

# AN13275

如何在新的 iMX8/8X 板上启用 Linux BSP L5.4

Rev. 1 — 2023年5月26日

应用笔记

## 文档信息

信息	内容
关键词	iMX8, Linux BSP, port
摘要	本应用笔记介绍了在新的自定义 i.MX8/8X 板上启用标准 Linux BSP L5.4 的一般过程，帮助用户快速移植标准 Linux BSP 版本代码到自定义 i.MX8/8X 板上，并提示用户注意那些需要修改的关键部分。



## 1 介绍

### 1.1 目标

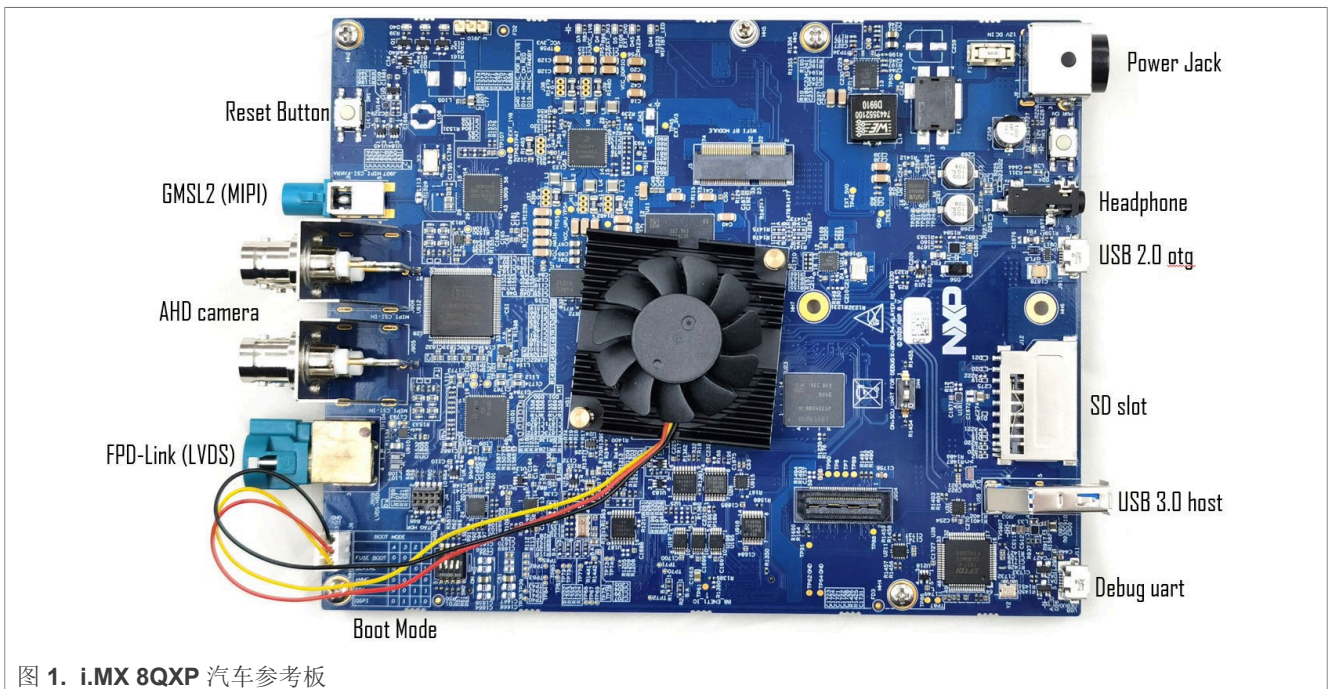
本应用笔记介绍了在新的自定义 i.MX 8/8X 板上启用标准 Linux BSP L5.4 的一般过程，帮助用户快速移植标准 Linux BSP 版本代码到自定义 i.MX 8/8X 板上，并提示用户注意那些需要修改的关键部分。

### 1.2 示例板

本应用笔记使用 i.MX 8QXP 汽车参考板作为示例板，因为标准的 Linux BSP 版本不支持该参考板。更多详细信息，请联系恩智浦代表。

该板的硬件设计基于 i.MX8QXP MEK 板，但有如下变化：

- i.MX 8QXP C0 芯片
- 三星汽车级 LPDDR4 和 eMMC5.1
- MIPI-CSI，配备 NVP6324 汽车 AHD 解决方案
- LVDS 显示器，配备 TI DS90UB947/948 Serdes（通过 FPD Link III），用于汽车应用
- MIPI-DSI 显示器，配备 Maxim 96752/96755 Serdes（通过 GMSL2），用于汽车应用
- 恩智浦 TJA1101 汽车 100 Mbps 以太网 PHY
- 用于 Carplay/AA 的 USB3.0 host 和用于调试的 USB2.0 OTG



### 1.3 Linux BSP 版本

本应用笔记以 L5.4.47\_2.2.0 Linux BSP 版本为例。如需查询所有 i.MX Linux BSP 的版本，请参见 Embedded Linux for i.MX Applications Processors。

以下章节介绍了移植SCFW、ATF、U-Boot 和 Linux 内核的一般过程。每项都可以独立编译，您可从以下链接下载软件包或源代码：

- SCFW  
[https://www.nxp.com/webapp/Download?colCode=L5.4.47\\_2.2.0\\_SCFWKIT-1.6.0&appType=license](https://www.nxp.com/webapp/Download?colCode=L5.4.47_2.2.0_SCFWKIT-1.6.0&appType=license)
- Arm Trusted Firmware (ATF)  
Git clone <https://github.com/nxp-imx/imx-atf> -b rel\_imx\_5.4.47\_2.2.0
- U-Boot  
Git clone <https://github.com/nxp-imx/uboot-imx> -b rel\_imx\_5.4.47\_2.2.0
- imx-mkimage  
Git clone <https://github.com/nxp-imx/imx-mkimage> -b rel\_imx\_5.4.47\_2.2.0
- Linux Kernel  
Git clone <https://github.com/nxp-imx/linux-imx> -b rel\_imx\_5.4.47\_2.2.0

## 2 生成 DDR 配置文件

i.MX 8/8X DDR 寄存器编程辅助工具（RPA）是 Excel 电子表格工具，用于为用户具体的DDR 配置（DDR 芯片类型、容量等）进行 DDR 初始化。RPA 以两种格式（参见单独的 Excel 工作表选项卡）启动 DDR 初始化：

- DDR 压力测试脚本  
这种格式专门用于 DDR 压力测试，首先复制“DDR压力测试脚本CBT”选项卡上的内容，再将其粘贴到一个文本文件中，以 .ds 文件扩展名命名该文档。请在执行 DDR 压力测试时使用此文件。
- DCD CFG 文件  
这种格式是 SCU 固件（SCFW）专用的配置文件。在这种情况下，用户复制“DCD CFG文件 CBT”选项卡上的内容，并将其粘贴到一个文本文件，以 .cfg 文件扩展名命名该文档并将该文件放入相应的 SCFW 板级文件目录中。

### 2.1 下载 RPA 工具

注：在所有情况下，RPA 修订参照都是最低 SCFW 版本，参见 [i.MX 8/8X 系列 DDR 工具版本](#) 表。在某些情况下，还可以进行 BSP 调节。

要获取最新的RPA，请参见以下链接：

- [i.MX8QM DDR 寄存器编程辅助工具 \(RPA\)](#)
- [i.MX8QXP/DXP/DX DDR 寄存器编程辅助工具 \(RPA\)](#)

要参照 L5.4.47\_2.2.0 BSP 和 SCFW 1.6.0，请在以下步骤中使用 MX8QXP\_C0\_B0\_LPDDR4\_RPA\_1.2GHz\_v14.xlsx RPA 版本。

### 2.2 使用 RPA 工具

要使用 RPA 工具为用户自定义板上的具体 DDR 生成新的 DDR 压力测试脚本和 DCD CFG 文件，请按以下步骤操作。

1. 从 DDR 供应商处获取所需的 DDR 数据手册  
要在 RPA 工具中填写 DRAM 参数，请使用 DDR 供应商提供的 DDR 数据手册，通常可从 DDR 供应商的网站下载。用户也可直接联系 DDR 供应商获取此数据手册。
2. 更新“寄存器配置”选项卡上的芯片信息表。

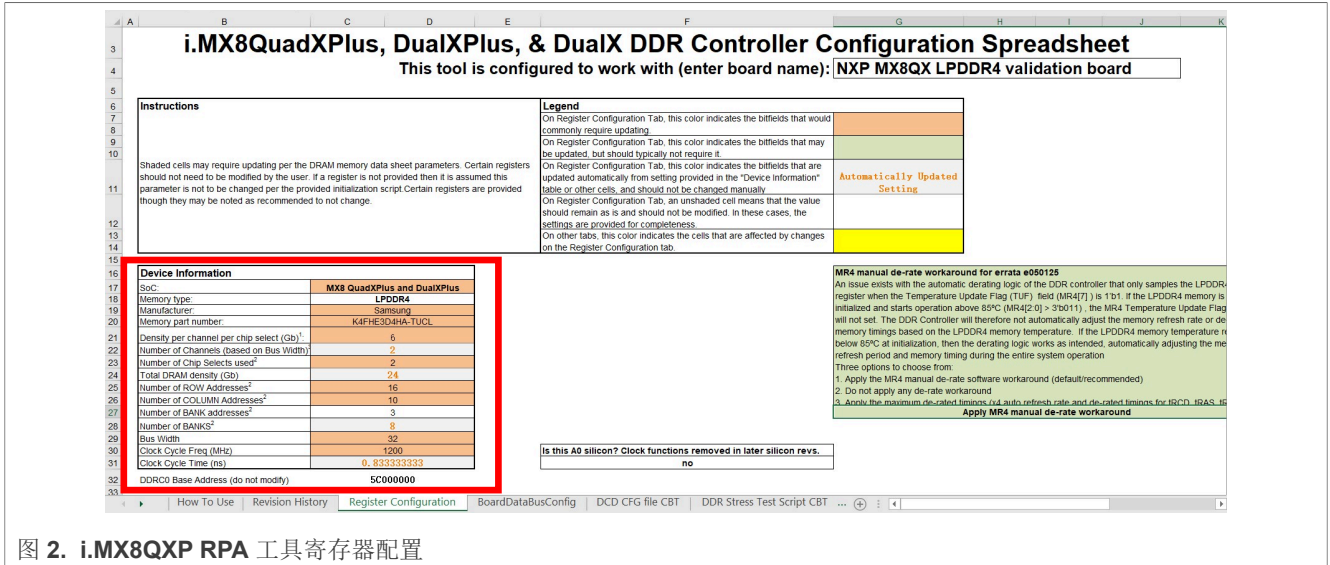


图 2. i.MX8QXP RPA 工具寄存器配置

在 图 2 突出显示的芯片信息表中更新以下信息：

- 制造商
- 存储器型号
- 每个片选信号通道的容量 (Gb)
- 片选信号数量
- 行地址的数量
- 列地址的数量
- BANK 地址的数量
- 总线宽度
- 时钟周期频率 (MHz)

其他参数将使用上述信息自动计算并填入表格。

3. 更新 “BoardDataBusConfig (板数据总线配置)” 选项卡上的数据总线映射。

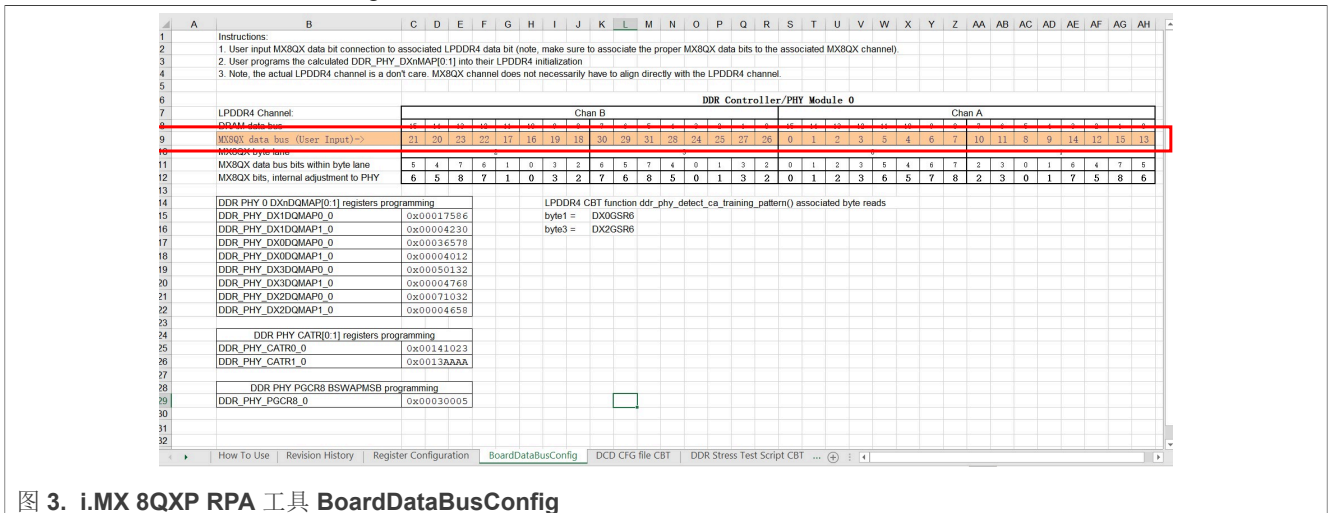


图 3. i.MX 8QXP RPA 工具 BoardDataBusConfig

通常，由于物理布局的限制，DDR 芯片和 SOC 之间数据引脚的物理连接并不是直接配对的。因此，我们需要一个映射表来记录 DDR 数据引脚的物理连接，并将这些信息放入 DDR 控制器的寄存器中，使 DDR 数据引脚能够正确进行逻辑连接。



在 图 3 突出显示的行中，根据硬件原理图更新 DDR 芯片和 SOC 之间数据引脚的物理映射。其他参数将根据用户的输入自动更新。

例如，从 图 4 的示例板示意图中，我们可以发现 DDR 芯片上的 DQ0\_A 引脚与 iMX 8QXP 上的 DDR\_DQ13 引脚相连，因此我们在圈出的单元格中输入 13，对于其他引脚，请遵循相同的方法。

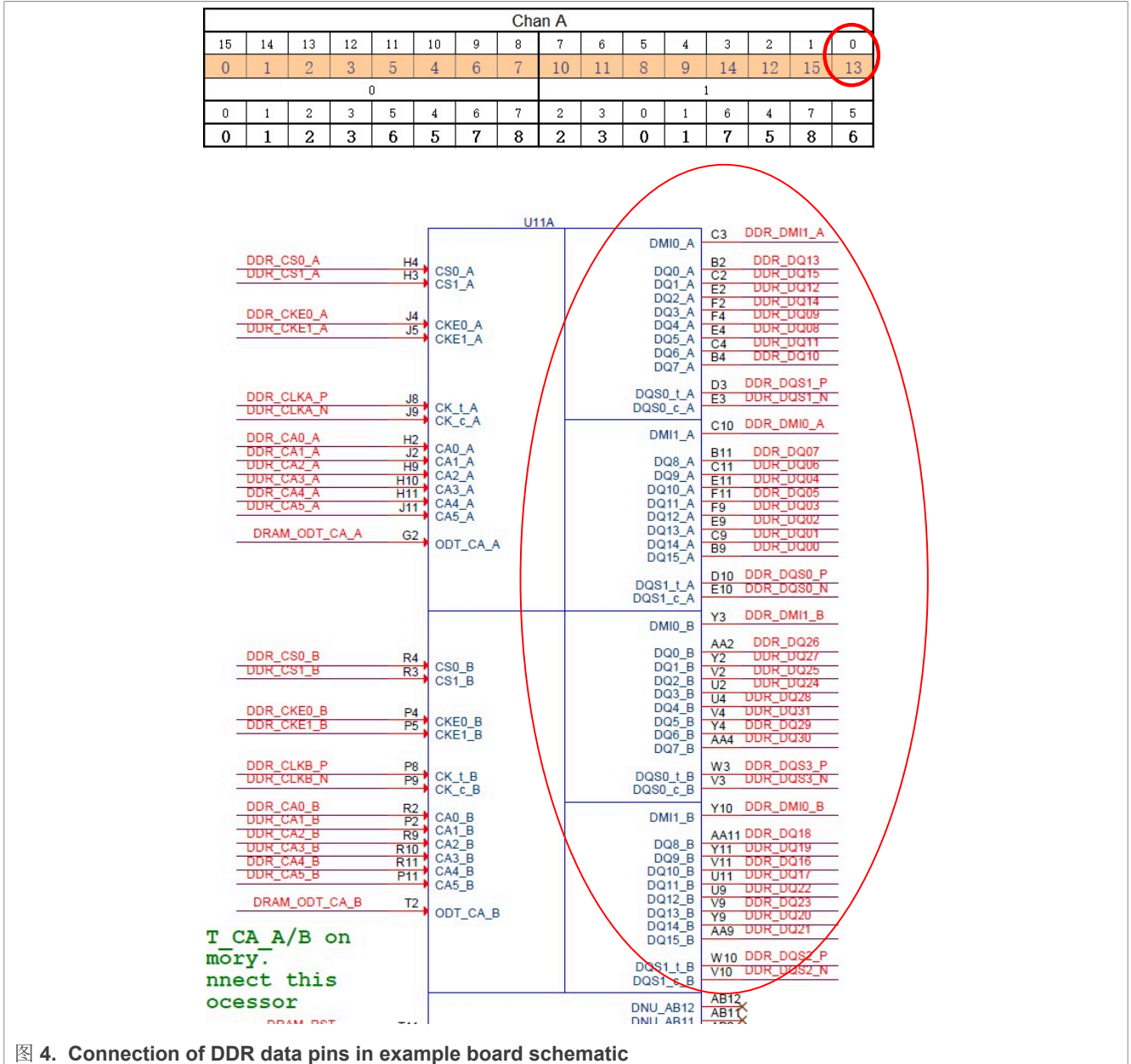


图 4. Connection of DDR data pins in example board schematic

- 将“DCD CFG文件CBT”和“DDR压力测试脚本CBT”选项卡中的文本复制到文件。  
单击RPA工具中的“DCD CFG文件CBT”选项卡，将所有文本复制到一个文件中，命名为 BOARD\_NAME.cfg。该文件稍后将在移植 SCFW 时使用。  
单击“DDR压力测试脚本CBT”选项卡，将所有文本复制到一个文件中，命名为 BOARD\_NAME.ds。该文件稍后将在 DDR 压力测试中使用。

### 3 SCFW 移植

系统控制器单元（SCU）对硬件的许多底层功能进行了抽象。在 SCU 上运行的软件称为 SC 固件（SCFW）。SCFW 提供以下功能和服务。

- 系统初始化和启动
- 系统控制器通信
- 电源管理
- 资源管理
- 引脚管理
- 计时器
- 中断
- 安全
- 其它

大多数 SCFW 代码仅在 SCFW 移植工具包中以目标文件格式提供，用户无法修改。但对于与板级相关的设置，SCFW 移植工具包提供了 `board.c` 文件的源代码，其中包含了与板级相关的初始化功能和自定义特性。本章重点介绍如何为新板级移植 `board.c` 文件。

#### 3.1 提取 SCFW 代码

要提取 SCFW 代码，请按以下步骤操作。

1. 从 [应用 \(Apps\)](#) 下载 SCFW 1.6.0 软件包 `imx-scfw-porting-kit-1.6.0.tar.gz`。
2. 解压文件。
3. 转到软件包文件夹。
4. 使用以下命令提取 SCFW 代码。

```
$chmod a+x imx-scfw-porting-kit-1.6.0.bin
$./imx-scfw-porting-kit-1.6.0.bin
```

5. 阅读并接受许可证后，提取 `imx-scfw-porting-kit-1.6.0` 文件夹中的 SCFW 代码。除了代码之外，还有在 `imx-scfw-porting-kit-1.6.0/doc/pdf` 文件夹提取的版本文档，包括版本说明、API 用户指南和更详细的移植指南。对于 i.MX8/8X 产品的新用户和开发人员，这些文档非常有用。如对 SCFW 有任何疑问，请先查看这些文档。
6. 使用以下命令提取 i.MX8QXP 专有的 SCFW 代码。

```
$cd imx-scfw-porting-kit-1.6.0/src/
$tar zxvf scfw_export_mx8qx_b0.tar.gz
```

代码位于 `imx-scfw-porting-kit-1.6.0/src/scfw_export_mx8qx_b0/` 路径。

我们可以将此路径设为 `SCFW_DIR`。

#### 3.2 创建板级文件

每块板都有自己独特的硬件设计，在 SCFW 层面可能有各种板操作。因此，SCFW 在 `SCFW_DIR/platform/board/` 下为每个受支持的板提供板级文件夹。

```
board_common.c  config.h  mx8qm_mek  mx8qx_auto  mx8qx_val  pmic.h
board_common.h  drivers  mx8qm_val  mx8qx_dxl_phantom  none
board.S  mx8dxl_evk  mx8qx_6layers  mx8qx_mek  pmic.c
```

板级文件夹包含以下部分：

- board.bom: 包括 PMIC 驱动的信息
- board.c: 与板相关的操作
- board.h: board.c 的标头文件，包括 board.c 中使用的宏定义
- Makefile: 用于编译 board.c 的 makefile
- dcd/: DDR 配置文件的文件夹，通常至少包含以下两个脚本：
  - ddr\_stress\_test\_parser.cfg: 用于为 DDR 压力测试编译 SCFW。
  - BOARD\_NAME.cfg: 在 [Using RPA tools](#) 中生成的 ddr 脚本，用于编译 SCFW 以供正常系统使用。

为了简化移植，用户可以直接从参考板级文件夹 mx8qx\_mek 复制这些文件，并根据自身要求修改。

由于具体修改取决于板的设计和最终产品的用例，本文档未列出详细的修改，但以下三点，通常需要用户在 board.c 文件中自行修改。

1. 在 board\_system\_config() 函数中

SCFW 提供的一个主要特性是资源分区。它将资源划分到不同的域以保护系统。默认情况下，我们将为 M4 内核创建一个分区，在 board.c 中的 board\_system\_config() 中执行。

通常，M4 的资源分区按以下步骤进行：

- a. 将所有资源标记为不可移动。

```
/* Mark all resources as not movable */
BRD_ERR(rm_set_resource_movable(pt_boot, SC_R_ALL, SC_R_ALL,
    SC_FALSE));
BRD_ERR(rm_set_pad_movable(pt_boot, SC_P_ALL, SC_P_ALL,
    SC_FALSE));
```

- b. 为 M4 内核创建一个新分区。

```
/* Allocate M4_0 partition */
BRD_ERR(rm_partition_alloc(pt_boot, &pt_m4_0, SC_FALSE, SC_TRUE,
    SC_FALSE, SC_TRUE, SC_FALSE));
```

- c. 将 M4 子系统的所有资源和引脚标记为可移动。

```
/* Mark all M4_0 subsystem resources as movable */
BRD_ERR(rm_set_subsys_rsrc_movable(pt_boot, SC_R_M4_0_PID0,
    SC_TRUE));
BRD_ERR(rm_set_pad_movable(pt_boot, SC_P_ADC_IN1, SC_P_ADC_IN2,
    SC_TRUE));
```

- d. 将 M4 内核需要使用的资源和引脚标记为可移动。

通常这部分需要根据板的设计和用例进行修改。

```
/* Move some resources not in the M4_0 subsystem */
BRD_ERR(rm_set_resource_movable(pt_boot, SC_R_SYSTEM,
    SC_R_SYSTEM, SC_TRUE));
BRD_ERR(rm_set_resource_movable(pt_boot, SC_R_IRQSTR_M4_0,
    SC_R_IRQSTR_M4_0, SC_TRUE));
BRD_ERR(rm_set_resource_movable(pt_boot, SC_R_MU_5B,
    SC_R_MU_5B, SC_TRUE));
```

```

/* Move some pads not in the M4_0 subsystem */
BRD_ERR(rm_set_pad_movable(pt_boot, SC_P_FLEXCAN0_RX,
    SC_P_FLEXCAN2_TX, SC_TRUE));
BRD_ERR(rm_set_pad_movable(pt_boot, SC_P_USB_SS3_TC1,
    SC_P_USB_SS3_TC1, SC_TRUE));
BRD_ERR(rm_set_pad_movable(pt_boot, SC_P_USB_SS3_TC3,
    SC_P_USB_SS3_TC3, SC_TRUE));
BRD_ERR(rm_set_pad_movable(pt_boot, SC_P_QSPI0A_DATA0,
    SC_P_COMP_CTL_GPIO_1V8_3V3_QSPI0B, SC_TRUE));

```

- e. 将所有标记为可移动的资源 and 引脚移动到 M4 分区。

```

/* Move everything flagged as movable */
BRD_ERR(rm_move_all(pt_boot, pt_boot, pt_m4_0, SC_TRUE,
    SC_TRUE));

```

- f. 为 M4 分区分配存储器区域。

通常，DDR 中的存储器区域需要根据板的 DRAM 大小进行调整。

```

/* Move M4_0 TCM */
BRD_ERR(rm_find_memreg(pt_boot, &mr_m4_0, 0x034FE0000ULL,
    0x034FE0000ULL));
BRD_ERR(rm_assign_memreg(pt_boot, pt_m4_0, mr_m4_0));

/* Reserve DDR for M4_0 */
BRD_ERR(rm_memreg_frag(pt_boot, &mr_m4_0,
    0x088000000ULL, 0x08FFFFFFFULL));
BRD_ERR(rm_assign_memreg(pt_boot, pt_m4_0, mr_m4_0));

/* Reserve FlexSPI for M4_0 */
BRD_ERR(rm_memreg_frag(pt_boot, &mr_m4_0, 0x08081000ULL,
    0x08180FFFULL));
BRD_ERR(rm_assign_memreg(pt_boot, pt_m4_0, mr_m4_0));

```

- g. 授予其他分区访问 M4 资源的权限。

这部分也可以根据用例进行自定义。

```

/* Allow all to access the SEMA42 */
BRD_ERR(rm_set_peripheral_permissions(pt_m4_0,
    SC_R_M4_0_SEMA42, SC_RM_PT_ALL, SC_RM_PERM_FULL));

```

有关上面使用的 SCFW API 的详细说明，请参见 [sc\\_fw\\_port.pdf](#) 中的第 16 章。

2. 在 `board_ioctl()` 函数中

在某些用例中，用户可能需要在 SCFW 中添加自己的板级函数或功能。请使用 `board.c` 中的 `board_ioctl()` 函数。

```

/*-----*/
/* Board IOCTL function */
/*-----*/
sc_err_t board_ioctl(sc_rm_pt_t caller_pt, sc_rsrc_t mu, uint32_t *parm1,
    uint32_t *parm2, uint32_t *parm3)

```

使用 SCFW API, `sc_misc_board_ioctl`, 从 Linux 或 M4 进入 `board.c` 中的 `board_ioctl()` 函数。



16.29.2.22 sc\_misc\_board\_ioctl()

```
sc_err_t sc_misc_board_ioctl (
    sc_ipc_t ipc,
    uint32_t * parm1,
    uint32_t * parm2,
    uint32_t * parm3 )
```

This function calls the board IOCTL function.

Parameters

in	ipc	IPC handle
in, out	parm1	pointer to pass parameter 1
in, out	parm2	pointer to pass parameter 2
in, out	parm3	pointer to pass parameter 3

Returns

Returns and error code (SC\_ERR\_NONE = success).

sc\_misc\_board\_ioctl() 函数几乎直接被传递给 board\_ioctl() 函数。通过指针传递和返回了三个参数，并返回一个错误代码。用户可以为这三个参数定义含义，并在 board\_ioctl() 函数中实施自己的功能。此调用未与任何资源关联，因此并不安全。MU (API 调用的来源) 被传入，而拥有该 MU 的分区号也被传入。这样可以实现某种程度的安全性。

3. 在 board\_parameter() 函数中

board.c 中的 board\_parameter() 函数，顾名思义，就是用来配置板级参数的。它包括 PCIe PLL 时钟源、KS1 模式设置和用于显示的展频功能。

修改每个参数的返回值，为板选择所需的配置。例如，要使用外部时钟作为 PCIe PLL 的源时钟，请将 board\_parameter() 设置如下：

```
case BOARD_PARM_PCIE_PLL :
    rtn = BOARD_PARM_RTN_EXTERNAL;
    break;
```

要使用内部时钟作为 PCIe PLL 的源时钟，请将 board\_parameter() 设置如下：

```
case BOARD_PARM_PCIE_PLL :
    rtn = BOARD_PARM_RTN_INTERNAL;
    break;
```

有关所有可用的参数配置，请参见 sc\_fw\_port.pdf 中的第 4.4.1 章“板参数配置”，或 SCFW\_DIR/platform/main/board.h 中的标头文件。

3.3 编译 SCFW

要编译 SCFW，请按以下步骤操作：

1. 设置构建环境

在 Linux 环境中，SCFW builds 是使用交叉编译器进行编译的。编译所用的工具链应从 [GNU Arm Embedded Toolchain Downloads \(下载 GNU Arm 嵌入式工具链\)](#) 中获取。使用的版本是 GNU Arm 嵌入式工具链：8-2018-q4-major December 20, 2018。请下载工具链源代码，并按照说明在 Linux 主机上安装工具链。安装完成后，需要将环境变量“TOOLS (工具)”设置为包含编译器目录的目录。例如，如果使用 GCC 4.9 交叉编译工具链并安装到 /home/example/gcc-arm-none-eabi-8-2018-q4-major，请将“TOOLS (工具)”设置为 /home/example。构建还需要 srec\_cat，通常可以在 Linux srecord 软件包中获取。cppcheck 软件包也很有用。

如果使用 `bash`，则按以下方式设置“TOOLS（工具）”环境变量：

```
$export TOOLS=<your path to dir holding the toolchain>
```

2. 编译代码

固件可以使用Makefile完整编译。命令格式为：

```
Usage: make TARGET OPTIONS
```

SCFW 的目标基于芯片，而不是型号。一些部件是其他芯片的phantom（例如 QP 是 QM 的 phantom），通过融合选项创建。phantom 运行时，SCFW 读取熔丝并调整，为其提供支持。有三大芯片目标：

- qm: i.MX8QM 芯片
- qx: i.MX8QX 芯片
- dxl: i.MX8DXL 芯片

它们在各自的 build 目录中生成镜像（scfw\_tcm.bin）。参见 [表 1](#)，了解可以在 make 命令行上指定的选项。

表 1. make 命令行上的选项

选项	动作
V=0	静默输出（默认）
V=1	详细输出
D=0	配置为无调试
D=1	配置为调试（默认）
DL=<level>	配置调试级别（0-5）
M=0	无调试监视器（默认）
M=1	有调试监视器
B=<board>	配置板（默认值=val）
U=<uart>	配置调试UART（默认值=0）
DDR_CON=<file>	指定不带扩展名的DDR配置文件
R=<srev>	芯片版本
T=<test>	运行测试，而不是启动下一个内核

本应用笔记以汽车用 i.MX 8QXP 板为示例板，因此板级文件夹名称为 mx8qx\_auto，DDR 脚本名称为 imx8qxp\_auto\_samsung3GB\_1.2GHz\_v14.cfg。编译命令为：

```
$make qx R=b0 B=auto M=1 U=2 DDR_CON=imx8qxp_auto_samsung3GB_1.2GHz_v14
```

注：对于 MX8QX，R=b0 适用于 B0 和 C0 芯片版本。换句话说，即使您正在为 C0 芯片而构建，并使用 C0 芯片，也仍然需要使用 R=b0。

如果编译成功，则可以在 SCFW\_DIR/build\_mx8qx\_b0/scfw\_tcm.bin 路径下找到 SCFW 二进制文件。

注：

对于 [章节 4](#) 中的 DDR 压力测试，需要使用带特殊 DDR 脚本 ddr\_stress\_test\_parser.cfg 的 SCFW。因此，除了标准的 SCFW 外，用户还需要使用以下命令为 DDR 压力测试编译特殊的 SCFW：

```
$make qx R=b0 B=auto DDR_CON=ddr_stress_test_parser
```

## 4 运行 DDR 压力测试

MX8 DDR 压力测试是一个软件应用，用于验证 i.MX8 系列板上的 DDR 接口。它是在 PC 端运行的程序，通过 USB 连接将测试镜像下载到 i.MX 系列处理器的内部 RAM。因为它需要访问 Windows Registry（Windows 注册表），用户必须在管理员模式下运行。在目标板上运行的测试镜像执行 DDR 压力测试。结果通过 A 核 UART 发送到 PC，并在日志窗口中显示。还有可以将输出保存到日志文件的选项。

MX8 DDR 压力测试有助于在非操作系统环境下验证板上的 DDR 稳定性。

要运行 DDR 压力测试工具，请按以下步骤操作：

1. 从 [i.MX 8/8X Family DDR Tools Release \(i.MX 8/8X系列 DDR 工具版\)](#) 下载 DDR 压力测试工具。下载并安装后，该工具位于安装路径下的 `mx8_ddr_stress_test_ER14` 文件夹。有关 DDR 压力测试工具的更多详细信息，请参见 `MX8_DDR_Tool_User_Guide.pdf`。

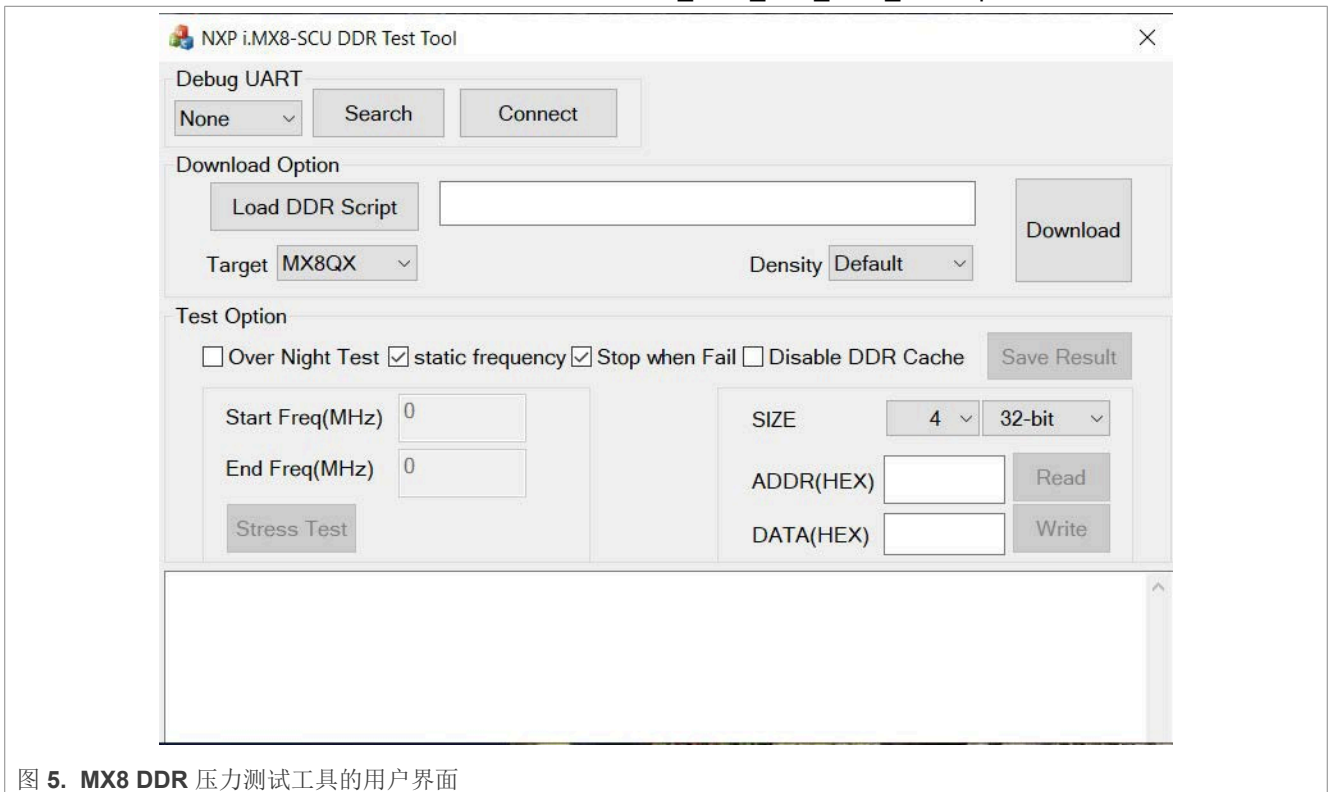


图 5. MX8 DDR 压力测试工具的用户界面

2. 准备以下两个文件：

- DDR 压力测试的 DDR 脚本

此文件在 [章节 2.2](#) 的 [步骤 4](#) 中生成。在本例中，DDR 脚本的名称为 `imx8qxp_auto_board.ds`，该文件可以放在 `mx8_ddr_stress_test_ER14\script\mx8qxp\imx8qxp_auto_board.ds` 文件夹中。

- 特殊的 SCFW

这个文件是在 [章节 3.3](#) 中生成，带有 DDR 脚本，`ddr_stress_test_parser.cfg`。将 SCFW 二进制文件的名称从 `scfw_tcm.bin` 改为 `mx8qxb0_scfw_download.bin`，并替换 `mx8_ddr_stress_test_ER14\bin\mx8qxb0_scfw_download.bin` 工具文件夹中的文件。

注：对于 MX8QX，此名称适用于 B0 和 C0 芯片版本。换句话说，即使您正在为 C0 芯片而构建，并使用 C0 芯片，您也仍然需要将 scfw\_tc.bin 重命名为 mx8qxb0\_scfw\_download.bin。

3. 将目标板连接到 PC 主机。
  - a. 将 i.MX 目标板配置为在 Serial Download（串行下载）模式/Manufacture（制造）模式下启动，并为该板通电。
  - b. 用 UART 线缆连接主机和 MX8 调试 UART。请注意，Win10 可能需要手动安装 COM 端口驱动程序（FTDI、Silab 等）。
  - c. 使用 USB 线缆连接主机与 MX8 目标板的 USB OTG 端口。HID 兼容的设备或 USB 输入设备将在 Device Manager（设备管理器）中显示。请注意，对于 MX8 USB OTG 连接，USB 线缆必须直接连接到主 PC 的 USB 端口，而不是通过 USB HUB。

4. 启动工具文件夹中的 MX8\_DDR\_Tester.exe。

注：对于 Win10，用鼠标右键单击 MX8\_DDR\_Tester.exe 并选择 Run as administrator（以管理员的身份）运行，以查看和选择可用的 COM 端口。

5. 按下“Debug UART（调试 UART）”区域的“Search（搜索）”按钮，选择连接到 MX8 Cortex A-Core Debug UART 的 UART 端口，然后按下“Connect（连接）”按钮。

注：如需查看并选择可用的 COM 端口，请在管理员模式下运行 DDR 压力测试。

6. 加载 DDR 初始化脚本，并选择正确的下载选项。

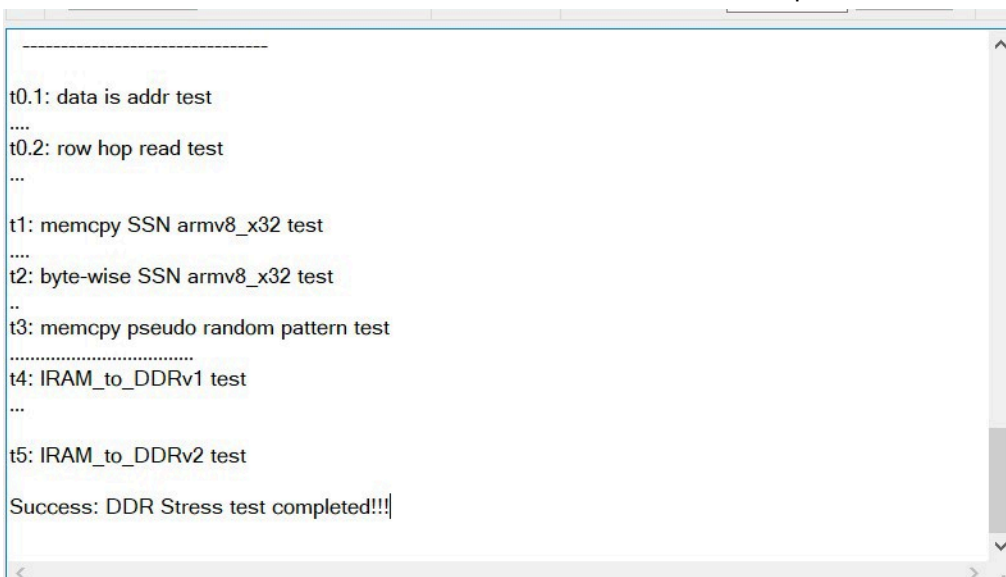
在本例中，我们选择 mx8\_ddr\_stress\_test\_ER14\script\mx8qx\mx8qxp\_auto\_board.ds 路径下的脚本。

7. 按下“Download（下载）”按钮，等待目标板就绪。

如果目标板成功启动，工具控制台上会显示 DDR 初始化信息。

8. 按下“Stress Test（压力测试）”按钮，使用所有默认设置：默认 DDR 频率、启用缓存、单循环 DDR 压力测试、出现错误时终止。

如果板成功通过 DDR 压力测试，则 Success: DDR Stress test completed!!! 的日志如下所示：



```
-----  
t0.1: data is addr test  
....  
t0.2: row hop read test  
...  
t1: memcpy SSN armv8_x32 test  
....  
t2: byte-wise SSN armv8_x32 test  
..  
t3: memcpy pseudo random pattern test  
.....  
t4: IRAM_to_DDRv1 test  
...  
t5: IRAM_to_DDRv2 test  
  
Success: DDR Stress test completed!!!
```

9. 选择“Over Night Test”，再次按下“Stress Test（压力测试）”按钮，DDR 压力测试的无限循环启动。通常，为了增加对 DDR 稳定性的信心，板需要通过超过 12 小时的 DDR 压力测试，甚至可能需要在高温/低温下重复相同的测试。



## 5 ATF 移植

ATF 是用于 Arm A-Profile 架构（Armv8-A 和 Armv7-A）安全环境软件的实施参考，包括在异常级别 3（EL3）执行的安全监视器。它为在 AArch32 或 AArch64 执行状态下生产安全环境启动和运行时固件提供了合适的起点。

ATF 采用 Arm 接口标准，包括：

- 电源状态协调接口（PSCI）
- 电源状态协调接口（PSCI）
- SMC 调用约定
- 系统控制和管理接口（SCMI）
- 软件委托异常接口（SDEI）

该代码旨在跨基于 Armv8-A 和 Armv7-A 架构的硬件平台和软件模型移植和重用。鼓励用户对 ATF 的任何安全环境代码自行进行安全验证，包括渗透测试。

对于 i.MX8 芯片，所有 i.MX8 板都需要 ATF。在为新的板级进行移植时，通常需要自定义两个部分：电源管理和资源分区。

### 5.1 电源管理

如上所述，ATF 为 Linux 系统提供 PSCI，作为电源管理机制。每个 SOC 平台都可以有特定的 PSCI 实施。下面以 i.MX8QXP 的 PSCI 实施为例。

按照 [章节 1.3](#) 将 ATF 源代码下载到 arm-trusted-firmware 文件夹后，i.MX8QXP 的 PSCI 实施代码位于 arm-trusted-firmware/plat/imx/imx8qx/imx8qx\_psci.c。

如下所示，iMX 8QXP 平台特有的各种 PSCI 操作都在 imx\_plat\_PSCI\_ops 结构中定义，映射到各个实施功能。

```
static const plat_psci_ops_t imx_plat_psci_ops = {
    .pwr_domain_on = imx_pwr_domain_on,
    .pwr_domain_on_finish = imx_pwr_domain_on_finish,
    .validate_ns_entrypoint = imx_validate_ns_entrypoint,
    .system_off = imx_system_off,
    .system_reset = imx_system_reset,
    .system_reset2 = imx_system_reset2,
    .pwr_domain_off = imx_pwr_domain_off,
    .pwr_domain_suspend = imx_domain_suspend,
    .pwr_domain_suspend_finish = imx_domain_suspend_finish,
    .get_sys_suspend_power_state = imx_get_sys_suspend_power_state,
    .validate_power_state = imx_validate_power_state,
    .pwr_domain_pwr_down_wfi = imx_pwr_domain_pwr_down_wfi,
};
```

由于在 i.MX 8 架构中，所有子系统的电源域都由 SCU 控制，因此大多数功能都在这个文件中实施，大多数操作都调用 SCFW API 来执行与电源相关的操作。如果用户要求使用某种特定的电源模式，可以在这里修改实施功能。

对于其他一些 i.MX 8 常用函数（如 system\_off 和 system\_reset），请参见 arm-trusted-firmware/plat/imx/common/imx8\_psci.c。

```

void __dead2 imx_system_off(void)
{
    sc_pm_set_sys_power_mode(ipc_handle, SC_PM_PW_MODE_OFF);
    wfi();
    ERROR("power off failed.\n");
    panic();
}

void __dead2 imx_system_reset(void)
{
    sc_pm_reboot(ipc_handle, SC_PM_RESET_TYPE_COLD);
    wfi();
    ERROR("system reset failed.\n");
    panic();
}

```

用户可能需要修改 `imx_system_reset` 函数。默认情况下，`imx_system_reset` 函数会调用 `sc_pm_reboot` SCFW API 进行分区重启，这意味着只会重启 A 核的分区，而 M4 的分区不会受到影响。

#### 13.4.3.24 sc\_pm\_reboot()

```

void sc_pm_reboot (
    sc_ipc_t ipc,
    sc_pm_reset_type_t type )

```

This function is used to reboot the caller's partition.

但在某些情况下，用户可能需要在 Linux 系统复位时对整个板进行重置。在这种情况下，在 `imx_system_reset` 函数中使用 `sc_pm_reset` SCFW API，而不是 `sc_pm_reboot`。但是请注意，只有 SC\_R\_SYSTEM 资源的所有者或具有 SC\_R\_SYSTEM 访问权限的分区才能调用 `sc_pm_reset` 重置整个板。

#### 13.4.3.19 sc\_pm\_reset()

```

sc_err_t sc_pm_reset (
    sc_ipc_t ipc,
    sc_pm_reset_type_t type )

```

This function is used to reset the system.

## 5.2 资源分区

对于 i.MX 8 芯片，除了电源管理，ATF 还有一个重要作用，即为 A 核的非安全环境创建资源分区。以 i.MX8QXP 为例，创建资源分区是在 `imx8_partition_resources` 函数中的 `arm-trusted-firmware/plat/imx/imx8qx/imx8qx_bl31_setup.c` 中完成的。

在 ATF 中为非安全环境分区创建和分配资源的过程与 [章节 3.2](#) 中 M4 的过程类似。通常按以下步骤操作：

1. 为非安全环境创建一个分区，并将父分区设置为 ATF 分区。

```
err = sc_rm_partition_alloc(ipc_handle, &os_part, false, false,
                           false, false, false);
if (err)
    ERROR("sc_rm_partition_alloc failed: %u\n", err);

err = sc_rm_set_parent(ipc_handle, os_part, secure_part);
if (err)
    ERROR("sc_rm_set_parent: %u\n", err);
```

2. 将需要保存在 ATF 分区中的所有资源标记为不可移动。

```
/* iterate through peripherals to give NS OS part access */
for (i = 0; i < ARRAY_SIZE(ns_access_allowed); i++) {
    err = sc_rm_set_peripheral_permissions(ipc_handle,
        ns_access_allowed[i], os_part, SC_RM_PERM_FULL);
    if (err)
        ERROR("sc_rm_set_peripheral_permissions: rsrc %u, \
            ret %u\n", ns_access_allowed[i], err);
}
```

secure\_rsrcs[] 阵列在 arm-trusted-firmware/plat/imx/imx8qx/include/sec\_rsrc.h 中定义，包含将留在 ATF 分区中的资源。如果用户有特殊要求，可以对其进行修改。

3. 为非安全环境分配存储器区域。根据是否实施了 OP-TEE 或 Trusty，存储器区域将相应地改变。  
注：

如果实施了 OP-TEE，

- 当 DRAM 等于或大于 2 GB 时，存储器区域 0xFE000000 - 0xFFFFFFFF 默认由 OP-TEE 使用。这在 arm-trusted-firmware/plat/imx/imx8qx/platform.mk 中设置，带 BL32\_BASE 和 BL32\_SIZE。
- 如果新板的 DRAM 小于 2 GB，则需要将 BL32\_BASE 修改为最高内存地址：BL32\_SIZE。例如，如果 DRAM 为 1 GB，则 BL32\_BASE 为 0xBE000000。

4. 将所有可移动资源和引脚移动到非安全环境分区。

```
/* move all movable resources and pins to non-secure partition */
err = sc_rm_move_all(ipc_handle, secure_part, os_part, true, true);
if (err)
    ERROR("sc_rm_move_all: %u\n", err);
```

5. 将某些资源的访问权限授予非安全环境分区。

```
/* set secure resources to NSI-movable */
for (i = 0; i < (ARRAY_SIZE(secure_rsrcs)); i++) {
    err = sc_rm_set_resource_movable(ipc_handle,
        secure_rsrcs[i], secure_rsrcs[i], false);
    if (err)
        ERROR("sc_rm_set_resource_movable: rsrc %u, ret %u\n",
            secure_rsrcs[i], err);
}
```

## 5.3 编译 ATF

要编译 ATF，请按以下步骤操作：

1. 设置构建环境

用于编译 ATF 的工具链与后面章节中用于编译 U-Boot 和 Linux 内核的交叉编译工具链相同。有关如何生成和安装工具链，请参见 i.MX Linux 用户指南（文档 [IMX LUG](#)）中第 4.5.12 章“如何在独立环境中构建 U-Boot 和 Kernel”。

[https://www.nxp.com/webapp/Download?colCode=L5.4.47\\_2.2.0\\_LINUX\\_DOCS](https://www.nxp.com/webapp/Download?colCode=L5.4.47_2.2.0_LINUX_DOCS)

2. 编译代码

- 如果没有实施 OP-TEE，请使用以下命令编译 ATF。

```
$ make PLAT=imx8qx bl31
```

- 如果实施了 OP-TEE，请使用以下命令编译 ATF。

```
$ make PLAT=imx8qx SPD=opteed bl31
```

如果编译成功，二进制文件将位于 `arm-trusted-firmware/build/imx8qx/release/bl31.bin` 中。

### 3. 启用调试打印

默认情况下，ATF 中未启用调试打印。要启用调试打印，用户需要在 `arm-trusted-firmware/ plat/imx/imx8qx/ include/platform_def.h` 中将 `DEBUG_CONSOLE` 和 `DEBUG_CONSOLE_A35` 定义为 1。

使用以下命令编译 ATF。

```
$ make DEBUG=1 PLAT=imx8qx SPD=opteed bl31
```

二进制文件将位于 `arm-trusted-firmware/build/imx8qx/debug/bl31.bin` 中。

## 6 U-Boot 移植

通用启动加载程序（通常简称为 U-Boot）是一种开源的主要启动加载程序，在嵌入式设备中使用，用于打包启动设备操作系统内核的指令。

U-Boot 既是第一阶段的启动加载程序，也是第二阶段的启动加载程序。它由系统的 ROM 或 BIOS 从受支持的启动设备（如 SD 卡、SATA 驱动、NOR flash（如使用 SPI 或 I<sup>2</sup>C）或 NAND flash）加载。如果有大小限制，U-Boot 会分为几个阶段：

- 平台加载小型 SPL（辅助加载程序），这是 U-Boot 的精简版。
- SPL 初始化硬件配置并加载更大、功能齐全的 U-Boot 版本。

无论是否使用 SPL，U-Boot 都会执行第一阶段（例如，配置存储器控制器和 SDRAM）和第二阶段启动（执行多个步骤，从各种必须配置的设备加载现代操作系统，为用户提供了一个菜单，以便其与启动流程进行交互和进行控制等）。

i.MX8 芯片支持 SPL 或非 SPL U-Boot。在 Linux BSP L5.4.47\_2.2.0 及未来版本中，SPL 默认是启用的。

### 6.1 为新的板级创建文件

按照 [章节 1.3](#) 中的步骤进行操作后，将 U-Boot 源代码下载到 `uboot-imx` 文件夹。

要为新板级移植 U-Boot，请为新板级创建 [表 2](#) 列出的文件。为了减少移植工作，用户可以从 MEK 参考板的文件中复制文件，并根据自己的具体要求修改。

在下文中，使用 `imx8qxp_auto` 作为板级名称，用于为新板级创建的文件。用户可以为自己的板级修改文件名。

表 2. U-Boot 中新板级所需的文件

文件位置	描述
<code>configs/imx8qxp_auto_defconfig</code>	汽车板的 <code>defconfig</code> 文件
<code>board/freescale/imx8qxp_auto/</code>	汽车板的板级文件夹
<code>board/freescale/imx8qxp_auto/Kconfig</code>	汽车板的 <code>Kconfig</code> 文件
<code>board/freescale/imx8qxp_auto/Makefile</code>	汽车板级文件夹中 C 文件的 <code>makefile</code>
<code>board/freescale/imx8qxp_auto/imx8qxp_auto.c</code>	实施与板级相关的初始化函数的板级文件，如 <code>board_init()</code>
<code>board/freescale/imx8qxp_auto/imximage.cfg</code>	用于配置 <code>imx8</code> 启动镜像的文件
<code>board/freescale/imx8qxp_auto/spl.c</code>	板级相关实施文件，用于 SPL 启动



表 2. U-Boot 中新板级所需的文件...续上页

文件位置	描述
board/freescale/imx8qxp_auto/uboot-container.cfg	用于创建容器镜像的文件，可由 SPL 加载
include/configs/imx8qxp_auto.h	汽车板的标头文件
arch/arm/mach-imx/imx8/snvs_security_sc_conf_8qxp_auto.h	汽车板 snvs 安全配置的标头文件
arch/arm/dts/fsl-imx8qxp-auto.dts	汽车板的设备树文件
arch/arm/dts/fsl-imx8qxp-auto-u-boot.dtsi	设备树包含为 SPL 生成 dts 的文件，请查看 doc/README.SPL，了解更多详细信息

## 6.2 修改新板级的文件

要修改新板级的文件，请按以下步骤操作：

1. 修改一些现有文件，在 U-Boot 中添加新的板级。
  - a. 在 arch/arm/mach-imx/imx8/Kconfig 中，添加 TARGET\_IMX8QXP\_AUTO 和汽车板的 Kconfig 文件。

```
diff --git a/arch/arm/mach-imx/imx8/Kconfig b/arch/arm/mach-imx/imx8/Kconfig
index 52387462c0..45529b2796 100644
--- a/arch/arm/mach-imx/imx8/Kconfig
+++ b/arch/arm/mach-imx/imx8/Kconfig
@@ -138,6 +138,11 @@ config TARGET_IMX8QXP_MEK
     select BOARD_LATE_INIT
     select IMX8QXP

+config TARGET_IMX8QXP_AUTO
+    bool "Support i.MX8QXP AUTO board"
+    select BOARD_LATE_INIT
+    select IMX8QXP
+
 config TARGET_IMX8QXP_LPDDR4_VAL
     bool "Support i.MX8QXP lpddr4 validation board"
     select BOARD_LATE_INIT
@@ -178,6 +183,7 @@ endchoice

source "board/freescale/imx8qm_mek/Kconfig"
source "board/freescale/imx8qxp_mek/Kconfig"
+source "board/freescale/imx8qxp_auto/Kconfig"
source "board/freescale/imx8qm_val/Kconfig"
source "board/freescale/imx8qxp_val/Kconfig"
source "board/freescale/imx8dxl_phantom_mek/Kconfig"
```

- b. 在 arch/arm/dts/Makefile 中，为汽车板添加 dtb 文件。

```
diff --git a/arch/arm/dts/Makefile b/arch/arm/dts/Makefile
index 81ac2569c6..ef3d285352 100644
--- a/arch/arm/dts/Makefile
+++ b/arch/arm/dts/Makefile
@@ -769,6 +769,7 @@ dtb-$(CONFIG_ARCH_IMX8) += \
    fsl-imx8qxp-ai_ml.dtb \
    fsl-imx8qxp-colibri.dtb \
    fsl-imx8qxp-mek.dtb \
+   fsl-imx8qxp-auto.dtb \
    fsl-imx8qxp-lpddr4-val.dtb \
    fsl-imx8qxp-lpddr4-val-gpmi-nand.dtb \
    fsl-imx8qxp-l7x17-val.dtb \
```

- c. 在 arch/arm/mach-imx/imx8/snvs\_security\_sc\_conf\_board.h 中，为汽车板的 snvs 安全配置标头文件添加 include。

```
diff --git a/arch/arm/mach-imx8/snvs_security_sc_conf_board.h b/arch/arm/mach-imx8/snvs_security_sc_conf_board.h
index 250952b7df..1c8bcb0980 100644
--- a/arch/arm/mach-imx8/snvs_security_sc_conf_board.h
+++ b/arch/arm/mach-imx8/snvs_security_sc_conf_board.h
@@ -10,6 +10,8 @@
#include "snvs_security_sc_conf_8qm_mek.h"
#elif CONFIG_TARGET_IMX8QXP_MEK
#include "snvs_security_sc_conf_8qxp_mek.h"
#elif CONFIG_TARGET_IMX8QXP_AUTO
#include "snvs_security_sc_conf_8qxp_auto.h"
#elif CONFIG_TARGET_IMX8DXL_EVK
#include "snvs_security_sc_conf_8dxl_evk.h"
#else
```

2. 对这些创建的文件进行必要的修改。在某些文件中，修改的工作量很少，例如只需更改板级名和文件路径。本文档不会列出所有的详细修改，而是重点关注用户自定义板级时通常需要多关注和考虑的地方。

a. 在 `configs/imx8qxp_auto_defconfig` 中

`defconfig` 文件是编译过程的重要配置文件，定义了板级所需的模块。

以我们的汽车为例，对比 `MEK` 板级的 `defconfig` 文件，对汽车板的 `defconfig` 进行以下更改。

```
-CONFIG_IMX_CONTAINER_CFG="board/freescale/imx8qxp_mek/uboot-
container.cfg"
-CONFIG_TARGET_IMX8QXP_MEK=y
-CONFIG_SYS_EXTRA_OPTIONS="IMX_CONFIG=board/freescale/imx8qxp_mek/
imximage.cfg"
-CONFIG_DEFAULT_DEVICE_TREE="fsl-imx8qxp-mek"
+CONFIG_IMX_CONTAINER_CFG="board/freescale/imx8qxp_auto/uboot-
container.cfg"
+CONFIG_TARGET_IMX8QXP_AUTO=y
+CONFIG_SYS_EXTRA_OPTIONS="IMX_CONFIG=board/freescale/imx8qxp_auto/
imximage.cfg"
+CONFIG_DEFAULT_DEVICE_TREE="fsl-imx8qxp-auto"
```

除了这些变化，还有与板级设计或要求有关的变化。

例如，在我们的汽车板上，`USB 3.0` 端口仅用作“`Host（主机）`”模式。在 `defconfig` 中删除对“`Gadget（设备）`”模式的支持，如下所示：

```
-CONFIG_USB_CDNS3_GADGET=y
+#CONFIG_USB_CDNS3_GADGET=y
```

注：默认情况下，`u-boot` 会为 `M4` 程序预留 `128 MB` 的存储器区域，从 `0x88000000` 到 `0x90000000`，这部分是通过 `defconfig` 中的 `CONFIG_BOOTAUX_RESERVED_MEM_BASE` 和 `CONFIG_BOOTAUX_RESERVED_MEM_SIZE` 定义的。要更改 `M4` 程序要使用的存储器区域大小，请根据 [章节 3.2](#) 修改此“`CONFIG（配置）`”和存储器区域分配。

要在新板级上启用功能和驱动程序，请将相关 `CONFIG` 添加到板级的 `defconfig` 文件中。

b. 在 `board/freescale/imx8qxp_auto/imx8qxp_auto.c` 中

该文件包含与板级相关的初始化函数，如 `board_init()`。用户可以在这里实施自己的板级特定的初始化函数，或为各种模块修改现有的初始化函数。例如，如果用户在初始化时需要设置 `GPIO` 引脚，就要在 `board_gpio_init()` 函数中添加这些 `GPIO` 引脚。然后要使用 `dm_gpio_lookup_name()` 找到目标 `GPIO`，使用 `dm_gpio_request()` 请求 `GPIO`，使用 `dm_gpio_set_dir_flags()` 设置 `GPIO` 引脚的方向和标志，最后使用 `dm_gpio_set_value()` 设置目标 `GPIO` 的输出。

```
static void board_gpio_init(void)
{
    struct gpio_desc desc;
    int ret;

    ret = dm_gpio_lookup_name("gpio@1a_3", &desc);
    if (ret)
        return;

    ret = dm_gpio_request(&desc, "bb_per_rst_b");
    if (ret)
        return;

    dm_gpio_set_dir_flags(&desc, GPIOD_IS_OUT);
    dm_gpio_set_value(&desc, 0);
    udelay(50);
    dm_gpio_set_value(&desc, 1);
}
```

c. 在 `include/configs/imx8qxp_auto.h` 中

这个板级的标头文件包含许多与板级相关的宏定义。要修改的两个最常见部分是 DRAM 大小和 ENV 设置。

DRAM 的大小通过以下命令定义：

```
#define PHYS_SDRAM_1 0x80000000
#define PHYS_SDRAM_2 0x880000000
#define PHYS_SDRAM_1_SIZE 0x80000000 /* 2 GB */
#define PHYS_SDRAM_2_SIZE 0x40000000 /* 1 GB */
```

`PHYS_SDRAM_1` 定义 DRAM 的低位基址，`PHYS_SDRAM_2` 定义高位基址。DRAM 的总大小为 `PHYS_SDRAM_1_SIZE + PHYS_SDRAM_2_SIZE`。

如果 DRAM 小于 2 GB，则 `PHYS_SDRAM_2_SIZE` 为 0。

用户需要根据板级上的 DDR 芯片修改 `PHYS_SDRAM_1_SIZE` 和 `PHYS_SDRAM_2_SIZE` 的定义值。

至于 ENV 设置，标头文件中已经定义了许多 ENV 设置。用户可以在 `CONFIG_EXTRA_ENV_SETTINGS` 中添加新的 ENV 设置，并根据需求修改现有的 ENV 设置，例如 `fdt_file` 和 `mmcargs`。

ENV 设置也可以在 U-Boot 控制台中通过 `setenv` 和 `saveenv` 命令动态更改。

d. 在 `arch/arm/dts/fsl-imx8qxp-auto.dts` 中

设备树架构首先在 Linux 内核中引入，并在 U-Boot 中实施。dts 文件是描述板级硬件组件的数据结构，以便系统可以使用和管理这些组件。

用户可以根据自己的板级设计添加或删除设备节点，以修改一些设备节点的参数，如时钟频率和控制引脚。

对于 SPL，还有一个设备树。要减少 SPL 的大小，请仅在其设备树中保留具有预重定位属性（`u-boot`、`dm-pre-reloc`、`u-boot`、`dm-spl`）的节点。用户可以查看 `arch/arm/dts/fsl-imx8qxp-mek-u-boot.dtsi` 示例。

## 6.3 编译 U-Boot

### 1. 设置构建环境

用于编译 U-Boot 的工具链与“Compiling ATF（编译 ATF）”中的交叉编译工具链相同。有关如何生成和安装工具链，请参见 *i.MX Linux User's Guide*（文档 [IMXLUG](#)）中第 4.5.12 章“如何在独立环境中构建 U-Boot 和 Kernel”。

[https://www.nxp.com/webapp/Download?colCode=L5.4.47\\_2.2.0\\_LINUX\\_DOCS](https://www.nxp.com/webapp/Download?colCode=L5.4.47_2.2.0_LINUX_DOCS)

## 2. 编译代码

要为目标板级构建U-Boot，请按以下步骤操作：

- a. 使用以下命令为板级生成配置文件。以imx8qxp\_auto board为例。

```
$ make imx8qxp_auto_defconfig
```

- b. 使用以下命令为目标板级生成 U-Boot。如果在配置文件中选择了 CONFIG\_SPL，则会生成 SPL 镜像。

```
$ make -j8
```

U-Boot 镜像位于 uboot-imx/u-boot.bin 中，SPL 镜像位于 uboot-imx/spl/u-boot-spl.bin 中。

## 7 构建 flash.bin 镜像

对于 i.MX 8 芯片，flash.bin 镜像（实际上是启动镜像容器集）可以包括 SCFW、SECO FW、M4 镜像、ATF 镜像、U-Boot 镜像和 SPL 镜像。ROM 代码从启动设备读取 flash.bin 镜像，并根据设置将其加载到不同的存储器地址。

为了简化生成 flash.bin 镜像的过程，可以使用 imx-mkimage 工具将上述镜像组合在一起，生成最终的 flash.bin 启动镜像，并刻入启动设备。

### 7.1 将镜像复制到 mkimage

在 [章节 1.3](#) 中完成以下步骤后，下载 imx-mkimage 文件夹中的 imx-mkimage 源代码。内容包括所有受支持的 iMX8 芯片的文件夹。

```
COPYING  iMX8DXL  iMX8QM  iMX8ULP  mkimage_imx8  scripts
iMX8dv  iMX8M  iMX8QX  Makefile  README  src
```

要生成 flash.bin 镜像，请将所有需要的镜像复制到目标芯片的文件夹中。以 imx8qxp 汽车板为例，按照以下步骤操作。

1. 将 [章节 3.3](#) 中生成的 SCFW scfw\_tcm.bin 复制到 imx-mkimage/iMX8QX/ folder 目录下。
2. 将 SECO FW mx8qxc0-ahab-container.img 复制到 imx-mkimage/iMX8QX/ 文件夹。使用以下命令下载镜像：

```
$wget https://www.nxp.com/lgfiles/NMG/MAD/YOCTO/imx-seco-3.7.1.bin
$chmod a+x imx-seco-3.7.1.bin
$cd imx-seco-3.7.1/firmware/seco//
```

每个 BSP 版本都有 SECO FW 版本。有关如何获得正确的 SECO FW 版本，请参见 i.MX Linux Release Notes（文档 [IMXLXRN](#)）。

3. 将 [章节 5.3](#) 中生成的 ATF 镜像 bi31.bin 复制到 imx-mkimage/iMX8QX/ 文件夹。
4. 将 [章节 6.3](#) 中生成的 U-Boot 镜像 u-boot.bin 和 SPL 镜像 u-boot-spl.bin 复制到 imx-mkimage/iMX8QX/ 文件夹。
5. 如果有 M4 程序，将 M4 镜像复制到 imx-mkimage/iMX8QX/ 文件夹，命名为 m4\_image.bin。有关如何编译 M4 镜像，请参见 [MCUXpresso SDK Builder](#)。



## 7.2 检查 makefile

flash.bin 镜像的 makefile 位于 imx-mkimage/iMX8QX/soc.mak 中。默认情况下，许多目标都是为通用用例定义的。如果目标满足其需求，用户可以使用这些目标直接构建 flash.bin 镜像，或者修改这些目标中的选项，甚至根据特定需求创建新目标。参见以下示例。

- flash

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

flash 目标包括 SCFW、SECO FW、ATF 和 U-Boot 镜像。u-boot-atf.bin 是由 ATF 镜像 bl31.bin 结合 u-boot.bin 镜像生成的。

```
u-boot-hash.bin: u-boot.bin
    ./$(MKIMG) -commit > head.hash
    @cat u-boot.bin head.hash > u-boot-hash.bin

u-boot-atf.bin: u-boot-hash.bin bl31.bin
    @cp bl31.bin u-boot-atf.bin
    @dd if=u-boot-hash.bin of=u-boot-atf.bin bs=1K seek=128
```

由于 flash 目标不包括 SPL 镜像，A核启动地址为 0x8000000，如上所示。

- flash\_regression\_linux\_m4

```
flash_regression_linux_m4: $(MKIMG) $(AHAB_IMG) scfw_tcm.bin u-boot-atf.bin m4_image.
bin
    ./$(MKIMG) -soc QX -rev B0 -append $(AHAB_IMG) -c -flags 0x00200000 -scfw scfw
    w_tcm.bin -ap u-boot-atf.bin a35 0x80000000 -p3 -m4 m4_image.bin 0 0x34FE0000 -out fl
    ash.bin
```

flash\_regression\_linux\_m4 目标添加 M4 图像与 flash 目标比较。对于这个目标，需要注意以下两个部分。

--flags 0x00200000 选项

这是在启动期间将传递给 SCFW 的启动标志。标志的定义参见 sc\_fw\_port.pdf。

表 3. 标志的定义

标志	位	含义
SC_BD_FLAGS_NOT_SECURE	16	初始启动分区不安全
SC_BD_FLAGS_NOT_ISOLATED	17	初始启动分区不是孤立的
SC_BD_FLAGS_RESTRICTED	18	初始启动分区是受限的
SC_BD_FLAGS_GRANT	19	初始启动分区授予对 SCFW 的访问权限
SC_BD_FLAGS_NOT_COHERENT	20	初始启动分区不连贯
SC_BD_FLAGS_ALT_CONFIG	21	备用 SCFW 配置（传递到 board.c）
SC_BD_FLAGS_EARLY_CPU_START	22	尽早启动一些 CPU
SC_BD_FLAGS_DDRTEST	23	DDR 压力测试的配置
SC_BD_FLAGS_NO_AP	24	即使 ROM 请求，也不启动 AP

如表 3 所示，-flags 0x00200000 表示设置了 SC\_BD\_FLAGS\_ALT\_CONFIG。这个标志在 board.c 的 board\_system\_config() 函数中使用。只有设置了该标志，SCU 才会为 M4 创建分区和分配资源。

-M4 内核的启动地址

M4 内核可根据用户的需求，从内部存储器 TCM、外部存储器 DRAM 或 NOR Flash 等外部设备启动。使用 M4 SDK 编译 M4 镜像时，每种方法都有特定的链接文件。用户需要根据选择的启动方式选择正确的链接文件，并根据链接文件中定义的地址调整此处的启动地址。

- 对于 flash\_regression\_linux\_m4\_dds 目标，假设从 TCM 启动。因此，启动地址为 0x34FE0000。
- 对于以 m4\_ddr 结尾的目标，假设是从 DDR 启动，启动地址为 0x88000000。
- 对于以 m4\_xip 结尾的目标，假设是从 QSPI NOR Flash 启动，启动地址为 0x08081000，如下所示。

```
flash_regression_linux_m4_ddr: $(MKIMG) $(AHAB_IMG) scfw_tcm.bin u-boot-atf.bin m4_image.bin
    ./$(MKIMG) -soc QX -rev B0 -append $(AHAB_IMG) -c -flags 0x00200000 -scfw scfw_tcm.bin -ap u-boot-atf.bin a35 0x80000000 -p3 -m4 m4_image.bin 0 0x88000000 -out flash.bin

flash_regression_linux_m4_xip : $(MKIMG) $(AHAB_IMG) scfw_tcm.bin u-boot-atf.bin m4_image.bin $(QSPI_HEADER)
    ./$(MKIMG) -soc QX -rev B0 -dev flexspi -append $(AHAB_IMG) -c -flags 0x00200000 -scfw scfw_tcm.bin -fileoff 0x80000 -p3 -m4 m4_image.bin 0 0x08081000 -fileoff 0x180000 -ap u-boot-atf.bin a35 0x80000000 -out flash.bin
    ./$(QSPI_PACKER) $(QSPI_HEADER)
```

- flash\_linux\_m4

```
flash_linux_m4: $(MKIMG) $(AHAB_IMG) scfw_tcm.bin u-boot-atf-container.img m4_image.bin u-boot-spl.bin
    ./$(MKIMG) -soc QX -rev B0 -dcd skip -append $(AHAB_IMG) -c -flags 0x00200000 -scfw scfw_tcm.bin -ap u-boot-spl.bin a35 0x00100000 -p3 -m4 m4_image.bin 0 0x34FE0000 -out flash.bin
    cp flash.bin boot-spl-container.img
    @flashbin_size=$(wc -c flash.bin | awk '{print $$1}'); \
    pad_cnt=$(( (flashbin_size + 0x400 - 1) / 0x400 ); \
    echo "append u-boot-atf-container.img at $$pad_cnt KB"; \
    dd if=u-boot-atf-container.img of=flash.bin bs=1K seek=$$pad_cnt;
```

flash\_linux\_m4 和 flash\_regression\_linux\_m4 的区别在于添加了 SPL 镜像。对于 SPL 启动，ROM 代码仅将 SPL 镜像加载到 OCRAM，从 OCRAM 启动 SPL 镜像后，它将尝试从启动设备读取剩余图像（即 u-boot-aft-container.img）并加载到 DDR。需要注意以下三个部分。

- u-boot-aft-container.img 镜像

该镜像由 ATF 镜像 b131.bin 和 U-Boot 镜像 u-boot-hash.bin 生成，如下所示。它还将包括 TEE 镜像 tee.bin（如果文件夹中有）。

```
u-boot-atf-container.img: b131.bin u-boot-hash.bin
    if [ -f tee.bin ]; then \
        if [ $(shell echo $(ROLLBACK_INDEX_IN_CONTAINER)) ]; then \
            ./$(MKIMG) -soc QX -sw_version $(ROLLBACK_INDEX_IN_CONTAINER) -rev B0 -c -ap b131.bin a35 0x80000000 -ap u-boot-hash.bin a35 0x80020000 -ap tee.bin a35 $(TEE_LOAD_ADDR) -out u-boot-atf-container.img; \
        else \
            ./$(MKIMG) -soc QX -rev B0 -c -ap b131.bin a35 0x80000000 -ap u-boot-hash.bin a35 0x80020000 -ap tee.bin a35 $(TEE_LOAD_ADDR) -out u-boot-atf-container.img; \
        fi; \
    else \
        ./$(MKIMG) -soc QX -rev B0 -c -ap b131.bin a35 0x80000000 -ap u-boot-hash.bin a35 0x80020000 -out u-boot-atf-container.img; \
    fi
```

- dcd 跳过选项

在非 SPL 启动方式中，ROM 代码将 A 核启动镜像从启动设备加载到 DRAM，因此 ROM 代码需要在加载前对 DDR 进行初始化。但在 SPL 启动方式中，ROM 代码只需将 SPL 镜像加载到 OCRAM 中，因此可以在 ROM 代码中跳过 DDR 初始化，稍后在 SCFW 中完成。“-dcd skip (-dcd 跳过)”选项在镜像容器中设置一个标志，因此当 ROM 代码读取镜像容器时知道它是如何配置的。

注：如果 M4 需要从 DDR 启动，ROM 代码仍然需要将 M4 镜像加载到 DRAM 中。在这种情况下，“-dcd skip”选项不适用，如下面的 flash\_linux\_m4\_ddr 目标所示。

```
flash_linux_m4_ddr: $(MKIMG) $(AHAB_IMG) scfw_tcm.bin u-boot-atf-container.img m4_image.bin u-boot-spl.bin
./$(MKIMG) -soc QX -rev B0 -append $(AHAB_IMG) -c -flags 0x00200000 -scfw scfw_tcm.bin -ap u-boot-spl.bin a35 0x00100000 -p3 -m4 m4_image.bin 0 0x88000000 -out flash.bin
cp flash.bin boot-spl-container.img
@flashbin_size=`wc -c flash.bin | awk '{print $$1}'`; \
pad_cnt=$((flashbin_size + 0x400 - 1) / 0x400); \
echo "append u-boot-atf-container.img at $$pad_cnt KB"; \
dd if=u-boot-atf-container.img of=flash.bin bs=1K seek=$$pad_cnt; \
```

有关“-dcd skip”选项和DDR初始化流程的更多详细信息，请参见 sc\_fw\_port.pdf 的第4.6章“DDR配置”。

-A 核的启动地址

在 SPL 启动方法中，由于 ROM 代码会将 SPL 镜像加载到 OCRM，A 核的启动地址也会更改为 OCRM 地址 0x00100000。

• flash\_linux\_m4\_xip

```
flash_linux_m4_xip: $(MKIMG) $(AHAB_IMG) scfw_tcm.bin u-boot-atf-container.img m4_image.bin u-boot-spl.bin
./$(MKIMG) -soc QX -rev B0 -dcd skip -append $(AHAB_IMG) -c -flags 0x00200000 -scfw scfw_tcm.bin -fileoff 0x80000 -p3 -m4 m4_image.bin 0 0x08081000 -fileoff 0x180000 -ap u-boot-spl.bin a35 0x00100000 -out flash.bin
cp flash.bin boot-spl-container.img
@flashbin_size=`wc -c flash.bin | awk '{print $$1}'`; \
pad_cnt=$((flashbin_size + 0x400 - 1) / 0x400); \
echo "append u-boot-atf-container.img at $$pad_cnt KB"; \
dd if=u-boot-atf-container.img of=flash.bin bs=1K seek=$$pad_cnt; \
./$(QSPI_PACKER) $(QSPI_HEADER)
```

flash\_linux\_m4\_xip 与 flash\_linux\_m4 之间的主要变化是 M4 从 QSPI NOR Flash 设备启动，而不是从 TCM 启动。除了上面提到的 M4 内核启动地址，还请注意 QSPI 标头文件。

从 QSPI/FSPI 设备启动 flash.bin 镜像，镜像中需要标头文件，以便 ROM 代码配置 QSPI/FSPI 设备。示例 QSPI/FSPI 标头文件在 imx-mkimage/scripts/fspi\_header 中提供。要选择 QSPI/FSPI 设备作为启动设备，请修改标头文件以适应这些设备。

例如，在示例标头文件中，我们可以看到偏移量 0x44 - 0x47 的值是 0x01010200。

```
01010200 /* Serial Nor, Single/Dual/Quad/Octal, SerialClkFreq 1 - 20MHz, 2 - 50MHz... */
```

要转换此设置，请参见 i.MX 8QuadMax Applications Processor Reference Manual (文档 [IMX8QMRM](#)) 的 **Chapter 5.8.3.3 FlexSPI configuration parameters**。

表 4. FlexSPI 配置参数

devicetype	0x044	1	1 - 串行 NOR
sflashPadType	0x045	j	1 - 单引脚 2 - 双引脚 4 - 四引脚 8 - 八引脚
serialClkFreq	0x046	1	芯片特定值，针对这种芯片 1 - 20 MHz 2 - 50 MHz 3 - SDR 为 166 MHz，DDR 为 200 MHz 4 - 80 MHz 5 - 100 MHz

表 4. FlexSPI 配置参数...续上页

			6 - 133 MHz
			7 - SDR 为 62 MHz, DDR 为 200 MHz
			其他值: 20 MHz
IutCustomSeqEnable	0x047	1	0 - 使用预定义的 LUT 序列索引和编号
			1 - 使用本模块中提供的 LUT 序列参数

我们可以看到，在示例标头文件中，

- deviceType 的值为 0x01，这是串行 NOR。
- sflashPadType 的值为 0x1，这是单引脚。
- serialClkFreq 的值为 0x02，即 50 MHz。
- lutCustomSeqEnable 的值为 0x00，即使用预定义的 LUT 序列索引和编号。

对于标头文件中的所有参数，请查看 i.MX 8QuadMax Applications Processor Reference Manual（文档 [IMX8QMRM](#)）的 **Chapter 5.8.3.3 FlexSPI configuration parameters** 中的定义，并为 FSPI 芯片设置正确的值。

### 7.3 生成 flash.bin 镜像

要生成 flash.bin 镜像，请在用户的 Linux 主机环境中使用 gcc 编译内部工具 mkimage\_imx8。mkimage\_imx8 的源代码位于 imx-mkimage/src。

由于使用以下命令生成 flash.bin 镜像时会自动编译，用户无需单独编译 mkimage\_imx8。

生成 flash.bin 镜像的命令为：

```
$make SOC=<SOC_TARGET> REV=<SOC_REV> [TARGET]
```

SOC\_TARGET 是目标芯片，SOC\_REV 是芯片反转，TARGET（目标）参见 [章节 7.2](#)。

注：

由于 iMX8QXP B0 和 C0 芯片的 SECO FW 不兼容，因此使用 REV 来指定 SECO FW 的版本。

- 对于 iMX8QXP B0 芯片，使用 SECO FW mx8qxb0-ahab-container.img。
- 对于 iMX8QXP C0 芯片，使用 SECO FW mx8qxc0-ahab-container.img。

以我们的 imx8qxp 汽车板为例，要用 M4 镜像和 SPL 镜像构建 flash.bin 镜像，使用以下命令：

```
$make SOC=iMX8QX REV=C0 flash_linux_m4
```

flash.bin 二进制文件在 imx-mkimage/iMX8QX/flash.bin 中生成。

### 7.4 烧录 flash.bin 镜像

对于 iMX8/8X 芯片，ROM 代码支持从以下启动设备启动：

- SD/MMC
- NAND FLASH
- FlexSPI NOR flash
- USB 2.0 OTG 和 USB 3.0（作为 2.0）支持串行下载器

通常，SD卡是验证生成的 flash.bin 镜像是否可以成功启动板级的最有效方法。

如果需要将 flash.bin 镜像烧录到 SD 卡上，请在 Linux 主机 PC 上插入 SD 卡。假设 SD 卡被识别为 /dev/sdx，使用以下命令将 flash.bin 烧录到SD卡中：

```
$sudo umount /dev/sdx*
$sudo dd if=flash.bin of=/dev/sdx bs=1k seek=32 conv=fsync && sync
```

关于如何将 flash.bin 镜像烧录到其他启动设备，请参见 i.MX Linux User Guide（文档 [IMXLUG](#)）的 **Chapter 4.4 Downloading images**。

## 8 Linux 内核移植

相对于前几章，用户对 Linux 内核移植更加熟悉。本章重点介绍 imx8qxp 汽车参考板的移植工作，说明为新的自定义板级添加新设备树文件和驱动程序的过程。

有关 Linux 文件系统移植，如添加新的用户空间工具或软件包，请参见 i.MX\_Yocto\_Project\_User's\_Guide.pdf。

### 8.1 为新的板级创建文件

要为新板级创建文件，请按以下步骤操作：

1. 在 arch/arm64/configs/ 中添加新的板级 defconfig。  
defconfig 文件定义了将包含在 Linux 内核中的组件。通常，根据硬件设计和软件要求，每块板都有专门的 defconfig 文件。  
imx 8qxp 汽车板与 MEK 参考板非常相似。因此，直接使用 arch/arm64/configs/imx\_v8\_defconfig 中的默认 defconfig 文件作为 defconfig 文件。对于其他 iMX 8 自定义板，请参考 imx\_v8\_defconfig 进行相应的修改。
2. 在 arch/arm64/boot/dts/freescale/ 中添加新的板级 dts。  
对于 imx 8qxp 汽车板，请创建以下 dts/dtsi 文件，文件可从 MEK 板的文件中复制，以减少工作量。

表 5. 在 Linux 内核中为汽车板创建的设备树文件

文件位置	描述
arch/arm64/boot/dts/freescale/imx8qxp-auto.dts	不带 rpmsg 的汽车板的 dts 文件，包括 imx8qxp.dtsi 和 imx8x-auto.dtsi。
arch/arm64/boot/dts/freescale/imx8qxp-auto-rpmsg.dts	带 rpmsg 的汽车板的 dts 文件，包括 imx8qxp-auto.dts 和 imx8x-auto-rpmsg.dtsi。
arch/arm64/boot/dts/freescale/imx8x-auto.dtsi	不带 rpmsg 的汽车板的 dtsi 文件，启用需要的设备节点。
arch/arm64/boot/dts/freescale/imx8x-auto-rpmsg.dtsi	带 rpmsg 的汽车板的 dtsi 文件，启用需要的设备节点。
arch/arm64/boot/dts/freescale/imx8qxp-auto-enet-tja1101.dts	启用 TJA1101 以太网 PHY 的汽车板的 dts 文件，其中包括 imx8qxp-auto.dts 和 imx8qxp-auto-enet-tja1101.dtsi。
arch/arm64/boot/dts/freescale/imx8qxp-auto-enet-tja1101.dtsi	启用 TJA1101 以太网 PHY 的汽车板的 dtsi 文件。



有 rpsmsg 和没有 rpsmsg 的 dts 的区别在于与 M4 内核相关的资源。如果使用了 M4 内核，那么就如前文所述那样，将一些外设接口资源分配给 SCFW 中的 M4 分区（如 I<sup>2</sup>C 和 Flexcan）。请在 dts 中禁用这些资源，或为 I<sup>2</sup>C 接口使用 imx\_rpsmsg\_i2c 等虚拟驱动程序。

3. 为板上的设备添加新的驱动程序。

对于 imx 8qxp 汽车板，在板设计中添加以下三个硬件组件：

- 带 NVP6324 汽车 AHD 解决方案的 MIPI-CSI
- 带有 TI DS90UB947/948 SerDes 的 LVDS 显示器（通过 FPD Link III），用于汽车应用
- 配备 Maxim 96752/96755 SerDes（通过 GMSL2）的 MIPI-DSI 显示器，用于汽车应用

参见 [表 6](#) 以了解要添加到内核代码的相应驱动程序。

表 6. 为 Linux 内核中的汽车板添加的驱动程序文件

文件位置	描述
drivers/gpu/drm/bridge/ds90ub94x.c	TI DS90UB947/948 SerDes 的驱动
drivers/gpu/drm/bridge/mx9675x.c	Maxim 96752/96755 SerDes 的驱动
drivers/media/platform/imx8/nvp6324/	NVP6324 的新文件夹
drivers/media/platform/imx8/nvp6324/Kconfig	NVP6324 驱动的 Kconfig
drivers/media/platform/imx8/nvp6324/Makefile	NVP6324 驱动的 Makefile
drivers/media/platform/imx8/nvp6324/nvp6324.h	NVP6324 驱动的头文件
drivers/media/platform/imx8/nvp6324/nvp6324_core.c	NVP6324 驱动的核心功能文件
drivers/media/platform/imx8/nvp6324/nvp6324_mipi.c	NVP6324 驱动的 MIPI 设置
drivers/media/platform/imx8/nvp6324/nvp6324_video.c	NVP6324 驱动的视频模式设置
drivers/media/platform/imx8/nvp6324/nvp6324_video_eq.c	NVP6324 驱动的视频活动队列设置

## 8.2 修改新板级的文件

要修改新板级的文件，请按以下步骤操作：

1. 编辑相关的 makefile，编译成内核镜像。在 imx8qxp 汽车板示例中，相关的 makefile 包括：

- arch/arm64/boot/dts/freescale/Makefile

添加 imx8qxp 汽车板的 dtb 文件。

```
diff --git a/arch/arm64/boot/dts/freescale/Makefile b/arch/arm64/boot/dts/freescale/Makefile
index 7b27addc9603..0097229dd187 100644
--- a/arch/arm64/boot/dts/freescale/Makefile
+++ b/arch/arm64/boot/dts/freescale/Makefile
@@ -143,7 +143,10 @@ dtb-$(CONFIG_ARCH_MXC) += imx8qxp-mek.dtb imx8qxp-mek-dsp.dtb imx8qxp-mek-ov5640
+ imx8qxp-lpddr4-val-lpspi.dtb imx8qxp-lpddr4-val-lpspi-slave.dtb \
+ imx8qxp-lpddr4-val-spdif.dtb imx8qxp-lpddr4-val-gpmi-nand.dtb imx8dxp-lpddr4-val.dtb \
+ imx8qxp-l7x17-val.dtb imx8dx-lpddr4-val.dtb imx8dx-l7x17-val.dtb \
+ imx8qxp-lpddr4-val-mlb.dtb
+ imx8qxp-lpddr4-val-mlb.dtb \
+ imx8qxp-auto.dtb \
+ imx8qxp-auto-rpsmsg.dtb \
+ imx8qxp-auto-enet-tjall101.dtb
dtb-$(CONFIG_ARCH_MXC) += imx8qxp-mek-dom0.dtb imx8qxp-mek-root.dtb \
+ imx8qxp-mek-inmate.dtb
dtb-$(CONFIG_ARCH_MXC) += imx8dx1-evk.dtb imx8dx1-evk-rpsmsg.dtb \
```

- drivers/gpu/drm/bridge/Makefile
- 添加 ds90ub94x 和 mx9675x 驱动程序。

```
diff --git a/drivers/gpu/drm/bridge/Makefile b/drivers/gpu/drm/bridge/Makefile
index 103466b50b2a..7d502f6a97aa 100644
--- a/drivers/gpu/drm/bridge/Makefile
+++ b/drivers/gpu/drm/bridge/Makefile
@@ -20,6 +20,8 @@ obj-$(CONFIG_DRM_TI_TFP410) += ti-tfp410.o
 obj-$(CONFIG_DRM_NWL_MIPI_DSI) += nwl-dsi/
 obj-y += cadence/
 obj-y += synopsys/
+obj-$(CONFIG_DRM_TI_DS90UB94x) += ds90ub94x.o
 obj-$(CONFIG_DRM_ITE_IT6263) += it6263.o
 obj-$(CONFIG_DRM_SEC_MIPI_DSIM) += sec-dsim.o
 obj-$(CONFIG_DRM_NXP_SEIKO_43WVFIG) += nxp-seiko-43wvfig.o
+obj-$(CONFIG_DRM_MAXIM_MAX9675x) += mx9675x.o
```

- drivers/media/platform/imx8/Makefile

包括 nvp6324 驱动的文件夹。

```
diff --git a/drivers/media/platform/imx8/Makefile b/drivers/media/platform/imx8/Makefile
index 15259cf9c13f..11a8877db03a 100644
--- a/drivers/media/platform/imx8/Makefile
+++ b/drivers/media/platform/imx8/Makefile
@@ -1,3 +1,4 @@
 obj-$(CONFIG_IMX8_MIPI_CSI2_YAV) += mxc-mipi-csi2_yav.o
 mxc-jpeg-encdec-objs := mxc-jpeg-hw.o mxc-jpeg.o
 obj-$(CONFIG_IMX8_JPEG) += mxc-jpeg-encdec.o
+obj-$(CONFIG_IMX8_NVP6324) += nvp6324/
```

2. 根据板级设计修改 dts 文件。例如，在 imx8qxp 汽车板上，LVDS0 上的连接就是这样设计的：  
ldb1 -> ds90ub947 -> ds90ub948 -> it6263 -> HDMI screen  
因此，imx8x-auto.dtsi 中的 ldb1 和 i2c0\_mipi\_lvds0 设备节点相应更改如下。

```

&i2c0_mipi_lvds0 {
    #address-cells = <1>;
    #size-cells = <0>;
    pinctrl-names = "default";
    pinctrl-0 = <&pinctrl_i2c0_mipi_lvds0>;
    clock-frequency = <100000>;
    status = "okay";

    lvds_ds0: ds90ub94x@c {
        compatible = "ti,ds90ub94x";
        reg = <0x0c>;
        clock-frequency = <400000>;

        port@0 {
            ds90ub94x_in: endpoint {
                remote-endpoint = <&lvds0_out>;
            };
        };

        port@1 {

            ds90ub94x_out: endpoint {
                remote-endpoint = <&it6263_in>;
            };
        };
    };

    lvds_bridge0: lvds-to-hdmi-bridge@4c {
        compatible = "ite,it6263";
        reg = <0x4c>;

        port {
            it6263_in: endpoint {
                remote-endpoint = <&ds90ub94x_out>;
            };
        };
    };
};

```

```

&ldb1_phy {
    status = "okay";
};

&ldb1 {
    status = "okay";

    lvds-channel@0 {
        fsl,data-mapping = "jeida";
        fsl,data-width = <24>;
        status = "okay";

        port@1 {
            reg = <1>;

            lvds0_out: endpoint {
                remote-endpoint = <&ds90ub94x_in>;
            };
        };
    };
};

```

连接顺序参见 dts 中的端点搭配。

对于其他板级的设计变更，dts 中也以类似的方法进行了相应修改。

3. 修改一些现有的驱动程序，为新板级添加特定功能或实施。

例如，在 imx8qxp 汽车板设计中，要连接 Maxim 96755 SerDes 上的 MIPI DSI 显示面板，请在 drivers/gpu/drm/panel/panel-simple.c 中添加此显示面板的设置，如下所示。

```
diff --git a/drivers/gpu/drm/panel/panel-simple.c b/drivers/gpu/drm/panel/panel-simple.c
index a834a39e335c..ec41932ca357 100644
--- a/drivers/gpu/drm/panel/panel-simple.c
+++ b/drivers/gpu/drm/panel/panel-simple.c
@@ -3144,6 +3144,31 @@ static const struct panel_desc arm_rtsm = {
     .bus_format = MEDIA_BUS_FMT_RGB888_1X24,
 };

+static const struct drm_display_mode max96752_1080p_mode = {
+    .clock = 148500,
+    .hdisplay = 1920,
+    .hsync_start = 1920 + 88,
+    .hsync_end = 1920 + 88 + 44,
+    .htotal = 1920 + 88 + 44 + 148,
+    .vdisplay = 1200,
+    .vsync_start = 1200 + 4,
+    .vsync_end = 1200 + 4 + 5,
+    .vtotal = 1200 + 4 + 5 + 36,
+    .vrefresh = 60,
+    .flags = DRM_MODE_FLAG_PVSYNC | DRM_MODE_FLAG_PHSYNC,
+};
+
+static const struct panel_desc max96752_1080p = {
+    .modes = &max96752_1080p_mode,
+    .num_modes = 1,
+    .bpc = 8,
+    .size = {
+        .width = 80,
+        .height = 60,
+    },
+    .bus_format = MEDIA_BUS_FMT_RGB888_1X24,
+};
+
static const struct of_device_id platform_of_match[] = {
@@ -3475,6 +3500,9 @@ static const struct of_device_id platform_of_match[] = {
     }, {
         .compatible = "winstar,wf35ltiacd",
         .data = &winstar_wf35ltiacd,
+    }, {
+        .compatible = "max,max96752-1080p",
+        .data = &max96752_1080p,
     }, {
         /* sentinel */
     }
};
```

### 8.3 编译 Linux 内核

要编译 Linux 内核，请按以下步骤操作：

#### 1. 设置构建环境。

用于编译 Linux 内核的工具链与用于编译 ATF 和 U-Boot 的交叉编译工具链相同。关于如何生成和安装工具链，请参见 i.MX Linux User's Guide（文档 [IMX LUG](#)）的 Chapter 4.5.12 How to build U-Boot and kernel。  
[https://www.nxp.com/webapp/Download?colCode=L5.4.47\\_2.2.0\\_LINUX\\_DOCS](https://www.nxp.com/webapp/Download?colCode=L5.4.47_2.2.0_LINUX_DOCS)

#### 2. 编译代码。

要为目标板构建 Linux 内核，请按以下步骤操作：

- a. 使用以下命令为板生成配置文件。在本例中，汽车板使用默认的 defconfig。

```
$ make imx_v8_defconfig
```

- b. 使用以下命令为目标板生成 Linux 内核，同时也生成了相关的 dtb 文件。

```
$ make -j8
```

- c. 编译完成的 Linux 内核镜像为 arch/arm64/boot/image，dtb 文件位于 arch/arm64/boot/dts/freescale/folder。

## 8.4 烧录 Linux 内核

- 如果 SD/MMC 已经被划分为 boot 分区和 rootfs 分区，请使用以下命令将内核镜像和 dtb 文件直接复制到启动分区。

```
$sudo mount /dev/sdx1 /mnt/boot/
$sudo cp Image /mnt/boot/
$sudo cp imx8qxp-auto.dtb /mnt/boot/
$sudo umount /dev/sdx1
```

- 如果 SD/MMC 尚未分区，请按照 i.MX Linux User's Guide（文档 [IMXLUG](#)）的 Chapter 4.3 Preparing an SD/MMC card to boot 中的步骤对 SD/MMC 进行分区，然后将内核镜像和 dtb 烧录到启动分区。

## 9 Note About the Source Code in the Document

Example code shown in this document has the following copyright and BSD-3-Clause license:

Copyright 2023 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

## 10 修订记录

[表 7](#) 总结了对本文档的修订。

表 7. 修订记录

版本号	日期	重大更新
1	2023年5月26日	更新了 <a href="#">章节 1.3</a>
0	2021年6月10日	初次发布



## 11 Legal information

### 11.1 Definitions

**Draft** — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

### 11.2 Disclaimers

**Limited warranty and liability** — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

**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.

**Suitability for use** — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer' s own risk.

**Applications** — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer' s sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer' s applications and products planned, as well as for the planned application and use of customer' s third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer' s applications or products, or the application or use by customer' s third party customer(s). Customer is responsible for doing all necessary testing for the customer' s applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer' s third party customer(s). NXP does not accept any liability in this respect.

**Terms and conditions of commercial sale** — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at <http://www.nxp.com/profile/terms>, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer' s general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

**Export control** — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

**Suitability for use in non-automotive qualified products** — Unless this data sheet expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer' s own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

**Translations** — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

**Security** — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. 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](mailto:PSIRT@nxp.com)) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

**NXP B.V.** - NXP B.V. is not an operating company and it does not distribute or sell products.

### 11.3 Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

**NXP** — wordmark and logo are trademarks of NXP B.V.

AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro,  $\mu$  Vision, Versatile — are trademarks and/or registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved.

i.MX — is a trademark of NXP B.V.

## 内容

<b>1</b>	<b>介绍 .....</b>	<b>2</b>
1.1	目标 .....	2
1.2	示例板 .....	2
1.3	Linux BSP 版本 .....	2
<b>2</b>	<b>生成 DDR 配置文件 .....</b>	<b>3</b>
2.1	下载 RPA 工具 .....	3
2.2	使用 RPA 工具 .....	3
<b>3</b>	<b>SCFW 移植 .....</b>	<b>6</b>
3.1	提取 SCFW 代码 .....	6
3.2	创建板级文件 .....	6
3.3	编译 SCFW .....	9
<b>4</b>	<b>运行 DDR 压力测试 .....</b>	<b>11</b>
<b>5</b>	<b>ATF 移植 .....</b>	<b>13</b>
5.1	电源管理 .....	13
5.2	资源分区 .....	14
5.3	编译 ATF .....	15
<b>6</b>	<b>U-Boot 移植 .....</b>	<b>16</b>
6.1	为新的板级创建文件 .....	16
6.2	修改新板级的文件 .....	17
6.3	编译 U-Boot .....	19
<b>7</b>	<b>构建 flash.bin 镜像 .....</b>	<b>20</b>
7.1	将镜像复制到 mkimage .....	20
7.2	检查 makefile .....	21
7.3	生成 flash.bin 镜像 .....	24
7.4	烧录 flash.bin 镜像 .....	24
<b>8</b>	<b>Linux 内核移植 .....</b>	<b>25</b>
8.1	为新的板级创建文件 .....	25
8.2	修改新板级的文件 .....	26
8.3	编译 Linux 内核 .....	29
8.4	烧录 Linux 内核 .....	30
<b>9</b>	<b>Note About the Source Code in the Document .....</b>	<b>30</b>
<b>10</b>	<b>修订记录 .....</b>	<b>30</b>
<b>11</b>	<b>Legal information .....</b>	<b>31</b>

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.

© 2023 NXP B.V.

All rights reserved.

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

Date of release: 2023年5月26日  
Document identifier: AN13275