Application Note

Redundant Boot With Unified Boot Mode Using a Single eMMC on TI TDA4x

Johnny Kim *Jacinto 7*

ABSTRACT

In automotive Advanced Driver Assistance Systems (ADAS), system reliability is critical — especially during the boot process. This application note presents a method for achieving redundant boot functionality using a single eMMC device on TI TDA4x Jacinto devices. The system uses embedded MultiMediaCard (eMMC) boot mode as primary and MultiMediaCard/Secure Disk (MMC/SD) boot mode as backup. Both paths share the same eMMC hardware without requiring physical boot mode changes or additional memory devices.

The proposed design makes sure that the system can recover and boot successfully even if the primary boot images are corrupted — for example, due to unexpected power loss during an update. This is accomplished by carefully organizing bootloader images, leveraging different boot modes, and preparing distinct boot paths within the same device. Validation is demonstrated by deliberately corrupting the primary image and observing the automatic fallback to the backup boot path. This approach offers a reliable, cost-effective, and maintenance-friendly redundant boot design for ADAS applications, providing system robustness and availability in real-world automotive environments.

Table of Contents

Trademarks

1 Introduction

TI's TDA4x Jacinto devices support booting from a variety of media such as Octal Serial Peripheral Interface (OSPI) flash memory, eMMC, universal asynchronous receiver-transmitter (UART), and so on. After power-on, the microcontroller unit (MCU) ROM code initiates the boot sequence by reading the physical boot mode, typically configured physically. Based on this configuration, the ROM code determines which boot media to access to locate the required boot images.

By default, the ROM code first attempts to load images from the primary boot media. If the primary boot images are valid and pass integrity and authentication checks, the system boots successfully using these images. However, if the primary boot images are missing or corrupted, the ROM code automatically proceeds to try the backup boot media. The boot process for the backup media follows the same sequence and validation steps as the primary. This dual-path boot mechanism provides the system with two opportunities to boot, greatly improving overall reliability and fault tolerance particularly critical for ADAS applications. The high-level boot sequence follows this process:

1. Power is applied, starting the boot process of the MCU ROM.
2. The MCU ROM code reads the boot mode pins to identify the primary boot source.
3. The MCU ROM code requests the Device Management and Security Controller (DMSC) to verify the integrity of the boot images.
4. If the integrity check passes, the system boots using the primary images.
5. If the integrity check fails, the MCU ROM code reads the boot mode pins again to identify the backup boot source.
6. The DMSC performs integrity checks on the backup images.
7. If successful, the system boots from the backup media.

This process provides redundant boot capability, enabling robust recovery in the event of image corruption or update failure.

The following sections of this application note describe how to configure this redundant boot mechanism using a single eMMC device, eliminating the need to switch physical boot modes or use multiple memory devices.

2 Physical Boot Mode DIP Switch Configuration

There are two sets of pins that the ROM code checks to determine the boot procedure: BOOTMODE pins and MCU_BOOTMODE pins. These pins are used to configure various settings such as POST configuration, PLL configuration, and more.

However, in this application note, the focus is specifically on the following boot-related configurations, as described in the corresponding tables:

- Primary Boot Mode A
- Primary Boot Mode B
- Primary Boot Mode Configuration
- Backup Boot Mode
- Backup Boot Mode Configuration

Table 2-1. MCU_BOOTMODE Pin Mapping

9	8	7	6	5	4	3	2	1	0
POST Config		Reserved	MCU Only	Primary Boot Mode A			PLL Configuration		

Table 2-2. BOOTMODE Pin Mapping

7	6	5	4	3	2	1	0
Backup Boot Mode Config	Primary Boot Mode Configuration			Backup Boot Mode			Primary Boot Mode B

On the TDA4VM EVM, these pins are physically mapped to DIP switches as follows:

- BOOTMODE[0:7] → BOOTMODE SW8[1:8]
- MCU_BOOTMODE[2:9] → BOOTMODE SW9[1:8]

Note

Configuration of the mapping occurs directly on the Common Processor Board for TDA4 devices, with variation possible based on the specifics of a custom board design.

An eMMC memory can be used in either eMMC boot mode or MMC/SD card boot mode. This application note proposes using eMMC boot mode as the primary boot mode and MMC/SD card boot mode as the backup boot mode, both utilizing a single eMMC device.

2.1 eMMC Boot Mode Switch for Primary Boot Mode

Table 2-3, Table 2-4, and Table 2-6 provide detailed information for configuring the primary boot mode as the first boot mode attempted after reset.

To configure eMMC boot as the primary boot mode, use the following pin settings:

1. Select eMMC Boot Mode
   - Primary Boot Mode B Pin: BOOTMODE[0] = 0b1
   - Primary Boot Mode A Pins: MCU_BOOTMODE[3:5] = 0b100

Table 2-3. Primary Boot Mode Selection When MCU Only = 0

Primary Boot Mode B Pin	Primary Boot Mode A Pins			Boot Mode Selected
0	MCU 5	MCU 4	MCU 3	
1	0	0	1	eMMC

2. Configure eMMC Boot Settings, see also Table 2-4 and Table 2-5
   - BOOTMODE[4:6] = 0b000 → 1.8V I/O, maximum bus width (8 bit on Port 0), using Port 0
3. MCU-only Configuration
   - MCU_BOOTMODE[6] = 0b0

Table 2-4. Primary Boot Mode Configuration

Primary Boot Mode Configuration Pins			Primary Boot Mode B Pin	Primary Boot Mode A Pins			Primary Boot Mode
6	5	4	0	MCU 5	MCU 4	MCU 3	
Port	Bus width	Voltage	1	0	0	1	eMMC

Table 2-5. eMMC Boot Configuration Fields

BOOTMODE Pins	Field	Value	Description
6	Port	0 1	Port 0 Port 1
5	Bus Width	0 1	Maximum width by port (8 bit on Port 0) 1 bit only
4	Voltage	0 1	1.8V 3.3V

As a result, configure the DIP switch settings as follows:

- BOOTMODE SW8[1:8]: 1XXX_000X
- MCU_BOOTMODE SW9[1:8]: X100_0XXX

Note

'X' indicates a "don't care" that does not affect the configuration in this context.

2.2 MMC/SD Boot Mode Switch for Backup Boot Mode

Table 2-6 and Table 2-7 detail information for configuring a backup boot mode when the primary boot mode fails or returns.

In the case of backup boot mode, configure the MMC/SD card boot mode instead of eMMC boot mode. To achieve this, the following pin settings are required:

1. Select MMC/SD Card Boot Mode using the Backup Boot Mode Pins as defined in Table 2-6.
   - BOOTMODE[1:3] = 0b101 → MMC/SD card boot mode

Table 2-6. Backup Mode Selection When MCU Only = 0

Backup Boot Mode Pins			Backup Boot Mode Selected
3	2	1	
1	0	1	MMC/SD Card

2. Configure MMC/SD Boot Mode Settings
   - BOOTMODE[4:5] = 0b00 → 4-bit bus width, file system (FS) mode
   - BOOTMODE[7] = 0b0 → Select Port 0

Table 2-7. MMC/SD Boot Configuration Fields

BOOTMODE Pins	Field	Value	Description
7	Port	0 1	Port 0 Port 1
5	Bus Width	0 1	4 bit 1 bit
4	FS/Raw	0 1	FS mode Raw Mode

As a result, the DIP switch settings for MMC/SD card backup boot mode are as follows:

- BOOTMODE SW8[1:8]: X101_00X0
- MCU_BOOTMODE SW9[1:8]: XXXX_XXXX (not relevant for backup boot configuration)

To support both primary and backup boot modes using a single eMMC device connected to MMC controller Port 0, use the following unified switch configuration:

- BOOTMODE SW8[1:8]: 1101_0000
- MCU_BOOTMODE SW9[1:8]: 0100_0000

3 Experiment

Two sets of boot images are required to validate the proposed redundant boot approach, one for primary boot and another for backup boot. The primary components used for validation include the following binaries:

- Secondary Bootloader (SBL)
- app (bootapp)
- TI Foundational Security (TIFS)

For simplicity, QNX OS is selected as the high-level operating system (OS). To boot into the QNX console, the following binaries are required:

- lateapp1, lateapp2
- atf_optee.appimage
- ifs_qnx.appimage

3.1 Prepare eMMC Boot Images for Primary Boot Mode

The following build commands can be used to generate boot images for eMMC boot, which serves as the primary boot mode:

```
cd ${PDK_PATH}/packages/ti/build
make -sj6 sbl_emmc_boot0_img BOARD=j721e_evm CORE=mcu1_0
make -sj6 boot_app_mmcsd_qnx HLOSBOOT=qnx BOARD=j721e_evm CORE=mcu1_0
```

Once the build process is complete, the following binaries are generated:

- tiboot3.bin (SBL)
- app (bootapp)

After building the binaries, flash them to the eMMC boot1 partition at the predefined offsets.

```
mmc dev 0 1
mmc partconf 0 1 1 1
mmc bootbus 0 2 0 0
fatload mmc 1 ${loadaddr} tiboot3.bin
mmc write ${loadaddr} 0x0 0x400
fatload mmc 1 ${loadaddr} tifs.bin
mmc write ${loadaddr} 0x400 0x1000
fatload mmc 1 ${loadaddr} app
mmc write ${loadaddr} 0x1400 0x2000
```

tiboot3.bin is flashed at 0x0, tifs.bin is flashed at 0x400, and app is flashed at 0x1400.

3.2 Prepare MMC/SD Boot Images for Backup Boot Mode

To support MMC/SD boot as the backup boot mode, a separate set of boot images must be prepared. Use the appropriate build commands to generate the required binaries. After building, the following files are produced:

- tiboot3.bin (SBL)
- app (bootapp)

```
cd ${PDK_PATH}/packages/ti/build
make -sj6 sbl_emmc_uda_img BOARD=j721e_evm CORE=mcu1_0
make -sj6 boot_app_mmcsd_qnx HLOSBOOT=qnx BOARD=j721e_evm CORE=mcu1_0
```

Copy these binaries to the eMMC User Data Area (UDA) partition, as the backup boot process loads the binaries from this region.

```
mount /dev/mmcblk0p1 ./emmc_uda_partition
export DST_DIR=/home/root/emmc_uda_partition

cp tiboot3.bin ${DST_DIR}/
cp app ${DST_DIR}/
cp tifs.bin ${DST_DIR}/
sync
```

Note

Modify the tiboot3.bin used for backup boot to correctly locate the tifs.bin and application within the eMMC UDA partition, rather than from the eMMC boot partition.

3.3 eMMC Memory Layout

Table 3-1. eMMC Layout for Primary and Backup Boot Mode

eMMC Layout	Binaries	Comment
Boot area partition for primary boot (raw mode)	tiboot3.bin	Offset: 0x0
	tifs.bin	Offset: 0x400
	app	Offset: 0x1400
User data area for backup boot (file system)	tiboot3.bin	
	tifs.bin	
	app	
User data area (file system)	lateapp1	Common for primary and backup boot mode
	lateapp2	
	atf_optee.appimage	
	ifs_qnx.appimage	

Primary and backup applications (bootapps) must load the specified binaries from the eMMC User Data Area (UDA) partition.

- lateapp1
- lateapp2
- atf_optee.appimage
- ifs_qnx.appimage

This arrangement provides a consistent runtime environment regardless of which boot path is taken.

3.4 Verify Redundant Boot

Primary and backup boot versions of *tiboot3.bin* differ slightly in where bootapp binaries are located. To verify the redundant boot mechanism, unique log messages are inserted into each version of *tiboot3.bin* to distinguish between primary and backup boot modes.

The system boots from the primary boot mode on the eMMC boot partition as shown in these sample log outputs.

```
SBL Revision: 01.00.10.01 (Jun 9 2024 - 13:16:51) from eMMC boot partition #1
TIFS ver: 9.2.4--v09.02.04 (Kool Koala)Starting Sciserver..... PASSED
 BOOT_APP (Jun 9 2024 - 13:16:54) from eMMC boot partition #1 in boot_app_main.c
MCU R5F App started at 7539 usecs
Loading BootImage
:
```

To simulate a failure and trigger the redundant boot, use the following commands to corrupt the *tiboot3.bin* in the eMMC boot partition:

```
mmc dev 0 1
mmc partconf 0 1 1 1
mmc bootbus 0 2 0 0
mw ${loadaddr} 0x00 0x1000
mmc write ${loadaddr} 0x0 0x400
```

After rebooting the system, the ROM code detects the corrupted primary image and automatically switches to the backup boot mode. This behavior can be confirmed through the unique log output associated with the backup *tiboot3.bin*.

```
SBL Revision: 01.00.10.01 (Jun 9 2024 - 13:46:16) from eMMC UDA partition
TIFS ver: 9.2.4--v09.02.04 (Kool Koala)Starting Sciserver..... PASSED
BOOT_APP (Jun 9 2024 - 13:46:18) from eMMC UDA partition in boot_app_main.c
MCU R5F App started at 7193 usecs
Loading BootImage
:
```

4 Summary

This application note demonstrates a robust and practical approach to implementing redundant boot using a single eMMC device on TI TDA4x Jacinto devices. The system is configured with eMMC boot as the primary mode and MMC/SD boot as the backup mode, thereby providing boot-time reliability without requiring hardware changes or multiple memory devices.

Key benefits of this approach include:

- Single physical boot mode configuration to support both primary and backup boot paths
- Shared eMMC hardware interface for both primary and backup boot
- Seamless fallback to backup images in case of corruption or update failure

Through image preparation, proper bootloader configuration, and validation through controlled image corruption, this method proves to be a reliable design for automotive systems where resilient boot behavior is critical. This design can be adopted in production environments to improve system availability, reduce field maintenance effort, and enhance overall safety and robustness in ADAS applications.

5 References

1. Texas Instruments, *TDA4VM Product Page*
2. Texas Instruments, *J721E DRA829/TDA4VM Processors Silicon Revision 2.0, 1.1 Technical Reference Manual*
3. Texas Instruments, *J721EXCPXEVM: Common processor board for Jacinto™ 7 processors*
4. Texas Instruments, *PROCESSOR-SDK-RTOS-J721E and PROCESSOR-SDK-QNX-J721E*

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), 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, regulatory 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 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.

TI objects to and rejects any additional or different terms you may have proposed.

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