Application Report

TDA4 Flashing Techniques

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

ABSTRACT

This application report focuses on the flashing techniques for the TDA4 family of devices. This includes flashing of customer boards while taking into account the constraints of the hardware interfaces and software tools available.

Project collateral mentioned in this document can be downloaded from the following URL: https://www.ti.com/lit/zip/spracy5.

Table of Contents

Trademarks

1 Introduction to Flashing Tools

For a particular boot media, given the hardware interfaces pinned out on the board, intent is to find a feasible solution to flash that particular boot media with bootloader and file system images.

Most software tools have some constraints that are listed in Table 1-1. From this, software tools that are supported for a particular boot media can be inferred.

Table 1-1. Software Tools Constraints

Software Tools	Hardware Interface	Boot Media			Constraints
		OSPI NOR	eMMC	QSPI NOR	
UNIFLASH	UART	L(1) Only bootloader images can be flashed. UBIFS cannot be flashed.	L Only eMMC boot0 (RAW) partition can be flashed.	L Only bootloader images can be flashed. UBIFS cannot be flashed.	• Support for UART boot mode. • MCU_UART0 should be pinned out on the board. • Flashing of only raw sectors supported for eMMC
CCS	JTAG	L Only bootloader images can be flashed. UBIFS cannot be flashed.	L Only eMMC user (UDA) can be flashed with tiny file system. Bootloader images should be already flashed.	L Only bootloader images can be flashed. UBIFS cannot be flashed.	• Presence of debugger interface. • Flashing speed depend on choice of JTAG.
Lauterbach	JTAG	Y(2)	Y	Y	• Licensed software
u-boot	UART, MMCSD, DFU boot	Y Using SD card, JTAG, Ethernet etc. with u-boot.	Y Using SD card, JTAG, Ethernet etc. with u-boot.	Y Using SD card, JTAG, Ethernet etc. with u-boot.	• u-boot should be flashed using other interfaces first.
DFU	USB	Y	Y	Y	

(1) L = Limited support
(2) Y = Full support

Note

UNIFLASH by default uses MCU_UART0.
Universal asynchronous receiver/transmitter (UART) boot to u-boot needs MCU_UART0 and one of the MAIN_UART to which u-boot will print.

1.1 Trace32/Lauterbach

Trace32/Lauterbach is a powerful tool when it comes to Arm®-based SoCs.

Trace32 can also be used to flash various memory types. JTAG interface needs to be pinned out on the board to use this approach. This enables customers who have Trace32 to flash boot images to eMMC and SPI memories.

Process of flashing involves CMM scripting that Lauterbach comprehends and a one time installation on Linux PC to connect to the TDA4. Using Trace32/Lauterbach to flash avoids dependency on a secondary boot media like UART/SD to burn images to the flash parts on the custom board.

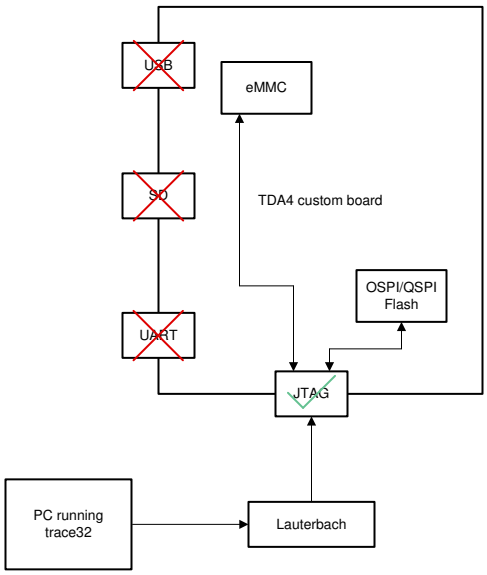

Figure 1-1. Trace32/Lauterbach Flashing

1.2 CCS-Based Flash Writer

Some custom boards only have a JTAG interface along with OSPI/QSPI flash. In absence of Trace32/ Lauterbach, there is a Code Composer Studio™-based software flash writer application to flash the on-board flashes.

The solution provides an application that runs on MCU R5 and then provides a user-friendly menu to write and erase to the OSPI/QSPI flash.

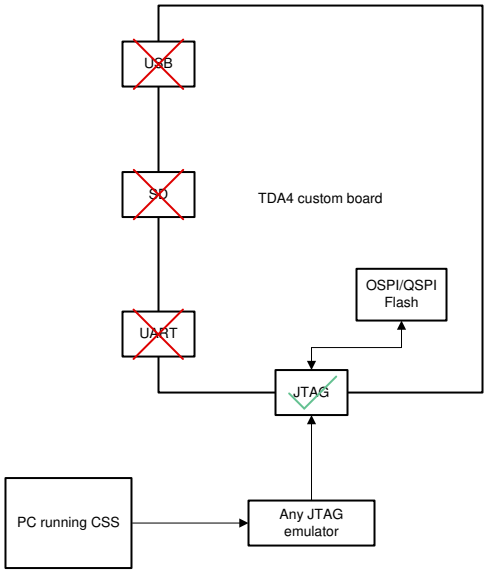

Figure 1-2. CCS-Based Flash Writer Flashing

1.3 Other Software Tools

- **UNIFLASH**
  - This tool can be used to flash RAW sectors for OSPI/QSPI and eMMC.
  - UART interface should be pinned out on the custom board.
  - UART boot mode should be supported.
- **dfu-util**
  - USB-based tool that can be used to flash both OSPI/QSPI and eMMC.
  - Only requirement is a USB interface.
- **U-boot**
  - Once u-boot is running on the board, u-boot utilities can be used to flash the boot media.
  - U-boot + SD card, U-boot + Ethernet, U-boot + CCS are some options that can be used to flash the eMMC and OSPI/QSPI.

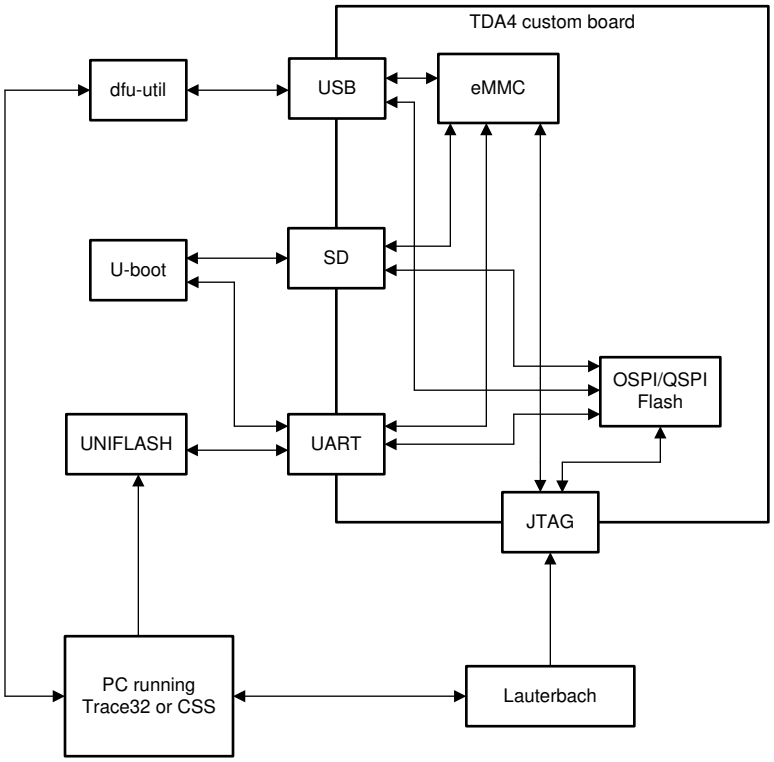

Figure 1-3. Other Software Tools

2 Flash Devices on TDA4

OSPI and eMMC flashes on popular choice of external flashes used on TDA4 boards. Figure 2-1 depicts the default layout for the flash in the SDK, which can be changed in case the custom applications need a different layout.

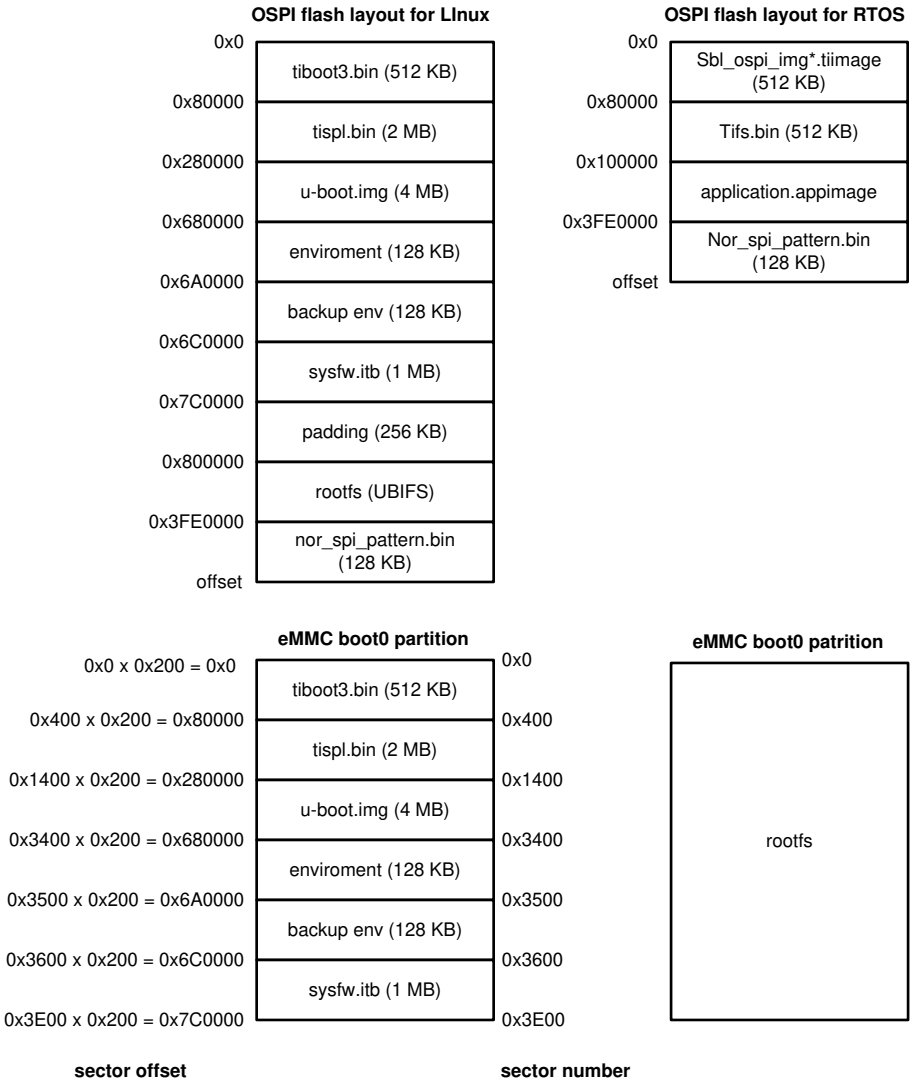

Figure 2-1. Flash Layout in SDK

2.1 Flashing OSPI and eMMC RAW Sectors

Figure 2-2 is useful to decide which software tools can be used to flash the OSPI flash or eMMC boot partitions given the hardware constrains on the TDA4 custom board.

For example, in a case where the customer wants to flash OSPI but has no MAIN UART, USB or SD card interface, Figure 2-2 clearly directs the customer to use JTAG with CCS or Lauterbach or to use UNIFLASH tool.

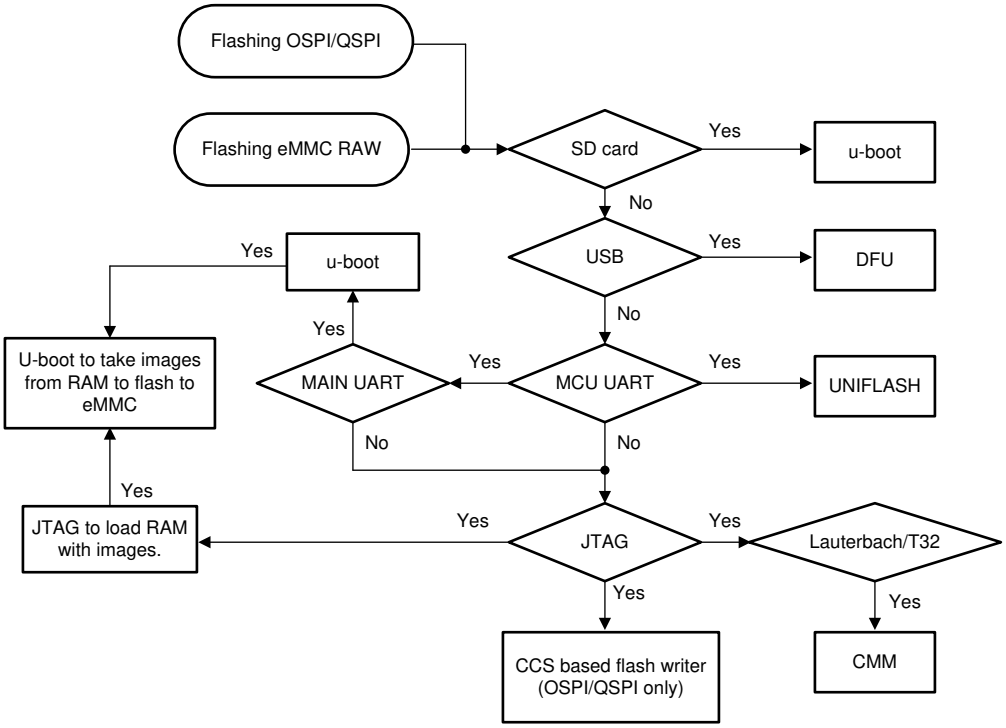

Figure 2-2. Flashing OSPI and eMMC RAW Sectors

2.2 Flashing eMMC User Partition

Figure 2-3 is useful to decide which software tools can be used flash the file system in eMMC user partitions given the hardware constraints on the TDA4 custom board.

For example, in a case where the customer wants to flash the eMMC user partition, Figure 2-3 clearly directs the hardware designer that MAIN UART will be required for Software Development in case JTAG interface, USB and SD card interface is not available.

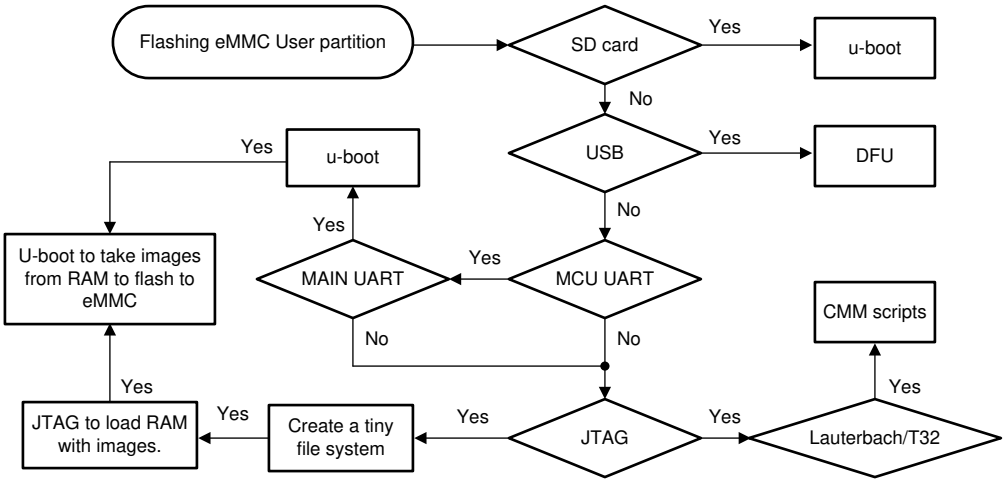

Figure 2-3. Flashing eMMC User Partition

3 Prerequisites for Flashing TDA4

3.1 Boot Switch Settings

Table 3-1. Boot Switching Settings

Mode	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 How to Generate a Tiny File System

Follow the instructions below to create a tiny file system. A default tisdk-tiny-image-j7-evm.tar.xz is also packaged with the SDK.

Note

Take the config file from the latest SDK installer. The below mentioned commands serve as a reference example. Also change the MACHINE based on the SoC being used.

```
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 Generating the eMMC tisdk-tiny-image.img

Follow the instructions below to create a tiny file system image (tisdk-tiny-image.img) from the tisdk-tiny-image-j7-evm.tar.xs.

```
# 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 Running Until u-boot

u-boot is a powerful tool to flash bootloader binaries. It can also be accompanied by JTAG to flash a file system.

The following sections discuss some of the techniques to boot u-boot on your board.

3.4.1 UART Boot Mode

UART boot is one of the peripheral boot modes supported on TDA4VM. It is very useful when primary boot media like SD interface is not available.

ROM supports booting from MCU_UART0 via X-Modem protocol. The entire UART-based boot process up to U-Boot (proper) prompt goes through different stages and uses different UART peripherals as follows:

WHO	Loading WHAT	Hardware Module	Protocol
Boot 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

For the detailed process for using UART boot mode, see the faq-tda4vm-detailed-step-for-uart-boot.

3.4.2 DFU Boot

DFU boot can be used to transfer images to the TDA4 board to boot till u-boot. Refer the instructions below to get to the u-boot prompt using DFU boot.

1. Change boot mode to DFU boot mode and power on. For boot switch settings, see Section 3.1.
2. On the HOST PC, run the below commands:

```
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. At this point you will have u-boot up, you can press any key to halt.

3.4.3 SD Boot or any Other Boot Mode

If you have SD boot or any other boot mode functional on the board, just use that to boot until u-boot and then you can use all the u-boot utilities.

3.5 Configuring Boot0 Partition and Partitioning eMMC

There needs to be a one-time configuration to give ROM access to the boot0 partition of eMMC.

```
# 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 Flashing

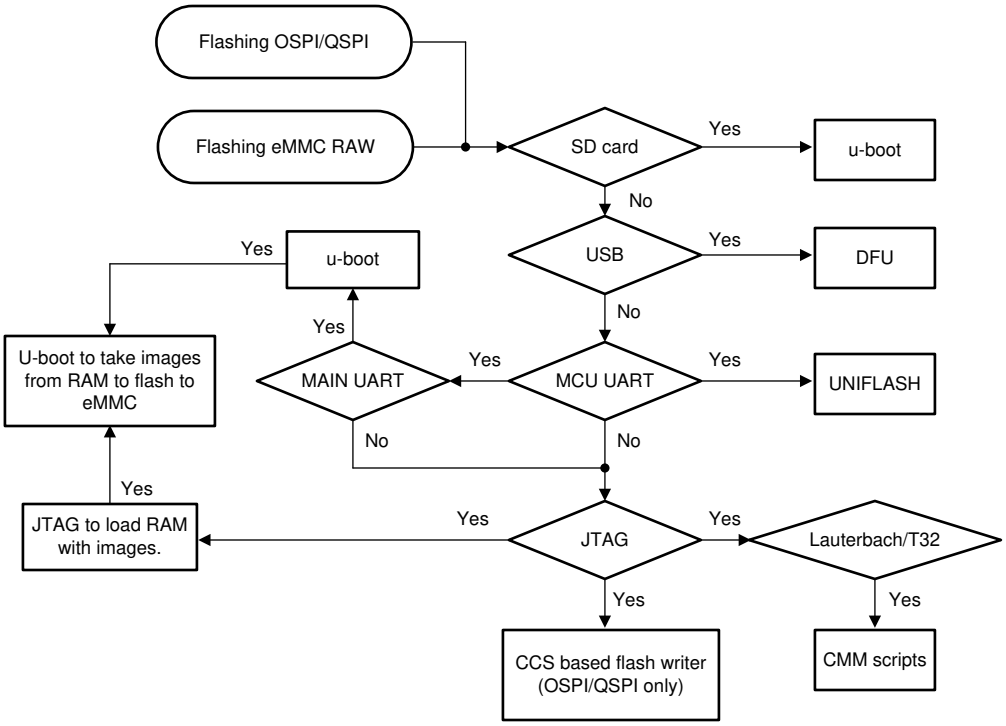

Figure 4-1. OSPI Flashing

4.1 Flashing Bootloader Binaries

The following tools can be used to flash the bootloader binaries to the OSPI flash:

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

4.1.1 TI UNIFLASH Tool

1. Download UNIFLASH tool from here.
2. Documentation for UNIFLASH for TDA4 can be referred from here.

4.1.1.1 Flashing Instructions

The following are examples to flash Linux bootloader images and RTOS images.

4.1.1.2 Linux Boot Binaries

```
# 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 Boot Binaries

```
# 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 is a host side implementation of the DFU specifications. The Device Firmware Upgrade feature can be used to program the firmware to OSPI.

4.2.1 Flashing Instructions

1. Connect Host PC to the MAIN UART of the board and connect minicom to the first instance.
2. Connect the USB Type C cable to the Host PC (Linux).
3. Change boot mode to DFU boot mode. For boot switch settings, see Section 3.1.
4. Run the following code on **HOST (PC)** and **TARGET (TDA4 EVM's u-boot prompt)**:

```
# 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. Change boot mode to OSPI boot mode and power on, you should be at the u-boot prompt. For boot switch settings, see Section 3.1.

Note

In the SDK7.3, the dfu_alt_info_ospi does not have flashing information for PHY tuning data. This will be updated in upcoming releases.

4.3 CCS/JTAG

Apply attached patch as mentioned in faq-how-to-flash-ospi-using-ccs-on-tda4x-dra82-evm on PDK, which is available as part of PSDKRA. This patch adds a utility that can be used to flash OSPI using CCS and JTAG interface.

4.3.1 Flashing Instructions

1. Apply 0001-Add-support-for-flashing-OSPI-flash-using-CCS.patch on the PDK.
2. Change flag USE_CCS to yes in the pdk/packages/ti/board/utils/uniflash/target/build/uart_make.mk file.
3. Rebuild flash programmer by using make -sj PLATFORM=j721e_evm board_utils_uart_flash_programmer from pdk/packages/ti/build.
4. Flash Writer binary would be available in pdk/packages/ti/board/utils/uniflash/target/bin/j721e_evm/ uart_j721e_evm_flash_programmer_release.xer5f.
5. Launch CCS and target configuration. For details on this step, see this.
6. From Scripting Console run loadJsFile("pdk/packages/ti/drv/Sciclient/tools/ccsLoadDmsc/j721e/launch.js") from the CCS scripting console.
7. Once Java script is completed, load flash writer code in MCU_Cortex®_R5_0 core and Run it.
8. Application runs and displays 3 menu options on the CCS console, Erase OSPI flash, Flashing file in OSPI and exit.
9. For erasing flash, select the start address and size to be erased.
10. For flashing a file, enter the file name with the full path, then enter offset where file is to be flashed, then load the file using CCS scripting console (command to load is printed on CCS console). Once file is loaded from scripting console, enter '0' on CCS console.
11. Press '2' on main menu to exit, after all files are flashed.

4.4 Trace32/Lauterbach

Bootloader binaries can be flashed to OSPI flash using Trace32 and Lauterbach. Use the DRA829/J721e/ TDA4VM specific cmm scripts as attached.

4.4.1 Flashing Instructions

1. The TDA4VM board needs to be in no boot mode to begin with.
2. Open the attached file dra82x-ospi-MT35XU512-snor.cmm using a text editor. Change the paths below with respect to the files on your setup:

```
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. Open your Linux terminal and run the command below. This should open the trace32 main window.

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

4. Run the file dra82x-ospi-MT35XU512-snor.cmm script and it would take a couple of minutes to burn. Switch to OSPI boot mode settings and boot to u-boot prompt.

4.5 u-boot

Once u-boot is running on the board, you can use u-boot commands to flash OSPI flash.

4.5.1 Flashing Instructions

For example, if u-boot is booted from MMCSD boot, then the following commands can be used to flash OSPI flash.

1. Boot to u-boot.
2. Run below commands from u-boot prompt.

```
# 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. Power off the board and power on after changing the dip switches to OSPI boot mode. For boot switch settings, see Section 3.1.

5 eMMC flashing

5.1 Flashing Bootloader Binaries

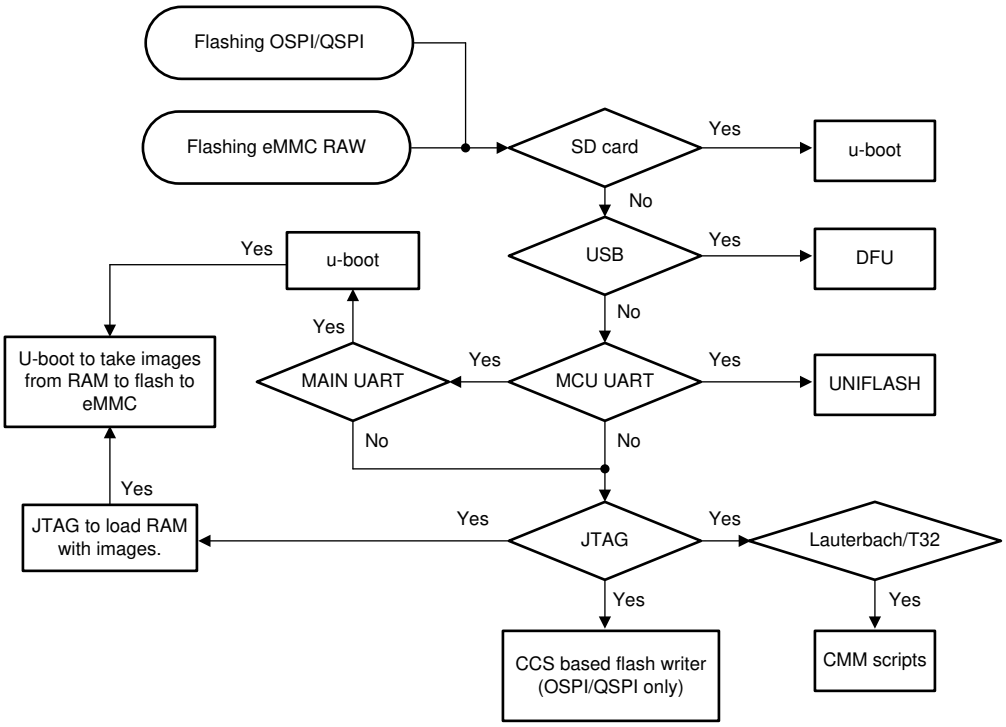

Figure 5-1. eMMC Flashing

5.1.1 TI UNIFLASH Tool

Using UNIFLASH, you can flash the eMMC boot0 partition. UNIFLASH documentation can be found here.

5.1.1.1 Flashing Instructions

1. Change bootmode to UART boot mode. For boot switch settings, see Section 3.1.
2. Run the commands below on the HOST (Windows or Linux, use dslite.bat or dslite.sh accordingly).

```
# 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
```

Note

MCU_UART0 should not be open in any serial port viewer while flashing. Additionally, on Windows PC, you need to manually plug out and plug back in the MCU UART cable before you execute the flashing steps.

3. Change bootmode to eMMC (boot) and power on. For boot switch settings, see Section 3.1.

5.1.2 Trace32/Lauterbach

Bootloader binaries can be flashed to eMMC using Trace32 and Lauterbach. Use the DRA829/J721e/TDA4VM specific cmm scripts as attached.

5.1.2.1 Flashing Instructions

1. Open the attached file dra82x-emmc.cmm using text editor. Change the paths below with respect to the files on your setup:

```
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. On your terminal, run the below. The above should open the trace32 main window.

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

3. Run the file dra82x-emmc.cmm script; it takes a couple of minutes to burn.
4. Switch to eMMC boot0 switch settings and boot to u-boot prompt.

5.1.3 dfu-util

dfu-util is a host side implementation of the DFU specifications. The Device Firmware Upgrade feature can be used to program the firmware to eMMC boot0 partition.

5.1.3.1 Flashing Instructions

1. Connect Host PC to the MAIN UART of the board and connect minicom to the first instance
2. Connect the USB Type C cable to the Host PC (Linux).
3. Change boot mode to DFU boot mode. For boot switch settings, see Section 3.1.
4. Run the below on **HOST (PC)** and **TARGET (TDA4 EVM's u-boot prompt)**.

```
# 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. Change boot mode to eMMC (boot0) boot mode and power on. You should be at the u-boot prompt. For boot switch settings, see Section 3.1.

5.2 u-boot

Once u-boot is running on the board, you can use u-boot commands like “mmc write” to flash eMMC boot0 partition.

5.2.1 Flashing Instructions

1. Boot to u-boot.
2. Run the commands below from the u-boot prompt.

```
# 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. Power off the board and power it on after change boot to eMMC(boot) boot mode. For boot switch settings, see Section 3.1.

5.3 eMMC UDA Partition Flashing With tinyrootfs

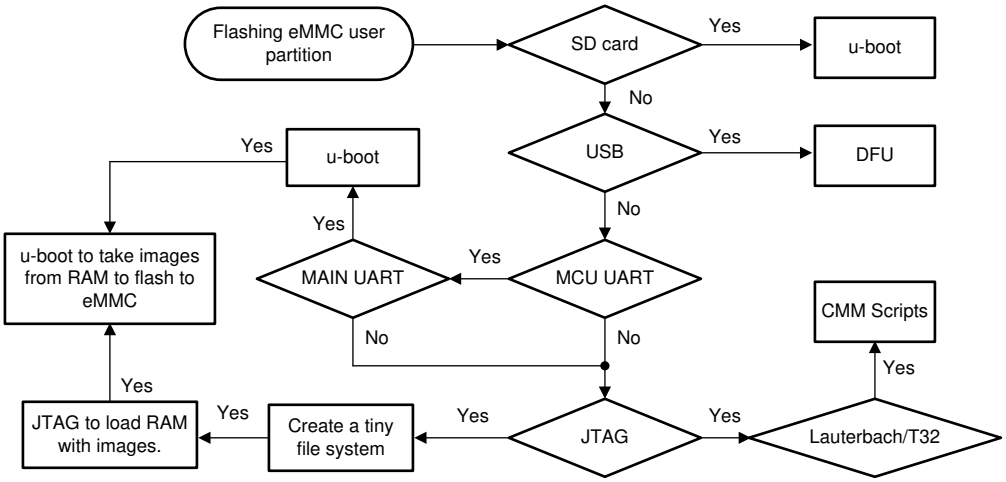

Figure 5-2. Rootfs Flashing to eMMC

5.3.1 dfu-util

As mentioned in the previous section, dfu-util can also be used to flash a file system to the eMMC UDA partition.

5.3.2 u-boot + CCS/JTAG

Linux tinyrootfs copying can be done via CCS and debugger.

Once you have the bootloader binaries (u-boot) flashed in the boot0 partition, then the u-boot utilities can be used along with CCS + JTAG to flash a tiny file system in the eMMC UDA partition

5.3.2.1 Flashing Instructions

1. Halt at u-boot.
2. Connect CCS using the XDS110/XDS560v2 and using the appropriate target ccxml (without GEL files).
3. Load the tisdk-tiny-image.img to the DDR address 0x80080000. In this document, see Section 3.2 for the tiny file system and then Section 3.3 to create the *.img.
   a. Launch the target ccxml and connect to CortexA72_0_0.
   b. Open Memory browser and load memory: View -> Memory Browser -> Load Memory.
   c. Select file tisdk-tiny-image.img and select File Type as Binary, click Next.
   d. Set Start Address as 0x80080000, click Finish.
   e. Depending on the JTAG used, the time taken to load this will vary. Expect 1 hour with XDS110 and 20 minutes with XDS560v2.
4. The image is loaded on the RAM, flash it using u-boot.

```
# 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
```

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (https:www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265