i.MX8 引导过程和创建可引导映像

原文: https://community.nxp.com/docs/DOC-343178

由 manuelrodriguez 于 2019-4-18 创建的文档

本文档旨在概述 i.MX8 引导过程，并引导您完成创建可引导映像的过程。

启动过程

退出重置状态后，i.MX8 ROM（存储在 i.MX8 的非易失性存储器中的固件）将根据启动模式引脚以确定即将使用的启动媒体/设备。
i.MX8 可以从以下启动设备启动：

- eMMC / SD 卡
- FlexSPI Flash
- NAND
- 串行下载（USB）-在芯片生产过程中所使用的模式，通过把映像下载到 RAM 并刷新板载启动设备来启动板卡。

下表显示了 i.MX8QXP 上的可选选项，i.MX8 读取控制引导模式的引脚，并根据配置选择相应的引导设备。

Table 5-8. Boot Device Selection

BOOT_MODE[3:0]	Boot Device
0010	eMMC Boot
0011	SD Boot
0100	NAND Boot - 128 pages in block
0101	NAND Boot - 32 pages in block
0110	FlexSPI Boot - 3B Read
0111	FlexSPI Boot - Hyperflash 3.0V

一旦标识启动设备，ROM 会配置并尝试在启动设备中的预定义地址读取映像，下表显示了预加载映像位于不同启动设备上的地址。

Table 5-22. 1st Image Container Offset Values

Boot Device	1st Image Container Offset
SD	32KB
eMMC	0, if located in the Boot Partition 32K, if located in the User Area
NAND	Right behind FCB/DBBT
FlexSPI	0x1000

ROM 将数据从上面的预定义地址（取决于所选的引导设备）加载到系统控制器单元（SCU）内部存储器（紧密耦合的存储器）中，并对其进行解析，以查找映像单元。它也可以通过 USB 下载映像来启动。

映像单元具有将所有映像加载到系统所需的所有信息，首先加载的映像是系统控制器固件（SCFW）和安全控制器固件（SECO）。

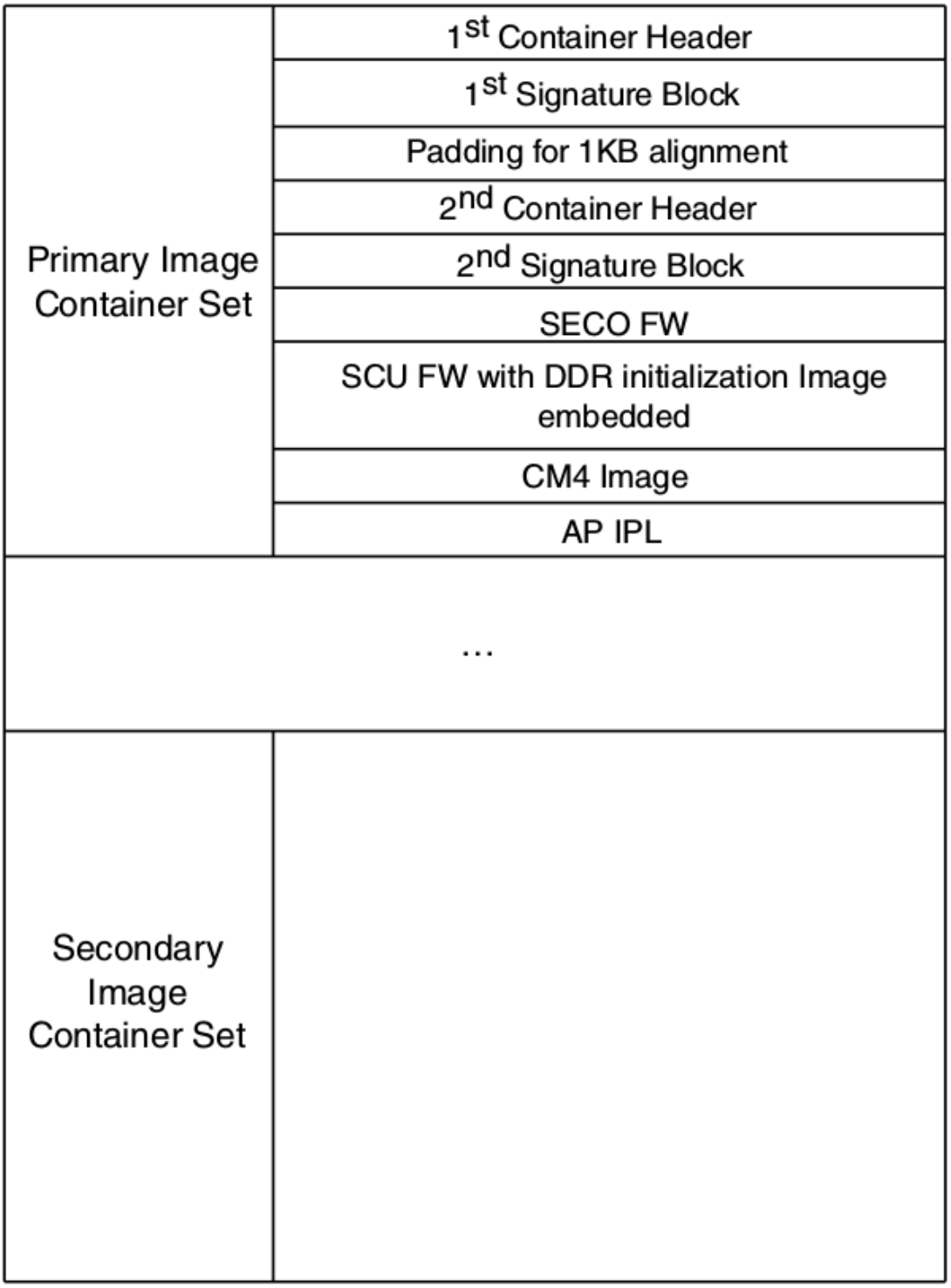


Figure 5-25. Typical Boot Image Layout

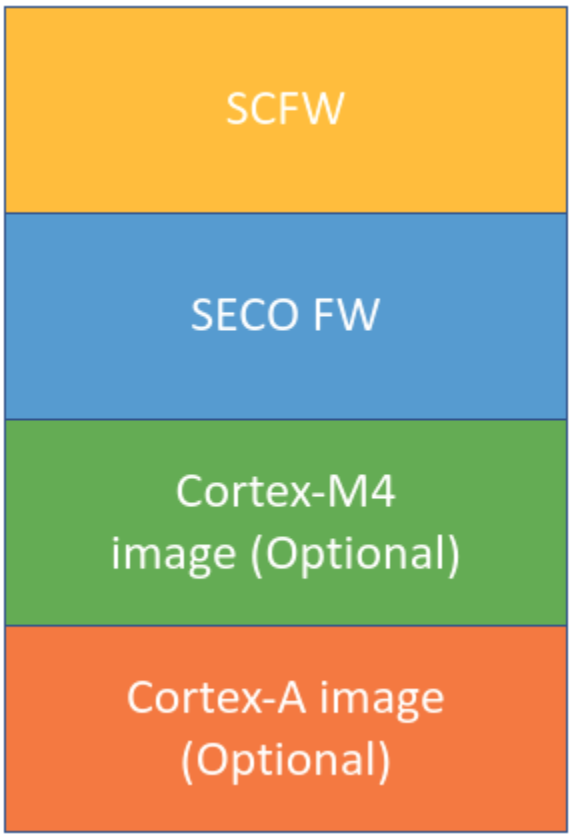
需要加载 SECO FW 来刷新设备中的 watchdog 计时器（喂狗）。如果未在 watchdog 计时器

到期之前加载 SECO FW，设备将被重置，这通常发生在设备无法从启动媒体中获取有效镜像的情况下。

一旦加载了 SCFW，ROM 就会跳转到 SCFW，并开始执行它。
然后，SCFW 初始化 DDR，并开始为 Cortex-M4（可选）和 Cortex-A 内核（可选）加载图像。
一旦将映像加载到它们的目标内存中，SCFW 就会启动内核，并将它们设置在他们的起始地址中。

创建可引导映像

作为扼要重述，可引导映像至少由系统控制器固件和安全控制器固件组成。它可以选择性地包含可用于 Cortex M4 内核（如果在 QM 设备的情况下，如果有多个）和 Cortex A 内核的映像。



引导仅包含 SCFW 和 SECO FW 的映像是可能的，这对将 SCFW 移植到目标板上的第一阶段可能是很有用的。仅使用 Cortex-M4 映像（裸机，FreeRTOS，AutoSAR ...），仅使用 Cortex-A 映像（U 引导或任何引导加载程序）或同时使用 Cortex-M4 和 Cortex-A 映像来引导映像也是可能的。 。

Mkimage 工具

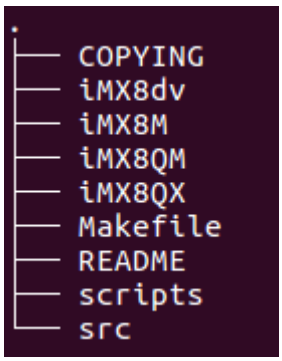
负责合并所有这些映像并为 i.MX8 创建启动映像的工具称为 mkimage。它可以在以下存储库中以源代码形式获得：

https://source.codeaurora.org/external/imx/imx-mkimage

mkimage 仅能在 Linux 中使用

因此，第一步是将 mkimage 存储库克隆到我们的计算机中，并签出最新的分支，在编写此文档时，最新版本为 4.14.98_02：
git clone https://source.codeaurora.org/external/imx/imx-mkimage
cd imx-mkimage
git checkout imx_4.14.98_2.0.0_ga

现在，您应该能够看到以下文件夹：



获取 SCFW

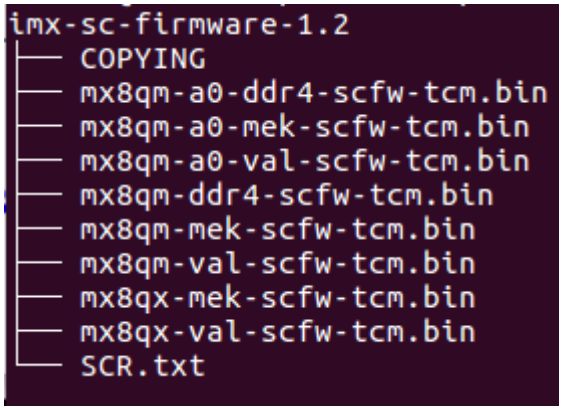
有了 mkimage 工具后，您需要使用一些实际的图像。如果使用的是自定义板，可能需要为其移植 SCFW 和 DDR 配置文件（取决于它与 NXP 参考板的距离）。

以下是有关 SCFW 基础知识的文档纲要，以及如何从头开始构建文档。如果需要入门移植方面的帮助，可以点击下面的链接。
https://community.nxp.com/docs/DOC-342654

如果您在 NXP 的参考板上尝试此操作，则可以使用预构建的 SCFW 二进制文件，可以通过 Yocto 项目的构建过程或下载移植工具包并按照以下步骤操作来获得它：
Download SECO FW binaries for release 4.14.98_02 here.

```
chmod a+x firmware-imx-8.1.bin
./firmware-imx-8.1.bin
```

系统将提示您接受许可协议，然后提取二进制文件：



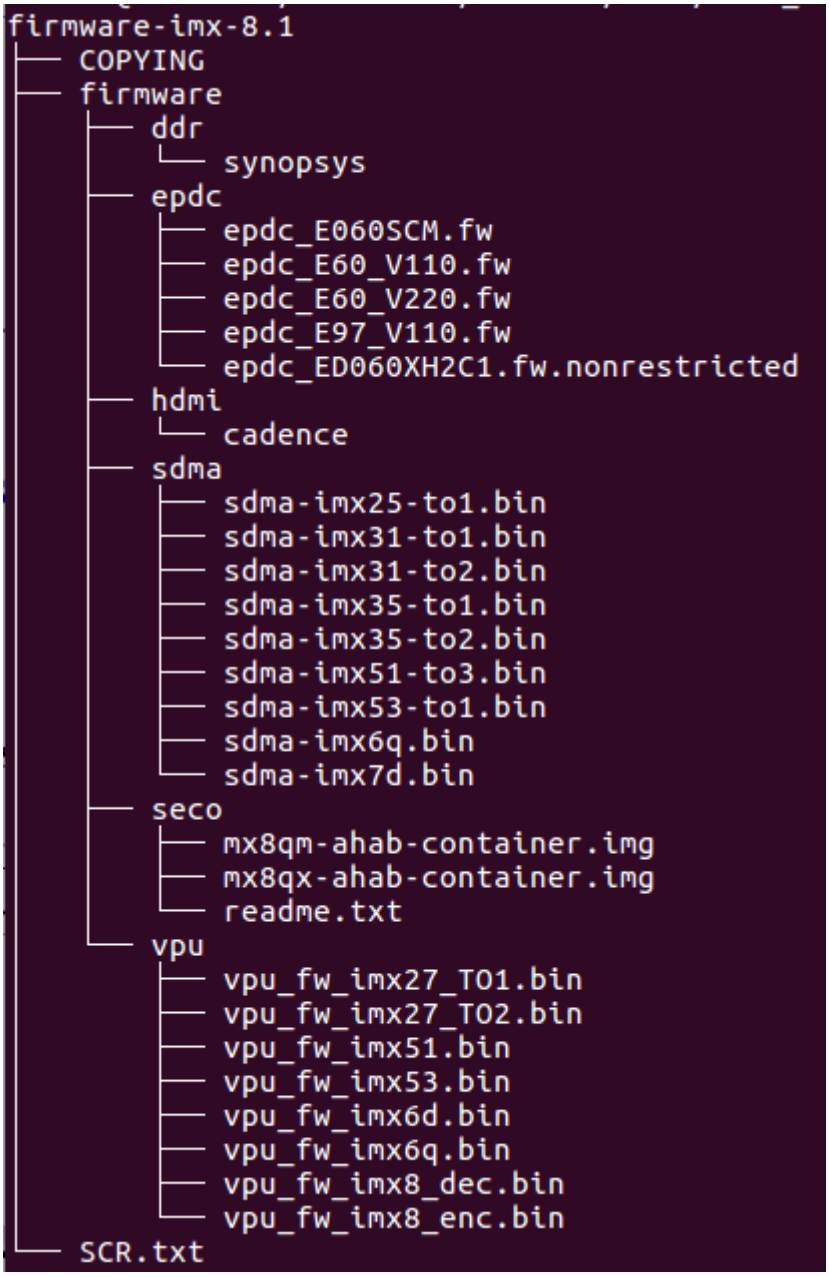
获得 SECO FW

安全控制器固件仅以二进制形式发布，可以从 NXP 网站获得。

Download SECO FW binaries for release 4.14.98_02 here.

```
chmod a+x firmware-imx-8.1.bin
./firmware-imx-8.1.bin
```

系统将提示您接受许可协议，然后提取二进制文件 ：

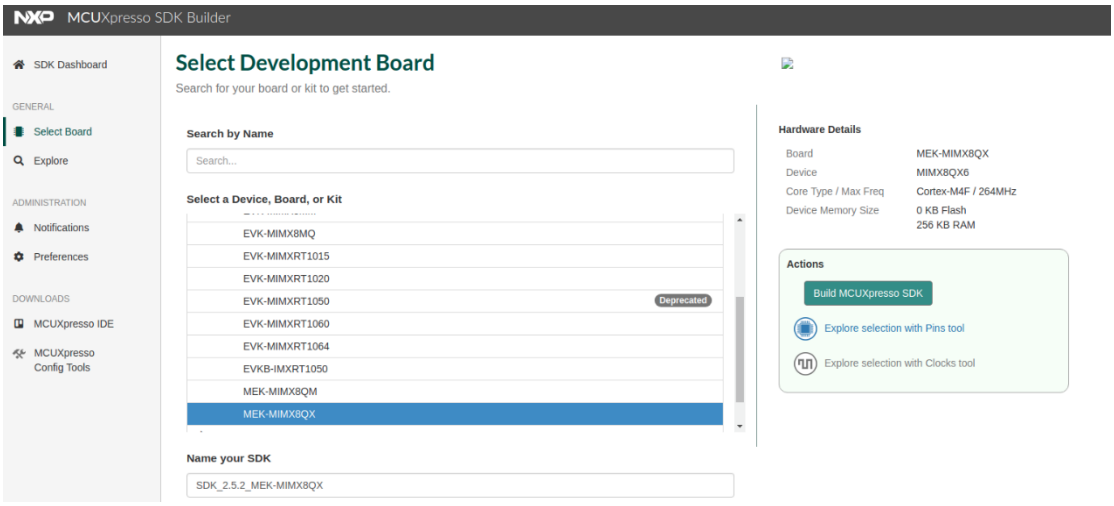


SECO FW 在固件/ seco 下

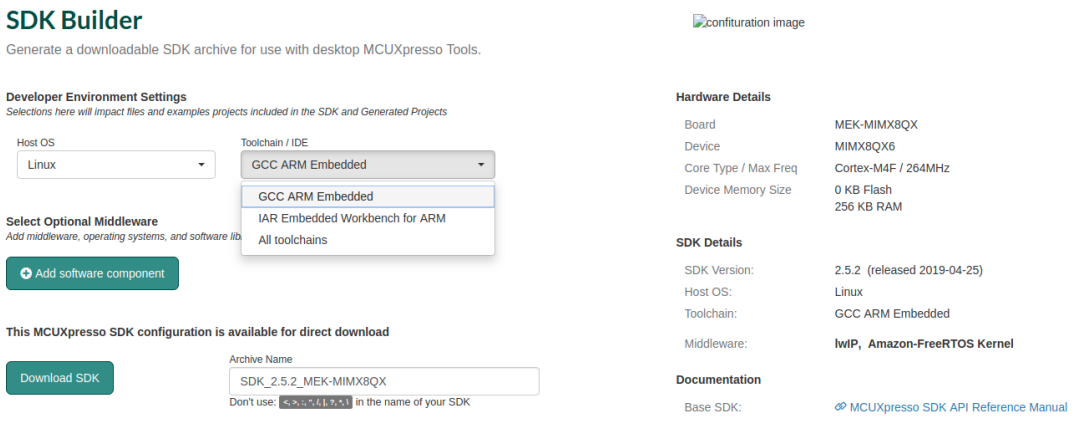
- mx8qm-ahab-container.img -----> 适用于 QM 设备的 SECO FW
- mx8qx-ahab-container.img ------>适用于 QXP 设备的 SECO FW

获取 Cortex-M4 的图像

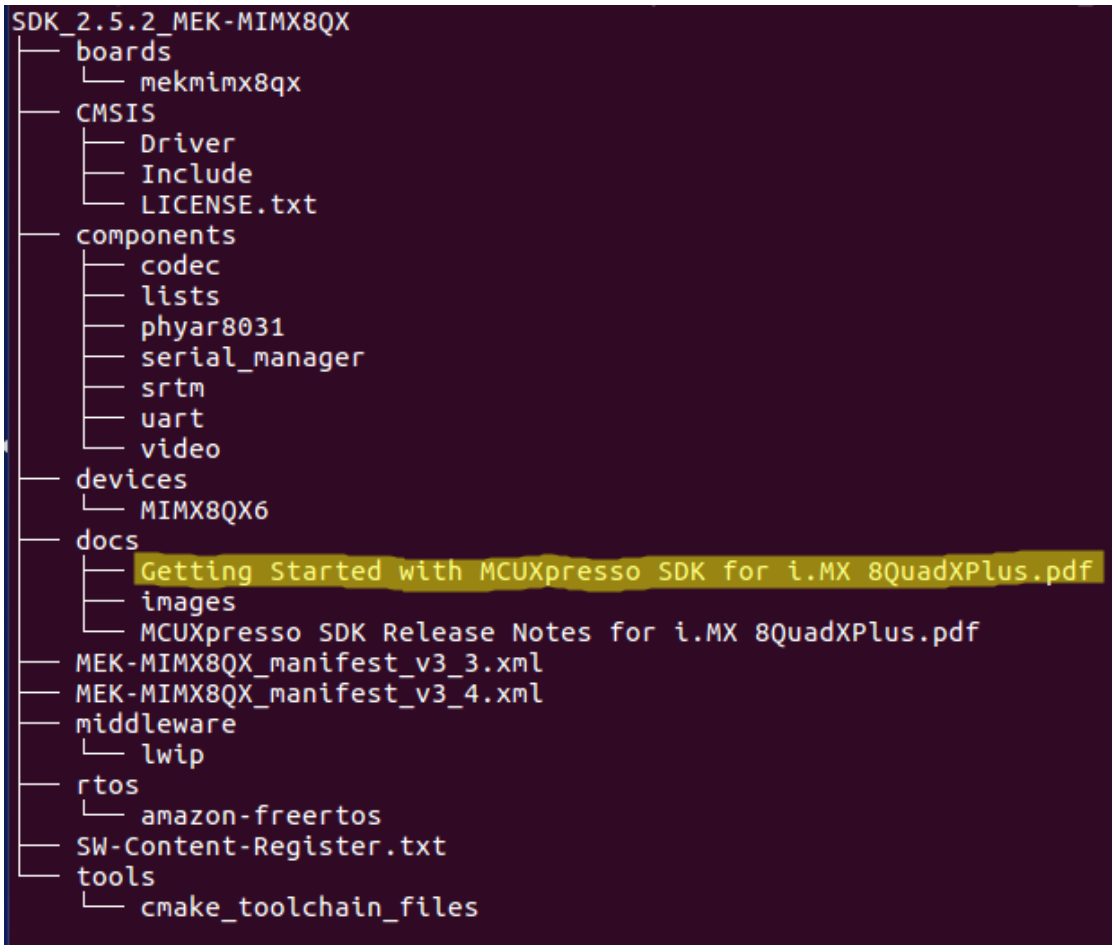
Cortex-M4 的图像可以用 SDK 生成：
https://mcuxpresso.nxp.com/en/select



只需选择要使用的设备，然后单击“ Build MCUXpresso SDK”，然后系统将提示您选择您的 IDE 和主机。



单击下载 SDK，包含该 SDK 的压缩文件将被下载到您的计算机上。现在，您只需要解压缩文件并按照入门文档中的步骤生成镜像。

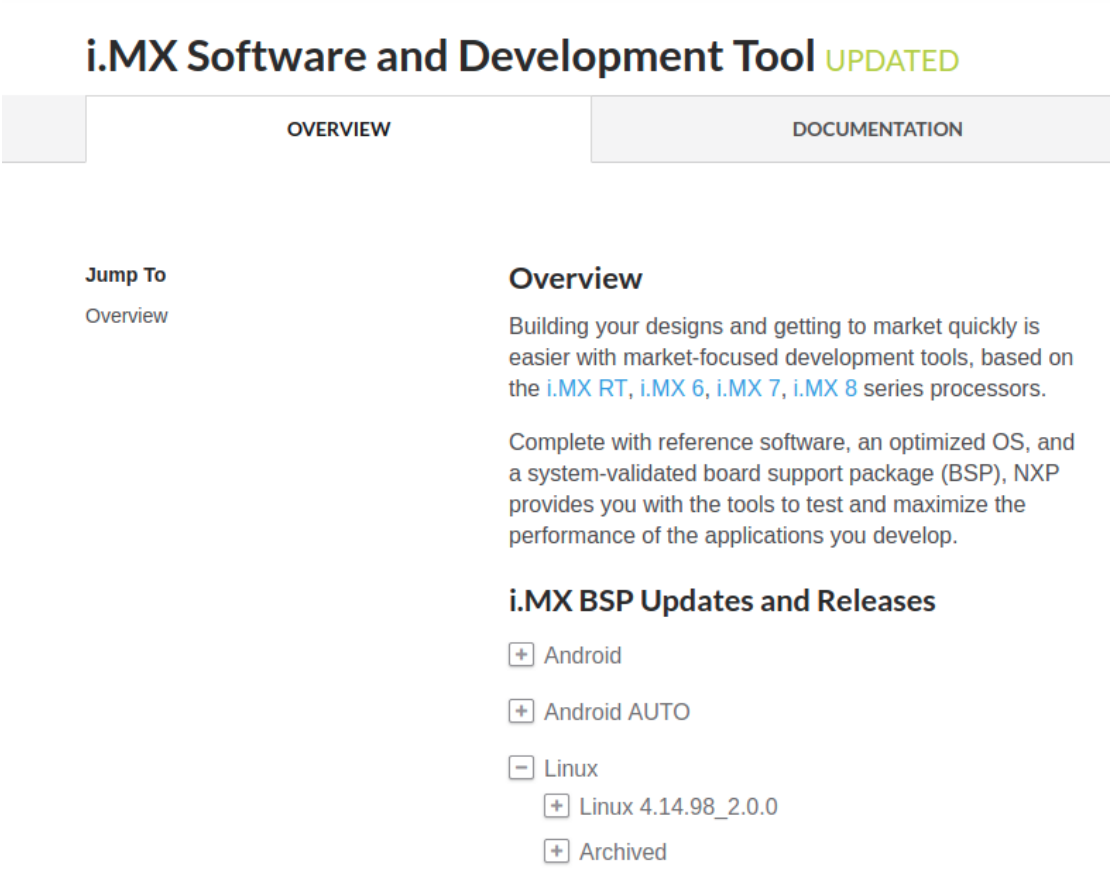


入门文档的内容是设置工具链的步骤和为 M4 生成映像的步骤。

本文档中还附带了用于 QM 和 QXP MEK 的 M4 二进制文件，示例在 M4 终端上输出 hello world 消息。

获取 Cortex-A 的图像

可以通过 Yocto BSP 获得用于 Cortex-A 内核的引导程序：



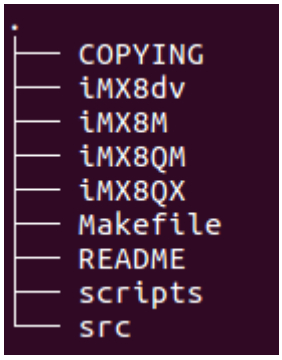
为 4.14.98 发行版生成映像的步骤在以下链接：
https://www.nxp.com/webapp/Download?colCode=imx-yocto-L4.14.98_2.0.0_ga

有关 Yocto BSP 的更多详细信息，可以在这里找到：
https://community.nxp.com/docs/DOC-94849

在此处随附了为 MEK 平台上的 Cortex-A 内核创建启动映像所需的所有二进制文件。

建立可启动映像

一旦所有必需的部件都已构建/获得，就可以创建可引导映像。
需要将 SCFW，SECO FW 和相应的 Cortex-M4 / A 映像复制到目标设备的文件夹，即，如果您正在为 i.MX8QX 变体构建映像，则将该变体的二进制文件复制到其文件夹：



以下是构建可启动映像所需的文件的列表：

- scfw_tcm.bin ----------系统控制器固件目标板的二进制文件
- mx8qm（qx）-ahab-container.image ----------适用于 QM 或 QXP 变体的安全控制器固件
- bl31.bin ---------- ---- ARM 可信固件二进制文件（如果将 u-boot 与 ATF 一起使用，则是必需的）**仅用于**通过 u-boot **创建 Cortex-A 映像**
- u-boot.bin ---------- -U-boot 二进制文件**（可选）**
- m4_image ---------- M4 二进制图像，QM 变体具有 2 个 Cortex-M4，在这种情况下，可能需要 M4 二进制文件 **（可选）**

将所需的二进制文件复制到所需的变体文件夹（在本示例中为 QXP 或 QM）后，即可开始构建一些映像。

在每个文件夹中包含的 soc.mak 文件中定义了用于构建不同映像的所有目标，该文件包含用于创建许多受支持的可引导映像的不同示例。

创建仅 SCFW 图像

用于创建仅 SCFW 映像的目标是 flash_b0_scfw，并且在每个变量的 soc.mak 文件下定义了该目标。
要从 imx-mkimage 目录为 QXP 调用此目标：

```
make SOC=iMX8QX flash_b0_scfw
```

要从 imx-mkimage 目录调用此目标：

```
make SOC=iMX8QM flash_b0_scfw
```

可以在下面看到 flash_b0_scfw 的目标定义。

QXP 的定义：

```
flash_scfw flash_b0_scfw: $(MKIMG) mx8qx-ahab-container.img
scfw_tcm.bin
```

```
 ./$(MKIMG) -soc QX -rev B0 -dcd skip -append mx8qx-ahab-
container.img -c -scfw scfw_tcm.bin -out flash.bin
```

QM 的定义：

```
flash_b0_scfw: $(MKIMG) mx8qm-ahab-container.img
scfw_tcm.bin
 ./$(MKIMG) -soc QM -rev B0 -dcd skip -append mx8qm-ahab-
container.img -c -scfw scfw_tcm.bin -out flash.bin
```

仅创建 Cortex-A 图像

仅用于创建 Cortex-A 图像的目标称为 flash_b0。

从 imx-mkimage 目录中为 QXP 调用目标：

```
make SOC=iMX8QX flash_b0
```

从 imx-mkimage 目录中调用 QM 目标：

```
make SOC=iMX8QM flash_b0
```

flash_b0 的目标定义如下所示。

QXP 的定义：

```
flash flash_b0: $(MKIMG) mx8qx-ahab-container.img
scfw_tcm.bin u-boot-atf.bin
 ./$(MKIMG) -soc QX -rev B0 -append mx8qx-ahab-container.img
-c -scfw scfw_tcm.bin -ap u-boot-atf.bin a35 0x80000000 -out
flash.bin
```

QM 的定义：

```
flash_b0: $(MKIMG) mx8qm-ahab-container.img scfw_tcm.bin u-
boot-atf.bin
 ./$(MKIMG) -soc QM -rev B0 -append mx8qm-ahab-container.img
-c -scfw scfw_tcm.bin -ap u-boot-atf.bin a53 0x80000000 -out
flash.bin
```

仅创建 Cortex-M4 图像

仅用于创建 Cortex-m4 图像的目标在 QXP 上称为 flash_b0_cm4。既然系统中有两个 M4，QM 有不同的目标。

从 imx-mkimage 目录中为 QXP 调用此目标：

```
make SOC=iMX8QX flash_b0_cm4
```

从 imx-mkimage 目录调用此 QM 目标：

```
// For Cortex-M4_0 only
make SOC=iMX8QM flash_b0_cm4_0
```

```
// For Cortex-M4_1 only
make SOC=iMX8QM flash_b0_cm4_1
// For both Cortex-M4_0 and Cortex-M4_1
make SOC=iMX8QM flash_b0_m4s_tcm
```

可以在下面看到 flash_b0_cm4 的目标定义。
QXP 的定义：

```
flash_cm4 flash_b0_cm4: $(MKIMG) mx8qx-ahab-container.img
scfw_tcm.bin m4_image.bin
 ./$(MKIMG) -soc QX -rev B0 -append mx8qx-ahab-container.img
-c -scfw scfw_tcm.bin -p1 -m4 m4_image.bin 0 0x34FE0000 -out
flash.bin
```

QM 的定义：

```
flash_b0_cm4_0: $(MKIMG) mx8qm-ahab-container.img
scfw_tcm.bin m4_image.bin
 ./$(MKIMG) -soc QM -rev B0 -dcd skip -append mx8qm-ahab-
container.img -c -scfw scfw_tcm.bin -p1 -m4 m4_image.bin 0
0x34FE0000 -out flash.bin

flash_b0_cm4_1: $(MKIMG) mx8qm-ahab-container.img
scfw_tcm.bin m4_image.bin
 ./$(MKIMG) -soc QM -rev B0 -dcd skip -append mx8qm-ahab-
container.img -c -scfw scfw_tcm.bin -p1 -m4 m4_image.bin 1
0x38FE0000 -out flash.bin

flash_b0_m4s_tcm: $(MKIMG) mx8qm-ahab-container.img
scfw_tcm.bin m40_tcm.bin m41_tcm.bin
 ./$(MKIMG) -soc QM -rev B0 -dcd skip -append mx8qm-ahab-
container.img -c -scfw scfw_tcm.bin -p1 -m4 m40_tcm.bin 0
0x34FE0000 -m4 m41_tcm.bin 1 0x38FE0000 -out flash.bin
```

上面的示例适用于从 TCM 引导的 M4 映像，M4 拥有能从 DDR 引导和执行的能力。它还能够从 SPI 存储器进行 XIP（就地执行）。有关此目标的示例，请查看 soc.mak。

使用 Cortex-A 和 Cortex-M4 镜像创建镜像

用软件为所有内核创建映像的目标称为 flash_linux_m4。
要从 imx-mkimage 目录为 QXP 调用此目标：

```
make SOC=iMX8QX flash_linux_m4
```

从 imx-mkimage 目录调用此 QM 目标：

```
make SOC=iMX8QM flash_linux_m4
```

可以在下面看到 flash_linux_m4 的目标定义。
QXP 的定义：

```
flash_linux_m4: $(MKIMG) mx8qx-ahab-container.img
scfw_tcm.bin u-boot-atf.bin m4_image.bin
 ./$(MKIMG) -soc QX -rev B0 -append mx8qx-ahab-container.img
-c -flags 0x00200000 -scfw scfw_tcm.bin -ap u-boot-atf.bin
a35 0x80000000 -p3 -m4 m4_image.bin 0 0x34FE0000 -out
flash.bin
```

QM 的定义：

```
flash_linux_m4: $(MKIMG) mx8qm-ahab-container.img
scfw_tcm.bin u-boot-atf.bin m4_0_image.bin m4_1_image.bin
 ./$(MKIMG) -soc QM -rev B0 -append mx8qm-ahab-container.img
-c -flags 0x00200000 -scfw scfw_tcm.bin -ap u-boot-atf.bin
a53 0x80000000 -p3 -m4 m4_0_image.bin 0 0x34FE0000 -p4 -m4
m4_1_image.bin 1 0x38FE0000 -out flash.bin
```

Flash 图片

这将创建一个名为 flash.bin 的可引导镜像，并将该镜像刷新到 SD 卡并在您的 MEK 上引导，只需执行以下操作：

```
sudo dd if=iMX8QX/flash.bin of=/dev/mmcblkX bs=1k seek=32
```

如果所需目标是 QM，就将 if = iMX8QX ...更改为 if = iMX8QM。
然后在“ of = / dev / mmcblkX”上匹配您的 SD 卡设备，您可以在插入 SD 卡之前和之后在控制台上键入 lsblk 来查看 SD 卡的枚举方式。
请记住，根据上面的信息，i.MX8 将在 SD 卡上以 32k 搜索镜像，这就是我们在此处像 SD 卡刷新镜像的原因。
有关更多示例，请查看 soc.mak 文件，该文件包含用于不同启动媒体（NAND / QSPI）以及不同配置和用法的示例。

额外资源

参考手册第 5 章系统引导
SCFW API 和端口文档
imx-mkimage README
System Controller Firmware 101

附件：..\..\..\bootable image.tar.gz