使用 Cortex-M4 SDK 与 i.MX 7ULP-EVK 上的 Cortex-A7 Android BSP 通信

原文：https://community.nxp.com/docs/DOC-341570

此文件展示了如何在 i.MX 7ULP-EVK 的 Cortex-A7 上运行 Android BSP 的同时，从 MCUXpresso SDK 运行多核通信示例。

尽管本文主要讨论多核演示，但类似的过程可以应用于运行 SDK 中的任何其他演示。

1.源代码

此文件基于以下版本：

硬件版本：imx7ulp-evk

Android 版本：Android O8.1.0 for i.MX 7ULP GA

MCUXpresso 版本：SDK2.4 for i.MX 7ULP GA

2.构建 Cortex-M4 SDK

SDK 软件包中至少有两个多核演示，分别是 rpmsg_lite_pingpong_rtos 和 rpmsg_lite_str_echo_rtos。它们位于：

<SDK_2.4.0_EVK-MCIMX7ULP_dir>/boards/evkmcimx7ulp/multicore_examples/

根据 SDK 入门指南，构建 rpmsg_lite_str_echo_rtos 演示。记住还要遵循文档的第 6 章，第 4 步，以生成 ram 可启动映像（sdk20-app.img）。

3.构建 Android BSP

3.1. RPMsg 内核模块

在构建 BSP 之前，将以下行添加到 BoardConfig.mk 文件（<android_build_dir>/device/fsl/evk_7ulp/BoardConfig.mk）：

```
BOARD_VENDOR_KERNEL_MODULES += \
  $(KERNEL_OUT)/drivers/net/wireless/qcacld-2.0/wlan.ko \
+ $(KERNEL_OUT)/drivers/rpmsg/imx_rpmsg_tty.ko
```

3.2. Cortex-M4 图像

将 SDK 映像文件（sdk20-app.img）复制到 Android 源代码中的以下目录：

```
$ cp <SDK_2.4.0_EVK-MCIMX7ULP_dir>/tools/imgutil/evkmcimx7ulp/sdk20-app.img \
  <android_build_dir>/vendor/nxp/fsl-proprietary/mcu-sdk/7ulp/sdk20-app.img
```

相应地更改 BoardConfig.mk 文件：

```
# Copy prebuilt M4 demo image:
PRODUCT_COPY_FILES += \
- vendor/nxp/fsl-proprietary/mcu-sdk/7ulp/imx7ulp_m4_demo.img:imx7ulp_m4_demo.img
+ vendor/nxp/fsl-proprietary/mcu-sdk/7ulp/sdk20-app.img:imx7ulp_m4_demo.img
```

完成这些更改后，按照 BSP 用户指南中的说明构建并刷新 Android。

4.启用多核通信

引导时，SoC 会自动加载 Cortex-M4 图像。

完成引导后，安装 imx_rpmsg_tty.ko 模块以创建多核通信通道：

```
$ su
$ insmod vendor/lib/modules/imx_rpmsg_tty.ko
```

要将消息从 Cortex-A7 发送到 Cortex-M4，请使用/ dev / ttyRPMSG *通道：

```
$ echo "MESSAGE" > /dev/ttyRPMSG*
```

/ dev / ttyRPMSG *是指在板上创建的 RPMsg 设备，因此请相应地更改编号。Cortex-M4 将回显从 Cortex-A7 收到的所有消息。

这是一个有关如何使用 Android 在 i.MX 上通信不同内核的简单示例，但是它可以用作 Android 多核应用程序的起点。