Application Note

TDA4 刷写技术

Karan Saxena, Keerthy J, Karthik R, and Brijesh Jadav

摘要

本应用报告重点介绍了 TDA4 系列器件的刷写技术。这包括在考虑硬件接口和可用软件工具限制的同时刷写客户电路板。

可以从以下 URL 下载本文档所提到的工程配套资料：https://www.ti.com/cn/lit/zip/spracy5。

内容

商标

1 刷写工具简介

对于特定的引导介质，考虑到硬件接口在电路板上有引脚输出，目的是找到一个可行的解决方案，用引导加载程序和文件系统映像刷写特定的引导介质。

大多数软件工具都有表 1-1 所列的一些约束条件。由此可以推断出特定引导介质支持的软件工具。

表 1-1. 软件工具约束条件

软件工具	硬件接口	引导介质			约束条件
		OSPI NOR	eMMC	QSPI NOR	
UNIFLASH	UART	L(1) 只能刷写引导加载程序映像。无法刷写 UBIFS。	L 只能刷写 eMMC boot0 (RAW) 分区。	L 只能刷写引导加载程序映像。无法刷写 UBIFS。	• 支持 UART 引导模式。 • MCU_UART0 在电路板上应具有引脚输出。 • 仅刷写 eMMC 支持的原始扇区
CCS	JTAG	L 只能刷写引导加载程序映像。无法刷写 UBIFS。	L 只能使用微型文件系统刷写 eMMC 用户 (UDA)。 应该已经刷写了引导加载程序映像。	L 只能刷写引导加载程序映像。无法刷写 UBIFS。	• 存在调试器接口。 • 刷写速度取决于 JTAG 的选择。
Lauterbach	JTAG	Y(2)	Y	Y	• 许可的软件
u-boot	UART、MMCSD、DFU 引导	Y 将 SD 卡、JTAG、以太网等与 u-boot 结合使用。	Y 将 SD 卡、JTAG、以太网等与 u-boot 结合使用。	Y 将 SD 卡、JTAG、以太网等与 u-boot 结合使用。	• 应首先使用其他接口来刷写 u-boot。
DFU	USB	Y	Y	Y	

(1) L = 有限支持
(2) Y = 完全支持

备注

默认情况下，UNIFLASH 使用 MCU_UART0。
通用异步接收器/发送器 (UART) 引导到 u-boot 需要 MCU_UART0 以及 u-boot 将打印到的其中一个 MAIN_UART。

1.1 Trace32/Lauterbach

对于基于 Arm® 的 SOC，Trace32/Lauterbach 是一款功能强大的工具。

Trace32 还可用于刷写各种类型的存储器。若要使用这种方法，JTAG 接口需要在电路板上具有引脚输出。这使具有 Trace32 的客户能够将引导映像刷写到 eMMC 和 SPI 存储器中。

刷写过程包括 Lauterbach 可以理解的 CMM 脚本，以及在 Linux PC 上一次性安装以连接到 TDA4。使用 Trace32/Lauterbach 进行刷写可避免依赖 UART/SD 等辅助引导介质将映像烧写到定制电路板上的刷写器件。

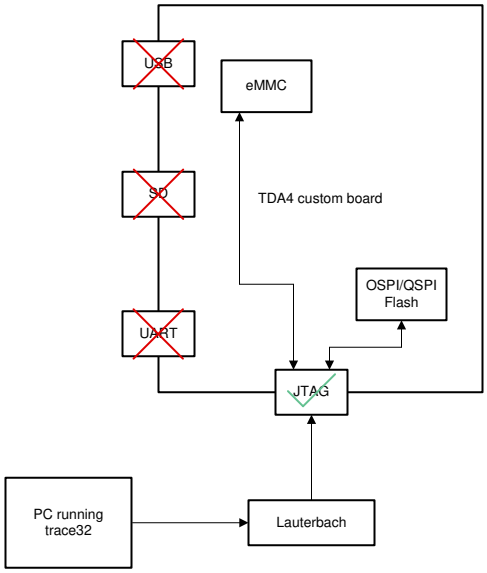

图 1-1. Trace32/Lauterbach 刷写

1.2 基于 CCS 的闪存写入器

一些定制电路板仅具有 JTAG 接口以及 OSPI/QSPI 闪存。在没有 Trace32/Lauterbach 的情况下，可以使用一个基于 Code Composer Studio™ 的软件闪存写入器应用程序来刷写板载闪存。

该解决方案提供了一个在 MCU R5 上运行的应用程序，然后提供一个对用户友好的菜单来写入和擦除 OSPI/QSPI 闪存。

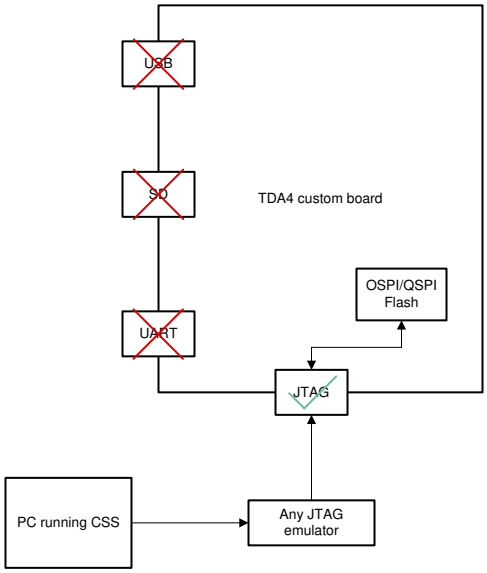

图 1-2. 基于 CCS 的闪存写入器刷写

1.3 其它软件工具

- **UNIFLASH**
  - 该工具可用于为 OSPI/QSPI 和 eMMC 刷写 RAW 扇区。
  - UART 在定制电路板上应具有引脚输出。
  - 应支持 UART 引导模式。
- **dfu-util**
  - 基于 USB 的工具，可用于刷写 OSPI/QSPI 和 eMMC。
  - 唯一的要求是使用 USB 接口。
- **U-boot**
  - u-boot 在电路板上运行后，u-boot 实用程序可用于刷写引导介质。
  - U-boot + SD 卡、U-boot + 以太网、U-boot + CCS 是一些可用于刷写 eMMC 和 OSPI/QSPI 的选项。

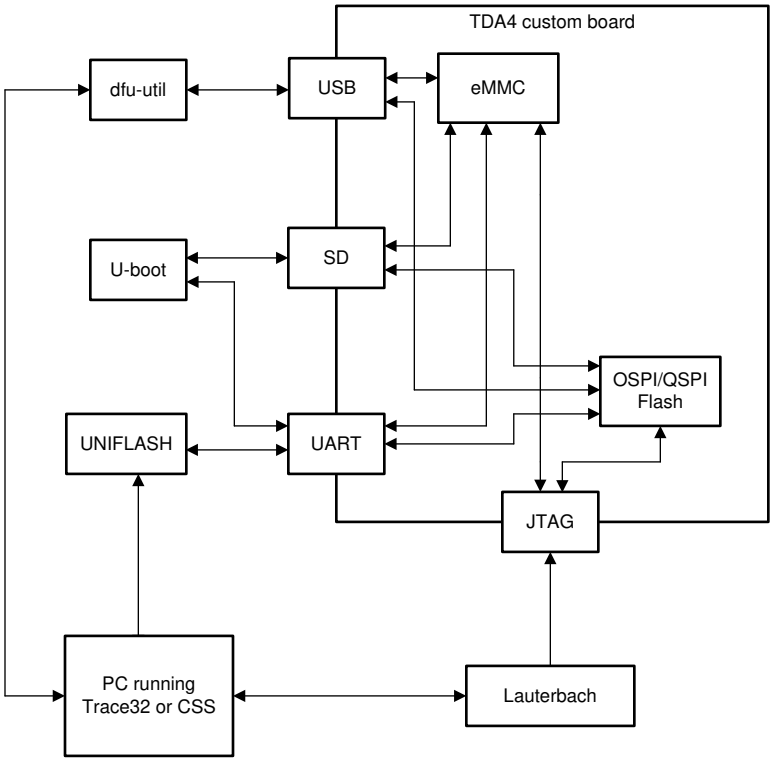

图 1-3. 其它软件工具

2 TDA4 上的闪存器件

OSPI 和 eMMC 闪存是 TDA4 板上常被选用的外部闪存。图 2-1 描述了 SDK 中闪存的默认布局，如果自定义应用需要不同的布局，可以对其进行更改。

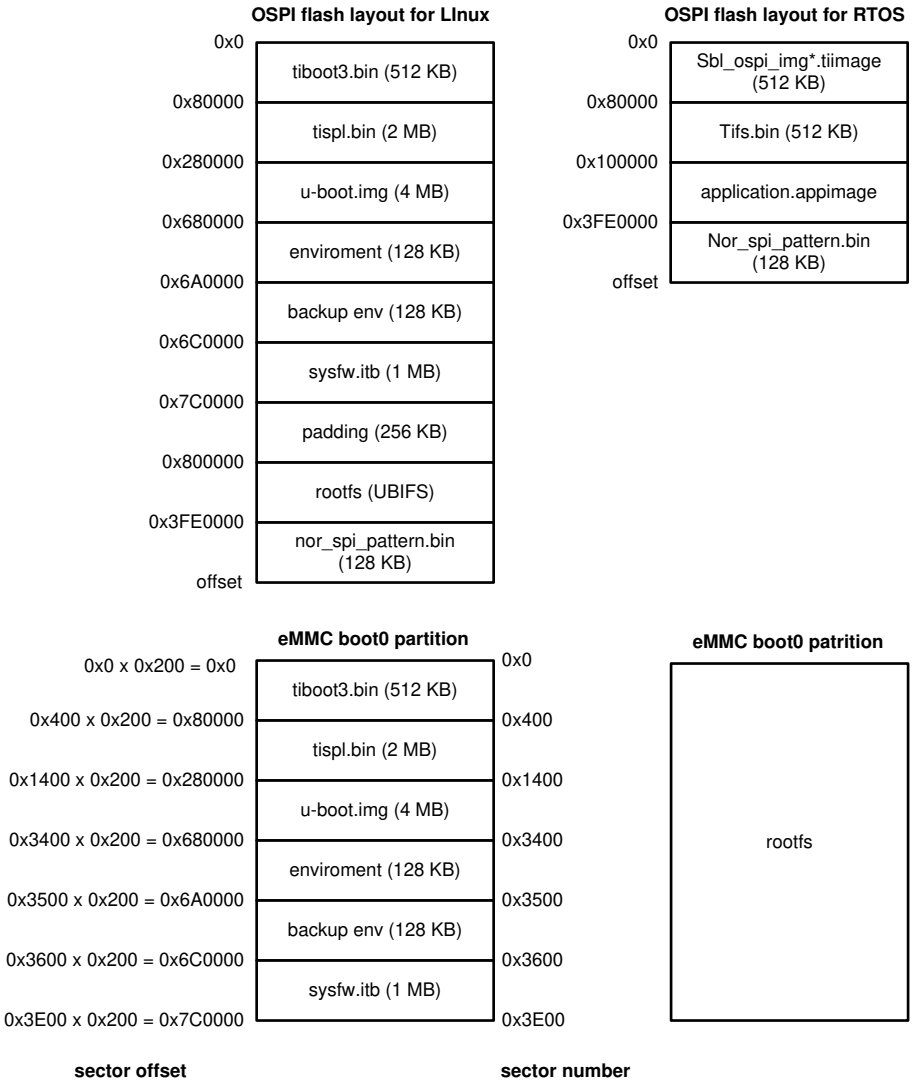

图 2-1. SDK 中的闪存布局

2.1 刷写 OSPI 和 eMMC RAW 扇区

考虑到 TDA4 定制电路板上的硬件限制，可以使用图 2-2 来确定哪些软件工具可用于刷写 OSPI 闪存或 eMMC 引导分区。

例如，如果客户希望刷写 OSPI 但没有 MAIN UART、USB 或 SD 卡接口，图 2-2 明确指示客户将 JTAG 与 CCS 或 Lauterbach 一同使用，或使用 UNIFLASH 工具。

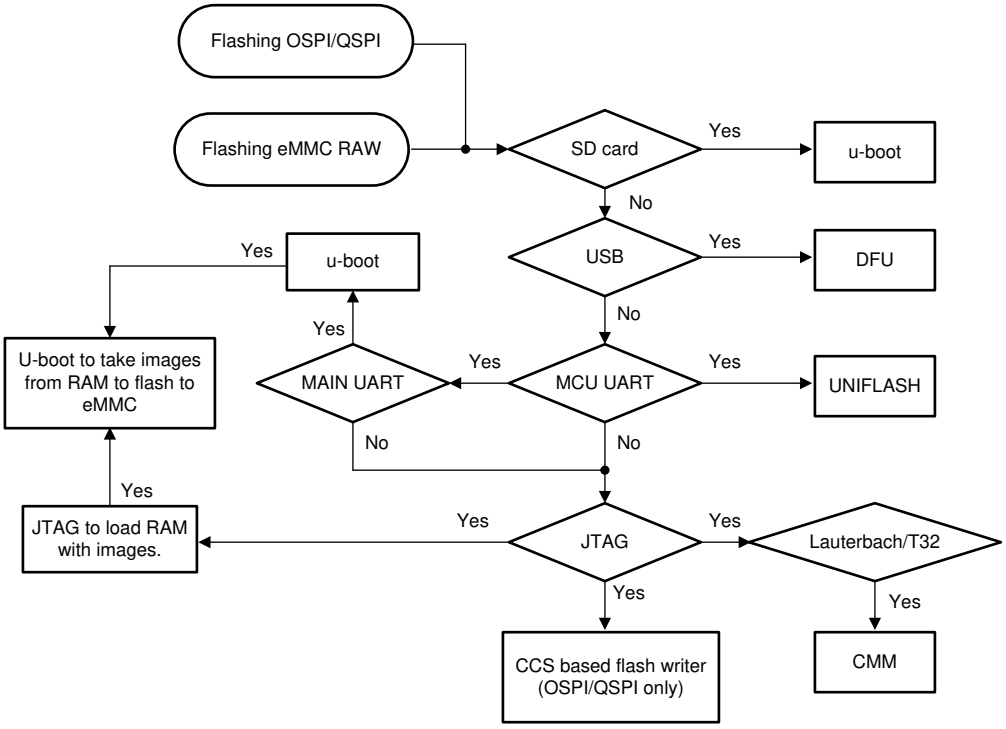

图 2-2. 刷写 OSPI 和 eMMC RAW 扇区

2.2 对 eMMC 用户分区进行刷写

考虑到 TDA4 定制电路板上的硬件限制，图 2-3 可用于确定哪些软件工具可用于在 eMMC 用户分区中刷写文件系统。

例如，在客户希望对 eMMC 用户分区进行刷写的情况下，图 2-3 向硬件设计人员明确指出：如果 JTAG 接口、USB 和 SD 卡接口不可用，软件开发将需要 MAIN UART。

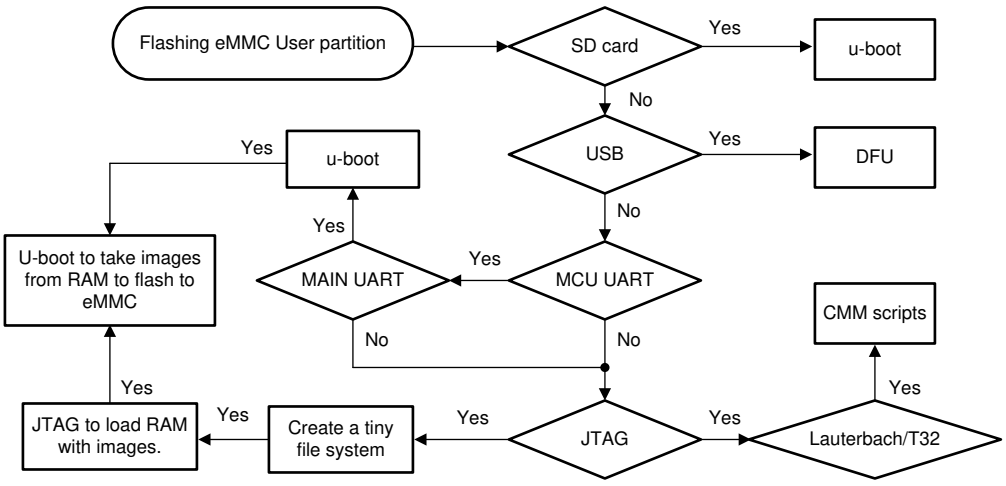

图 2-3. 对 eMMC 用户分区进行刷写

3 刷写 TDA4 的必要条件

3.1 引导开关设置

表 3-1. 引导开关设置

模式	SW8[1:8]	SW9[1:8]	SW3[1:10]
OSPI	0000_0000	0100_0000	**0**xxx_xxxx_xx
eMMC (boot0)	1000_0000	0100_0000	xxxx_xxxx_xx
eMMC(user)	1000_0000	0000_0000	xxxx_xxxx_xx
MMCSD	1000_0010	0000_0000	xxxx_xxxx_xx
NO BOOT	1000_1000	0111_0000	xxxx_xxxx_xx
UART	0000_0000	0111_0000	xxxx_xxxx_xx
DFU	1000_0000	0010_0000	0101_0010_10

3.2 如何生成微型文件系统

按照以下说明创建微型文件系统。默认的 tisdk-tiny-image-j7-evm.tar.xz 也与 SDK 打包在一起。

备注

从更新的 SDK 安装程序获取配置文件。下面提到的命令用作参考示例。此外，根据所使用的 SoC 更改 MACHINE。

```
git clone git://arago-project.org/git/projects/oe-layersetup.git tisdk
cd tisdk/

# Copy config file from the 7.00 installer yocto-build/configs/psdkla/psdkla-07_00_00.txt
./oe-layertool-setup.sh -f psdkla-07_00_00.txt

# export proxy
export ftp_proxy=http://webproxy.ext.ti.com:80
export http_proxy=http://webproxy.ext.ti.com:80
export https_proxy=http://webproxy.ext.ti.com:80

cd build
# Edit config file if you want to reuse downloads folder
vi conf/local.conf
. conf/setenv

# this is the path to the toolchains for ARMv7 and ARMv8
TOOLCHAIN_BASE=/sdk/tools MACHINE=j7-evm bitbake -k tisdk-tiny-image

# build image can be found here
la -la tisdk/build/arago-tmp-external-arm-glibc/deploy/images/j7-evm/tisdk-tiny-image-j7-evm.tar.xz
```

3.3 生成 eMMC tisdk-tiny-image.img

按照以下说明从 tisdk-tiny-image-j7-evm.tar.xs 创建一个微型文件系统映像 (tisdk-tiny-image.img)。

```
# Below code needs to run on the HOST machine

# Create an empty image file, seek value will create an image of that size.Typically 100MB should
be enough.
# Size will should be slightly greater than size of untarred tisdk-tiny-image-j7-evm.tar.xz + size
(DTBs + Kernel image)
dd if=/dev/null of=tisdk-tiny-image.img bs=1M seek=100

# Add a filesystem to it
mkfs.ext4 -F tisdk-tiny-image.img

# Mount it on your local machine to copy DTB and Kernel image
mkdir example_mnt_pt
sudo mount -t ext4 -o loop tisdk-tiny-image.img /home/karan/yocto-build/example_mnt_pt

# Untar tisdk-tiny-image-j7-evm.tar.xz on the mounted file system which is currently empty
cd example_mnt_pt
sudo tar xvf ../tisdk-tiny-image-j7-evm.tar.xz
```

```
# Copy DTB, DTBOs and Kernel Image for your platform
sudo cp /media/karan/rootfs/boot/Image* ./boot
sudo cp /media/karan/rootfs/boot/*dtb* ./boot
sync

# Unmount the image, now the tisdk-tiny-image.img is ready
cd ../
sudo umount /home/karan/yocto-build/example_mnt_pt
```

3.4 运行直至 u-boot

u-boot 是一款用于刷写引导程序二进制文件的强大工具。它还可以与 JTAG 一同刷写文件系统。

以下几节将讨论在电路板上引导 u-boot 的一些技术。

3.4.1 UART 引导模式

UART 引导是 TDA4VM 支持的外设引导模式之一。当诸如 SD 接口等主引导介质不可用时，它会非常有用。

ROM 支持通过 X-Modem 协议从 MCU_UART0 引导。基于 UART 的引导直至进入 U-boot（正确）提示符的整个过程会经历不同的阶段，并使用不同的 UART 外设，如下所示：

加载者	加载内容	硬件模块	协议
引导 ROM	tiboot3.bin (R5 SPL)	MCU_UART0	X-Modem
R5 SPL	sysfw.itb	MCU_UART0	Y-Modem
R5 SPL	tispl.bin (A72 SPL)	MAIN_UART0	Y-Modem
A72 SPL	u-boot.img	MAIN_UART0	Y-Modem

有关使用 UART 引导模式的详细过程，请参阅 faq-tda4vm-detailed-step-for-uart-boot。

3.4.2 DFU 引导

DFU 引导可用于将映像传输至 TDA4 板以进行引导，直至进入 u-boot。请参考以下说明，以使用 DFU 引导进入 u-boot 提示符。

1. 将引导模式更改为 DFU 引导模式并接通电源。有关引导开关设置，请参阅节 3.1。
2. 在主机 PC 上，运行以下命令：

```
HOST $ sudo dfu-util -l
HOST $ sudo dfu-util -R -a bootloader -D <PATH_TO_BIN>/tiboot3.bin
HOST $ sudo dfu-util -R -a sysfw.itb -D <PATH_TO_BIN>/sysfw.itb
HOST $ sudo dfu-util -R -a tispl.bin -D <PATH_TO_BIN>/tispl.bin
HOST $ sudo dfu-util -R -a u-boot.img -D <PATH_TO_BIN>/u-boot.img

# At this point, the u-boot will start executing.Halt at the u-boot prompt (u-boot logs will
appear on the MAIN UART 1st instance)
```

3. 此时，u-boot 将启动，用户可以按任意键停止。

3.4.3 SD 引导或任何其他引导模式

如果电路板上有 SD 引导或任何其他引导模式功能，只需使用该功能进行引导，直到进入 u-boot，然后就可以使用所有 u-boot 实用程序。

3.5 配置 Boot0 分区和对 eMMC 进行分区

需要进行一次性配置，以便 ROM 能够访问 eMMC 的 boot0 分区。

```
# Run the below from u-boot prompt

# Write partition table to eMMC
setenv mmcdev 0
gpt write mmc 0 ${partitions}

#one time only per board, to give ROM access to the boot partition, the following commands must be
used for the first time
mmc partconf 0 1 1 1
mmc bootbus 0 2 0 0
```

4 OSPI 刷写

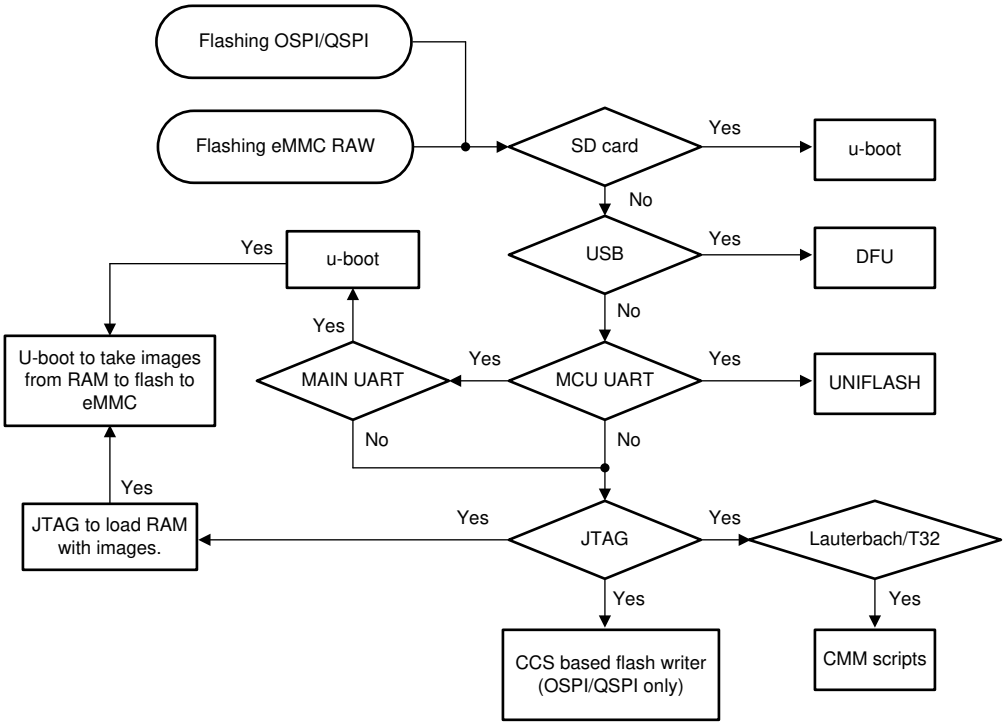

图 4-1. OSPI 刷写

4.1 刷写引导加载程序二进制文件

以下工具可用于将引导加载程序二进制文件刷写到 OSPI 闪存：

- UNIFLASH 工具
- dfu-util
- CCS/JTAG
- Trace32/Lauterbach
- U-boot

4.1.1 TI UNIFLASH 工具

1. 从此处下载 UNIFLASH 工具。
2. 可从此处查阅适用于 TDA4 的 UNIFLASH 的文档。

4.1.1.1 刷写说明

以下是刷写 Linux 引导加载程序映像和 RTOS 映像的示例。

4.1.1.2 Linux 引导二进制文件

```
# Send UART flash programmer
sudo ./dslite.sh --mode processors -c /dev/ttyUSB1 -f /home/karan/ti/uniflash_6.1.0/processors/
FlashWriter/j721e_evm/uart_j721e_evm_flash_programmer_release.tiimage -i 0

# Flash R5 SPL
sudo ./dslite.sh --mode processors -c /dev/ttyUSB1 -f /home/karan/sdk7_3/board-support/prebuilt-
images/tiboot3.bin -d 3 -o 0

# Flash A72 SPL
sudo ./dslite.sh --mode processors -c /dev/ttyUSB1 -f /home/karan/sdk7_3/board-support/prebuilt-
images/tispl.bin -d 3 -o 80000

# Flash u-boot
sudo ./dslite.sh --mode processors -c /dev/ttyUSB1 -f /home/karan/sdk7_3/board-support/prebuilt-
images/u-boot.img -d 3 -o 280000

# Flash sysfw.itb
sudo ./dslite.sh --mode processors -c /dev/ttyUSB1 -f /home/karan/sdk7_3/board-support/prebuilt-
images/sysfw.itb -d 3 -o 6C0000

# Flash phy training data
sudo ./dslite.sh --mode processors -c /dev/ttyUSB1 -f /home/karan/sdk7_3/ti-processor-sdk-rtos-
j721e-evm-07_03_00_07/pdk_jacinto_07_03_00_29/packages/ti/board/src/flash/nor/ospi/
nor_spi_patterns.bin -d 3 -o 3FE0000
```

4.1.1.3 RTOS 引导二进制文件

```
# Send UART flash programmer
sudo ./dslite.sh --mode processors -c /dev/ttyUSB1 -f /home/karan/ti/uniflash_6.1.0/processors/
FlashWriter/j721e_evm/uart_j721e_evm_flash_programmer_release.tiimage -i 0

# Flash R5 SPL
sudo ./dslite.sh --mode processors -c /dev/ttyUSB1 -f /home/karan/sdk7_3/board-support/prebuilt-
images/tiboot3.bin -d 3 -o 0

# Flash A72 SPL
sudo ./dslite.sh --mode processors -c /dev/ttyUSB1 -f /home/karan/sdk7_3/board-support/prebuilt-
images/tispl.bin -d 3 -o 80000

# Flash u-boot
sudo ./dslite.sh --mode processors -c /dev/ttyUSB1 -f /home/karan/sdk7_3/board-support/prebuilt-
images/u-boot.img -d 3 -o 280000

# Flash sysfw.itb
sudo ./dslite.sh --mode processors -c /dev/ttyUSB1 -f /home/karan/sdk7_3/board-support/prebuilt-
images/sysfw.itb -d 3 -o 6C0000

# Flash phy training data
sudo ./dslite.sh --mode processors -c /dev/ttyUSB1 -f /home/karan/sdk7_3/ti-processor-sdk-rtos-
j721e-evm-07_03_00_07/pdk_jacinto_07_03_00_29/packages/ti/board/src/flash/nor/ospi/
nor_spi_patterns.bin -d 3 -o 3FE0000
```

4.2 dfu-util

dfu-util 是 DFU 规格的主机端实现。器件固件升级功能可用于将固件编程到 OSPI。

4.2.1 刷写说明

1. 将主机 PC 连接到电路板的 MAIN UART，并将 minicom 连接到第一个实例。
2. 将 USB Type C 电缆连接到主机 C (Linux)。
3. 将引导模式更改为 DFU 引导模式。有关引导开关设置，请参阅节 3.1。
4. 在**主机 (PC)** 和**目标 (TDA4 EVM 的 u-boot 提示符)** 上运行以下代码：

```
# This will download the images to the board but not flash them to OSPI
# These first set of steps are optional if you have u-boot running on the board already
HOST $ sudo dfu-util -l
HOST $ sudo dfu-util -R -a bootloader -D <PATH_TO_BIN>/tiboot3.bin
HOST $ sudo dfu-util -R -a sysfw.itb -D <PATH_TO_BIN>/sysfw.itb
HOST $ sudo dfu-util -R -a tispl.bin -D <PATH_TO_BIN>/tispl.bin
HOST $ sudo dfu-util -R -a u-boot.img -D <PATH_TO_BIN>/u-boot.img

# At this point, the u-boot will start executing.Halt at the u-boot prompt (u-boot logs will
appear on the MAIN UART 1st instance)
TARGET => env default -f -a
TARGET => saveenv
TARGET => setenv dfu_alt_info ${dfu_alt_info_ospi}

TARGET => dfu 0 sf 0:0

# This does the actual flashing to OSPI flash
HOST $ sudo dfu-util -l
HOST $ sudo dfu-util -a tiboot3.bin -D <PATH_TO_BIN>/tiboot3.bin
HOST $ sudo dfu-util -a tispl.bin -D <PATH_TO_BIN>/tispl.bin
HOST $ sudo dfu-util -a u-boot.img -D <PATH_TO_BIN>/u-boot.img
HOST $ sudo dfu-util -a sysfw.itb -D <PATH_TO_BIN>/sysfw.itb
```

5. 将引导模式更改为 OSPI 引导模式并接通电源，现在系统应该处于 u-boot 提示符下。有关引导开关设置，请参阅节 3.1。

备注

在 SDK7.3 中，dfu_alt_info_ospi 没有适用于 PHY 调优数据的刷写信息。这将在即将发布的版本中进行更新。

4.3 CCS/JTAG

在 PDK 上应用 faq-how-to-flash-ospi-using-ccs-on-tda4x-dra82-evm 中提及的附加补丁，该补丁作为 PSDKRA 的一部分提供。该补丁增加了一个实用程序，旨在使用 CCS 和 JTAG 接口刷写 OSPI。

4.3.1 刷写说明

1. 在 PDK 上应用 0001-Add-support-for-flashing-OSPI-flash-using-CCS.patch。
2. 在 pdk/packages/ti/board/utils/uniflash/target/build/uart_make.mk 文件中将 USE_CCS 标志更改为“yes”。
3. 使用 make -sj PLATFORM=j721e_evm board_utils_uart_flash_programmer 从 pdk/packages/ti/build 重新编译闪存编程器。
4. Flash Writer 二进制文件在 pdk/packages/ti/board/utils/uniflash/target/bin/j721e_evm/uart_j721e_evm_flash_programmer_release.xer5f 中提供。
5. 启动 CCS 和目标配置。有关此步骤的详细信息，请查看此处。
6. 在 Scripting Console 中，从 CCS Scripting Console 中运行 loadJsFile("pdk/packages/ti/drv/Sciclient/tools/ccsLoadDmsc/j721e/launch.js")。
7. 完成 Java 脚本后，在 MCU_Cortex®_R5_0 内核中加载闪存写入器代码并运行它。
8. 应用程序在 CCS 控制台上运行并显示 3 个菜单选项：擦除 OSPI 闪存、在 OSPI 中刷写文件以及退出。
9. 若要擦除闪存，请选择要擦除的起始地址和大小。
10. 若要刷写文件，请输入文件名以及完整路径，接着输入要刷写文件的偏移量，然后使用 CCS Scripting Console 加载文件 (加载命令显示在 CCS 控制台上)。从脚本控制台加载文件后，在 CCS 控制台上输入“0”。
11. 所有文件都完成刷写后，按主菜单上的“2”以退出。

4.4 Trace32/Lauterbach

可以使用 Trace32 和 Lauterbach 将引导加载程序二进制文件刷写到 OSPI 闪存。使用随附的 DRA829/J721e/TDA4VM 特定 cmm 脚本。

4.4.1 刷写说明

1. TDA4VM 板首先需要处于无引导模式。
2. 使用文本编辑器打开附加的文件 dra82x-ospi-MT35XU512-snor.cmm。根据安装程序中的文件更改以下路径：

```
O "/opt/t32/cmm-richard/cmm-dra/cmm-tda4_dra829/x_gel_to_cmm_public/j7es_m3.cmm"

DO "/opt/t32/cmm-richard/cmm-dra/cmm-tda4_dra829/x_gel_to_cmm_public/J721E.cmm"

Data.load.binary /home/keerthy/work/ti-processor-sdk-linux-automotive-j7-evm-07_00_01/board-
support/prebuilt-images/tiboot3.bin 0x50000000

Data.load.binary /home/keerthy/work/ti-processor-sdk-linux-automotive-j7-evm-07_00_01/board-
support/prebuilt-images/tispl.bin 0x50080000

Data.load.binary /home/keerthy/work/ti-processor-sdk-linux-automotive-j7-evm-07_00_01/board-
support/prebuilt-images/u-boot.img 0x50280000

Data.load.binary /home/keerthy/work/ti-processor-sdk-linux-automotive-j7-evm-07_00_01/board-
support/prebuilt-images/sysfw.itb 0x506C0000
```

3. 打开 Linux 终端并运行下面的命令。这将打开 trace32 主窗口。

```
cd /opt/t32/bin/pc_linux64
./t32marm
```

4. 运行文件 dra82x-ospi-MT35XU512-snor.cmm 的脚本，这需要几分钟的时间来完成烧写。切换至 OSPI 引导模式设置，并引导至 u-boot 提示符。

4.5 u-boot

u-boot 在电路板上运行后，用户就可以使用 u-boot 命令来刷写 OSPI 闪存。

4.5.1 刷写说明

例如，如果 u-boot 是通过 MMCSD 引导功能进行引导的，则可以使用以下命令来刷写 OSPI 闪存。

1. 引导至 u-boot。
2. 从 u-boot 提示符运行以下命令。

```
# Run below commands at u-boot prompt

=> sf probe
=> fatload mmc 1 ${loadaddr} tiboot3.bin
=> sf update ${loadaddr} 0x0 0x${filesize}
=> fatload mmc 1 ${loadaddr} tispl.bin
=> sf update ${loadaddr} 0x80000 0x${filesize}
=> fatload mmc 1 ${loadaddr} u-boot.img
=> sf update ${loadaddr} 0x280000 0x${filesize}
=> fatload mmc 1 ${loadaddr} sysfw.itb
=> sf update ${loadaddr} 0x6C0000 0x${filesize}
=> fatload mmc 1 ${loadaddr} nor_spi_patterns.bin
=> sf update ${loadaddr} 0x3FE0000 0x${filesize}
```

3. 将 dip 开关切换至 OSPI 引导模式后，关闭电路板电源，然后接通电源。有关引导开关设置，请参阅节 3.1。

5 eMMC 刷写

5.1 刷写引导加载程序二进制文件

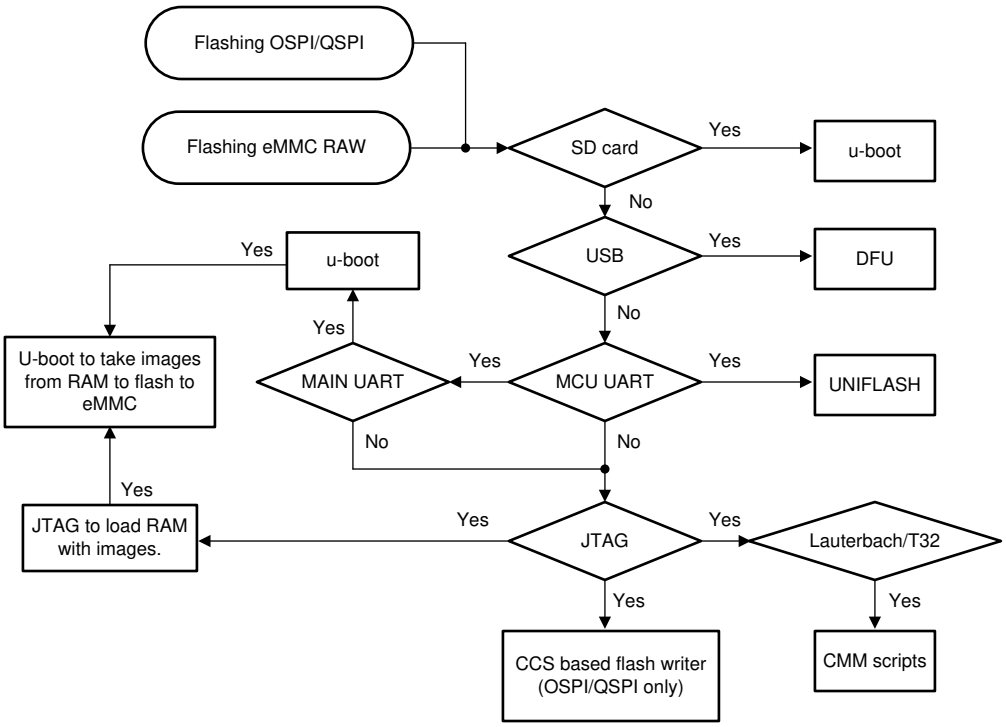

图 5-1. eMMC 刷写

5.1.1 TI UNIFLASH 工具

使用 UNIFLASH，可以刷写 eMMC boot0 分区。可在此处找到 UNIFLASH 文档。

5.1.1.1 刷写说明

1. 将引导模式更改为 UART 引导模式。有关引导开关设置，请参阅节 3.1。
2. 在主机（Windows 或 Linux，相应地使用 dslite.bat 或 dslite.sh）上运行以下命令。

```
# Send UART flash programmer

sudo ./dslite.sh --mode processors -c /dev/ttyUSB1 -f /home/karan/ti/uniflash_6.1.0/processors/
FlashWriter/j721e_evm/uart_j721e_evm_flash_programmer_release.tiimage -i 0

sudo ./dslite.sh --mode processors -c /dev/ttyUSB1 -f /home/karan/sdk7_3/board-support/prebuilt-
images/tiboot3.bin -d 4 -o 0

sudo ./dslite.sh --mode processors -c /dev/ttyUSB1 -f /home/karan/sdk7_3/board-support/prebuilt-
images/tispl.bin -d 4 -o 80000

sudo ./dslite.sh --mode processors -c /dev/ttyUSB1 -f /home/karan/sdk7_3/board-support/prebuilt-
images/u-boot.img -d 4 -o 280000

sudo ./dslite.sh --mode processors -c /dev/ttyUSB1 -f /home/karan/sdk7_3/board-support/prebuilt-
images/sysfw.itb -d 4 -o 6C0000
```

备注

进行刷写时，MCU_UART0 不应在任何串行端口查看器中打开。此外，在 Windows PC 上，在执行刷写步骤之前，需要手动拔出 MCU UART 电缆，然后重新插入。

3. 将引导模式更改为 eMMC（引导）并接通电源。有关引导开关设置，请参阅节 3.1。

5.1.2 Trace32/Lauterbach

可以使用 Trace32 和 Lauterbach 将引导加载程序二进制文件刷写到 eMMC。使用随附的 DRA829/J721e/TDA4VM 特定 cmm 脚本。

5.1.2.1 刷写说明

1. 使用文本编辑器打开附加的文件 dra82x-emmc.cmm。根据安装程序中的文件更改以下路径：

```
DO "/opt/t32/cmm-richard/cmm-dra/cmm-tda4_dra829/x_gel_to_cmm_public/j7es_m3.cmm"

DO "/opt/t32/cmm-richard/cmm-dra/cmm-tda4_dra829/x_gel_to_cmm_public/J721E.cmm"

FLASHFILE.LOAD /home/keerthy/work/ti-processor-sdk-linux-automotive-j7-evm-07_00_01/board-
support/prebuilt-images/tiboot3.bin 0x0

FLASHFILE.LOAD /home/keerthy/work/ti-processor-sdk-linux-automotive-j7-evm-07_00_01/board-
support/prebuilt-images/tispl.bin 0x80000

FLASHFILE.LOAD /home/keerthy/work/ti-processor-sdk-linux-automotive-j7-evm-07_00_01/board-
support/prebuilt-images/u-boot.img 0x280000

FLASHFILE.LOAD /home/keerthy/work/ti-processor-sdk-linux-automotive-j7-evm-07_00_01/board-
support/prebuilt-images/sysfw.itb 0x6C0000
```

2. 在终端上，运行以下命令。上述命令应打开 trace32 主窗口。

```
cd /opt/t32/bin/pc_linux64
./t32marm
```

3. 运行文件 dra82x-emmc.cmm 脚本；这需要几分钟时间来进行烧写。
4. 切换至 eMMC boot0 开关设置，并引导至 u-boot 提示符。

5.1.3 dfu-util

dfu-util 是 DFU 规格的主机端实现。器件固件升级功能可用于将固件编程到 eMMC boot0 分区。

5.1.3.1 刷写说明

1. 将主机 PC 连接到电路板的 MAIN UART，并将 minicom 连接到第一个实例。
2. 将 USB Type C 电缆连接到主机 C (Linux)。
3. 将引导模式更改为 DFU 引导模式。有关引导开关设置，请参阅节 3.1。
4. 在**主机 (PC)** 和**目标 (TDA4 EVM 的 u-boot 提示符)** 上运行以下命令。

```
# This will download the images to the board but not flash them to eMMC
# These first set of steps are optional if you have u-boot running on the board already
HOST $ sudo dfu-util -l
HOST $ sudo dfu-util -R -a bootloader -D <PATH_TO_BIN>/tiboot3.bin
HOST $ sudo dfu-util -R -a sysfw.itb -D <PATH_TO_BIN>/sysfw.itb
HOST $ sudo dfu-util -R -a tispl.bin -D <PATH_TO_BIN>/tispl.bin
HOST $ sudo dfu-util -R -a u-boot.img -D <PATH_TO_BIN>/u-boot.img

# At this point, the u-boot will start executing.Halt at the u-boot prompt (u-boot logs will
appear on the MAIN UART 1st instance)
TARGET => env default -f -a
TARGET => setenv mmcdev 0
TARGET => setenv bootpart 0
TARGET => saveenv
TARGET => setenv dfu_alt_info ${dfu_alt_info_emmc}

# one time only per board
TARGET => gpt write mmc 0 ${partitions}

TARGET => dfu 0 mmc 0

# This does the actual flashing to the eMMC boot0 partition
HOST $ sudo dfu-util -l
HOST $ sudo dfu-util -a tiboot3.bin.raw -D <PATH_TO_BIN>/tiboot3.bin
HOST $ sudo dfu-util -a tispl.bin.raw -D <PATH_TO_BIN>/tispl.bin
HOST $ sudo dfu-util -a u-boot.img.raw -D <PATH_TO_BIN>/u-boot.img
HOST $ sudo dfu-util -a sysfw.itb.raw -D <PATH_TO_BIN>/sysfw.itb

# Flashing a tiny file system to eMMC User partition
```

```
HOST $ sudo dfu-util -a rootfs -D <PATH_TO_CREATED_TINYFS>/tinyrootfs.img

#one time only per board, to give ROM access to the boot partition, the following commands must
be used for the first time
TARGET => mmc partconf 0 1 1 1
TARGET => mmc bootbus 0 2 0 0
```

5. 将引导模式更改为 eMMC (boot0) 引导模式并接通电源。现在系统应该处于 u-boot 提示符下。有关引导开关设置，请参阅节 3.1。

5.2 u-boot

u-boot 在电路板上运行后，用户就可以使用“mmc write”等 u-boot 命令刷写 eMMC boot0 分区。

5.2.1 刷写说明

1. 引导至 u-boot。
2. 从 u-boot 提示符运行以下命令。

```
# Run below commands at u-boot prompt

=> mmc dev 0 1
=> fatload mmc 1 ${loadaddr} tiboot3.bin
=> mmc write ${loadaddr} 0x0 0x400
=> fatload mmc 1 ${loadaddr} tispl.bin
=> mmc write ${loadaddr} 0x400 0x1000
=> fatload mmc 1 ${loadaddr} u-boot.img
=> mmc write ${loadaddr} 0x1400 0x2000
=> fatload mmc 1 ${loadaddr} sysfw.itb
=> mmc write ${loadaddr} 0x3600 0x800
```

3. 将引导更改为 eMMC(boot) 引导模式后，关闭电路板电源，然后接通电源。有关引导开关设置，请参阅节 3.1。

5.3 使用 tinyrootfs 在 eMMC UDA 分区中进行刷写

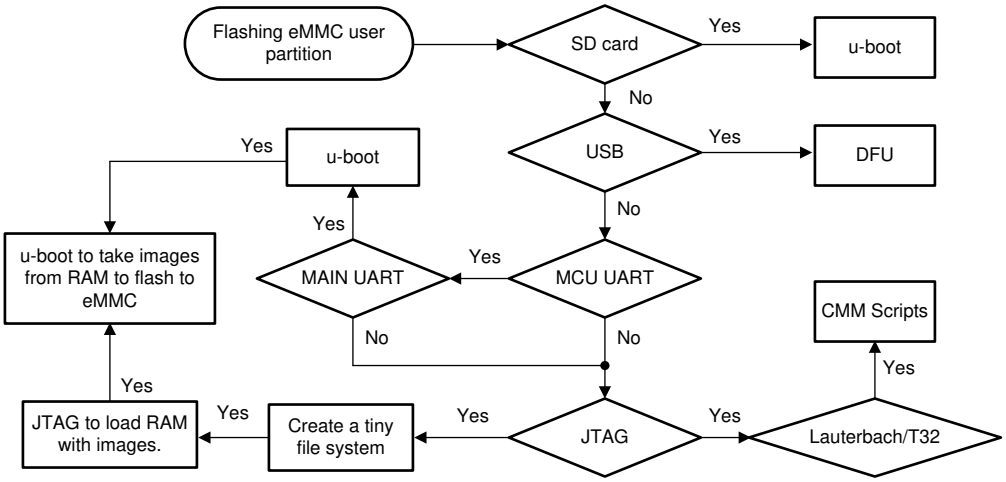

图 5-2. Rootfs 刷写到 eMMC

5.3.1 dfu-util

如前一节所述，dfu-util 还可用于将文件系统刷写到 eMMC UDA 分区。

5.3.2 u-boot + CCS/JTAG

Linux tinyrootfs 复制可以通过 CCS 和调试器来执行。

一旦在 boot0 分区中刷写了引导加载程序二进制文件 (u-boot)，u-boot 实用程序就可以与 CCS + JTAG 一同用于在 eMMC UDA 分区中刷写微型文件系统

5.3.2.1 刷写说明

1. 在 u-boot 处停止。
2. 使用 XDS110/XDS560v2 和适当的目标 ccxml（不含 GEL 文件）连接 CCS。
3. 将 tisdk-tiny-image.img 加载到 DDR 地址 0x80080000。在本文档中，请参阅节 3.2 了解微型文件系统，然后参阅节 3.3 创建 *.img。
   a. 启动目标 ccxml 并连接到 CortexA72_0_0。
   b. 打开存储器浏览器并加载存储器：“View”-> “Memory Browser”-> “Load Memory”。
   c. 选择文件 tisdk-tiny-image.img，并选择“Binary”作为“File Type”，然后点击“Next”。
   d. 将“Start Address”设置为 0x80080000，然后点击“Finish”。
   e. 根据使用的 JTAG，加载所需的时间会有所不同。XDS110 预计需要 1 小时，XDS560v2 预计需要 20 分钟。
4. 映像被加载到 RAM 中，并使用 u-boot 进行刷写。

```
# Run the below command from u-boot prompt
=> part start mmc 0 rootfs

# what ever the above command outputs, use that below (eg - 0x22)
=> mmc write 0x80080000 0x22 0x32000

# after the above completes
=> boot
```

重要声明和免责声明

TI 提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、某特定用途方面的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他安全、安保或其他要求。这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的应用。严禁对这些资源进行其他复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。您应全额赔偿因在这些资源的使用中对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，TI 对此概不负责。

TI 提供的产品受 TI 的销售条款 (https:www.ti.com/legal/termsofsale.html) 或 ti.com 上其他适用条款/TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。

邮寄地址：Texas Instruments, Post Office Box 655303, Dallas, Texas 75265

重要声明和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、某特定用途方面的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他功能安全、信息安全、监管或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的应用。严禁对这些资源进行其他复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。您应全额赔偿因在这些资源的使用中对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，TI 对此概不负责。

TI 提供的产品受 TI 的销售条款或 ti.com 上其他适用条款/TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

邮寄地址：Texas Instruments, Post Office Box 655303, Dallas, Texas 75265