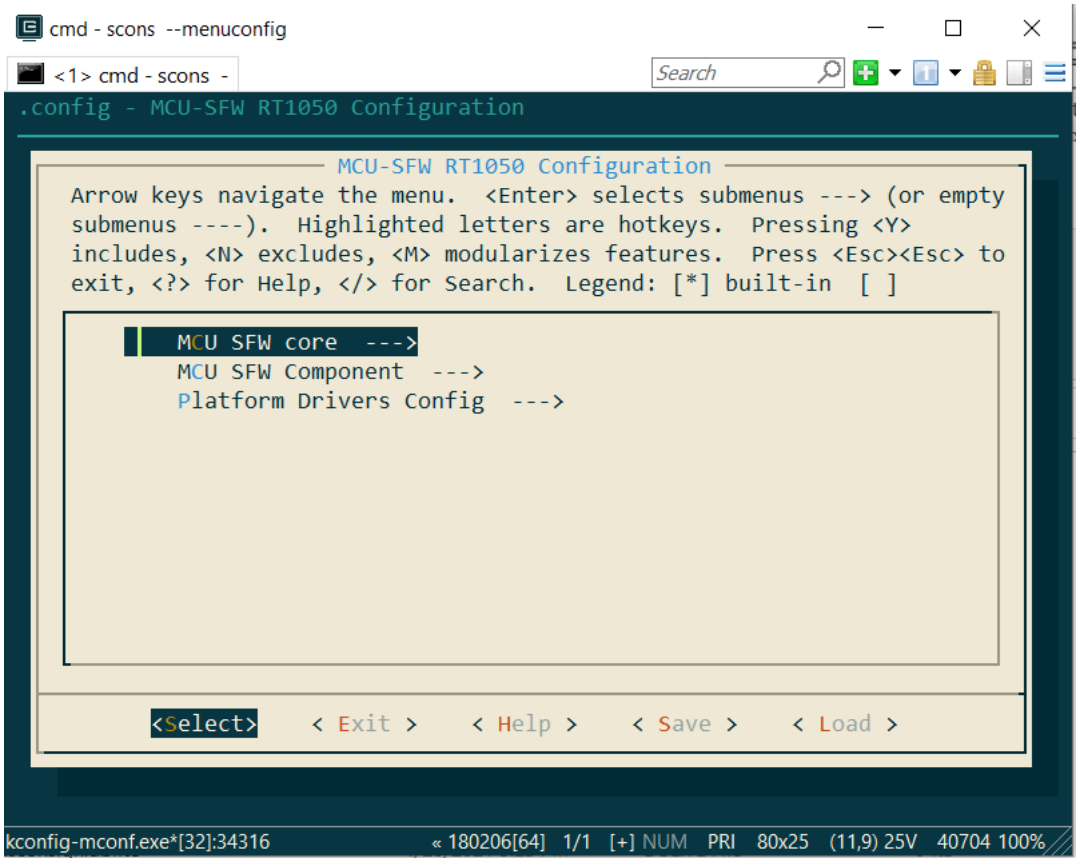


图132. Menuconfig

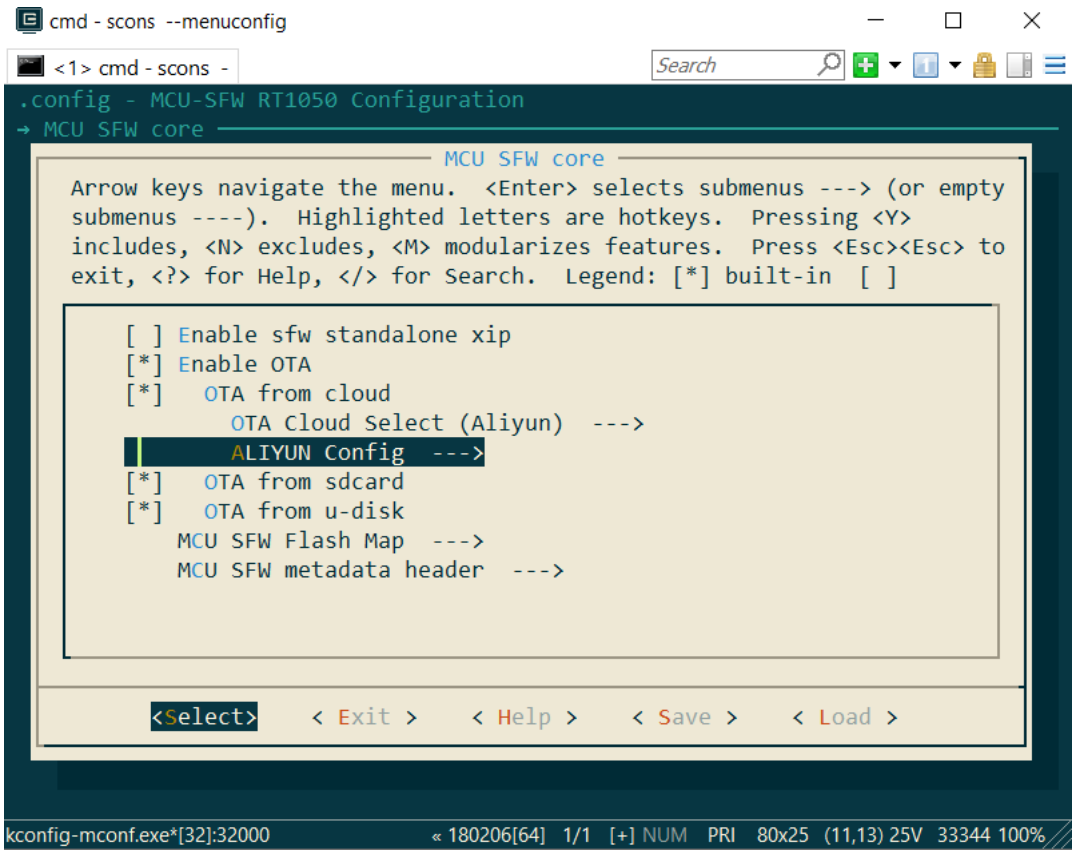


图133. 设置阿里云OTA信息

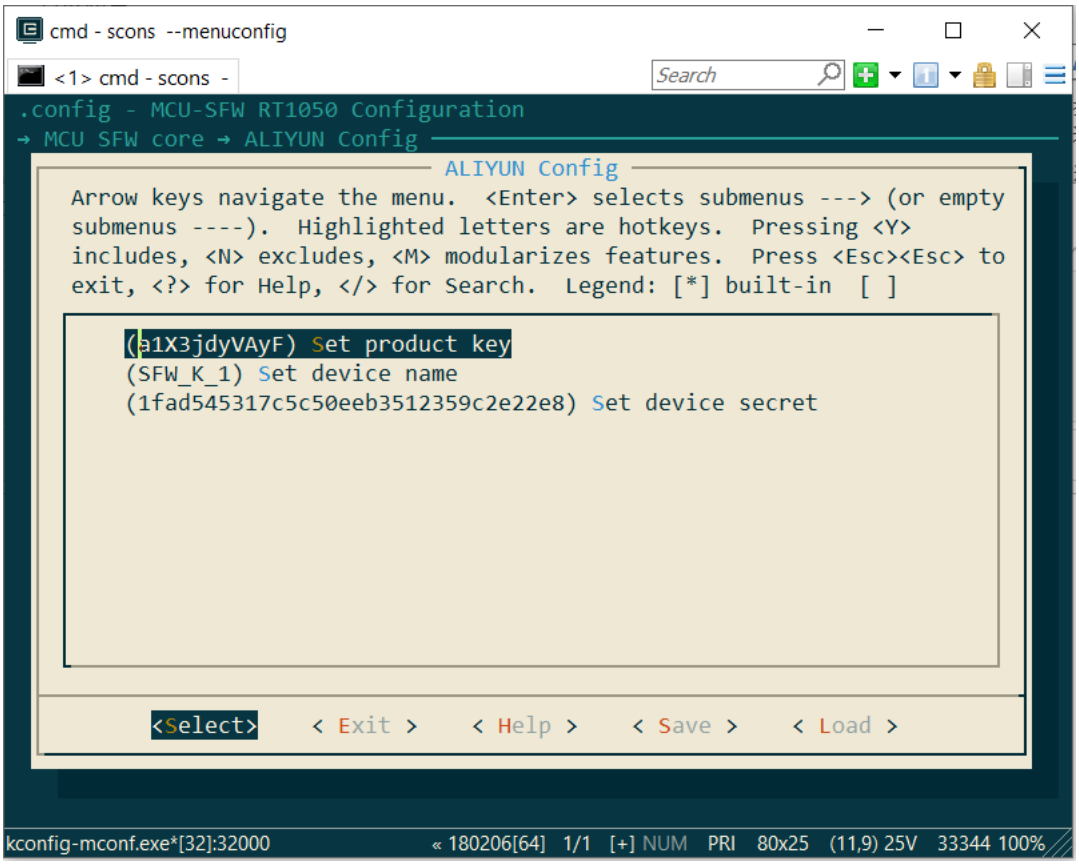


图134. 设置三要素

3. 在脚本中，选择 `MCU SFW Component -> secure -> enable mbedtls`，并将mbedtls配置文件改为 `ksdk_mbedtls_config.h`

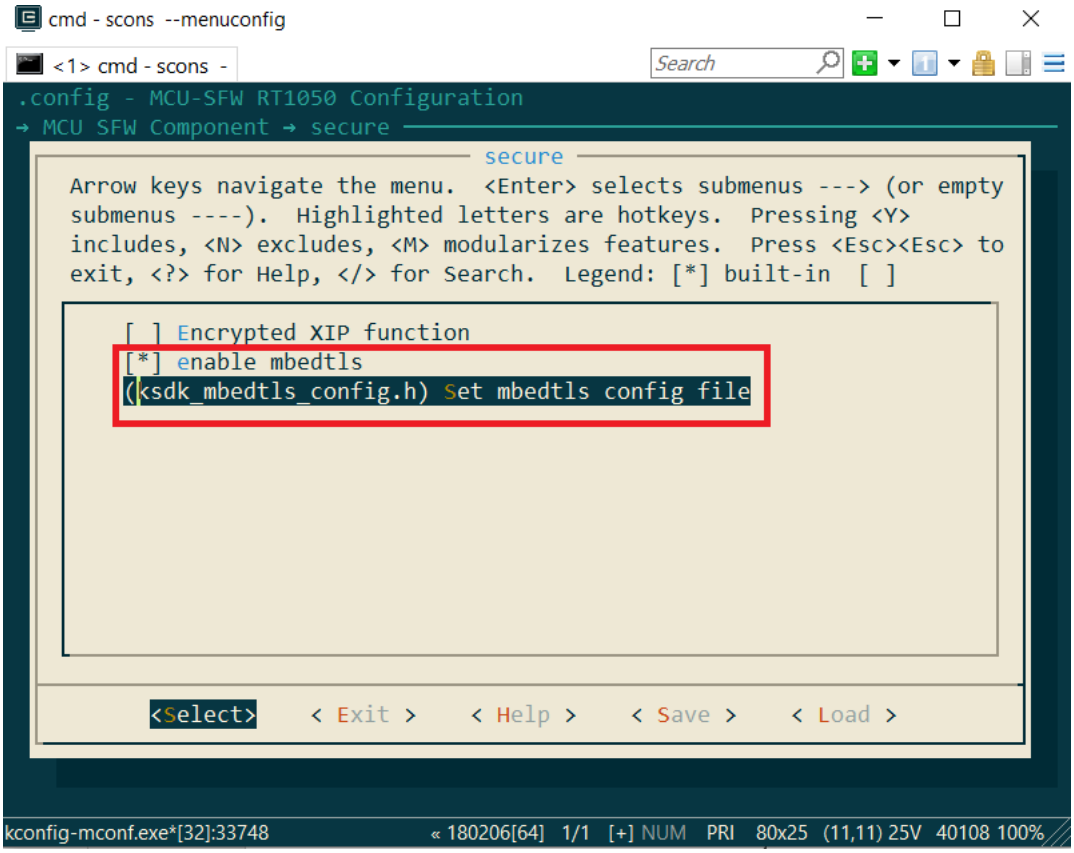


图135. 设置mbedtls配置文件

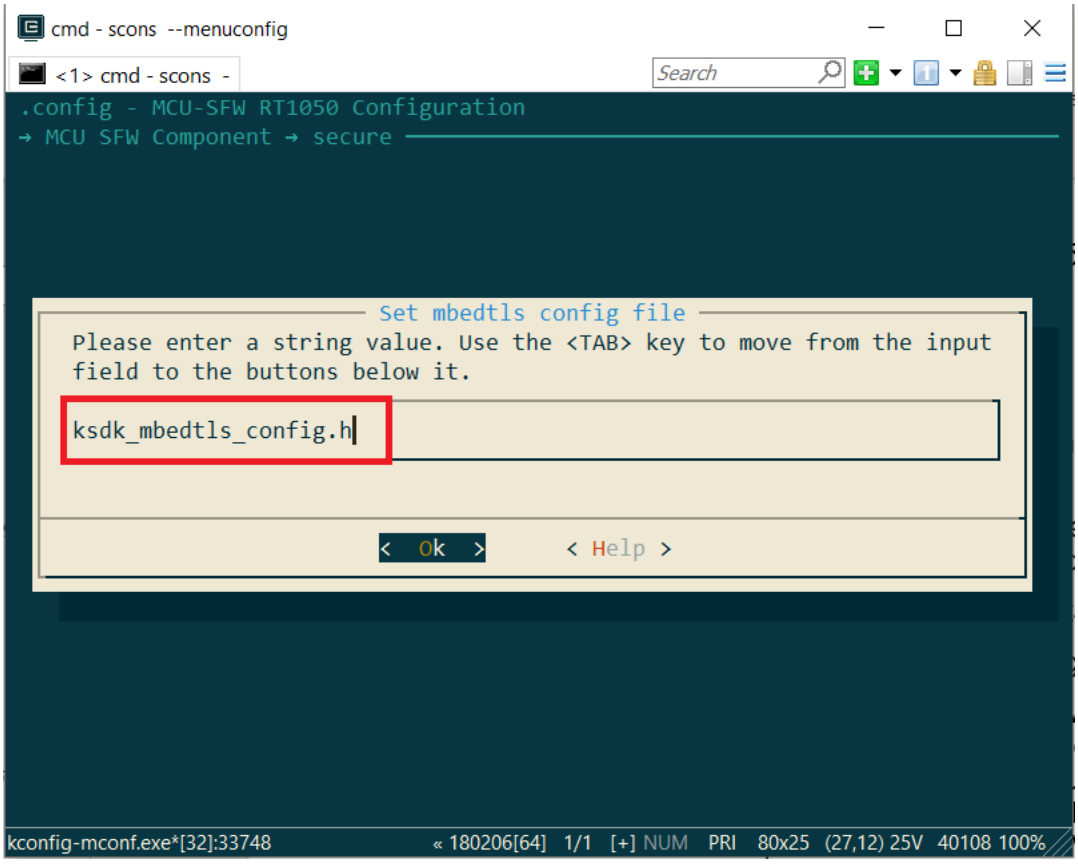


图136. `ksdk_mbedtls_config.h`

保存设置并使用命令 `scons –ide=iar` 来生成iar项目。

7.3.2.1 修改 `cur_version`，以便进行测试

1. 进入 `sfw/target/evkmimxrt1064` 路径，双击批处理文件 `env.bat`
2. 输入 `scons –ide=iar` 命令来生成iar项目。

> **注意**
> 请参考第 2 节，以生成keil或gcc项目。

3. 进入 `sfw/target/evkmimxrt1064/iar` 路径，打开sfw.eww项目。
4. 进入options，选择 `generate additional output`，并选择“Raw binary”格式。

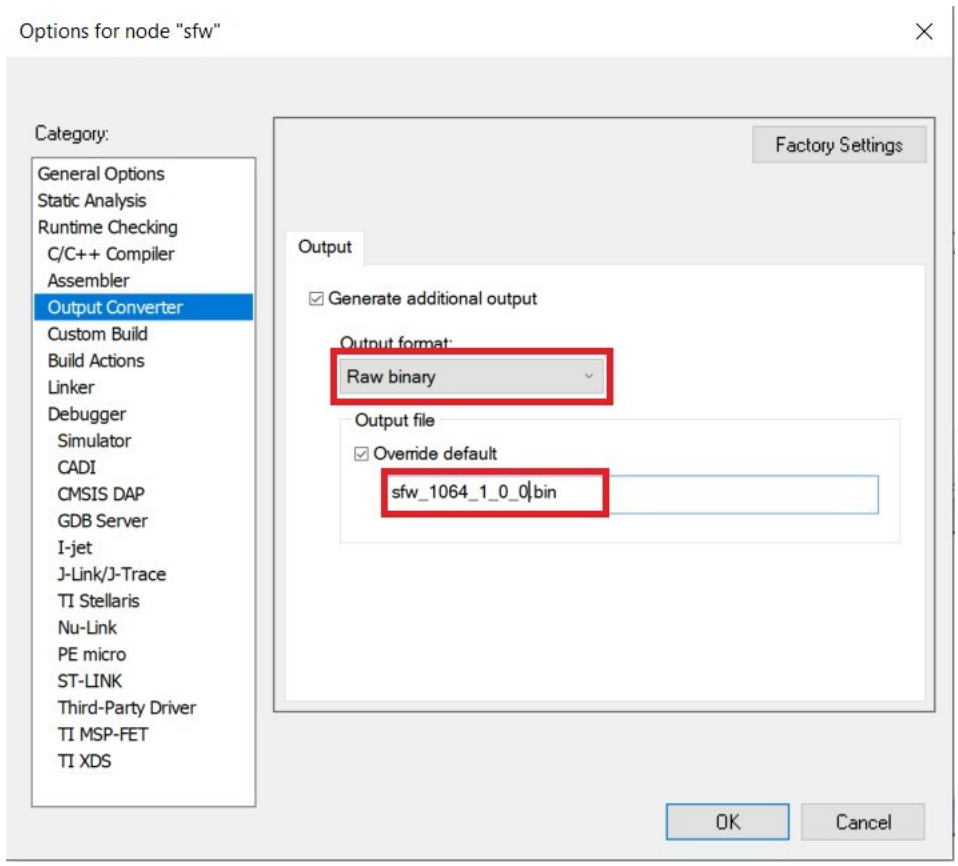


图137. 输出文件

5. 在 `sfw/firmware/Aliyun_ota/fota_basic_demo.c` 文件中查看当前版本。将 `cur_version` 设为1.0.0。

```
342     cur_version = "1.0.0";  //更改为所需要更新的版本，如1.1.0
```

图138. 设置版本

6. 将bin文件改为 `sfw_1064_100.bin`，然后点击“make”按钮来构建项目。bin文件将在 `/sfw/target/evkmimxrt1064/iar/build/iar/Exe` 目录下生成。
7. 将cur_version改为“1.4.0”，并将bin文件名改为 `sfw_1064_140.bin`。生成它。
8. 将生成的bin文件复制到 `sbl/component/secure/mcuboot/scripts` 文件夹中。
9. 使用下面的命令用RSA签名 `sfw_1064_100.bin` 和 `sfw_1064_140.bin` 工程源文件。然后 `1064_ali_100.bin` 和 `1064_ali_140.bin` 就会生成。

```
python imgtool.py sign --key sign-rsa2048-priv.pem --align 4 --version "1.0.0" --header-size
 0x400 --pad-header --slot-size 0x100000 --max-sectors 32 sfw_1064_100.bin 1064_ali_100.bin
python imgtool.py sign --key sign-rsa2048-priv.pem --align 4 --version "1.4.0" --header-size
 0x400 --pad-header --slot-size 0x100000 --max-sectors 32 sfw_1064_140.bin1064_ali_140.bin
```

7.3.2.2 在云端创建OTA任务

1. 在左边的“Maintenance”（监控运维）目录下，选择“OTA Update”（OTA更新）。

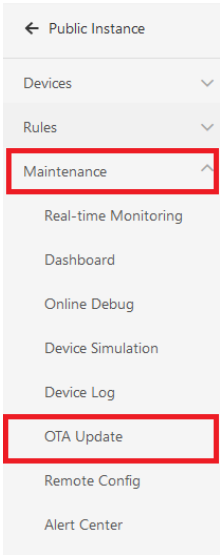


图139. OTA升级

2. 点击“Add Update Package”（添加升级包）按钮来添加升级包，输入升级包名称，选择相应的产品模块，“Update package version”（升级包版本）应与上传的bin文件的版本相对应，然后点击上传确认。现在使用的是1.4.0版本号。

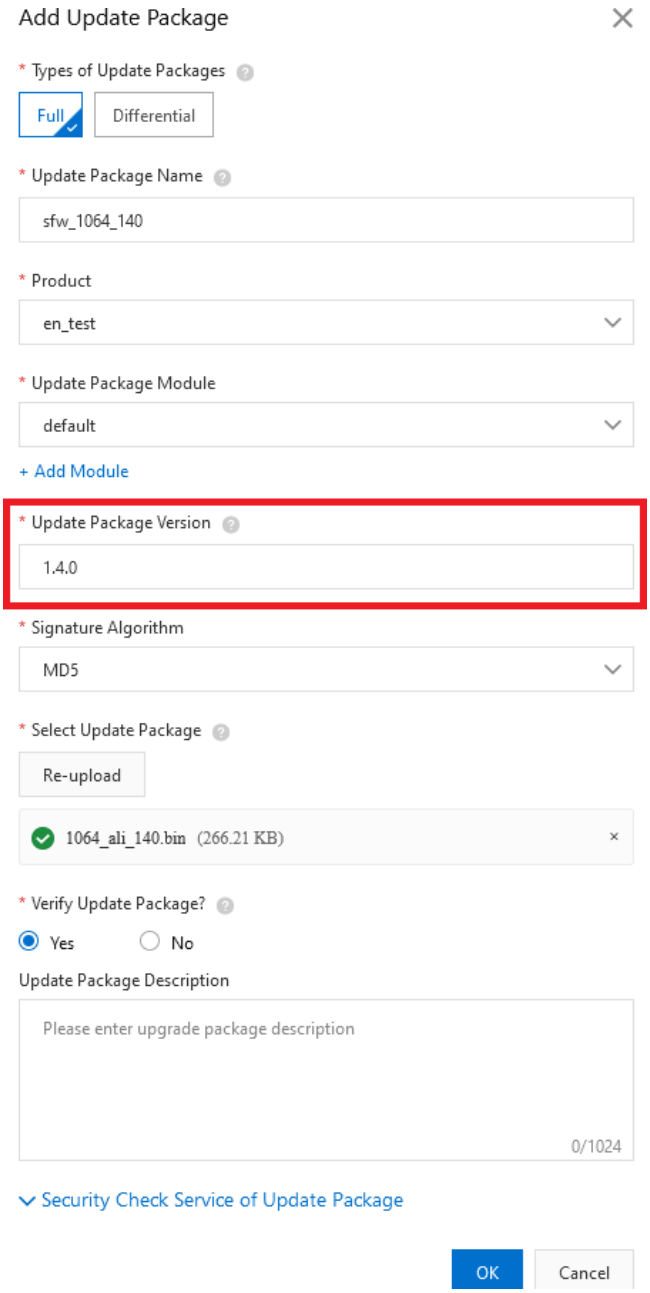


图140. 升级包信息

7.3.2.3 运行该应用程序

1. 使用 `MCUBootUtility` 工具将之前生成的 `1064_ali_100.bin` 下载到电路板的slot1。slot1的默认位置是 flash_offset+0x100000 到 flash_offset+0x200000，整个slot大小为1MB。

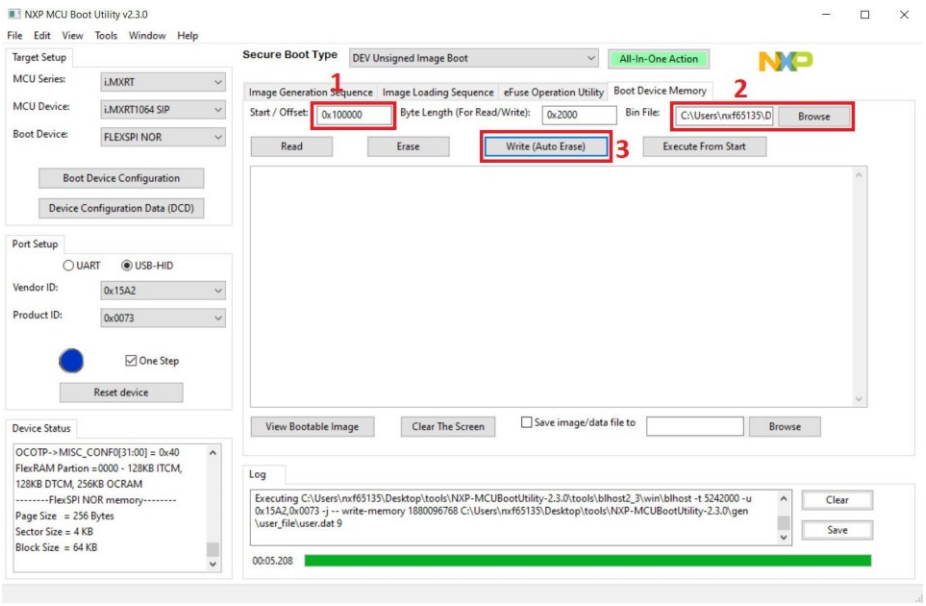


图141. 使用映像工具下载bin文件

2. 准备引导加载程序。

   在SBL项目中，进入路径 `sbl/target/evkmimxrt1064`。在Scons的menuconfig界面中禁用 `Enable single image function` 选项和 `Enable mcu isp support` 选项，以禁用单一源文件模式和禁用MCU ISP支持。

   编译SBL项目并将其下载到目标电路板。

   注意

   a. 如果使用新的签名密钥，请同时修改 `sign-rsa2048-pub.c`

   b. 通过DAPLink的拖放功能来编程SBL映像，可能会擦除整个闪存。

3. 插上以太网线。运行该项目，调试控制台显示如下所示的日志。

   日志“The image now in PRIMARY_SLOT slot”和“Getting IP address from DHCP”显示：slot1的映像已经启动成功。“IPv4地址：”和“版本：1.0.0”显示：网络连接成功，阿里云获得当前设备版本1.0.0。

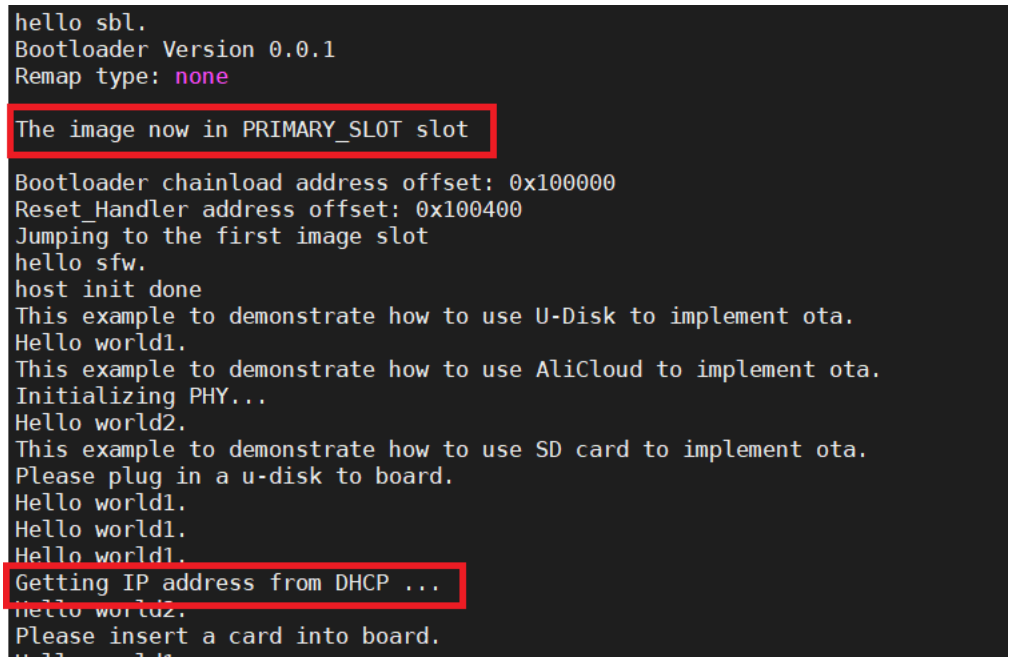


图142. 运行sbl项目

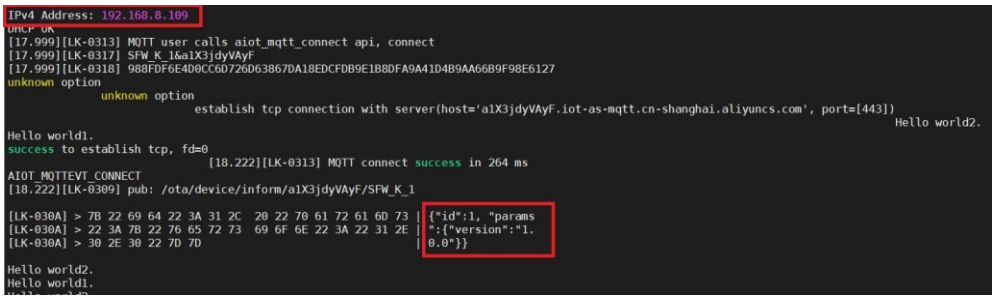


图143. 跳转到第一个工程

4. 在Web中验证升级包。点击“Verify”（验证）按钮，填写需要升级的版本号，并选择要测试的设备。可以省略升级超时间。然后点击“OK”。

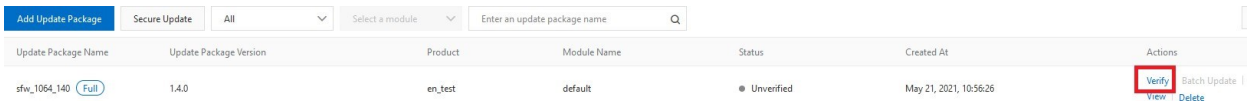


图144. 验证OTA包

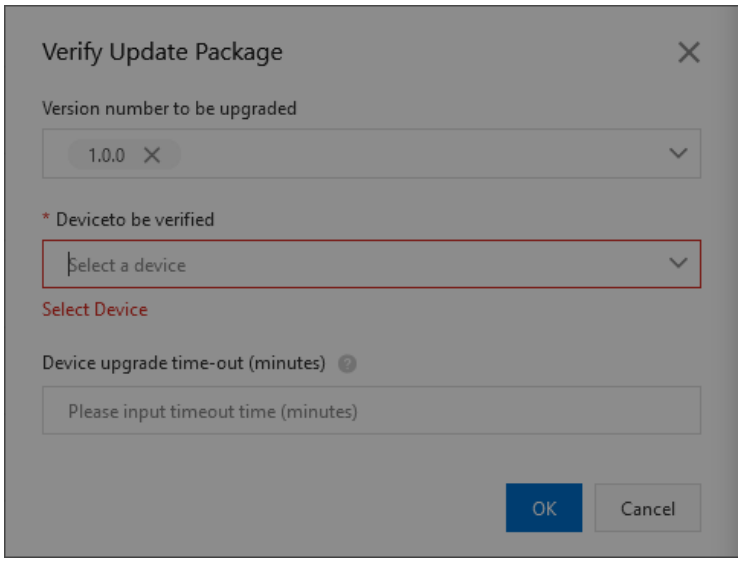


图145. 验证升级包信息

5. 调试控制台显示OTA进度。

下面是上传的软件包及其版本的信息。

```
[21.888][LK-0309] pub: /ota/device/upgrade/a1X3jdyVAyF/SFW_K_1

[LK-030A] < 7B 22 63 6F 64 65 22 3A  22 31 30 30 30 22 2C 22 | {"code":"1000","
[LK-030A] < 64 61 74 61 22 3A 7B 22  73 69 7A 65 22 3A 32 37 | data":{"size":27
[LK-030A] < 32 36 30 30 2C 22 64 69  67 65 73 74 53 69 67 6E | 2600,"digestSign
[LK-030A] < 22 3A 22 45 34 45 4F 72  42 6D 4C 37 77 55 46 6B | ":"E4EOrBmL7wUFk
[LK-030A] < 41 78 39 66 6E 57 67 62  39 35 33 30 67 2F 37 51 | Ax9fnWgb9530g/7Q
[LK-030A] < 64 48 78 4F 6C 51 31 32  6C 4C 63 7A 6E 46 45 4F | dHxOlQ12lLcznFEO
[LK-030A] < 46 4C 62 58 5A 66 42 4F  58 76 76 30 55 77 63 6A | FLbXZfBOXvv0Uwcj
[LK-030A] < 73 7A 65 75 56 6B 66 72  77 38 30 38 62 6B 76 43 | szeuVkfrw808bkvC
[LK-030A] < 38 33 57 71 52 4A 58 6C  2F 6D 67 6A 73 50 36 54 | 83WqRJXl/mgjsP6T
[LK-030A] < 45 33 62 6A 41 51 7A 50  32 7A 2F 2F 45 52 67 36 | E3bjAQzP2z//ERg6
[LK-030A] < 31 56 6B 4B 7A 37 6B 70  75 44 58 48 6D 59 50 66 | 1VkKz7kpuDXHmYPf
[LK-030A] < 35 37 6B 6A 49 32 54 53  42 65 6A 51 77 50 71 6E | 57kjI2TSBejQwPqn
[LK-030A] < 58 2F 2F 44 59 53 56 2B  76 49 78 63 37 6D 6E 32 | X//DYSV+vIxc7mn2
[LK-030A] < 71 57 73 35 6A 43 49 69  6B 37 41 42 72 33 31 58 | qWs5jCIik7ABr31X
[LK-030A] < 7A 5A 34 67 48 36 55 77  53 69 2B 53 50 62 72 56 | zZ4gH6UwSi+SPbrV
[LK-030A] < 6A 32 48 4E 6C 49 31 6C  4F 43 4B 4C 37 59 57 55 | j2HNlI1lOCKL7YWU
[LK-030A] < 61 65 38 76 51 58 61 43  62 56 50 73 72 2F 4F 79 | ae8vQXaCbVPsr/Oy
[LK-030A] < 50 64 68 5A 4B 51 52 62  39 4A 62 78 54 38 72 61 | PdhZKQRb9JbxT8ra
[LK-030A] < 4F 55 43 64 52 37 37 68  57 39 65 45 6D 6B 50 47 | OUCdR77hW9eEmkPG
[LK-030A] < 68 2F 6A 50 57 37 39 45  50 30 6F 6D 53 38 33 4B | h/jPW79EP0omS83K
[LK-030A] < 49 4A 54 71 34 56 6D 45  6B 4E 44 72 5A 49 69 4D | IJTq4VmEkNDrZIiM
[LK-030A] < 71 74 68 4A 45 79 38 54  59 71 72 46 79 59 46 5A | qthJEy8TYqrFyYFZ
[LK-030A] < 64 6E 77 79 63 48 50 4E  53 64 41 4E 37 37 71 4D | dnwycHPNSdAN77qM
[LK-030A] < 6B 66 5A 52 66 4F 4E 49  4E 66 6C 55 39 78 6A 33 | kfZRfONINflU9xj3
[LK-030A] < 68 39 79 6A 36 2B 63 79  77 3D 3D Hello world1.
22 2C 22 73 69 | h9yj6+cyw==","si
[LK-030A] < 67 6E 22 3A 22 35 32 61  35 63 30 32 61 65 37 61 | gn":"52a5c02ae7a
[LK-030A] < 66 63 64 64 63 35 33 62  38 63 66 36 34 62 61 34 | fcddc53b8cf64ba4
[LK-030A] < 36 31 34 34 64 22 2C 22  76 65 72 73 69 6F 6E 22 | 6144d","version"
[LK-030A] < 3A 22 31 2E 34 2E 30 22  2C 22 75 72 6C 22 3A 22 | :"1.4.0","url":"
[LK-030A] < 68 74 74 70 73 3A 2F 2F  69 6F 74 78 2D 6F 74 61 | https://iotx-ota
[LK-030A] < 2E 6F 73 73 2D 63 6E 2D  73 68 61 6E 67 68 61 69 | .oss-cn-shanghai
[LK-030A] < 2E 61 6C 69 79 75 6E 63  73 2E 63 6F 6D 2F 6F 74 | .aliyuncs.com/ot
[LK-030A] < 61 2F 66 66 35 64 39 30  33 32 33 34 37 39 33 31 | a/ff5d9032347931
[LK-030A] < 39 37 65 61 37 63 31 62  66 64 61 36 37 31 38 64 | 97ea7c1bfda6718d
[LK-030A] < 30 39 2F 63 6B 6F 6D 61  33 6D 35 71 30 30 30 30 | 09/ckoma3m5q0000
[LK-030A] < 33 61 38 65 34 65 35 30  77 38 38 6B 2E 62 69 6E | 3a8e4e50w88k.bin
```

图146. 获取软件包信息

```
[LK-030A] < 61 74 75 72 65 3D 59 64  59 69 6C 75 6F 77 4E 58 | ature=YdYiluowNX
[LK-030A] < 25 32 42 70 6C 61 77 52  59 66 64 4C 4B 4A 70 47 | %2BplawRYfdLKJpG
[LK-030A] < 74 5A 59 25 33 44 22 2C  22 73 69 67 6E 4D 65 74 | tZY%3D","signMet
[LK-030A] < 68 6F 64 22 3A 22 4D 64  35 22 2C 22 6D 64 35 22 | hod":"Md5","md5"
[LK-030A] < 3A 22 35 32 61 35 63 30  32 61 65 37 61 66 63 64 | :"52a5c02ae7afcd
[LK-030A] < 64 63 35 33 62 38 63 66  36 34 62 61 34 36 31 34 | dc53b8cf64ba4614
[LK-030A] < 34 64 22 7D 2C 22 69 64  22 3A 31 36 32 30 38 37 | 4d"},"id":162087
[LK-030A] < 35 37 33 35 37 30 38 2C  22 6D 65 73 73 61 67 65 | 5735708,"message
[LK-030A] < 22 3A 22 73 75 63 63 65  73 73 22 7D            | ":"success"}

OTA target firmware version: 1.4.0, size: 272600 Bytes
unknown option
            establish tcp connection with server(host='iotx-ota.oss-cn-shanghai.aliyuncs.com', port=[80])
                                                                                                    Hel
success to establish tcp, fd=1
                          [22.222][LK-040B] > GET /ota/ff5d903234793197ea7c1bfda6718d09/ckoma3m5q00003a8e4
LTAI4G1TuWwSirnbAzUHfL3e&Signature
[22.333][LK-040B] > Host: iotx-ota.oss-cn-shanghai.aliyuncs.com
[22.333][LK-040B] > Accept: text/html, application/xhtml+xml, application/xml;q=0.9, */*;q=0.8
[22.333][LK-040B] > Range: bytes=0-
[22.333][LK-040B] > Content-Length: 0
[22.333][LK-040B] >
[22.333][LK-0309] pub: /ota/device/progress/a1X3jdyVAyF/SFW_K_1

[LK-030A] > 7B 22 69 64 22 3A 32 2C  20 22 70 61 72 61 6D 73 | {"id":2, "params
[LK-030A] > 22 3A 7B 22 73 74 65 70  22 3A 22 30 22 2C 22 64 | ":{"step":"0","d
[LK-030A] > 65 73 63 22 3A 22 22 7D  7D                     | esc":""}}
```

图147. 目标固件版本

下面显示：下载请求已被成功发送，下载过程开始。

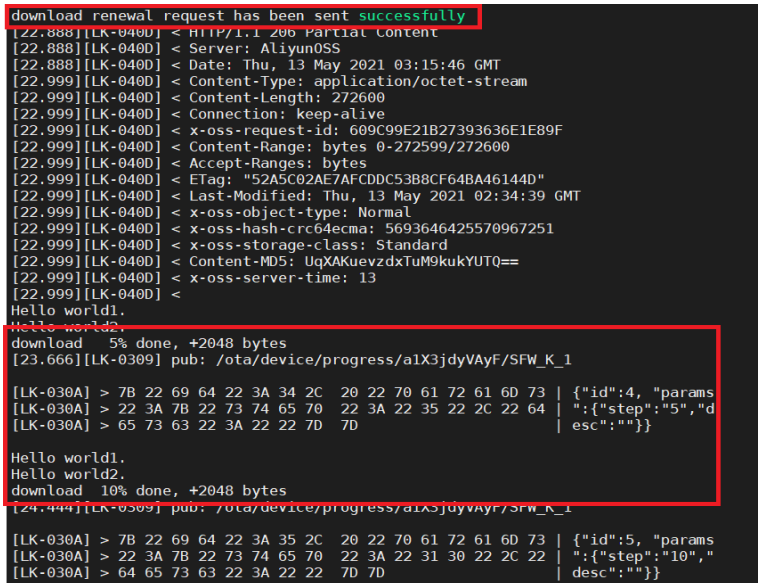


图148. 下载请求

下面显示的是下载进度完成，系统复位操作开始。

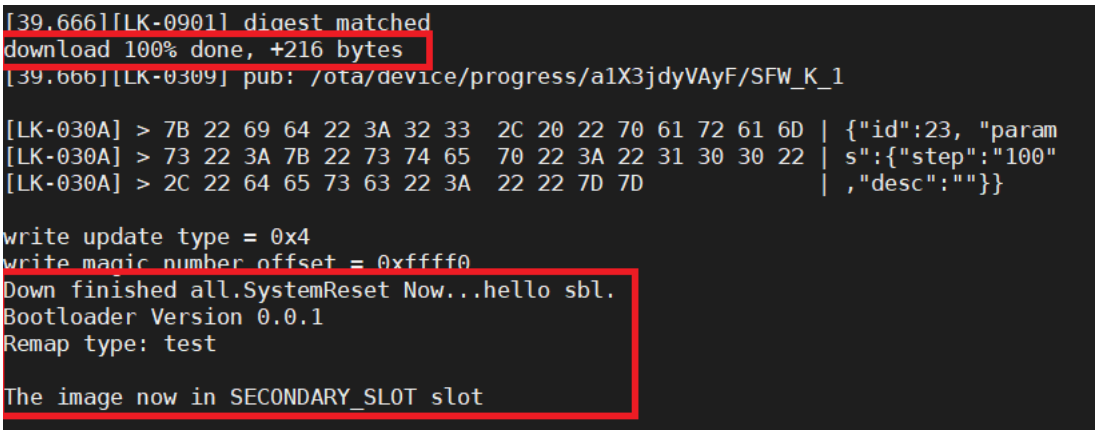


图149. 全部下载完成

下面显示当前软件包的版本是1.4.0。在阿里云网站上，OTA信息显示验证成功。

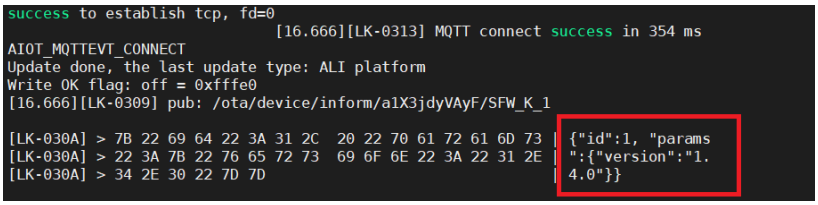


图150. 运行新工程文件

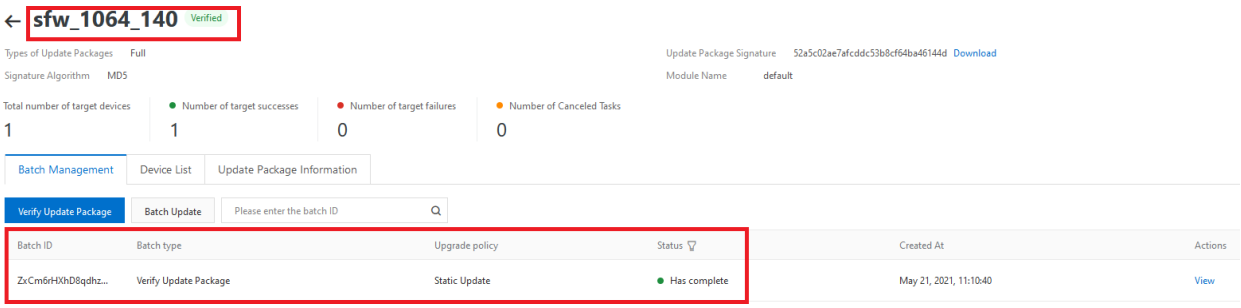


图151. 阿里云物联网平台显示OTA的成功

7.4 安全FOTA

本节介绍如何制作启用了安全功能的OTA固件的步骤。

7.4.1 EVKMIMXRTxxxx平台的安全启动演示

本节讲述如何启用安全启动和生成有签名的image的步骤。本演示以EVKMIMXRT1060硬件平台为例，这些步骤也可以应用于其他i.MX RT平台。

7.4.1.1 生成密钥和证书

要启用ROM安全启动，需要生成密钥和证书。请按以下步骤生成它们。

1. 检索并安装MCUXpresso Secure Provisioning工具
2. 运行这个工具，点击切换处理器的按钮，选择MIMXRT1060。如果要选择一个不同系列的处理器，请创建一个新的工作区。

注意

用管理员模式打开MCUXpresso Secure Provisioning工具，否则一些重要的材料将不会被生成。

注意

对于EVKMIMXRT1010平台，选择MIMXRT1015来生成密钥。

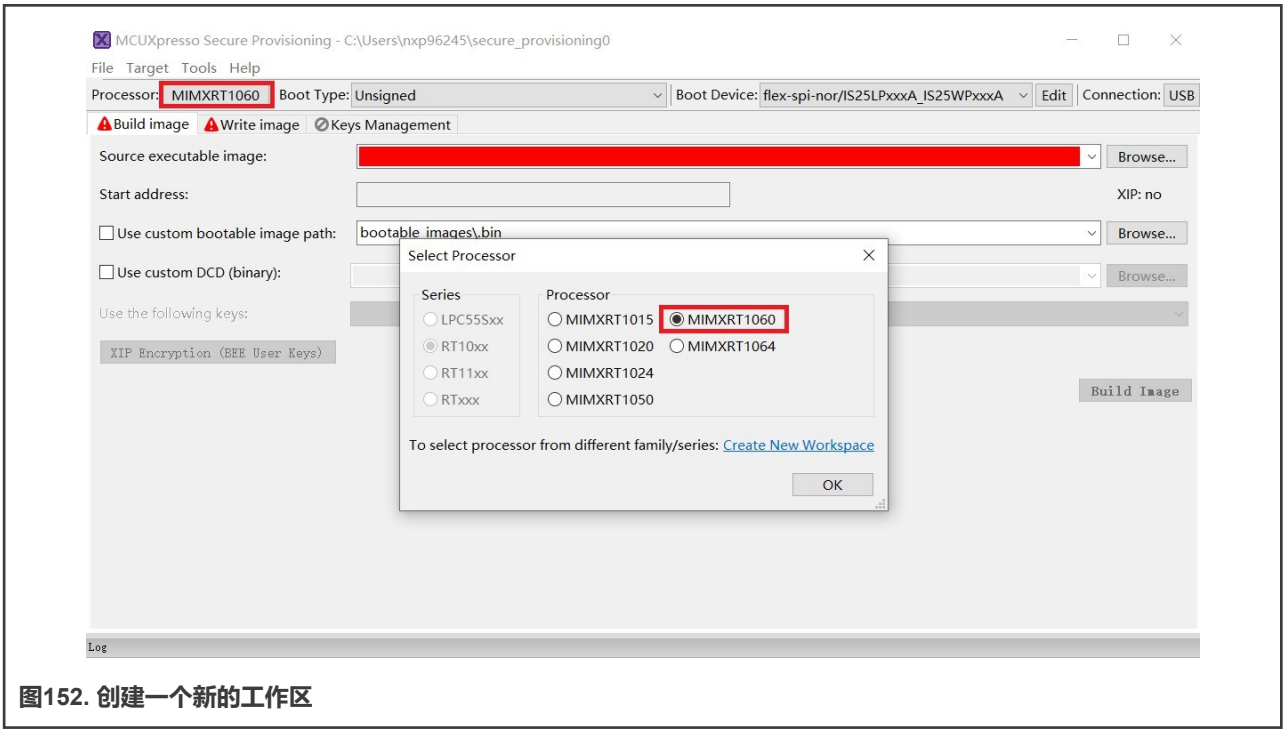


图152. 创建一个新的工作区

3. 选择启动类型为“Authenticated”（HAB）。

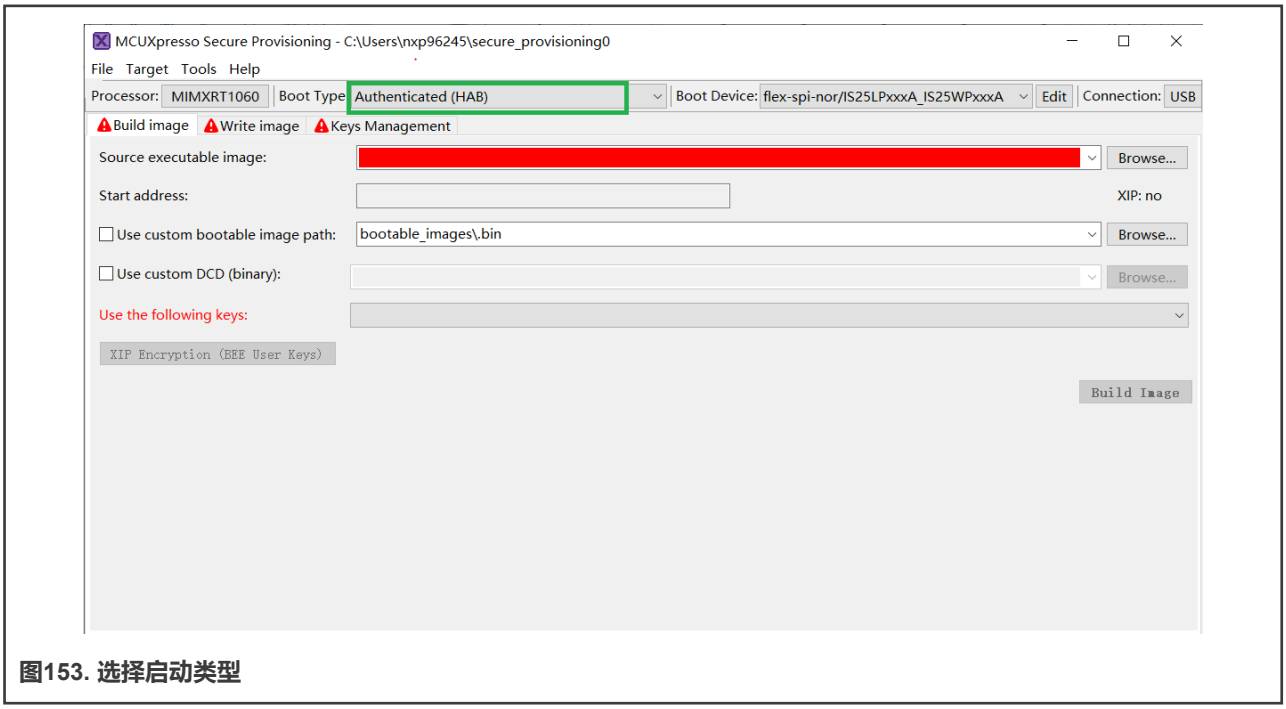


图153. 选择启动类型

4. 在“Keys Management”视图中，点击“Generated keys”，然后在该菜单中指定所有参数。

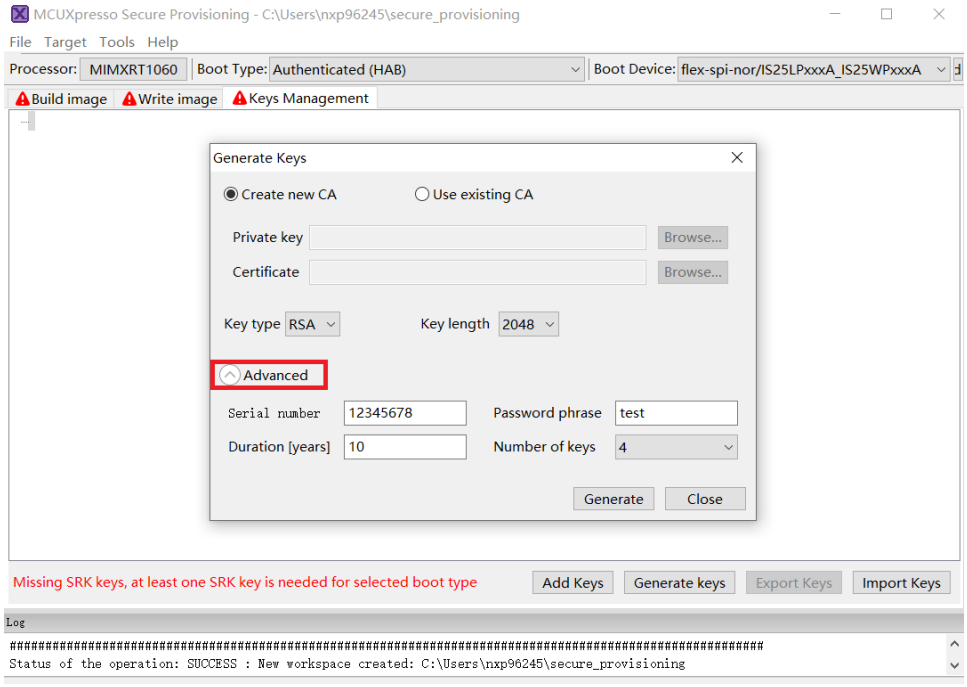


图154. 生成密钥（EVKMIMXRTRT10xx、EVKMIMXRTRT11xx）

5. 点击“Generate”，OpenSSL的输出显示在进度窗口。

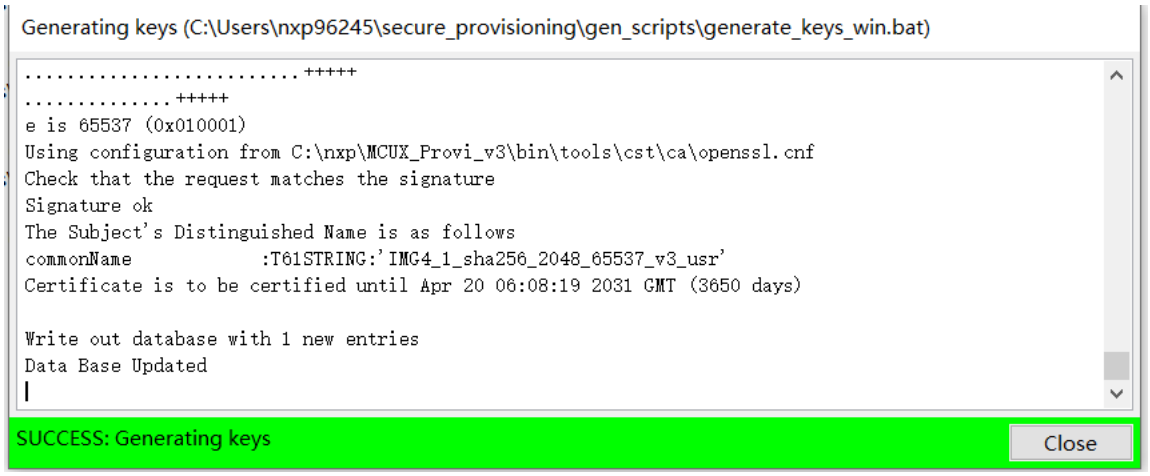


图155. 生成密钥 - 成功

6. 你可以在工作区目录下的“keys”和“crts”文件夹中找到生成的密钥和证书，将“keys”、“crts”和“gen_hab_certs”文件夹复制到 `sbl/target/evkmimxrt1060/secure` 文件夹中。

7.4.1.2 SBL镜像准备

要生成一个有签名的可启动SBL镜像，步骤如下：

1. 在目标开发板的项目文件夹下运行 `env.bat`
2. 运行 `scons --menuconfig` 进入 `MCU SBL Core` 菜单，选择 `Enable ROM` 来启用验证 `sbl`

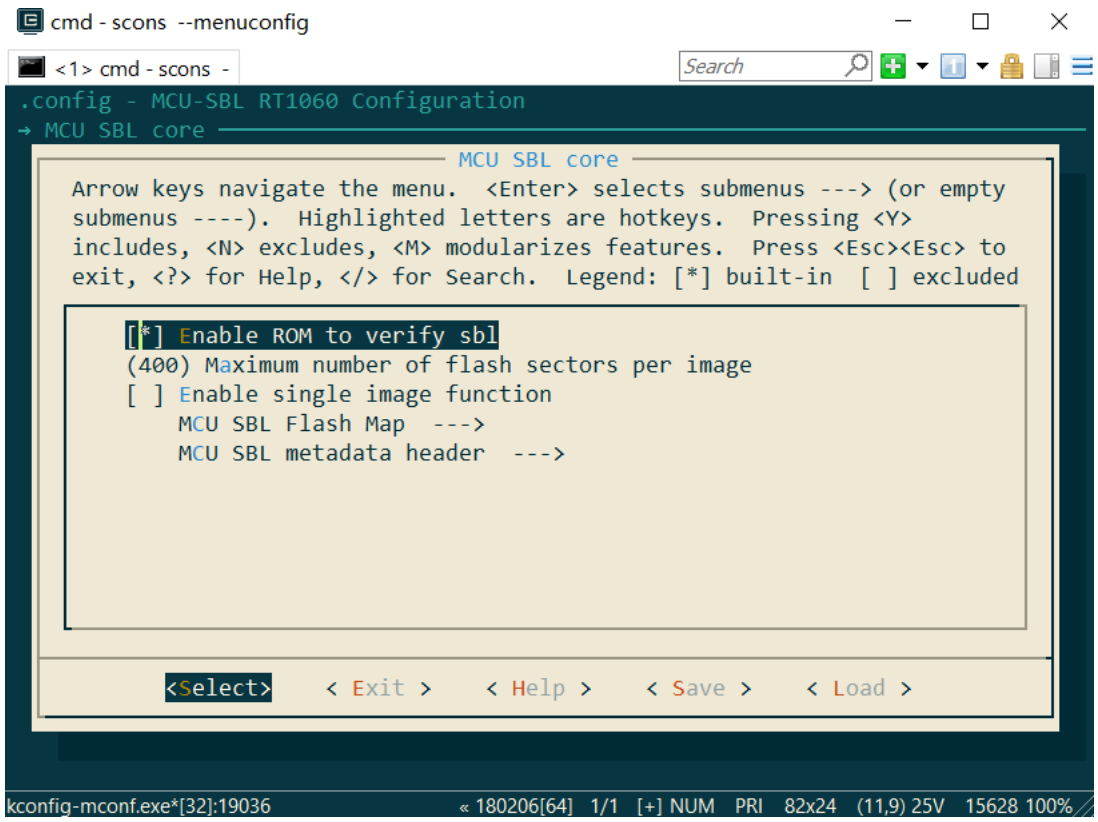


图156. 启用ROM来验证sbl

3. 进入菜单 `MCU SBL Component > secure > selected signing method`，选择一种应用程序的签名类型，保存并退出menuconfig。

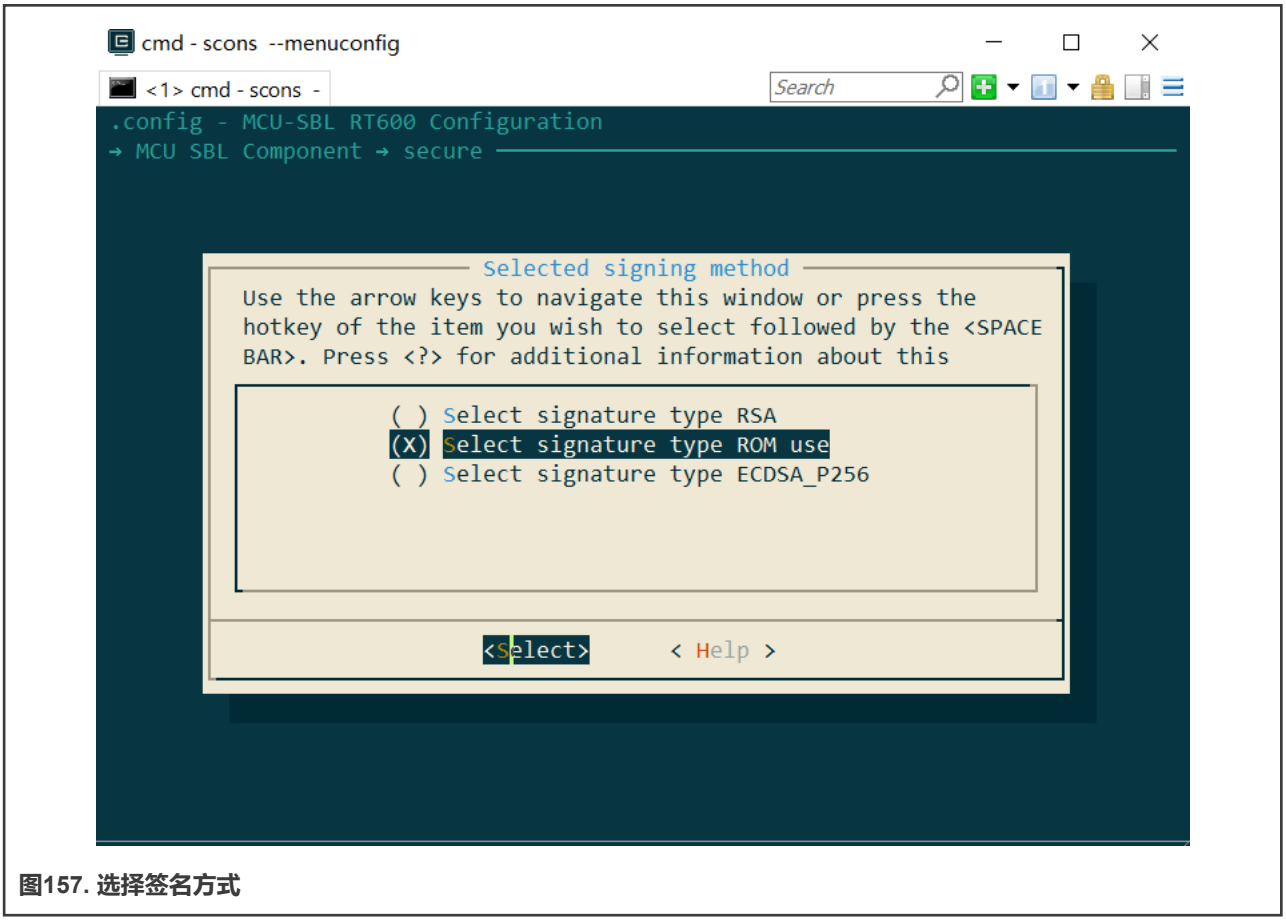


图157. 选择签名方式

4. 运行 `scons --ide=iar` 命令来生成IAR工程文件，或者运行 `scons --ide=mdk5` 来生成Keil工程文件。
5. 配置选项使IAR工程生成`.srec`格式的image，然后构建项目。对于Keil工程，用户可以通过运行以下命令生成srec格式的映像：

```
fromelf.exe --m32combined --output "$L@L.srec" "#L"
```

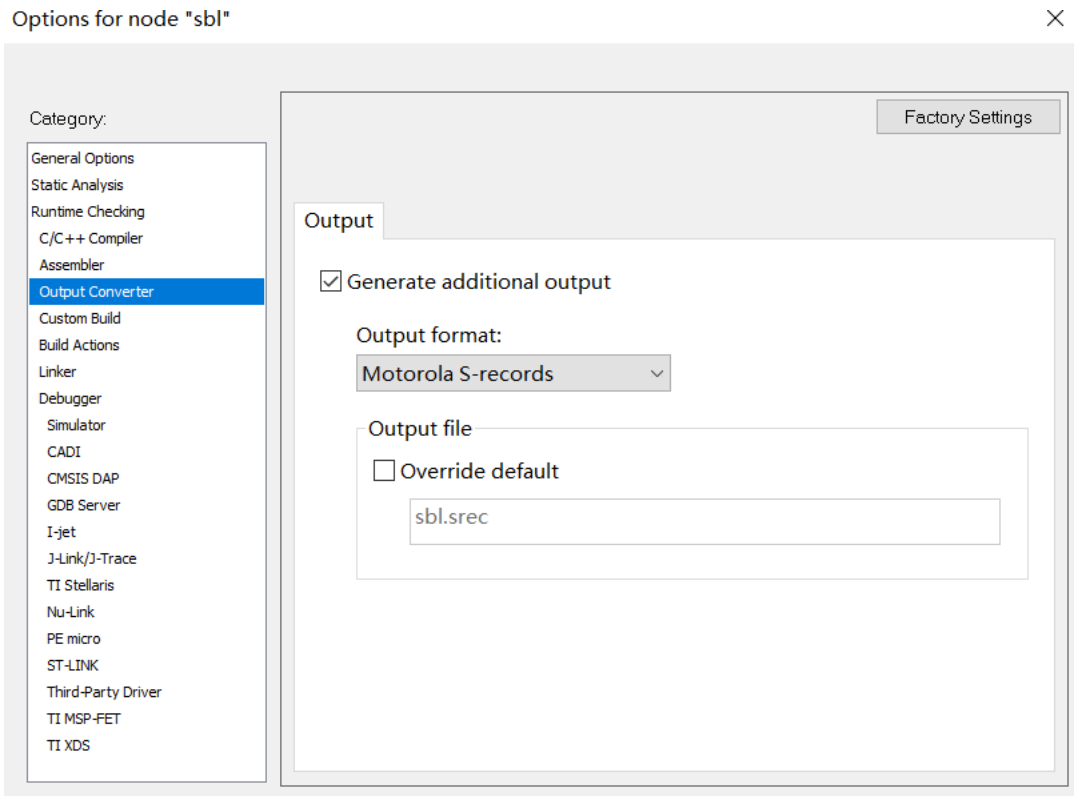


图158. 更新定义的符号

7.4.1.3 应用程序image准备

生成应用程序image的步骤如下：

1. 打开链接器文件 `sfw\target\xxxx\board\link\MIMXRTXXXX_flexspi_nor.icf`（以IAR链接器文件为例），并在链接器文件中做如下修改：

```
define symbol m_interrupts_start       = BOOT_FLASH_ACT_APP + 0x2000;
define symbol m_interrupts_end         = BOOT_FLASH_ACT_APP + 0x2000 + 0x3FF;

define symbol m_text_start             = BOOT_FLASH_ACT_APP + 0x2000 + 0x400;
define symbol m_text_end               = BOOT_FLASH_CAND_APP - 0x1000;
```

2. 在SFW的一个目标板文件夹下运行 `env.bat`
3. 进入菜单 `MCU SFW core`，不选中菜单 `Enable sfw standalone xip`，保存并退出menuconfig。

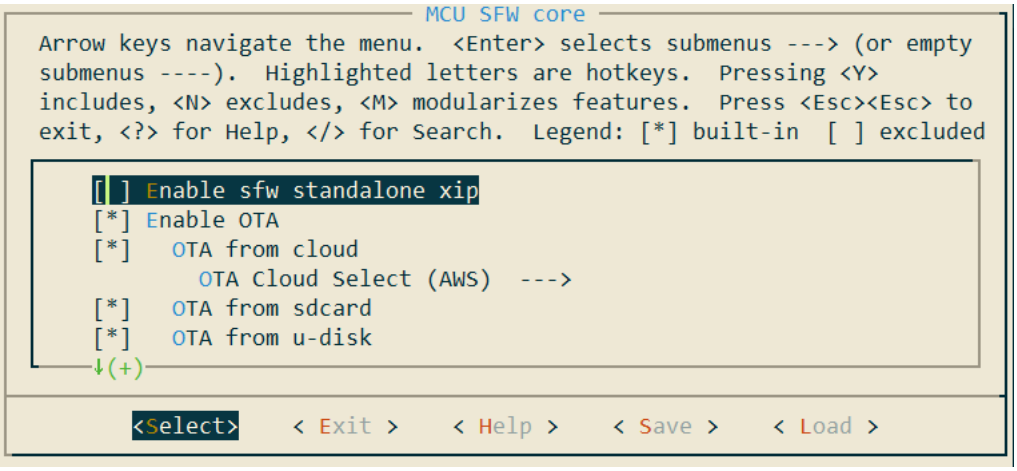


图159. 不勾选sfw单独运行

4. 运行 `scons --ide=iar` 来生成IAR工程文件，或者运行 `scons --ide=mdk5` 来生成SFW的Keil工程文件。
5. 配置选项以生成 `.srec` 格式的image，然后构建项目。

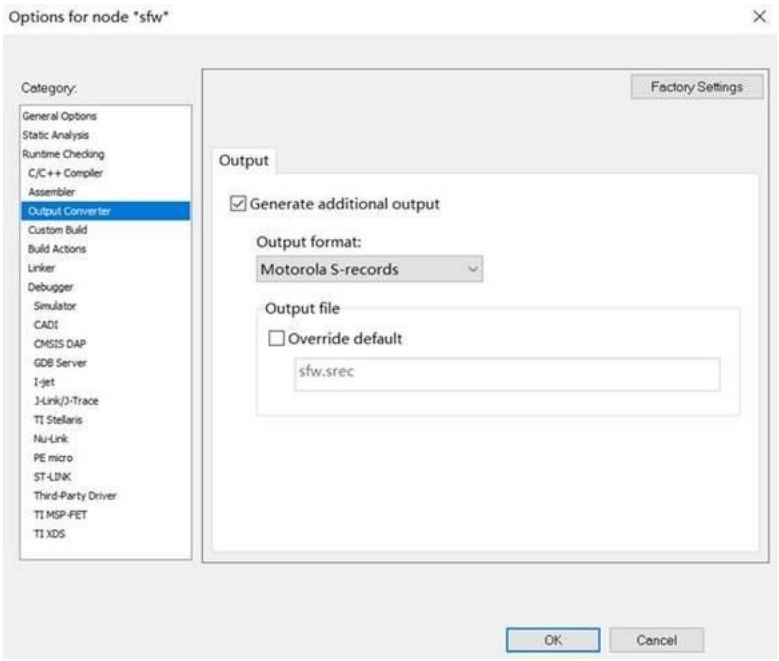


图160. 输出二进制格式

注意

对于其他签名方式，要生成一个binary格式的应用程序。

7.4.1.4 烧写OCOTP（eFuse）

下面是一个如何对SRK表进行烧写并启用HAB关闭模式的示例。

注意

在开发阶段，对于大多数用例，设备可能处于HAB打开模式下。

1. 在 sbl/target/evkmimxrt1060/secure 中找到并打开脚本 program_ocotp.bat，然后为工具elftosb、cst、blhost等设置正确的安装路径：

```
SET "PATH=C:\nxp\MCUX_Provi_v3\bin\tools\elftosb\win;%PATH%"
SET "PATH=C:\nxp\MCUX_Provi_v3\bin\tools\sdphost\win;%PATH%"
SET "PATH=C:\nxp\MCUX_Provi_v3\bin\tools\blhost\win;%PATH%"
SET "PATH=C:\nxp\MCUX_Provi_v3\bin\tools\cst\mingw32\bin;%PATH%"
```

2. 以十六进制模式查看 gen_hab_certs 文件夹下的 SRK_fuses.bin 文件。

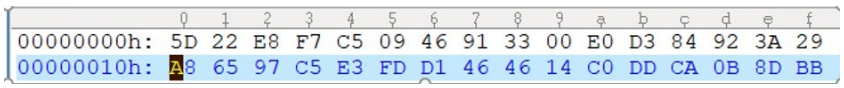


图161. 十六进制的SRK哈希值

3. 位于文件中的电子熔丝（eFuse）值将被烧写到SoC上的 SRK_HASH eFuse字段。这个哈希值必须按以下顺序（第一个字（word）写到第一个eFuse索引）烧写到SoC eFuses：

   f7e8225d, 914609c5, d3e00033, 293a9284, c59765a8, 46d1fde3, ddc01446, bb8d0bca。请注意数据的字节顺序。

   如果用户用脚本 program_ocotp.bat 对eFuses进行编程，记得要根据脚本中用户的 SRK_fuses.bin 更新SRK的哈希值。用于对eFuses进行编程的命令在默认情况下被注释掉了。需要手动启用它。对于其他平台，SRK哈希表的eFuse OCOTP索引可能不同。请参考所使用的恩智浦处理器的熔丝图。

```
blhost.exe -u 0x15A2,0x0073 -j -- efuse-program-once 0x18 f7e8225d
blhost.exe -u 0x15A2,0x0073 -j -- efuse-program-once 0x19 914609c5
blhost.exe -u 0x15A2,0x0073 -j -- efuse-program-once 0x1a d3e00033
blhost.exe -u 0x15A2,0x0073 -j -- efuse-program-once 0x1b 293a9284
blhost.exe -u 0x15A2,0x0073 -j -- efuse-program-once 0x1c c59765a8
blhost.exe -u 0x15A2,0x0073 -j -- efuse-program-once 0x1d 46d1fde3
blhost.exe -u 0x15A2,0x0073 -j -- efuse-program-once 0x1e ddc01446
blhost.exe -u 0x15A2,0x0073 -j -- efuse-program-once 0x1f bb8d0bca
```

4. 使用以下命令启用HAB关闭模式。在生产阶段，需要启用HAB关闭模式并签名SBL image。

   ```
   blhost.exe -u 0x15A2,0x0073 -j -- efuse-program-once 0x06 00000002
   ```

5. 通过命令efuse-read-once来验证eFuse的值。

   ```
   blhost.exe -u 0x15A2,0x0073 -j -- efuse-read-once 0x18
   ```

6. 把在EVKMIMXRT1060电路板上SW7-4开关置于“开”，进入串行下载模式，然后运行脚本 program_ocotp.bat。

7.4.1.5 运行SBL和应用程序

要生成签名过的SBL和应用程序，请按照以下步骤操作。然后将它们烧写到目标板上。

1. 将 sbl.srec 和 sfw.srec 复制到 sbl/target/evkmimxrt1060/secure 文件夹中。如果你选择RSA或ECDSA签名类型，请使用 sfw.bin。
2. 使板子EVKMIMXRT1060进入Serial Downloader模式，并连接USB OTG和DEBUG USB端口。
3. 在scons终端里输入 cd secure 进入到文件夹secure下。
4. 运行 sign_sbl_app.bat 来生成最终签名的SBL和应用程序。如果想生成用于升级的应用程序image，记得在imgtool.py命令行中修改参数版本号。
5. 通过脚本 sign_sbl_app.bat 或其他工具下载签名的SBL和应用程序。这个脚本将应用程序的image下载到slot1。如果之前测试过OTA功能，SBL可能仍会启动slot2中的应用程序，而不是运行刚下载到slot1中的程序。

6. 切换（SW7-3 开，SW7-4 关）到正常的启动模式，然后复位电路板。你可以在终端看到图示的输出。

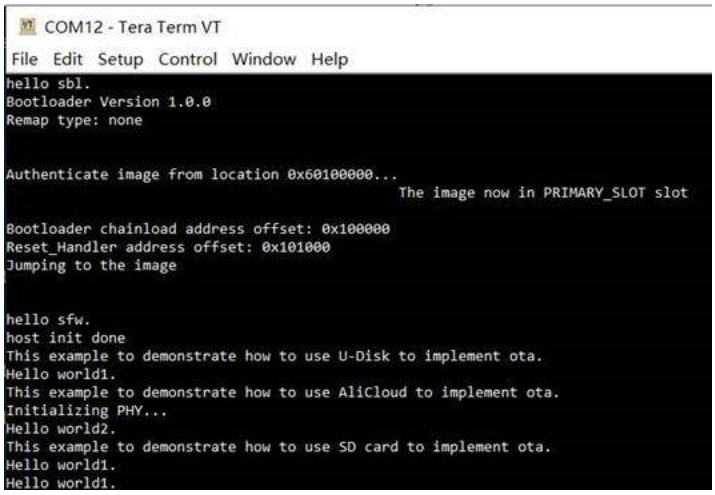


图162. 终端日志

7.4.2 EVKMXRTxxxx平台的加密XIP启动演示

本节讲述启用XIP加密验签启动的步骤。本演示以EVKMIMXRT1060硬件平台为例。

7.4.2.1 SBL镜像准备

要生成签名的可启动SBL镜像，步骤如下：

1. 在目标板文件夹下运行`env.bat`。
2. 运行 `scons --menuconfig`，以进入菜单“MCU SBL Core”，选择“Enable ROM to verify sbl”。

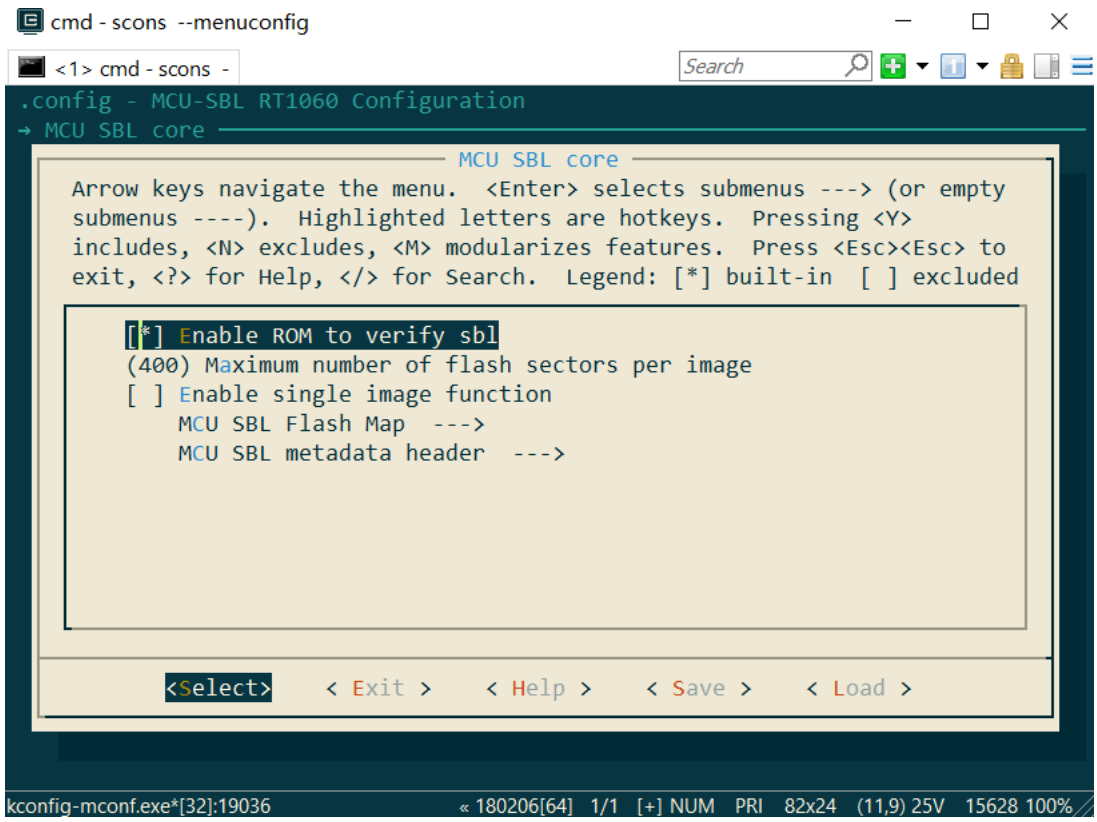


图163. 启用ROM来验证sbl

3. 进入菜单 `MCU SBL Component > secure`，选择“Encrypted XIP”功能，保存并退出menuconfig。

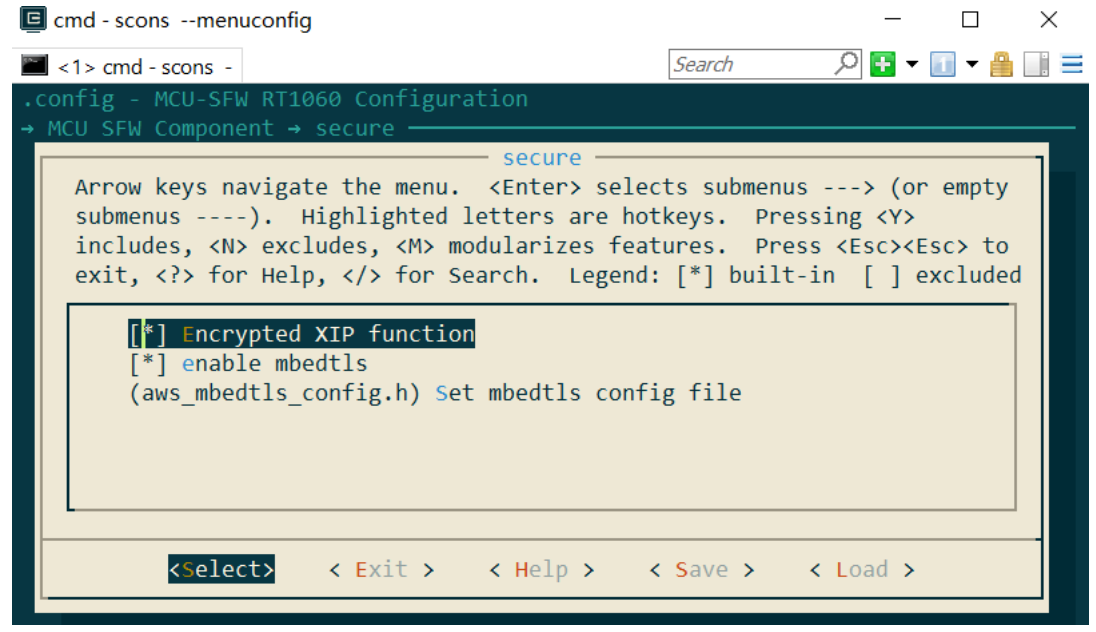


图164. 启用SBL中的密钥上下文升级

4. 运行 `scons --ide=iar` 命令来生成IAR工程文件。

5. 配置工程以生成.srec 格式的image，然后构建项目。对于Keil工程，通过运行 `fromelf.exe --m32combined --output "$L@L.srec"`来生成srec格式的image。

7.4.2.2 应用程序镜像准备

要生成用于加密的应用程序image，步骤如下：

1. 打开链接器文件 `sfw\target\xxxx\board\link\MIMXRTXXXX_flexspi_nor.icf`（以IAR链接器文件为例），对链接器文件进行如下修改。

```
define symbol m_interrupts_start       = BOOT_FLASH_ACT_APP + 0x2000;
define symbol m_interrupts_end         = BOOT_FLASH_ACT_APP + 0x2000 + 0x3FF;

define symbol m_text_start             = BOOT_FLASH_ACT_APP + 0x2000 + 0x400;
define symbol m_text_end               = BOOT_FLASH_CAND_APP - 0x1000;
```

2. 在目标平台文件夹下运行 `env.bat`
3. 进入菜单 `MCU SFW core`，不勾选菜单 `Enable sfw standalone xip`。
4. 进入菜单 `MCU SFW component > secure`，勾选菜单 `Encrypted XIP function`，然后保存并退出menuconfig。

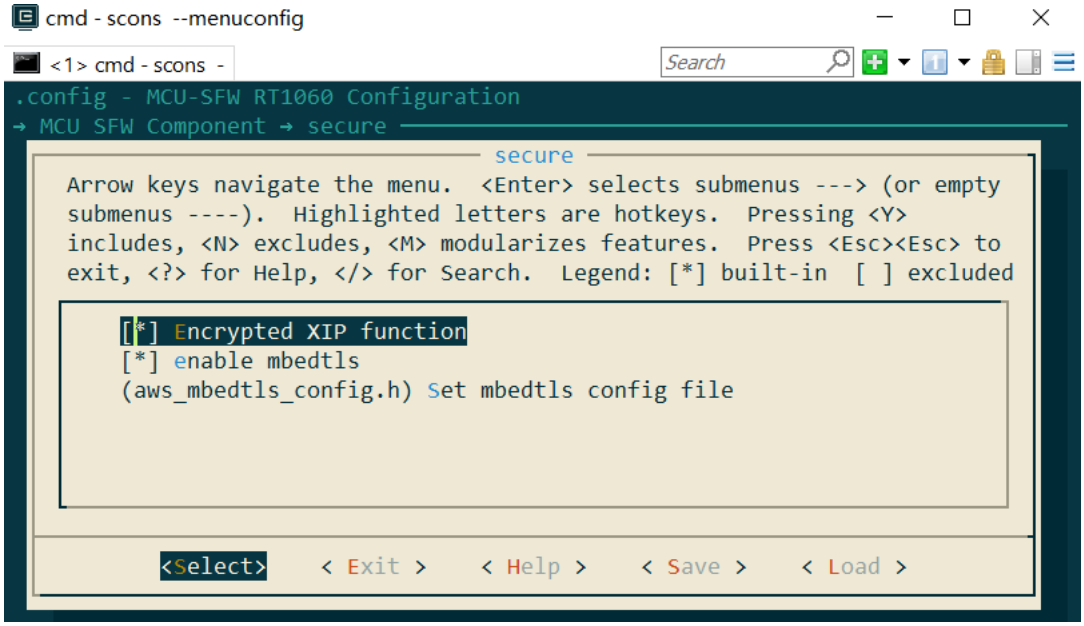


图165. 在SFW中启用密钥升级功能

5. 生成SFW工程文件。
6. 如果应用程序支持OTA，请在函数 `enable_image()`之后调用函数 `update_key_context()`。
7. 配置选项，以生成srec格式的映像，然后构建项目。

注意

对于RSA或ECDSA签名方式，在步骤1中不要改变链接器文件，需要生成binary格式的映像。

7.4.2.3 烧写KEK（eFuse）

KEK（加密密钥的密钥）作为BEE的输入密钥用来解密KIB。

在测试阶段，你可以不写KEK用efuse默认值0来做KEK。下面是一个示例，如何为EVKMIMXRT1060烧写KEK到efuse SW_GP2，将下面的命令添加到 `program_ocotp.bat` 文件中。

```
 :: kek=00112233445566778899aabbccddeeff
blhost.exe -u 0x15A2,0x0073 -j -- efuse-program-once 0x29 ccddeeff
blhost.exe -u 0x15A2,0x0073 -j -- efuse-program-once 0x2a 8899aabb
blhost.exe -u 0x15A2,0x0073 -j -- efuse-program-once 0x2b 44556677
blhost.exe -u 0x15A2,0x0073 -j -- efuse-program-once 0x2c 00112233
```

KEK作为EVKMIMXRT1010或EVKMIMXRT1170中的OTFAD密钥。要按照如下所示的字节序烧写：`0xffeeddcc,0xbbaa9988,0x77665544,0x33221100`。

7.4.2.4 运行SBL和应用程序

按照以下步骤，生成签名和加密的SBL和应用程序。

1. 将 `sbl.srec` 和 `sfw.srec` 复制到 `sbl/target/evkmimxrt1060/secure` 文件夹中。
2. 编辑 `sign_enc_sbl_app.bat` 并设置KEK和加密的区域。
3. 使EVKMIMXRT1060板进入Serial Downloader模式，连接USB OTG和DEBUG USB端口。
4. 运行 `sign_enc_sbl_app.bat` 来生成最终签名加密的SBL和应用程序1。
5. 通过这个脚本或其他工具下载SBL和应用程序。
6. 切换（SW7-3 ON，SW7-4 OFF）到正常启动模式，设置SW5-1为ON，然后复位电路板。可以在终端看到如下输出。

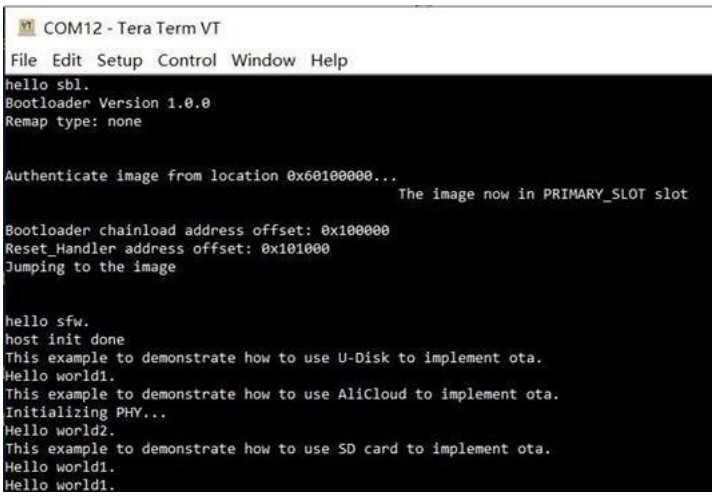


图166. 终端日志

7.4.2.5 应用程序OTA镜像准备

要生成经过签名和加密的升级应用程序，请执行以下步骤：

1. 按照 7.4.2.2 节所述生成工程文件。
2. 构建镜像并将其重命名为 `sfw2.srec` 或 `sfw2.bin`
3. 打开脚本 `sign_enc_sfw2.bat`，设置KEK和加密区域。
4. 运行 `sign_enc_sfw2.bat`，文件 `sfw_2_enc.bin` 是最终使用的镜像。

7.4.3 LPC55S69平台的安全启动演示

本节讲述启用XIP加密验证启动的步骤。本演示以LPC55S69（1B版本）硬件平台为例。

7.4.3.1 生成密钥和证书

要启用ROM安全启动，需要生成密钥和证书。请按照以下步骤生成它们。

1. 安装MCUXpresso Secure Provisioning工具。
2. 运行这个工具，点击 `File > New Workspace`，然后选择处理器LPC55S69，以创建一个新的工作区。

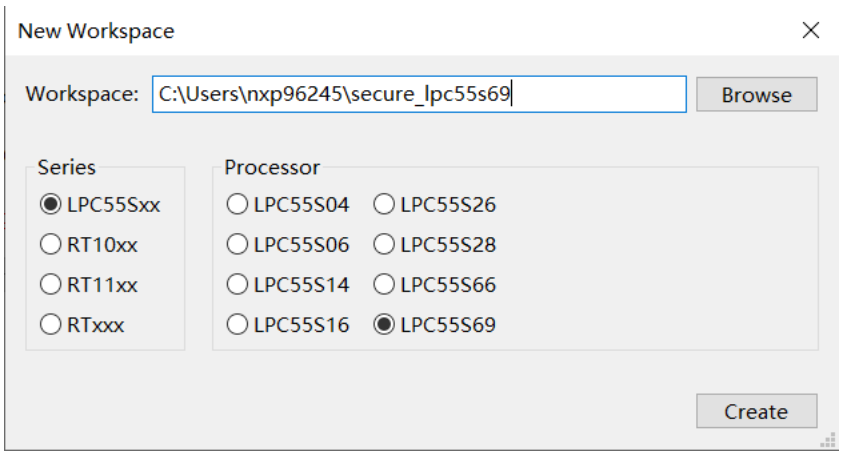


图167. 创建一个新的工作区

3. 选择启动类型为“Encrypted (PRINCE) and Signed”。

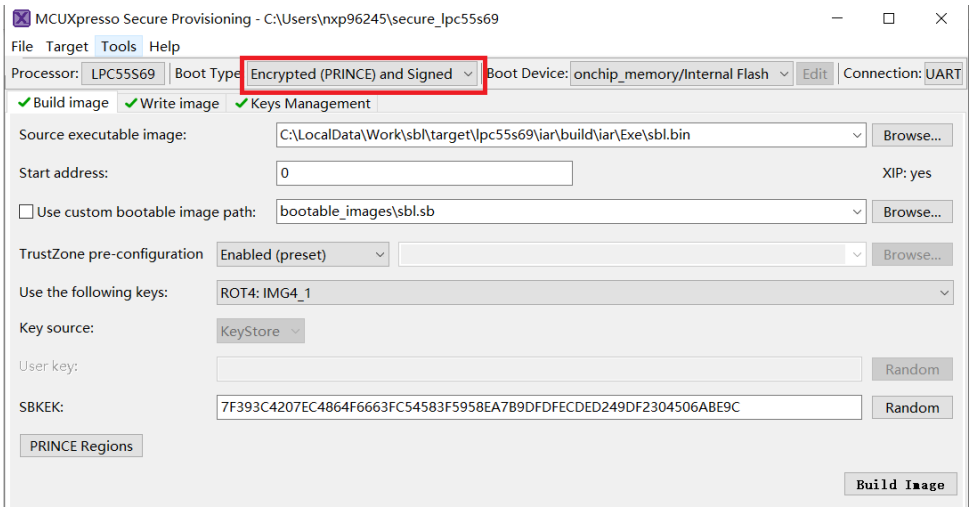


图168. 选择启动类型

4. 在“Keys Management”视图中，点击“Generated keys”按钮，然后在该菜单中指定所有参数。

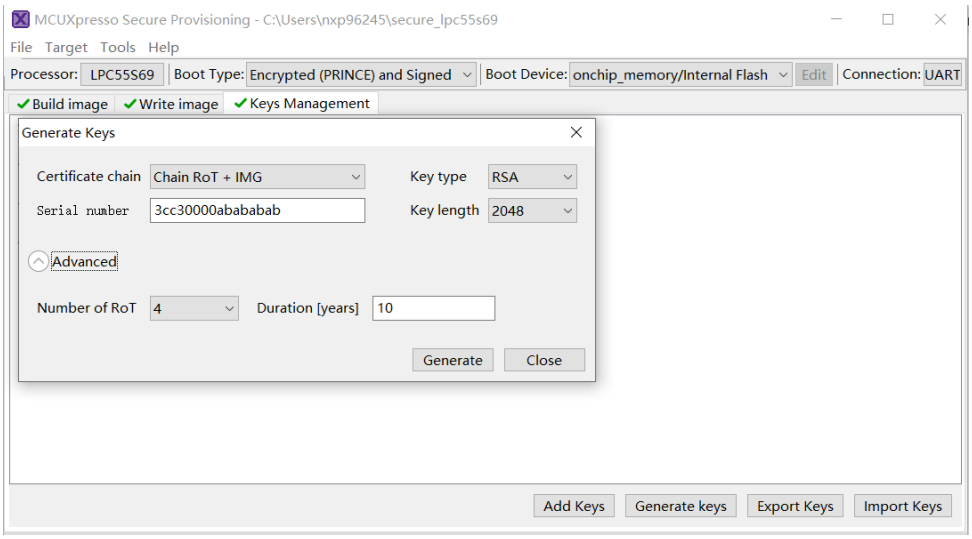


图169. 图7-141：生成密钥

5. 点击“Generate”按钮，OpenSSL输出显示在进度窗口中。

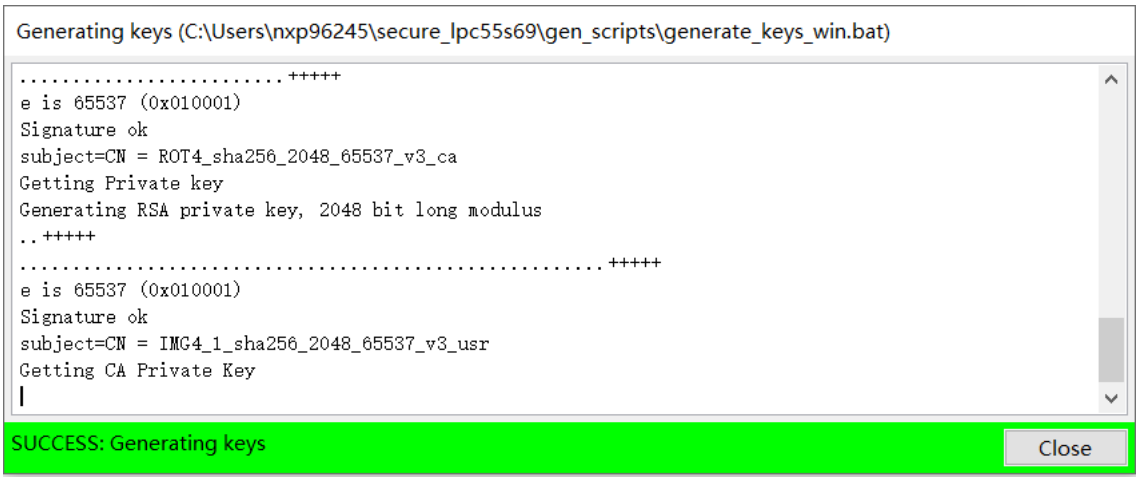


图170. 生成密钥 - 成功

6. 你可以在工作区目录下的 `keys` 和 `crts` 文件夹中找到生成的密钥和证书。

7.4.3.2 SBL映像准备

下面的步骤描述创建SBL镜像的过程。

1. 在目标板文件夹下运行 `env.bat`。
2. 进入菜单 `MCU SFW Core >`，选择一种应用程序签名类型，保存并退出menuconfig。

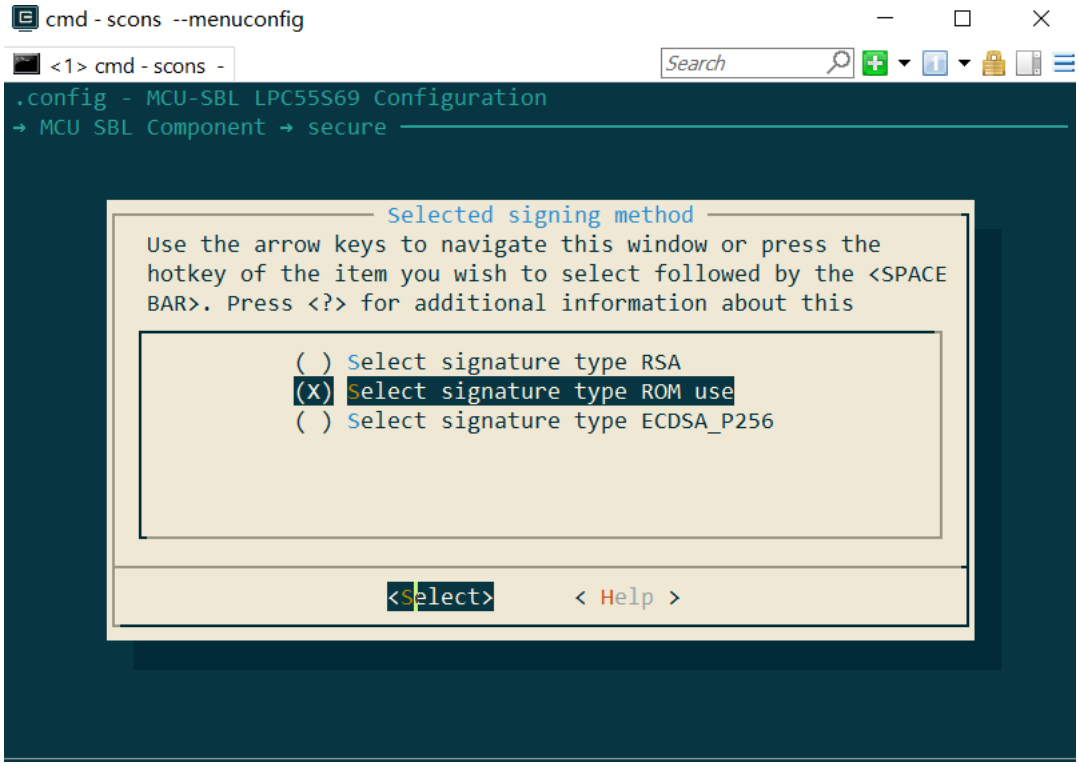


图171. 选择签名方式

3. 运行 `scons --ide=iar` 命令来生成IAR工程文件。
4. 用IDE构建binary镜像，它是没有签名加密的普通镜像。

7.4.3.3 应用程序镜像准备

请按照下面的步骤创建签名镜像。

1. 在SFW repo下的目标板文件夹下运行 `env.bat`
2. 进入菜单 `MCU SFW core`，不勾选菜单 `Enable sfw standalone xip`，保存并退出menuconfig。

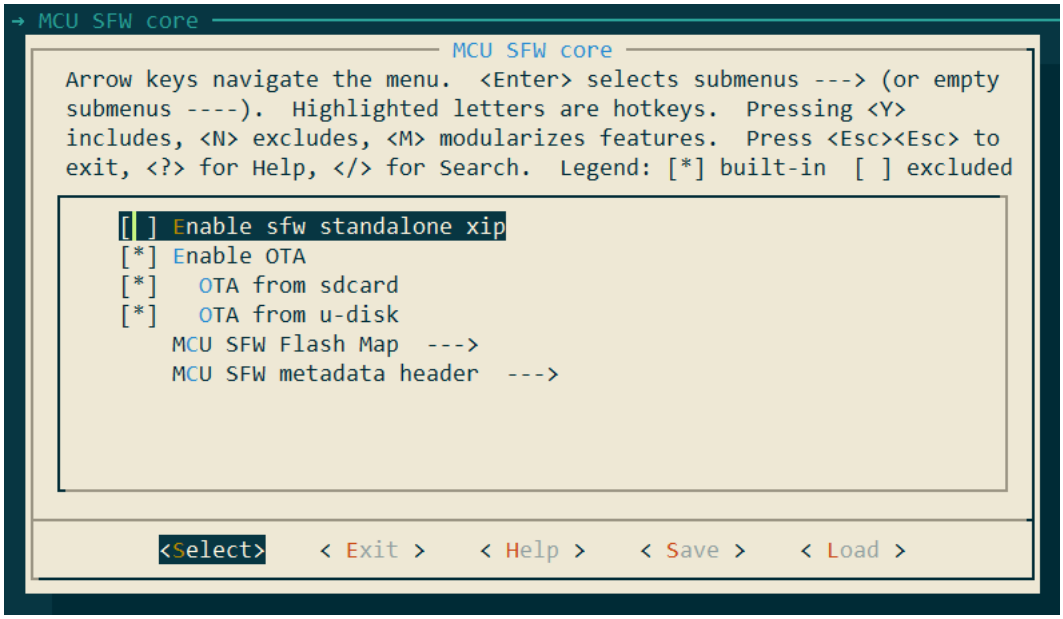


图172. 不勾选启用sfw单机版

3. 按照下面的示例，生成一个IAR工程文件：

   ```
   scons --ide=iar
   ```

4. 在 `Project > Options > Output Converter` 中，勾选“Generate additional output”并选择 `Raw binary` output format。

5. 构建该工程。

7.4.3.4 签名和编程加密的SBL

本节讲述如何使用工具MCUXpresso Secure Provisioning Tool (SPT) 构建和编程验证的SBL映像。

1. 在“Build Image”视图中，在“Source executable image”框中选择印象。
2. 在“Start address”标签中，输入0。
3. 在 TrustZone pre-configuration 框中选择 Enabled(present)。
4. 对于“use the following keys”，任意选择一组密钥链，例如 `ROT1: IMG1_1`。
5. 输入SBKEK或用“Random”按钮生成一个。

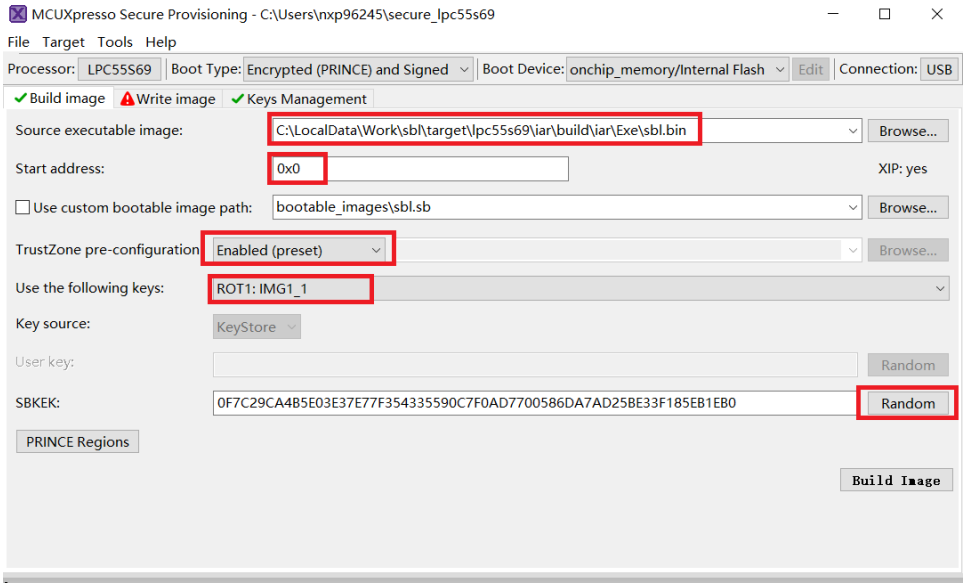


图173. 为构建映像设置参数

6. 配置PRINCE。在PRINCE区域设置SBL image的加密长度。如果不加密image，请跳过这一步。在“Boot Type”中选择“Signed”。

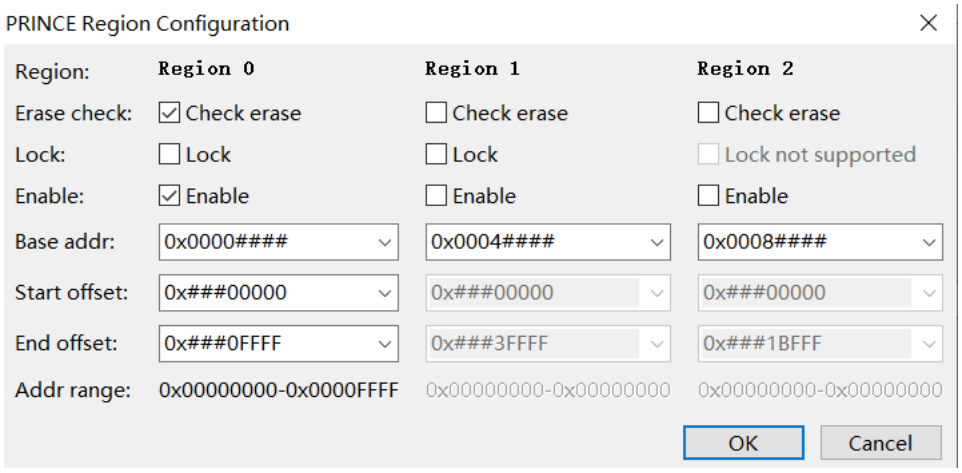


图174. 设置加密区域

7. 点击“Build image”按钮，构建镜像的结果会显示在进度窗口中。

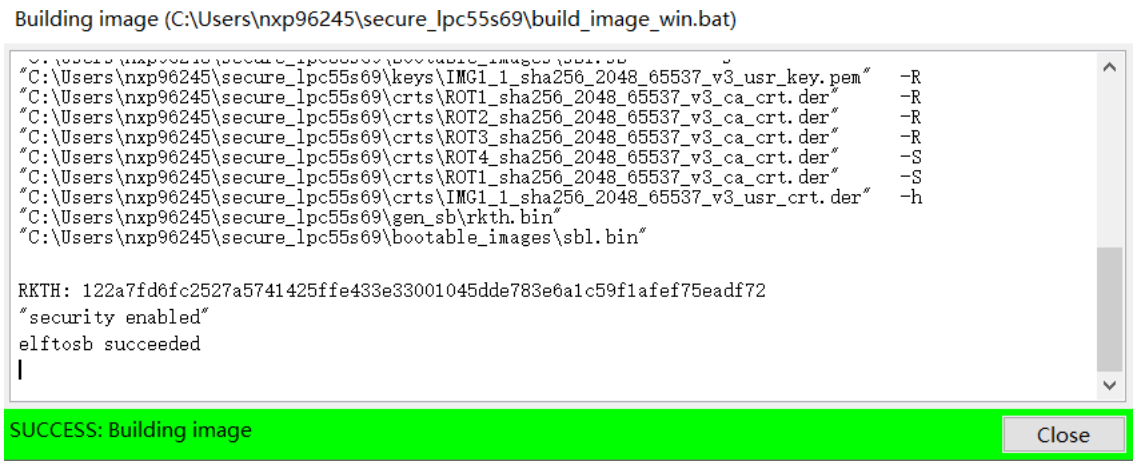


图175. 生成启动映像

8. 确保电路板已经连接好，按下开发板ISP按键同时复位开发板，使处理器进入ISP模式。
9. 点击“USB”标签，根据所选端口将连接设置为USB或UART，并测试与处理器的连接。

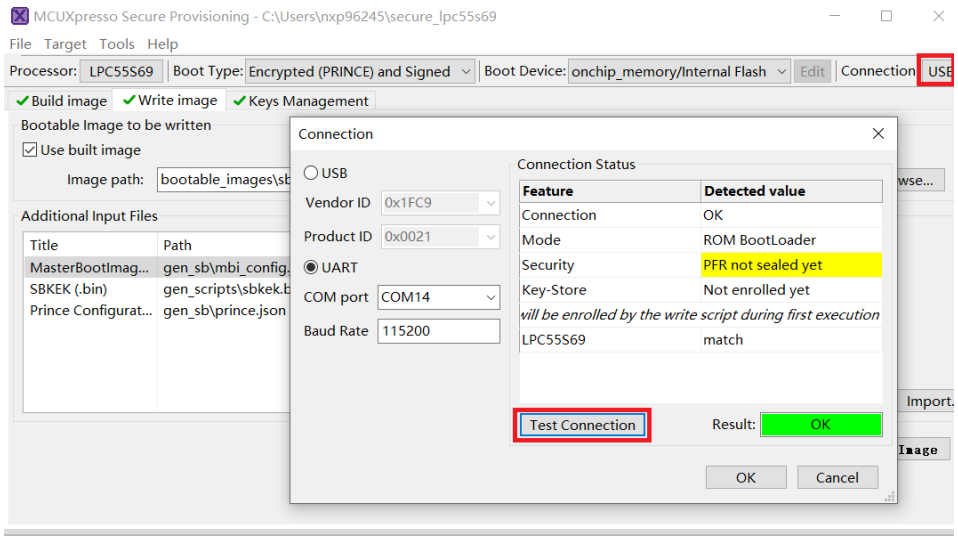


图176. 连接对话框

10. 确保“Use built image”的复选框被选中。
11. 点击“Write Image”。

注意

勾选“Enable security”（启用安全）复选框会使芯片锁住CMPA区域。

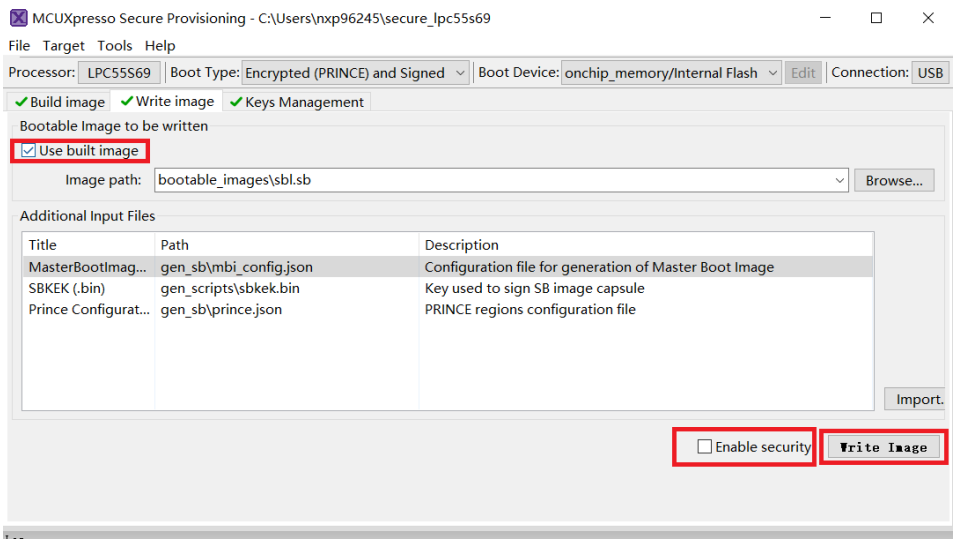


图177. 编程SBL image

12. 进度窗口会显示构建image的过程。

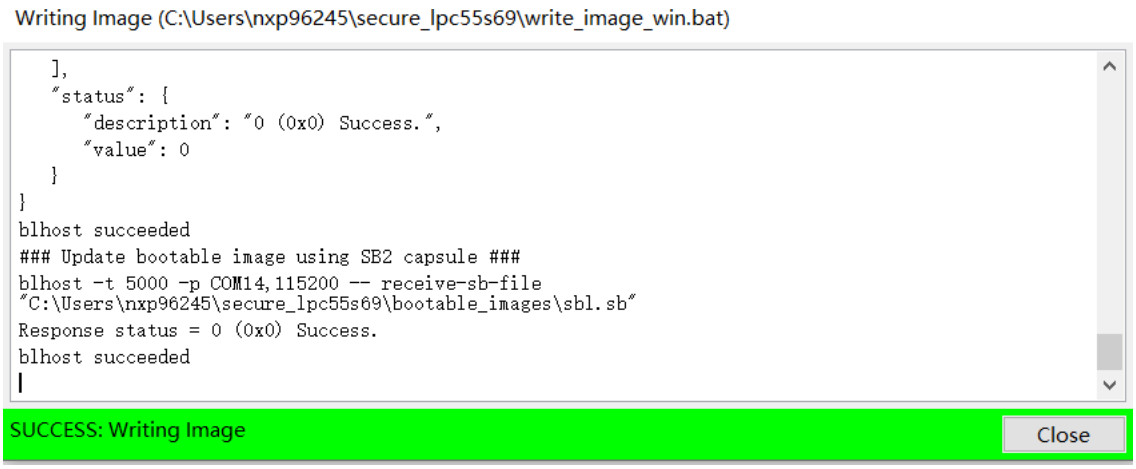


图178. 写入映像信息 - 成功

7.4.3.5 签名和烧写应用程序image

可以通过BootRom的ISP模式烧写SFW印象，也可以通过SBL的ISP模式烧写。按照以下步骤生成一个签名的SFW映像：

1. 将 `sfw.bin` 复制到 *sbl/target/lpc55s69/secure* 文件夹中。
2. 编辑 `mbi_config.json` 以设置正确的密钥和证书路径，或从MCUXpresso Secure Provisioning lpc55s69工作区复制 `keys` 和 `crts` 到这里。

```
"rootCertificate0File": "./crts/ROT1_sha256_2048_65537_v3_ca_crt.der",
"rootCertificate1File": "./crts/ROT2_sha256_2048_65537_v3_ca_crt.der",
"rootCertificate2File": "./crts/ROT3_sha256_2048_65537_v3_ca_crt.der",
"rootCertificate3File": "./crts/ROT4_sha256_2048_65537_v3_ca_crt.der",
"mainCertPrivateKeyFile": "./keys/IMG1_1_sha256_2048_65537_v3_usr_key.pem",
"chainCertificate0File0": "./crts/ROT1_sha256_2048_65537_v3_ca_crt.der",
"chainCertificate0File1": "./crts/IMG1_1_sha256_2048_65537_v3_usr_crt.der"
```

3. 打开 `signed_enc_sfw.bat`，为工具 `elftosb` 和 `blhost` 设置正确的安装路径。

```
SET "PATH=C:\nxp\MCUX_Provi_v3\bin\tools\elftosb\win;%PATH%"
SET "PATH=C:\nxp\MCUX_Provi_v3\bin\tools\blhost\win;%PATH%"
```

4. 在 `sign_enc_sfw.bat` 中设置正确的com端口并配置PRINCE区域。
5. 在scons环境中输入 `cd secure` 进入文件夹secure。
6. 运行 `sign_enc_sfw.bat` 来生成最终有签名的应用程序image。
7. 生成签名镜像后，按任意键配置PRINCE加密区域，可以修改脚本跳过配置PRINCE这一步，然后准备下载签名的映像。
8. 使LPC55S69 EVK开发板进入ISP模式。
9. 下载映像，如果写入操作成功，复位开发板。

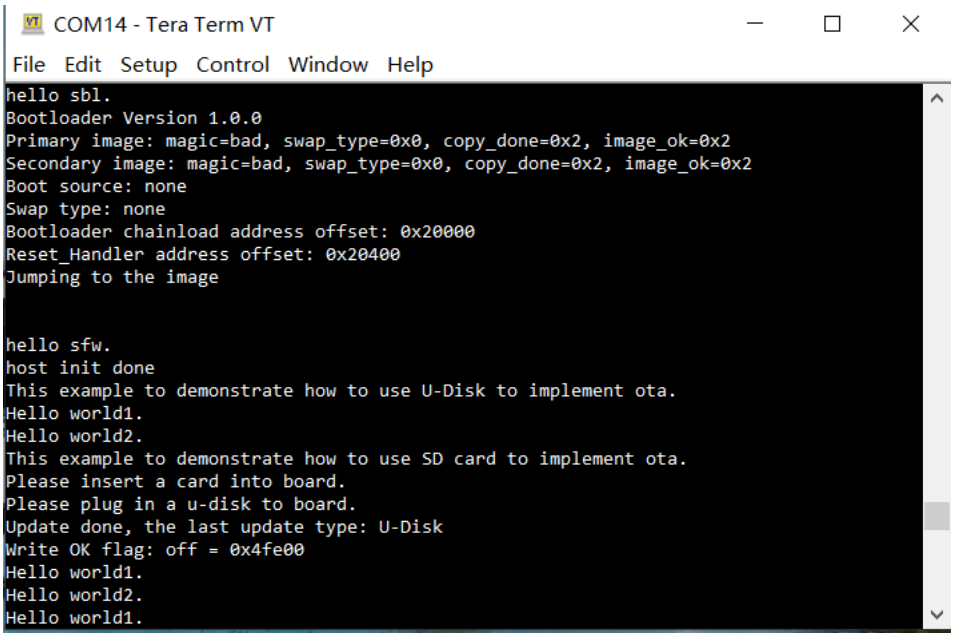


图179. 终端日志

注意

加密后的应用程序镜像需要按8k字节对齐。

7.4.4 EVKMIMXRTxxx平台的安全启动演示

本节以MIMXRT685S硬件平台作为示例介绍启用XIP签名加密的步骤。本节也适用于MIMXRT595S平台。

7.4.4.1 生成密钥和证书

要启用ROM安全启动，需要生成密钥和证书。请按照以下步骤生成它们。

1. 安装MCUXpresso Secure Provisioning工具。
2. 运行这个工具，点击 `File > New Workspace`，然后选择处理器MIMXRT685来创建一个新工作区。

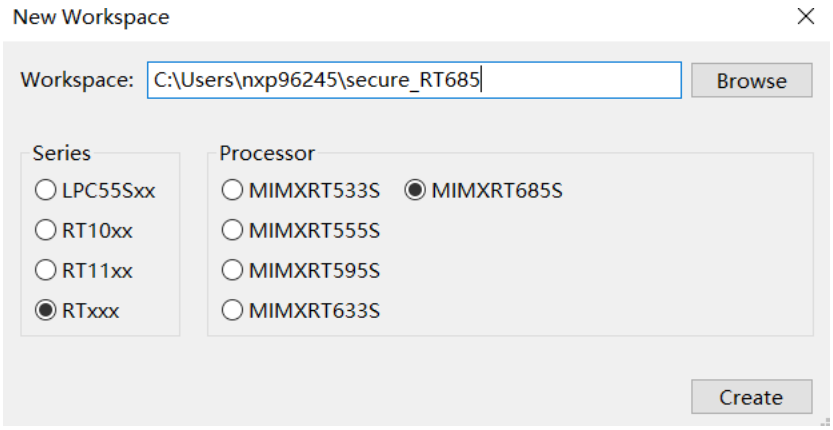


图180. 创建一个新的工作区

3. 选择“Boot Type”为“Signed”。

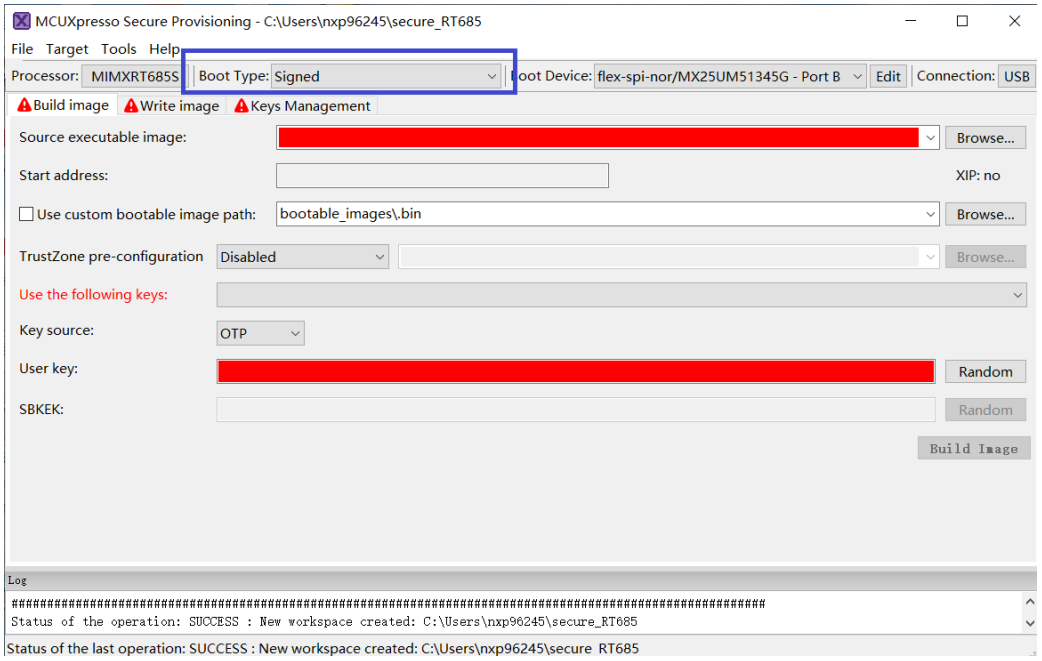


图181. 选择启动类型

4. 在“Keys Management”视图中，点击“Generated keys”按钮，然后在该菜单中设置参数。

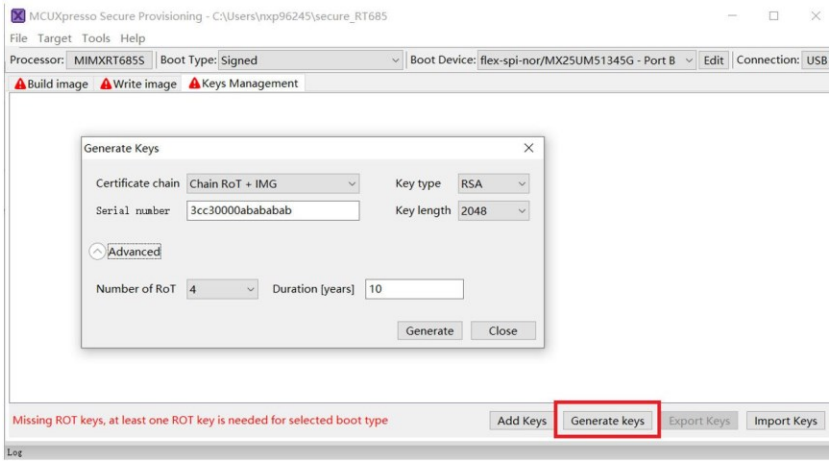


图182. 生成密钥

5. 点击“Generate”按钮。OpenSSL输出会显示在进度窗口中。

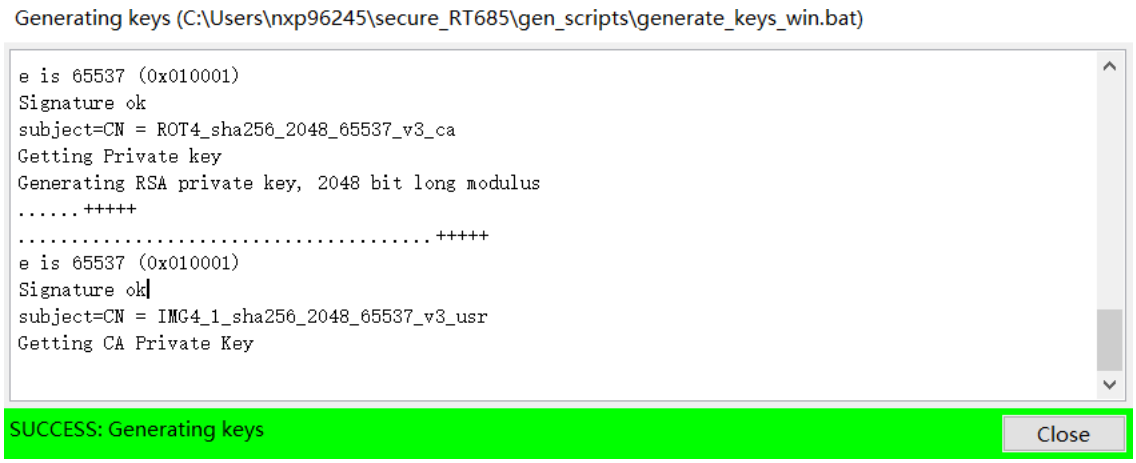


图183. 生成密钥 - 成功

6. 你可以在工作区目录的 `keys` 和 `crts` 文件夹中找到生成的钥匙和证书。将文件夹“keys”和“crts”复制到文件夹 `sbl/target/evkmimxrt600/secure`。

7.4.4.2 SBL镜像准备

下面描述创建SBL镜像的步骤。

1. 运行 `sbl/target/evkmimxrt600` 文件夹下的 `env.bat`。
2. 运行 `scons –menuconfig` 以进入菜单 `MCU SBL Core`，然后选择 `Enable ROM to verify sbl`。

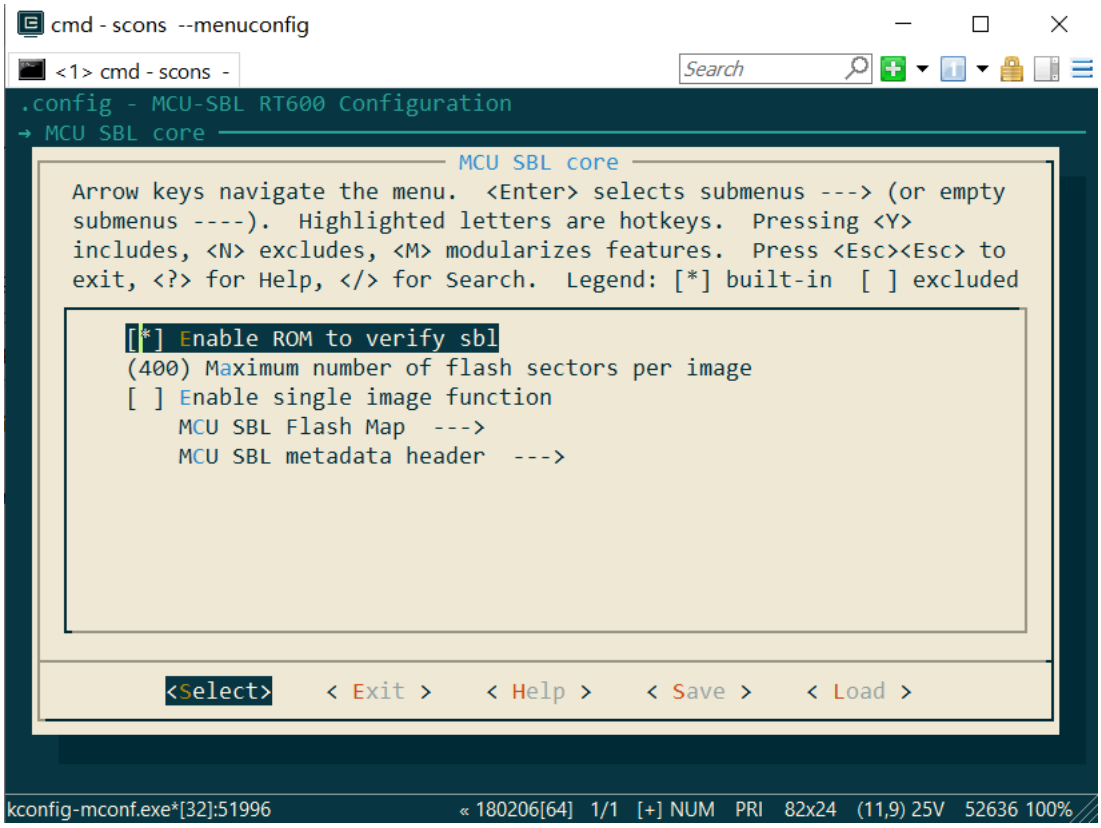


图184. 启用ROM来验证sbl

3. 进入菜单 `MCU SBL Component > secure > selected signing method`，选择一种应用程序的签名类型，保存并退出menuconfig。

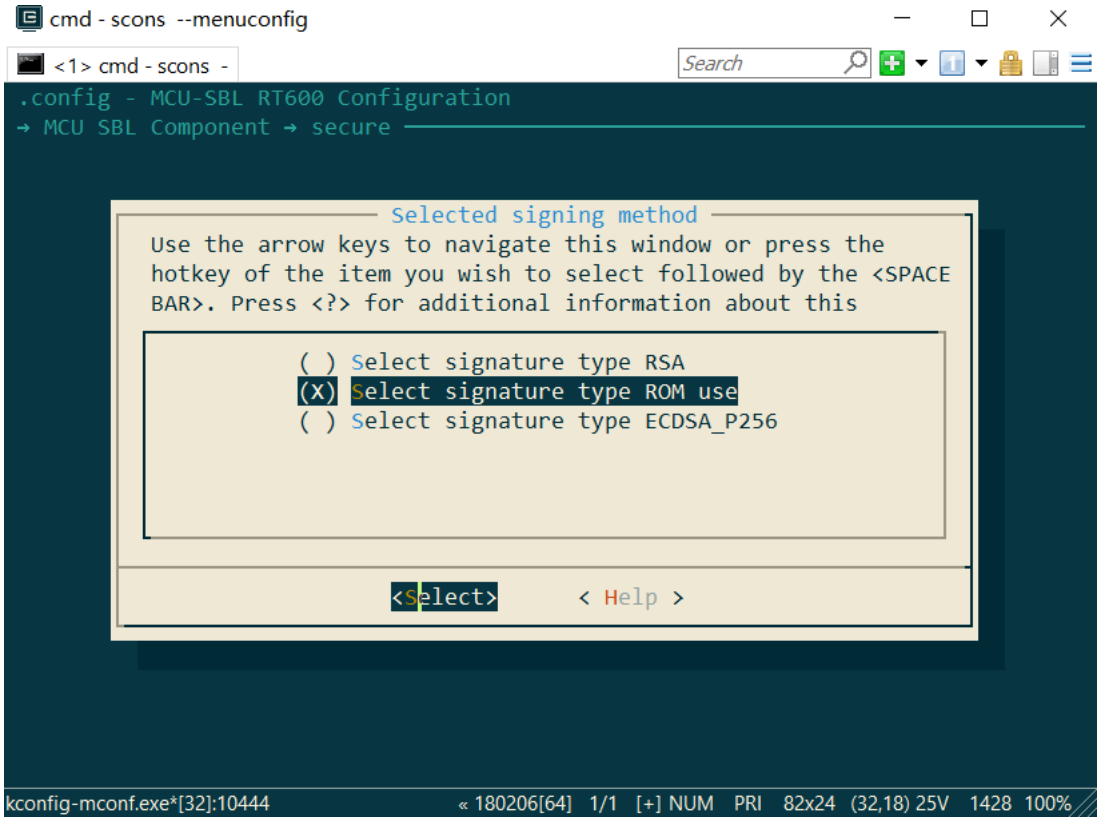


图185. 选择签名方式

4. 运行 `scons --ide=iar` 命令来生成IAR工程文件，或者运行 `scons --ide=mdk5` 来生成Keil工程文件。
5. 配置选项以生成binary格式的image，然后构建工程。

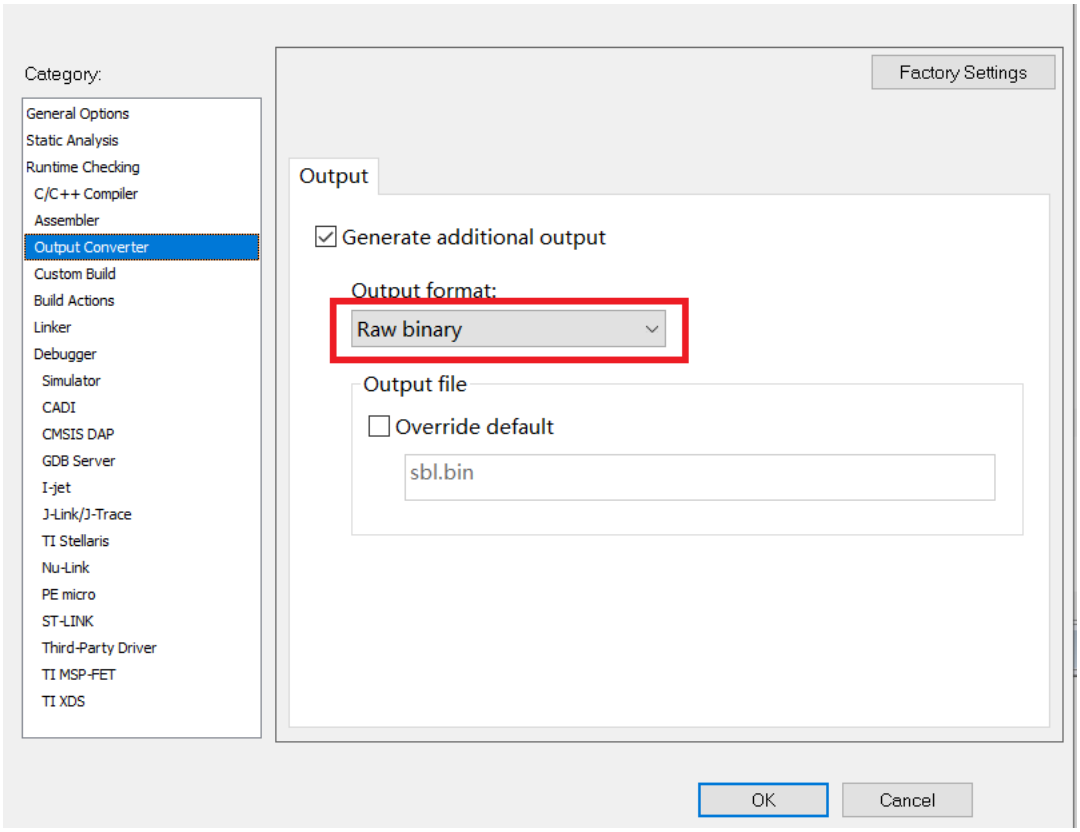


图186. 设置输出格式

7.4.4.3 应用程序镜像准备

要生成应用程序的明文镜像，步骤如下。

1. 运行 `sfw/target/evkmimxrt600` 文件夹下的 `env.bat`。
2. 运行命令 `scons --menuconfig`。
3. 进入菜单 `MCU SFW core`，不勾选菜单 `Enable sfw standalone xip`。

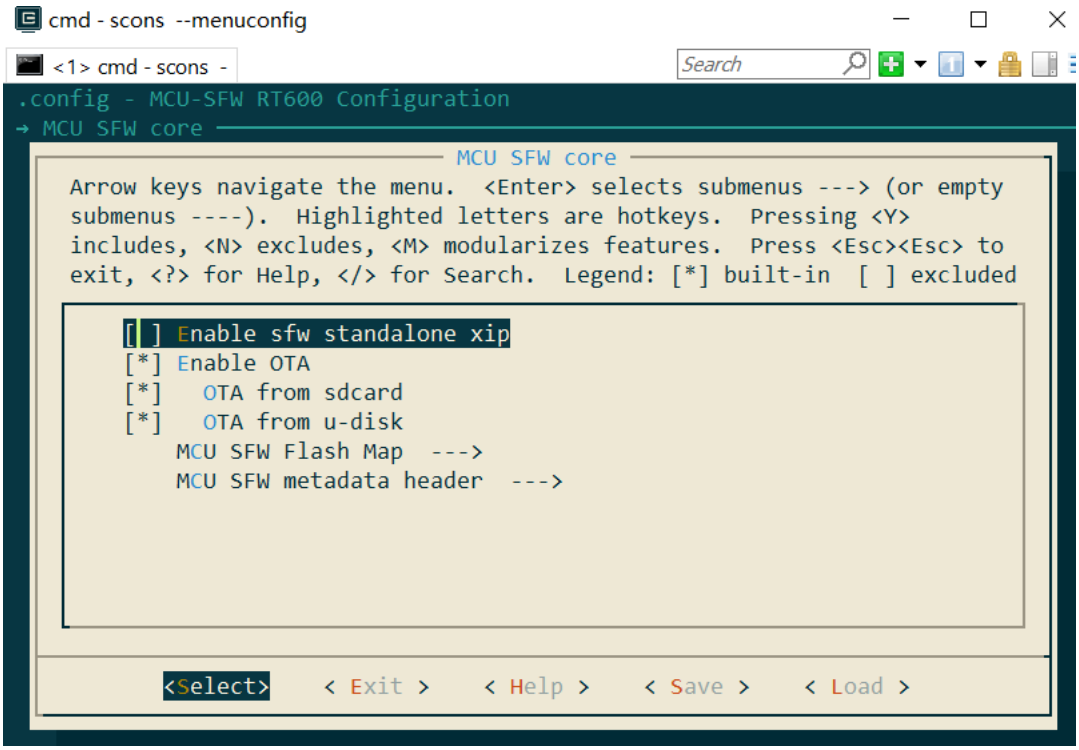


图187. 不勾选单独运行sfw选项

4. 对于要加密的image，进入菜单 `MCU SFW component > secure`，勾选菜单 `Encrypted XIP function`，然后保存并退出menuconfig。

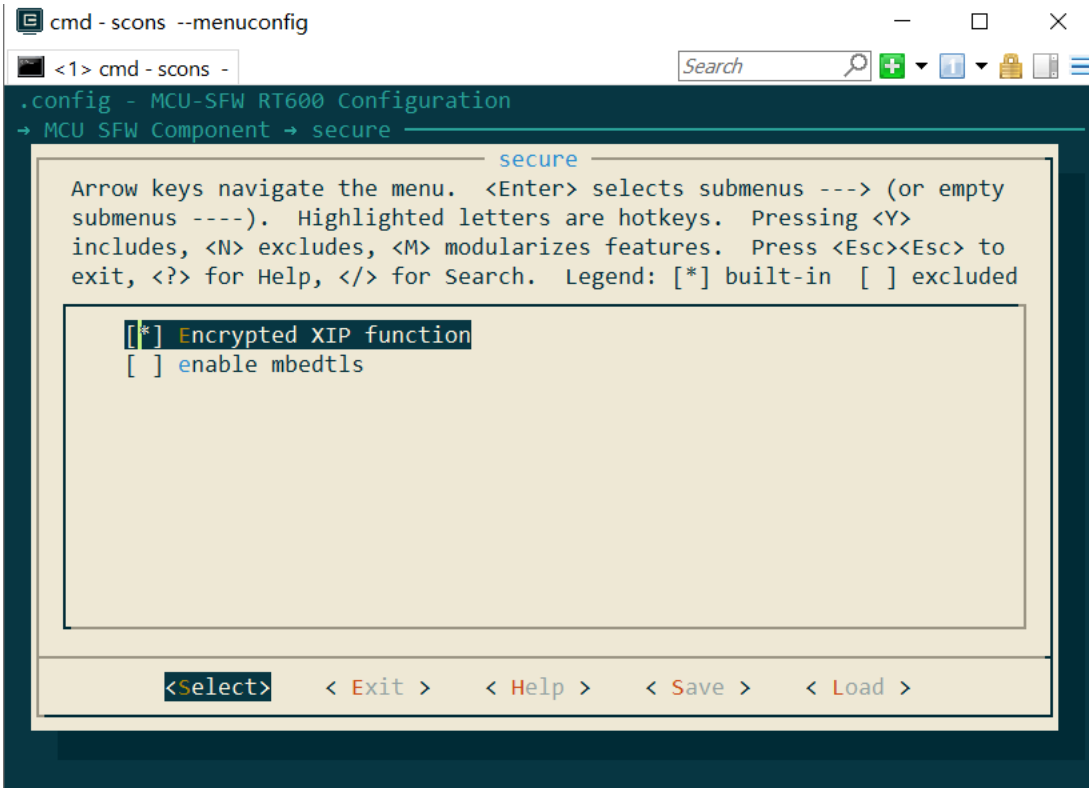


图188. 启用sfw中的密钥升级功能

5. 生成SFW工程文件。
6. 配置选项以生成binary image，然后构建项目。

7.4.4.4 运行已签名的SBL和SFW

按以下步骤生成签名的SBL和应用程序，然后将它们烧写到目标板上。

1. 将 `sbl.bin` 和 `sfw.bin` 复制到 `sbl/target/evkmimxrt600/secure` 文件夹中。
2. 拨动ISP引脚开关（ISP0开，ISP1关，ISP2关），使evkmimxrt600电路板进入ISP模式。
3. 连接器J5（LINK USB）连接一个micro USB数据线。
4. 在scons环境中输入`cd secure`进入文件夹secure。
5. 运行 `sign_sbl_app.bat` 来生成最终签名的SBL和应用程序。如果你想生成用于升级的应用程序image，需要在 `imgtool.py` 命令行中修改参数版本号。
6. 通过脚本 `sign_sbl_app.bat` 或其他工具下载签名的SBL和应用程序。这个脚本将应用程序的image下载到slot1。如果之前测试过OTA程序，SBL可能仍然从slot2中的应用程序启动。
7. 参考 7.4.4.6 节对OTP进行烧写。
8. 切换到正常的启动模式，然后复位电路板。可以在终端上看到图示的输出。

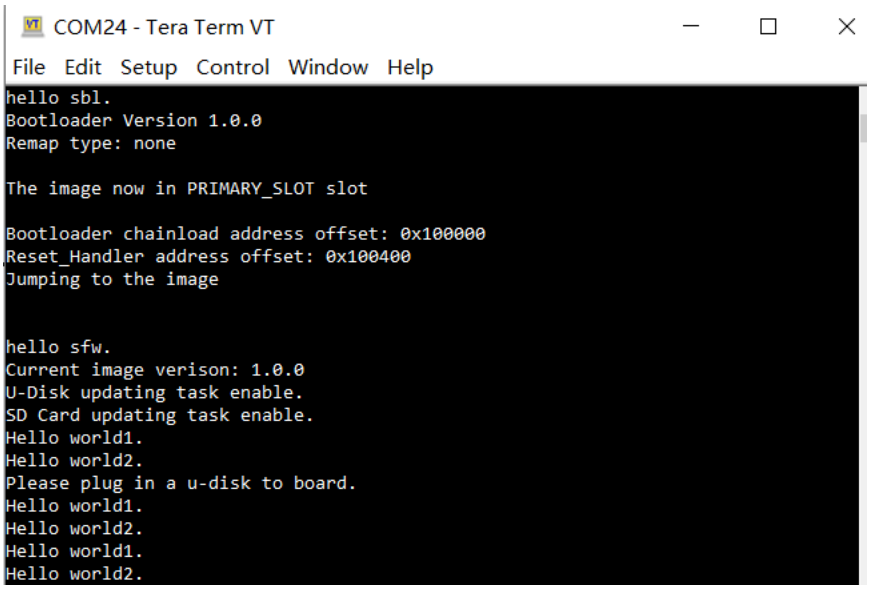


图189. 终端日志

7.4.4.5 运行加密的SBL和SFW

按照以下步骤，生成签名和加密的SBL和应用程序。

1. 将 `sbl.bin` 和 `sfw.bin` 复制到文件夹 `sbl/target/evkmimxrt600/secure` 中。
2. 编辑 `sign_enc_sbl_app.bat` 并设置KEK和加密的区域。
3. 拨动ISP引脚（ISP0开，ISP1关，ISP2关）使evkmimxrt600电路板进入ISP模式。
4. 将micro USB线缆插入到Link USB (J5)的接头J5。
5. 在scons环境下输入 `cd secure` 进入文件夹secure。
6. 运行 `sign_enc_sbl_app.bat` 来生成最终使用的签名加密的SBL和应用程序。
7. 通过这个脚本或其他工具下载SBL和应用程序。
8. 参照 7.4.4.6 节对OTP进行烧写。
9. 切换到正常的启动模式，然后复位开发板。可以在终端上看到如下输出。

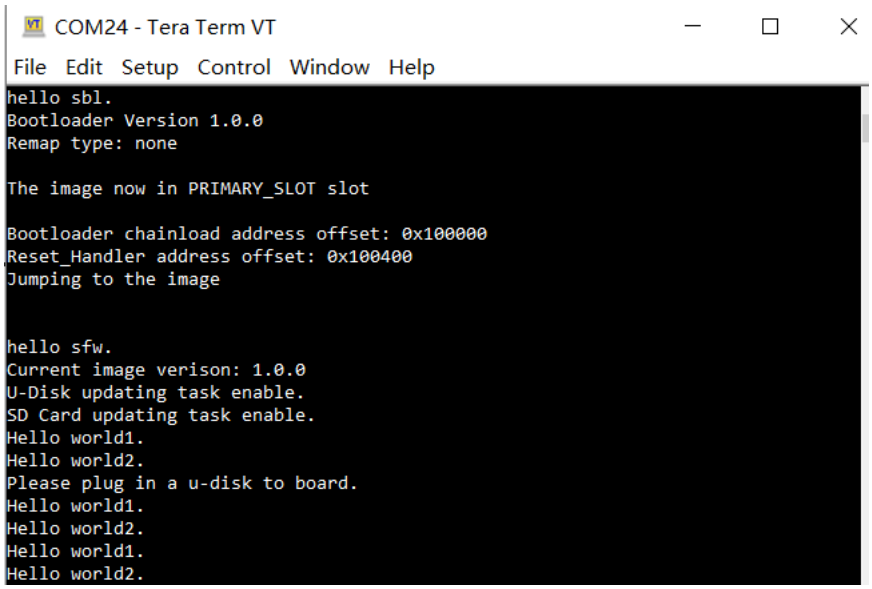


图190. 终端日志

7.4.4.6 烧写OTP（eFuse）

下面是一个对SRK表进行烧写并启用安全启动的示例。在开发阶段，用户可以使用shadow寄存器来测试安全启动。

1. 在 `sbl/target/evkmimxrt600/secure` 中找到并打开脚本 `program_ocotp.bat`，然后设置工具blhost的安装路径和正确的com端口。

```
SET "PATH=C:\nxp\MCUX_Provi_v3\bin\tools\blhost\win;%PATH%"
SET com_port=COMX,115200
```

2. 在生成SBL镜像的日志中查找RKT哈希值（RKTH）。

```
15. Output the root certificates SHA256 hash (RKTH).
        Success.
        RKTH: 8b8123193c27489fe835e104be046187dbc5507c310de41b469e68d5842decc0
```

图191. 十六进制的RKTH值

3. 日志中显示的RKTH值是用来烧录到OTP Fuses的。elftosb产生的RKTH值是 big-endian（大端序）格式的。为了在OTP中正确存储该值，需要将提供给efuse-program-once命令的参数的字节顺序换成little-endian（小端）格式。例如，1923818b, 9f48273c, 04e135e8, 876104be, 7c50c5db, 1be40d31, d5689e46, c0ec2d84。

```
blhost -p COMx,115200 -t 15000 -- efuse-program-once 0x78 1923818b
blhost -p COMx,115200 -t 15000 -- efuse-program-once 0x79 9f48273c
blhost -p COMx,115200 -t 15000 -- efuse-program-once 0x7A 04e135e8
blhost -p COMx,115200 -t 15000 -- efuse-program-once 0x7B 876104be
blhost -p COMx,115200 -t 15000 -- efuse-program-once 0x7C 7c50c5db
blhost -p COMx,115200 -t 15000 -- efuse-program-once 0x7D 1be40d31
blhost -p COMx,115200 -t 15000 -- efuse-program-once 0x7E d5689e46
blhost -p COMx,115200 -t 15000 -- efuse-program-once 0x7F c0ec2d84
```

注意

如果用户使用脚本program_ocotp.bat对eFuse进行编程，记得要根据用户的日志更新RKTH值。默认情况下，用于为efuses编程的命令被注释掉了。用户需要手动来开启它。

4. 用户可以使用以下命令启用安全启动：

```
blhost -p COMx,115200 -t 15000 -- efuse-program-once 0x60 0x900000
```

5. 用户可以通过命令efuse-read-once来验证eFuse的值。

```
blhost -p COMx,115200 -t 15000 -- efuse-read-once 0x60
```

6. 启用OTFAD并配置OTP_MASTER_KEY、OTFAD_SEED efuses，用于image解密。

```
// program OTP_MASTER_KEY
blhost -p COMx,115200 -t 15000 -- efuse-program-once 0x70 ccddeeff
blhost -p COMx,115200 -t 15000 -- efuse-program-once 0x71 8899aabb
blhost -p COMx,115200 -t 15000 -- efuse-program-once 0x72 44556677
blhost -p COMx,115200 -t 15000 -- efuse-program-once 0x73 00112233
blhost -p COMx,115200 -t 15000 -- efuse-program-once 0x74 0c0d0e0f
blhost -p COMx,115200 -t 15000 -- efuse-program-once 0x75 08090a0b
blhost -p COMx,115200 -t 15000 -- efuse-program-once 0x76 04050607
blhost -p COMx,115200 -t 15000 -- efuse-program-once 0x77 00010203
// program OTFAD SEED
blhost -p COMx,115200 -t 15000 -- efuse-program-once 0x6c 62184d50
blhost -p COMx,115200 -t 15000 -- efuse-program-once 0x6d d5ae8d29
blhost -p COMx,115200 -t 15000 -- efuse-program-once 0x6e bf6af264
blhost -p COMx,115200 -t 15000 -- efuse-program-once 0x6f 3a72eb7f
// enable OTFAD boot
blhost -p COMx,115200 -t 15000 -- efuse-program-once 0x6a 00001000
// config flash settings for MIMXRT685-EVK
blhost -p COMx,115200 -t 15000 -- efuse-program-once 0x61 314000
```

7. 拨动ISP引脚（ISP0开，ISP1关，ISP2关）使evkmimxrt600开发板进入ISP下载模式，然后运行脚本`program_ocotp.bat`。

7.4.4.7 应用程序OTA镜像准备

按照以下步骤生成用于升级的签名加密的应用程序。

1. 按照 7.4.4.3 节的方法构建image。
2. 将其重命名为 `sfw2.bin` 并复制到 `sbl/target/evkmimxrt600/secure` 文件夹中。
3. 打开脚本 `script sign_enc_sfw2.bat`，设置KEK和加密区域。
4. 运行 `sign_enc_sfw2.bat`，文件 `sfw_2_enc.bin` 是最终使用的image。

第八章
已知的问题

没有

第九章
修订历史

表8. 修订历史

修订	日期	说明
0	2021年8月26日	初版发布为开放源代码
1.1.0	2021年11月8日	1. 支持RT500、RT600的签名和加密功能 2. 优化回滚流程

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: 08 November 2021

Document identifier: MCUOTASBLSFWUG

arm