Application Note

OSPI and QSPI Flash Integration Guide for TDA4xx Devices

Karthikeyan S, Sudheer Kumar, Vinit Patel, Tarun Puvvada

ABSTRACT

This application note describes the OSPI (Octal Serial Peripheral Interface) subsystem on TDA4x devices and provides technical guidance for resolving boot, flash migration, and PHY tuning issues encountered during system bring-up. It outlines OSPI controller configuration requirements, Direct Access (DAC) and Indirect Access (INDAC) mode behavior, flash vendor compatibility considerations, and required driver updates within the Processor SDK RTOS and PDK environment.

This document is intended for software developers, bootloader (SBL) engineers, and platform integration teams working with TDA4x devices.

By consolidating validated configurations, known limitations, and structured debug practices, this document helps reduce integration effort, accelerate board bring-up, and ensure reliable OSPI boot and high-speed flash operation.

Trademarks

1 Introduction

The boot and initialization sequence of TDA4x devices relies on external QSPI and OSPI NOR/NAND flash memory for storing boot images and application binaries. The integrated OSPI controller supports single, dual, quad, and octal data protocols, along with SDR and DDR transfer modes. It provides multiple access mechanisms, including Direct Access Mode (DAC), Indirect Access Mode (INDAC), and STIG operations, each with defined operational characteristics and limitations.

In QSPI boot mode, the ROM code initializes the OSPI peripheral in Direct Access Mode (DAC) and reads the boot image from the selected QSPI flash device. The boot image is copied into on-chip RAM, after which execution is transferred to the loaded image.

Misconfiguration of addressing modes, device size parameters, PHY tuning, clock settings, or flash-specific command sequences can result in boot failures, restricted address access, data corruption, or unstable high-speed operation.

This application note consolidates validated EVM flash configurations, OSPI access mode behavior, clock frequency constraints, PHY tuning requirements, and flash migration considerations. It also documents commonly observed failure scenarios during board bring-up and SBL integration, along with recommended configuration updates and structured debug methodologies.

2 OSPI/QSPI Flash Devices Used on TDA4x EVMs

The flash devices listed in Table 1 are used on TI evaluation modules (EVMs) and have been validated internally with the Processor SDK RTOS.

Table 2-1. Supported Flash Devices on TDA4x EVMs

SoC	Flash Type	Vendor	Device	Density	Interface
J721E (TDA4VM)	OSPI NOR	Micron	MT35XU512ABA1G12	512Mb	OSPI0
J721E (TDA4VM)	QSPI	Micron	MT25QU512ABB	512Mb	OSPI1
J7200(DRA821)	OSPI NOR	Cypress	S28HS512TGABHM01	512Mb	OSPI0
J721S2 (TDA4VL)	OSPI NOR	Cypress	S28HS512TGABHM01	512Mb	OSPI0
J721S2 (TDA4VL)	OSPI NAND	Winbond	W35N01JWTBAG	1Gb	OSPI0
J784S4 (TDA4VH)	OSPI NOR	Cypress	S28HS512TGABHM01	512Mb	OSPI0
J784S4 (TDA4VH)	OSPI NAND	Winbond	W35N01JWTBAG	1Gb	OSPI0
J722S(TDA4AEN)	OSPI NOR	Cypress	S28HS512TGABHM01	512Mb	OSPI0
J722S(TDA4AEN)	OSPI NAND	Winbond	W35N01JWTBAG	512Mb	OSPI0

Note

The devices listed above reflect flash components used on TI EVMs. Compatibility of other OSPI flash devices requires verification of command set, timing parameters, addressing mode, and protocol support within the OSPI driver framework.

3 OSPI Flash Selection Considerations for Custom Boards

Selecting an OSPI flash device requires verification that the device command set, supported protocols (STR/DTR), opcode definitions, dummy-cycle requirements, addressing modes (3-byte or 4-byte), and timing specifications are fully compatible with both the SoC OSPI controller and the OSPI driver implementation.

Flash devices from different vendors may vary in JEDEC identification values, read/write command sets, DTR behavior, and PHY compatibility. These differences must be reflected in the OSPI driver configuration.

Guidelines for selecting supported OSPI flash devices for the Jacinto 7 family are available in the Texas Instruments E2E Support Forums FAQ titled:

1. TI E2E FAQ, Is There a Guide to Choosing the Right OSPI Flash Parts That Are Supported on Jacinto

4 OSPI Access Modes

The OSPI controller supports multiple access modes for communication with external flash devices.

4.1 Direct Access (DAC)

Direct Access (DAC) mode provides a memory-mapped interface to the external flash. In this mode, read accesses are automatically triggered when the processor accesses the mapped address space.

This mode is commonly used for:

- High-performance read operations
- Execute-In-Place (XIP)
- Code execution directly from external flash

In DAC mode, the flash memory appears in the system memory map.

4.2 Indirect Access (INDAC)

Indirect Access (INDAC) mode transfers data between the OSPI controller and flash using an internal SRAM buffer.

In this mode:

- The flash address is explicitly programmed through controller registers.
- Data is transferred via the OSPI internal SRAM
- Transfers are initiated by software.

INDAC mode is typically used for flash page program (write) operations and bulk data transfers. Erase commands are commonly issued using STIG mode.

4.3 Software Triggered Instruction (STIG)

Software Triggered Instruction (STIG) mode is used to send specific command sequences directly to the flash device without using memory-mapped access or large data transfers.

This mode is typically used for:

- Flash register access
- Status polling
- Device configuration commands
- Erase command issuance

4.4 Execute-In-Place (XIP)

Execute-In-Place (XIP) allows code execution directly from serial flash memory without copying the code to RAM.

In the Processor SDK RTOS OSPI driver implementation:

- XIP is supported only in read mode.
- XIP operates in conjunction with DAC mode.
- The FlsXipEnable parameter controls enabling or disabling XIP functionality.

4.5 Recommended OSPI Access Modes

Flash Read

Use DAC mode with PHY enabled, optionally combined with pipeline and DMA, for optimal performance. XIP can be enabled if code execution from flash is required.

Flash Write

Use INDAC mode without PHY (recommended). Write throughput is primarily limited by the flash device program time.

Flash Erase

Use STIG mode without PHY.

Flash Register Access

Use STIG mode without PHY.

5 OSPI Clock Frequency Support

The OSPI clock frequency parameter defines the operating clock for the OSPI peripheral. The following clock frequencies are supported on TDA4x platforms.

133 MHz (133333333U) - Fully supported across all OSPI operating modes, including Direct Access (DAC) and Indirect Access (INDAC), for both read and write operations.

166 MHz (166666666U) - Fully supported in INDAC mode for read and write operations. In DAC mode only read operations are supported at this frequency.

200 MHz (200000000U) - Operation at this frequency has not been validated by TI.

6 OSPI Flashing and Boot Support

Application software images can be programmed into non-removable OSPI flash devices on TDA4x platforms using TI UniFlash. UniFlash supports programming OSPI flash devices during development and bring-up. OSPI-specific flashing steps, supported configurations, and boot requirements are documented in the Processor RTOS SDK User Guide

For a comprehensive description of the supported flashing techniques on TDA4x devices, refer to the following application note SPRACY5 – Flashing Techniques for TDA4 Devices.

Note

TDA4x devices provide multiple OSPI controller instances (OSPI0 and OSPI1). However, only one OSPI instance should be actively used for flash access at a time, depending on the board configuration.

7 OSPI Migration and Required Updates

Migrating from the default EVM-supported OSPI NOR flash device to an alternative flash device (for example, from Cypress to Micron) can lead to functional issues if vendor-specific differences are not properly handled. Flash devices from different vendors may differ in JEDEC identification values, command opcodes, dummy cycle requirements, addressing modes, and DTR behavior.

If the OSPI driver configuration is not updated to reflect these differences, the system may encounter read/write failures, data corruption, or SBL boot failures during early boot. When migrating to a new OSPI NOR flash device, the OSPI flash driver must be updated with the correct device-specific parameters, including:

- Manufacturer ID and Device ID
- Read/write and erase opcodes
- Dummy cycle configuration
- Addressing mode (3-byte vs 4-byte)
- DTR enable/disable settings

TI provides a dedicated E2E FAQ that documents common flash migration scenarios and describes the required steps to add support for new OSPI flash devices in the Processor SDK RTOS

Migrating from the default EVM-supported OSPI NOR flash device to an alternative device (for example, from Cypress to Micron) requires careful validation of vendor-specific parameters. Differences across flash vendors may include:

- JEDEC manufacturer and device identification values
- Read, write, and erase command opcodes
- Dummy cycle requirements
- Addressing mode (3-byte vs 4-byte)
- DTR (Double Transfer Rate) support and enable sequence
- Sector size and erase block size

If the OSPI driver configuration is not updated to reflect these device-specific parameters, the system may experience:

- Read or write failures
- Data corruption
- PHY tuning failures
- SBL boot failure during early initialization

When adding support for a new OSPI NOR flash device, the flash configuration table in the PDK OSPI driver must be updated with the correct device parameters. This typically involves modifying the flash device configuration structures located under:

PDK Path:

<pdk>packages/ti/board/src/flash/nor/ospi

All command opcodes, dummy cycles, addressing mode selection, and DTR configuration must match the flash vendor datasheet.

Additional Resources:

The following E2E FAQs provide step-by-step guidance for adding support for new OSPI flash devices in TI SDKs:

1. TI E2E FAQ, How to add support for a new flash device in RTOS SDK(TDA4VM / Jacinto family)

2. TI E2E FAQ, Guidelines for configuring custom OSPI flash using SysConfig in MCU+ SDK(J722S EVM)

8 Verifying OSPI Configuration Using SDK Driver Examples

To validate the OSPI configuration and isolate configuration-related issues, it is recommended to first verify OSPI functionality using the OSPI driver examples provided in the Processor SDK RTOS.

The OSPI_Flash_Dma_TestApp_freertos example is located under:

PDK Path:

<pdk>/packages/ti/drv/spi/test/ospi_flash/src

This example can be built and executed using Code Composer Studio (CCS). Refer to the Processor SDK RTOS User Guide for instructions on building the PDK and configuring CCS for development.

Running the example under CCS enables:

- Single-step debugging
- Inspection of OSPI configuration parameters
- Tracing of OSPI driver API calls

This approach is effective for identifying incorrect configuration settings, improper clock configuration, unsupported flash parameters, or incorrect driver API usage that may contribute to observed failures.

Guidance for debugging the Secondary Bootloader (SBL) using CCS is available in the Texas Instruments E2E Support Forums FAQ titled Debugging SBL Boot in RTOS SDK.

If the OSPI driver example executes successfully without errors, this indicates that the following platform-level configurations are correct:

- OSPI hardware connectivity
- Pin multiplexing configuration
- Clock configuration
- Baseline OSPI driver initialization

Successful execution of the driver example provides a known-good reference point before integrating OSPI functionality into custom applications or bootloader configurations.

9 Common OSPI Issues on TDA4x Devices

This section summarizes commonly observed OSPI-related issues encountered during board bring-up, flash migration, and bootloader integration on TDA4x platforms. Each issue includes the observed behavior, root cause, and recommended resolution steps.

9.1 DAC Addressing Limitation

Problem Description

When using large OSPI flash devices (for example, 128 MB or 256 MB), the system may fail to access memory beyond the 64-MB boundary when operating in Direct Access (DAC) mode during boot or runtime.

Observed Symptoms

- Bootloader fails to load images stored above address 0x04000000
- Flash reads return incorrect data beyond the 64-MB offset
- Boot process fails when application images are located in higher flash regions

Root Causes

Several configuration issues can lead to this behavior.

Cause 1: 3-Byte Addressing Mode Enabled

The OSPI NOR driver may operate in 3-byte addressing mode, which limits the accessible address space to 64 MB.

Cause 2: OSPI Device Size Configuration

The OSPI controller device size configuration may be set to 64 MB in the default SDK configuration.

Cause 3: Boot Configuration Register Limit

Boot configuration registers may restrict OSPI boot access to 64 MB by default.

Cause 4: FSS Memory Region Mapping Limitation

The default FSS configuration maps only Region1 (~64 MB) into the R5F address space. Larger flash sizes require additional mapping using the RAT (Region Address Translation) module.

Resolution

Verify and update the following configurations

Enable 4-Byte Addressing Mode

Update the NOR driver to use 4-byte addressing mode.

PDK Path:

<pdk>/packages/ti/board/src/flash/nor/ospi/nor_xspi.c

Update address byte configuration from 3 bytes to 4 bytes.

Update OSPI Device Size Configuration

Update the CS0 device size during SBL initialization.

PDK Path:

<pdk>/packages/ti/csl/src/ip/ospi/V0/priv/csl_ospi.c

API:

CSL_ospiSetDevSize()

Example configuration:

/* For accessing 1h = size of 1Gb */

```
regVal |= (0x1 << CSL_OSPI_FLASH_CFG_DEV_SIZE_CONFIG_REG_MEM_SIZE_ON_CS0_FLD_SHIFT);
CSL_REG32_WR(&pRegs->DEV_SIZE_CONFIG_REG, regVal);
/* For accessing 1h = size of 2Gb */
regVal |= (0x2 << CSL_OSPI_FLASH_CFG_DEV_SIZE_CONFIG_REG_MEM_SIZE_ON_CS0_FLD_SHIFT);
CSL_REG32_WR(&pRegs->DEV_SIZE_CONFIG_REG, regVal);
```

Update Boot Configuration Register

Configure S0_BOOT_SIZE in the boot configuration register:

CTRLMMR_MCU_FSS_CTRL (0x40F040A0)

Set the value to 0x1 to enable access beyond 64 MB.

Configure RAT Mapping for Large Flash

When accessing large flash devices, configure the RAT (Region Address Translation) module to map extended flash regions into the R5F address space.

Example configuration:

```
*(volatile uint32_t *)(0x40F90024h + (reg_id * 0x10)) = 0xE0000000u;
*(volatile uint32_t *)(0x40F90028h + (reg_id * 0x10)) = 0x00000000u;
*(volatile uint32_t *)(0x40F9002Ch + (reg_id * 0x10)) = 0x00000005u;
*(volatile uint32_t *)(0x40F90020h + (reg_id * 0x10)) = (1 << 31) | (0x1C << 0);
```

Add this configuration during early boot (for example, in main()) to enable access to the full 256-MB OSPI flash space.

9.2 PHY Tuning Issue

Problem Description

At higher OSPI clock frequencies, particularly in Octal DDR mode (≥133 MHz), read operations may become unstable if PHY tuning is not enabled. This can result in intermittent read failures, data corruption, or SBL boot hangs during early system initialization.

Root Cause

High-speed OSPI operation requires precise data sampling alignment between the controller and the external flash device. Board-level routing delays, signal integrity variations, and timing skew can cause sampling errors when operating without PHY calibration.

The OSPI PHY tuning algorithm compensates for these timing variations by calibrating read sampling delay using a predefined calibration pattern stored in flash memory.

Required Configuration

To enable PHY tuning:

- Program the PHY calibration pattern at the beginning of the last erase sector of the OSPI flash.
- Enable PHY mode in the OSPI configuration (DAC mode only).
- Configure the calibration pattern memory region as non-cacheable.
- Ensure PHY is used only for read operations.

Operational Constraints

- PHY mode is supported only in Direct Access Mode (DAC).
- PHY is supported only for read operations.
- When PHY is enabled, the internal clock divider is bypassed and RCLK is directly driven to the flash device.

Clock Divider Requirements Without PHY:

- SDR mode requires baudRateDiv ≥ 4
- DDR mode requires baudRateDiv ≥ 8

Maximum Supported Frequency

The maximum operating frequency of the OSPI interface depends on whether PHY tuning is enabled. PHY tuning compensates for board-level signal delays and timing variations between the OSPI controller and the external flash device, allowing reliable operation at higher clock frequencies.

Operation Without PHY Tuning:

When PHY tuning is disabled, the supported operating frequencies are limited by interface timing constraints:

- DDR mode: up to 25 MHz
- SDR mode: up to 50 MHz

Operation With PHY Tuning

When PHY tuning is enabled, the controller can operate at higher clock frequencies.

- Up to 166 MHz, depending on the flash device and access mode.

Operating beyond validated frequencies or without proper PHY tuning may result in unstable reads, data corruption, or boot failures

Recommended PHY Tuning Pattern Locations

For systems using the same flash configuration as the TI evaluation modules (EVMs), the PHY tuning pattern is programmed at the following addresses:

Table 9.2-1. PHY Tuning Pattern Locations

SOC	Type	Address
J721S2 / J784S4	Cypres OSPI NOR	0x3FC0000
J721E	Micron OSPI NOR	0x3FE0000
J721S2 / J784S4	Winbond OSPI NAND	0x7FC0000

The tuning pattern must be placed at the beginning of the final erase block (last sector) of the flash device. Failure to place the PHY tuning pattern at the correct location may result in PHY calibration failure.

Validation Recommendation

It is recommended to validate PHY configuration using the OSPI DMA test application provided in the Processor SDK RTOS. Stable high-speed read operation across multiple power cycles confirms correct PHY configuration.

10 QSPI Boot Support Considerations

On TDA4x devices, the secondary OSPI interface (OSPI1) is commonly used to connect Quad-SPI (QSPI) flash devices on TI evaluation modules (EVMs) and custom boards. Although OSPI1 is an OSPI-capable controller, it may operate in QSPI mode depending on board-level pin routing and the type of flash device connected.

The Processor SDK RTOS does not provide default Secondary Bootloader (SBL) support for booting from QSPI flash devices. As a result, systems that use QSPI flash for boot storage require additional driver configuration.

To enable QSPI flash support, the PDK OSPI driver must be updated to match the characteristics of the selected flash device. This typically involves updating the flash configuration table, command opcodes, and initialization parameters to align with the device datasheet.

Proper driver configuration ensures correct flash initialization and reliable boot operation when using QSPI devices.

11 Summary

This document provides consolidated guidance for configuration and debug of the OSPI subsystem on TDA4x devices using Processor SDK RTOS. It documents validated flash configurations, supported clocking schemes, access mode constraints, PHY requirements, and large address space considerations.

Common bring-up issues are analyzed with root-cause explanations and corrective actions, enabling deterministic boot behavior and reliable high-speed OSPI operation across supported devices and SDK releases.

12 References

1. TI E2E FAQ, Shmoo Plot Generation for OSPI PHY Tuning
2. Texas Instruments, OSPI Performance Numbers (Jacinto 7 Datasheet)
3. Texas Instruments, OSPI Driver Configuration to support QSPI flash

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, 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 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, TI’s General Quality Guidelines, 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. Unless TI explicitly designates a product as custom or customer-specified, TI products are standard, catalog, general purpose devices.

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

Last updated 10/2025