Application Note

xSPI Custom Flash Debug Guide for MCU+SDK

Vaibhav Kumar, Nikhil Jain

ABSTRACT

This application note provides guidance on bringing up a custom flash part on TI Sitara™ processors using MCU+SDK. The document addresses the challenges faced by developers working on custom boards that use flash part different from those available on TI evaluation modules (EVMs).

The document outlines the steps required to integrate, configure and validate a new flash part to verify reliable operation with the TI software ecosystem. The target audience includes engineers and developers who are designing custom hardware and need to enable flash communication through an SPI/QSPI/OSPI interface.

Following this application note helps reduce bring-up time and minimizes common integration issues, enabling faster development and debugging of custom flash configurations.

Table of Contents

1 Introduction......2
2 Terminology......2
3 Understanding Boot Mode and Flash Compatibility......2
4 Flash Integration and SysConfig Setup......3
4.1 FLASH Parameters......4
4.2 OSPI Parameters......8
5 Common Bring-up Issues and Debugging......9
5.1 Boot Failure......9
5.2 Known Errata......9
5.3 Flash Initialization Failure......9
5.4 Flash Read Failure......10
5.5 Flash Program Failure......10
6 Checklist for Requesting OSPI and FLASH Support......11
7 Summary......11
8 References......11

Trademarks

1 Introduction

In embedded systems, external flash memory is widely used to store bootloaders and application code. Sitara processors such as AM64x, AM62x, AM62P and AM62Dx support multiple serial interfaces such as OSPI and QSPI for connecting external NOR/NAND flash devices. During custom board bring-up using MCU+SDK, users often face challenges while interfacing a new flash device, such as configuring SysConfig parameters, reading device IDs, or enabling PHY mode.

This application note explains how to configure flash parameters using SysConfig, interpret common error logs, and resolve issues during flash identification and PHY enablement.

2 Terminology

QSPI: Quad Serial Peripheral Interface

OSPI: Octal Serial Peripheral Interface

xSPI: Extended Serial Peripheral Interface

NVCR: Non-Volatile Configuration Registers

VCR: Volatile Configuration Registers

DDR: Dual Data Rate

SDR: Single Data Rate

QE: Quad Enable Bit

DAC: Direct Access Controller

3 Understanding Boot Mode and Flash Compatibility

The various boot modes, for Serial NOR/NAND flash, supported by ROM can be found in the technical reference manual here. Each boot mode describes the protocol ROM uses to issue read commands. The protocol varies for each boot mode. For example, for SPI (protocol 1S-1S-1S) boot mode, the ROM issues the read command 0x03(8 bits) followed by a 3-byte (24 bits) address and no dummy cycles.

A flash is considered to be boot capable, if as per the boot mode's description. The flash supports the following parameters:

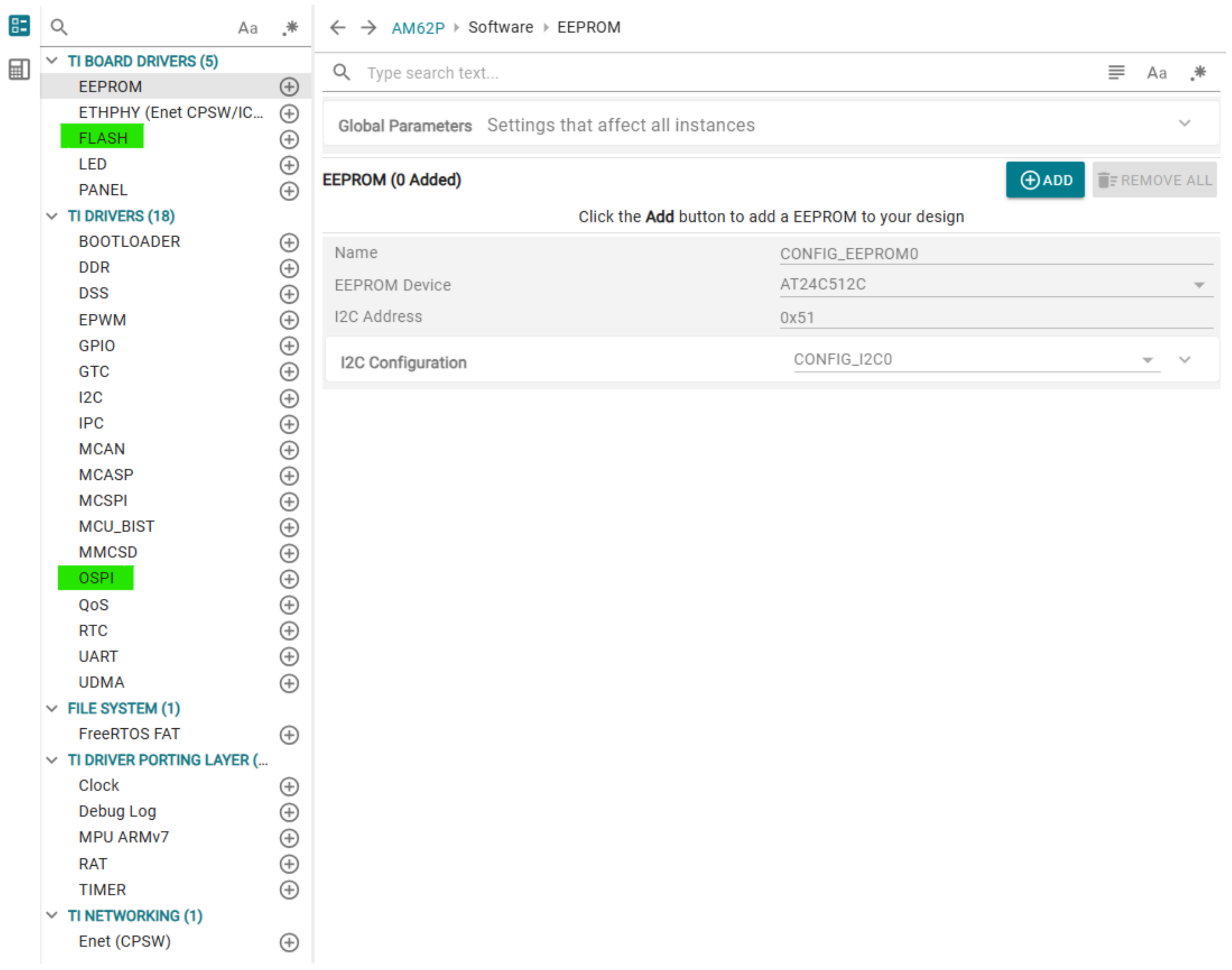
1. The protocol.
2. The read command.
3. The number of bytes (3-byte/4-byte) for read command and address.
4. The number of dummy cycles for reading.

For a custom flash, to get these values, see the flash specific datasheet. In the flash datasheet, these values are contained under a section named as *Command Set*, *Command Table* , *Instruction Set*, *Instruction Table*, *Transaction Set*, *Transaction Table*.

4 Flash Integration and SysConfig Setup

SysConfig is a configuration tool designed to simplify hardware and software configuration challenges to accelerate software development.

In MCU+SDK, for a custom flash, the section which needs modification is shown in Figure 4-1.



Name CONFIG_EEPROM0
EEPROM Device AT24C512C
I2C Address 0x51
I2C Configuration CONFIG_I2C0

Figure 4-1. SysConfig GUI Showing FLASH and OSPI Sections

To obtain the custom flash configuration values, follow the guide here. This guide currently works for Serial NOR flash. For Serial NAND flash, the configuration values must be obtained from the datasheet of the flash.

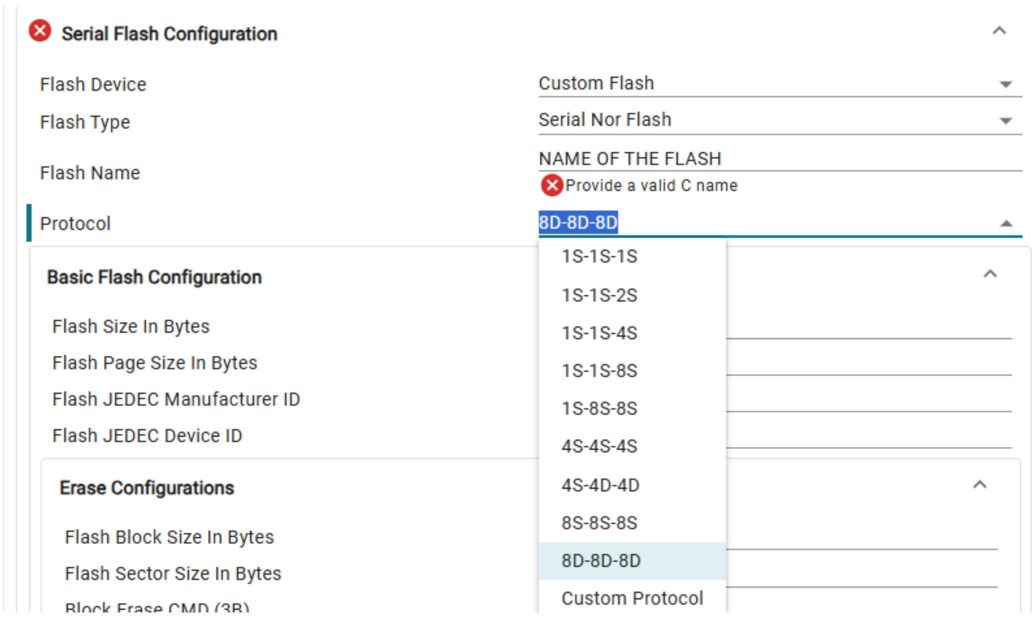
Upon obtaining the values, the FLASH section can be filled up with the obtained values. These values can be filled up manually or uploaded in a json format. Verify the values obtained by running the OSPI Flash Diagnostics example with the values mentioned in the flash datasheet. The following sections show the various parameters in FLASH and OSPI sections.

4.1 FLASH Parameters

For each of the parameters, click on the *?* to learn about the description.

Description for Figure 4-2:

1. **Flash Device:** Choose custom flash if the flash is not one of the default flashes on the TI EVM.
2. **Flash Name:** Write the name of the flash in use, for example, *W25Q512JV*.
3. **Protocol:** Choose one of the listed protocols which the flash also supports. This configures the number of lines to be used for Command, Address and Data respectively. It also specifies the data rate, that is DDR (Dual Data Rate) or SDR (Single Data Rate). For example, 4S-4D-4D, means 4 lines is used for Command, Address and Data respectively. For Command the data is latched on either of rising/falling edge, but for Address and Data, the data is latched on both the rising and falling edges.


Serial Flash Configuration

Flash Device: Custom Flash
Flash Type: Serial Nor Flash
Flash Name: NAME OF THE FLASH
Provide a valid C name
Protocol: 8D-8D-8D

1S-1S-1S
1S-1S-2S
1S-1S-4S
1S-1S-8S
1S-8S-8S
4S-4S-4S
4S-4D-4D
8S-8S-8S
8D-8D-8D
Custom Protocol

Basic Flash Configuration

Flash Size In Bytes
Flash Page Size In Bytes
Flash JEDEC Manufacturer ID
Flash JEDEC Device ID

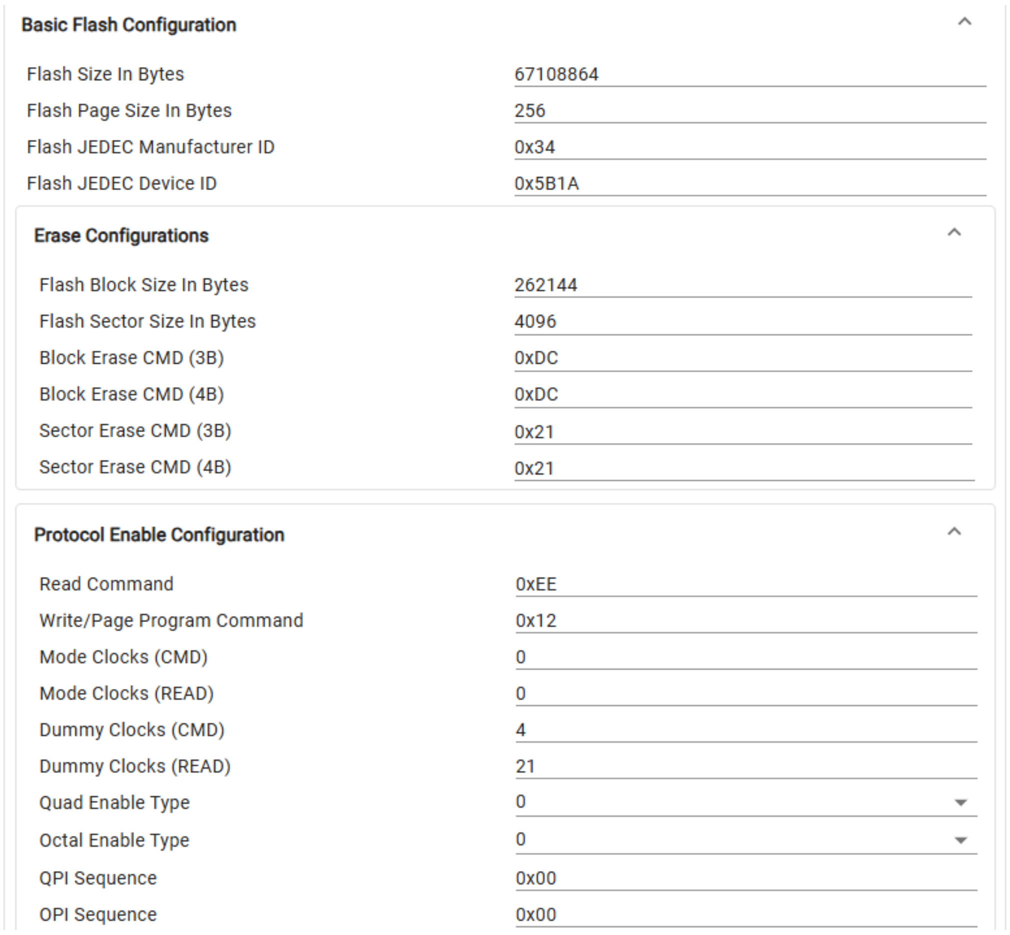
Erase Configurations

Flash Block Size In Bytes
Flash Sector Size In Bytes
Block Erase CMD (3B)

Figure 4-2. Basic Flash Configuration

Description for Figure 4-3:

1. **Flash JEDEC Manufacture ID:** This value is the same across a specific manufacturer.
2. **Flash JEDEC Device ID:** This value varies across the same manufacturer's flash variant. For example, a Winbond flash has different device ID for flashes which operate in 1.8V and 3.6V respectively. There are other parameters as well on which the flash part varies and can be found under the flash datasheet.
3. **Dummy Clocks (CMD)** and **Dummy Clocks (READ):** Dummy Clocks is used to synchronize flash reads. These values can be obtained from the flash datasheet as well. Look out for sections named as *Dummy Cycles*/*Dummy Clocks*. Please note, this field specifies the value which is set at OSPI Controller's end. To set dummy cycle/clock value at flash's end, the flash configuration register needs to be written to. Please refer to the **Dummy Cycle Configuration** field.
4. **Quad Enable Type:** Some QSPI flashes have a QE (Quad Enable) bit. This bit needs to be enabled, for protocols like 1S-1S-4S, 4S-4S-4S and 4S-4D-4D.
5. **QPI Sequence** and **OPI Sequence:** To enable 4-4-4 mode and 8-8-8 mode respectively, these fields are used.
6. For Flash's Block Size and Sector Size, refer to the flash datasheet. The commands for the same can be obtained from the diagnostics log, but always good to verify from the datasheet.
7. SysConfig allows users to set Sector Size and Block Size, in case the flash supports different sector/block sizes, user can specify certain sector/block size based on the application.
8. The (3B) and (4B) refers to 3 byte and 4 byte addressing mode of the flash. Basically, this specifies how many address bytes must be sent. For a Flash which is less than 16 MB, only 3 bytes addressing is enough to cater to the entire flash region.


Basic Flash Configuration

Flash Size In Bytes	67108864
Flash Page Size In Bytes	256
Flash JEDEC Manufacturer ID	0x34
Flash JEDEC Device ID	0x5B1A

Erase Configurations

Flash Block Size In Bytes	262144
Flash Sector Size In Bytes	4096
Block Erase CMD (3B)	0xDC
Block Erase CMD (4B)	0xDC
Sector Erase CMD (3B)	0x21
Sector Erase CMD (4B)	0x21

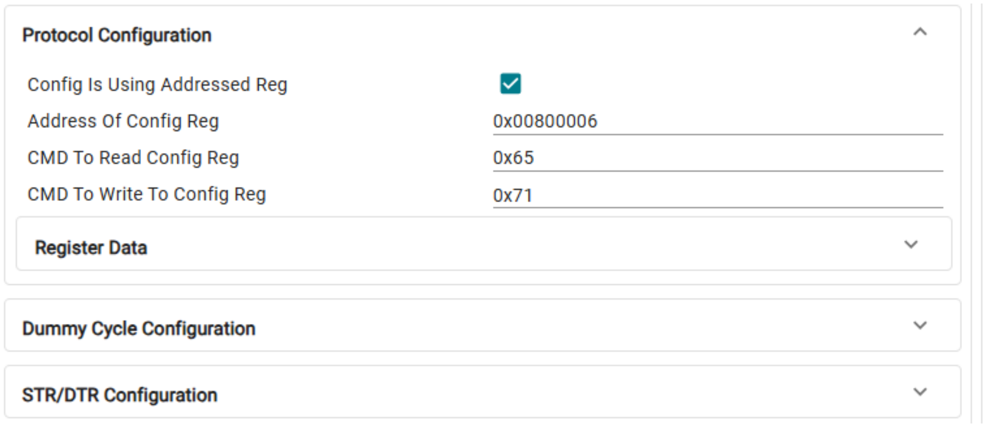
Protocol Enable Configuration

Read Command	0xEE
Write/Page Program Command	0x12
Mode Clocks (CMD)	0
Mode Clocks (READ)	0
Dummy Clocks (CMD)	4
Dummy Clocks (READ)	21
Quad Enable Type	0
Octal Enable Type	0
QPI Sequence	0x00
OPI Sequence	0x00

Figure 4-3. Basic Flash Erase and Protocol Configuration

Description for Figure 4-4:

1. **Protocol Configuration:** Protocol usually specifies the number of lines to be used for command, address and data. For the flash to operate in a specific protocol, this section needs to be filled up.
2. **Dummy Cycle Configuration:** For the flash to operate in a certain protocol, the flash configuration register needs dummy cycles to be configured. This is the flash's side setting and the OSPI Controller's setting is defined under the field ***Dummy Clocks (READ)*** .
3. **STR/DTR Configuration:** This can be set to make the flash operate in SDR/DDR mode.


Protocol Configuration

Config Is Using Addressed Reg	✓
Address Of Config Reg	0x00800006
CMD To Read Config Reg	0x65
CMD To Write To Config Reg	0x71

Register Data

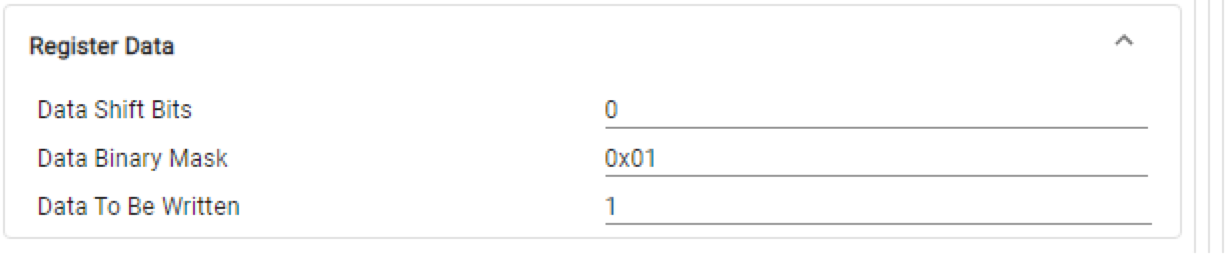
Dummy Cycle Configuration

STR/DTR Configuration

Figure 4-4. Protocol, Dummy Cycle and STR/DTR Configuration

Description for Figure 4-5:

Let's understand this carefully. Suppose the value of a configuration register is 10111101b. If the bits[5:2], needs to be updated with 1001b then, the values in Data Shift Bits, Data Binary Mask and Data To Be Written will be 2, 0x3C and 9 respectively. On the original value 10111101b, mask operation is carried, with the logic (value & ~(Data Binary Mask)). This results in the value being 10000001b. Now the shift operation will be performed as (value | (Data To Be Written << Data Shift Bits)). The final value becomes 10100101b and hence gets written to the configuration register.



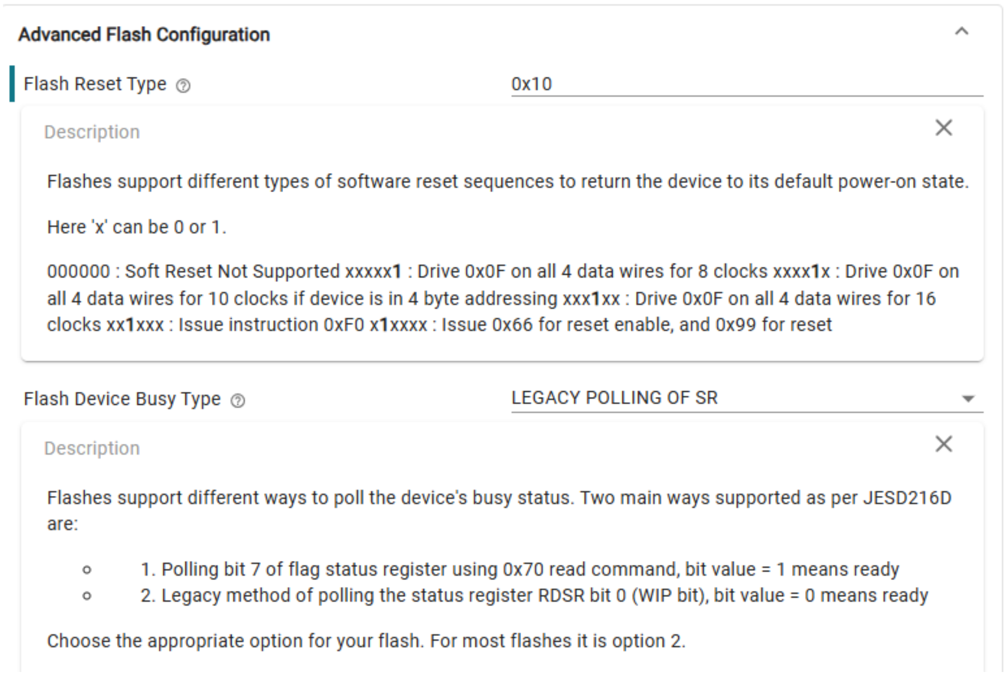
Register Data

Data Shift Bits	0
Data Binary Mask	0x01
Data To Be Written	1

Figure 4-5. Register Data Field Configuration

Description for Figure 4-6:

Fields are self-explanatory. Flash datasheet can be referred to set the correct values.



Advanced Flash Configuration

Flash Reset Type: 0x10

Description

Flashes support different types of software reset sequences to return the device to its default power-on state.

Here 'x' can be 0 or 1.

000000 : Soft Reset Not Supported xxxxx**1** : Drive 0x0F on all 4 data wires for 8 clocks xxxx**1**x : Drive 0x0F on all 4 data wires for 10 clocks if device is in 4 byte addressing xxx**1**xx : Drive 0x0F on all 4 data wires for 16 clocks xx**1**xxx : Issue instruction 0xF0 x**1**xxxx : Issue 0x66 for reset enable, and 0x99 for reset

Flash Device Busy Type: LEGACY POLLING OF SR

Description

Flashes support different ways to poll the device's busy status. Two main ways supported as per JESD216D are:

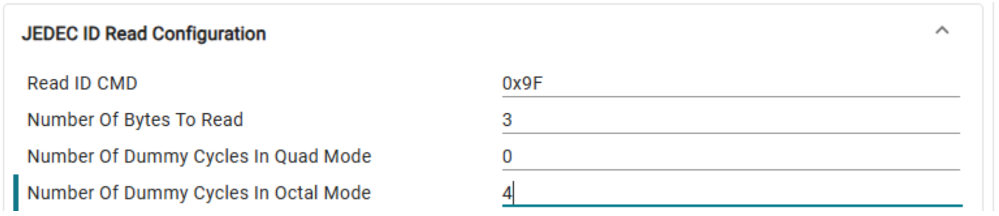
- 1. Polling bit 7 of flag status register using 0x70 read command, bit value = 1 means ready
- 2. Legacy method of polling the status register RDSR bit 0 (WIP bit), bit value = 0 means ready

Choose the appropriate option for your flash. For most flashes it is option 2.

Figure 4-6. Advanced Flash Configuration

Description for Figure 4-7:

All of these values can be found in the Flash Datasheet. Refer to the Read Device ID/Read Manufacture ID Transaction Table from the flash datasheet.



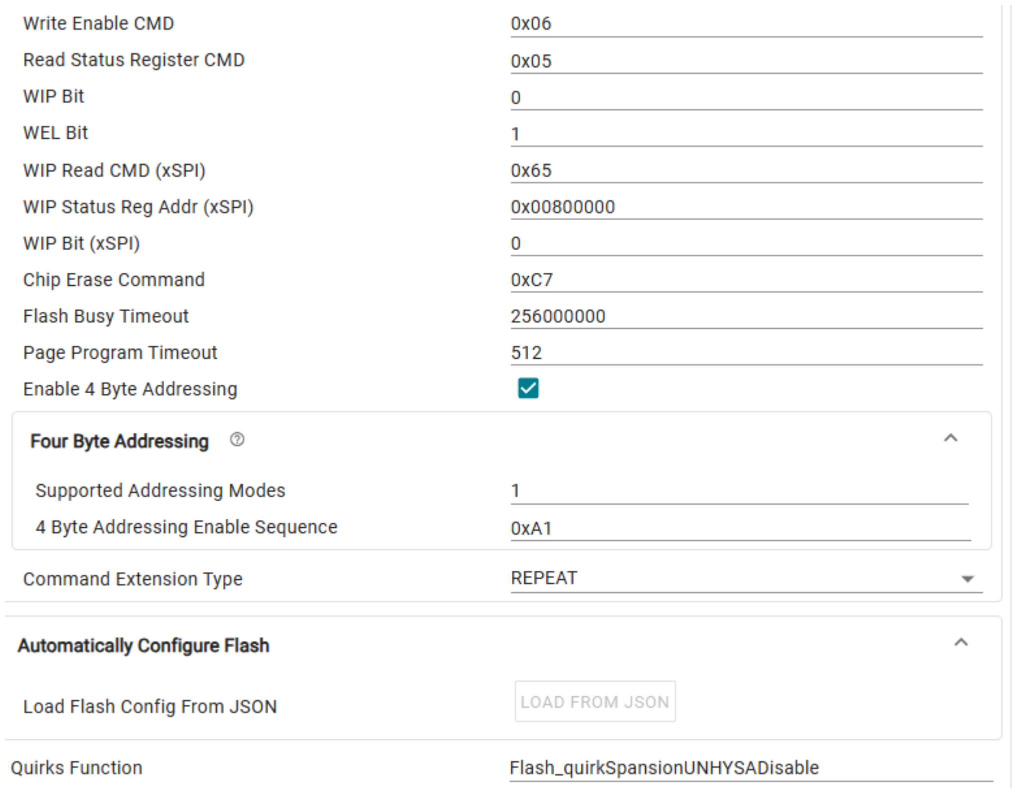
JEDEC ID Read Configuration

Read ID CMD	0x9F
Number Of Bytes To Read	3
Number Of Dummy Cycles In Quad Mode	0
Number Of Dummy Cycles In Octal Mode	4

Figure 4-7. JEDEC ID Read Configuration

Description for Figure 4-8:

1. **WIP Bit** and **WEL Bit:** These bits refer to Write in progress and Write Enable Latch bits in the Flash Status Register.
2. **Four Byte Addressing:** As explained earlier, this is needed if the flash is of size > 16MB, and it supports specific commands for (4B) addressing.
3. **Command Extension Type:** Assume the command to be sent is 0x9F. So, in case of:
   a. NONE, 0x9F is sent.
   b. REPEAT: 0x9F, 0x9F is sent.
   c. INVERSE: 0x9F, 0x60 (Inverse of 0x9F) is sent.
4. Quirks Function is used to define a custom API, which performs additional changes to the configuration register of the flash. Currently, this is defined as *Flash_quirkSpansionUNHYSADisable*. This function basically disables the Hybrid mode of the flash(S28HS512T) present on the TI EVM.



Write Enable CMD	0x06
Read Status Register CMD	0x05
WIP Bit	0
WEL Bit	1
WIP Read CMD (xSPI)	0x65
WIP Status Reg Addr (xSPI)	0x00800000
WIP Bit (xSPI)	0
Chip Erase Command	0xC7
Flash Busy Timeout	256000000
Page Program Timeout	512
Enable 4 Byte Addressing	✓
Four Byte Addressing	
Supported Addressing Modes	1
4 Byte Addressing Enable Sequence	0xA1
Command Extension Type	REPEAT
Automatically Configure Flash	
Load Flash Config From JSON	LOAD FROM JSON
Quirks Function	Flash_quirkSpansionUNHYSADisable

Figure 4-8. Advanced Flash Configuration (continued)

4.1.1 Recommended Approach

The MCU+ SDK provides default examples that can be found at the following path: MCU_PLUS_SDK_INSTALL_PATH/examples/drivers/ospi. One such example is OSPI Flash IO. More details about this example can be found here.

Running this example successfully makes sure that all basic flash operations – erase, write and read - are functioning correctly. This also confirms that the protocol configuration between the controller and the flash device is set up properly and that the flash parameters configured in SysConfig are working as expected.

Once a particular combination of flash parameters and protocol has been verified using the OSPI Flash IO example, these validated settings can then be reused in other applications such as UART Uniflash, bootloaders (like SBL OSPI), and similar projects.

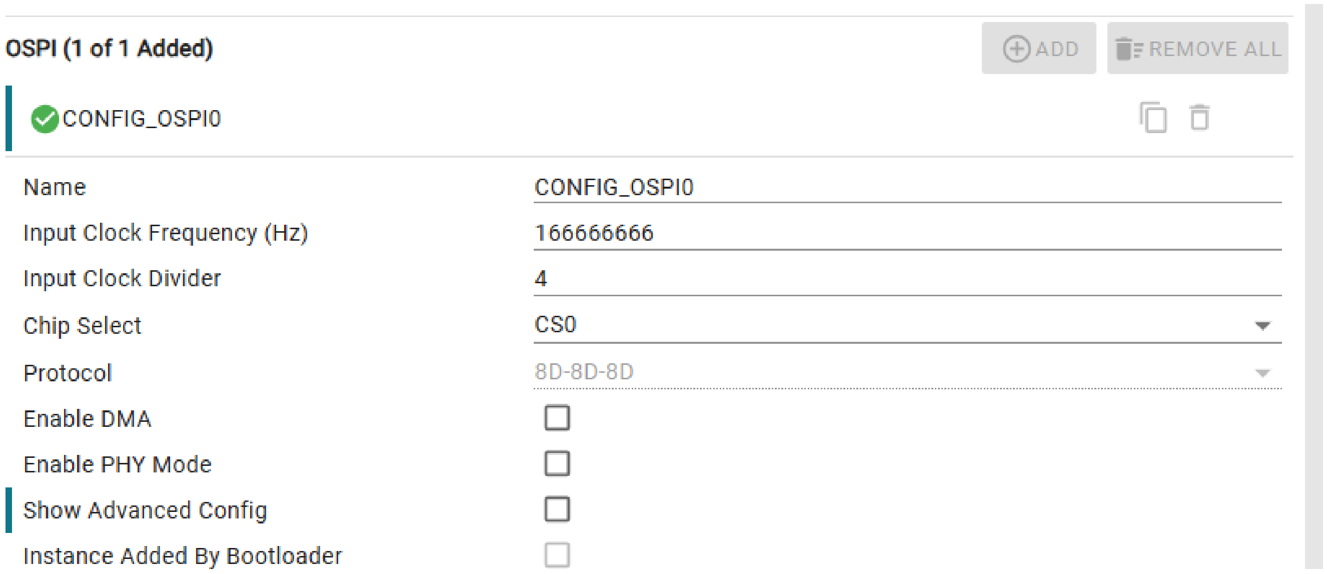
In summary, the OSPI Flash IO example serves as a reliable reference point for testing and configuring flash parameters, which can then be applied across multiple applications.

4.2 OSPI Parameters

For each of the parameters, click on the *?* to learn about the description.

Description for Figure 4-9:

1. **Input Clock Frequency (Hz):** This is the reference clock frequency (RCLK).
2. **Input Clock Divider:** This is the divider which divides the reference clock frequency. OSPI Output Clock = Input Clock Frequency / Input Clock Divider.
3. **Chip Select:** By default, CS0 is used by the OSPI Controller.
4. **Protocol:** This field gets populated from the FLASH section's ***Protocol***.
5. **Enable DMA:** DMA transfers happen for transaction size greater than 1KB.
6. **Enable PHY Mode:** Enables PHY mode (for clocks greater than 50MHz). When unchecked, TAP mode is used.



Name	CONFIG_OSPI0
Input Clock Frequency (Hz)	166666666
Input Clock Divider	4
Chip Select	CS0
Protocol	8D-8D-8D
Enable DMA	☐
Enable PHY Mode	☐
Show Advanced Config	☐
Instance Added By Bootloader	☐

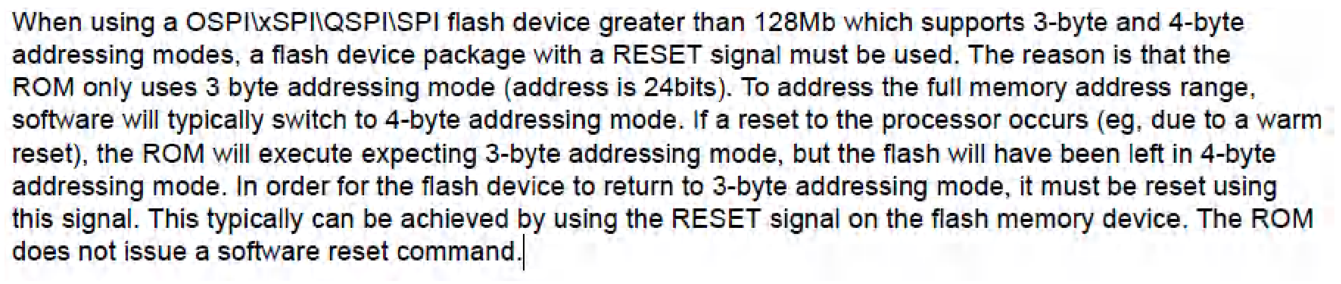
Figure 4-9. OSPI Configuration

5 Common Bring-up Issues and Debugging

Some common issues while working with the FLASH, using MCU+SDK, are as follows.

5.1 Boot Failure

Boot failure around ROM issuing a 3-byte address and flash being left in 4-byte addressing mode is common. To read more about this, please refer to Figure 5-1 from the AM62P TRM.



When using a OSPI\xSPI\QSPI\SPI flash device greater than 128Mb which supports 3-byte and 4-byte addressing modes, a flash device package with a RESET signal must be used. The reason is that the ROM only uses 3 byte addressing mode (address is 24bits). To address the full memory address range, software will typically switch to 4-byte addressing mode. If a reset to the processor occurs (eg, due to a warm reset), the ROM will execute expecting 3-byte addressing mode, but the flash will have been left in 4-byte addressing mode. In order for the flash device to return to 3-byte addressing mode, it must be reset using this signal. This typically can be achieved by using the RESET signal on the flash memory device. The ROM does not issue a software reset command.

Figure 5-1. AM62P TRM Section about RESET Signal

Some serial NAND flashes fail to boot in Serial NAND Boot mode, as the Parameter Page in the flash is not present at the offset 0x1. For a serial NAND flash, the parameter page must be present at the offset 0x1, for it to successfully boot in serial NAND Boot mode. The information about the offset of the parameter page can be obtained from the datasheet of the flash.

For Serial NOR Flash, always make sure to verify the commands from the Boot Mode description. This is mentioned in Understanding Boot Mode and Flash Compatibility.

When using QSPI NOR Flash, a common boot mode to boot is the QSPI Boot mode. Please note, for QSPI Boot mode, apart from matching the boot mode description with the flash datasheet, the QE bit must be checked for. QE is the Quad Enable Bit. For ROM to issue 1S-1S-4S correctly, the QSPI flash must already have the QE bit set, as the ROM does not set this bit. There are QSPI flashes, which have QE bit set from factory by default. For the ones which do not have QE bit set, TI recommends setting the QE bit prior to booting in QSPI boot mode.

5.2 Known Errata

Some of the TI Sitara Processors have known errata around booting and flashing. See *References* for the link to Processor specific errata.

1. i2351: All TI Sitara^{TM} Processors.
2. i2307: AM64x, AM62x Processors.
3. i2366: AM62Ax, AM64x, AM62Dx, AM62x Processors.
4. i2420: AM62Ax, AM64x, AM62Dx, AM62x Processors.
5. i2189: All TI Sitara Processors.
6. i2249: All TI Sitara Processors.
7. i2383: All TI Sitara Processors.

5.3 Flash Initialization Failure

In some cases, a hang is seen inside Flash_norOspiOpen function, while most of the times, there is a log printed as *FLASH open failed for instance 0 !!!*. This indicates that the flash has not been initialized properly. To get past this step, see the following tips.

5.3.1 FLASH and OSPI SysConfig Values

Revisit the OSPI and FLASH values, and see if any value has been written incorrectly.

5.3.2 Flash Device and Manufacture ID Read Failure

To debug READ ID failures, please read through the following FAQ.

5.3.3 PHY Failure

There are two types of failure with respect to PHY tuning. When ***Enable PHY Mode*** is checked, then these two failures can be encountered. Before starting this, learn about OSPI PHY Tuning and PHY Tuning Attack Vector.

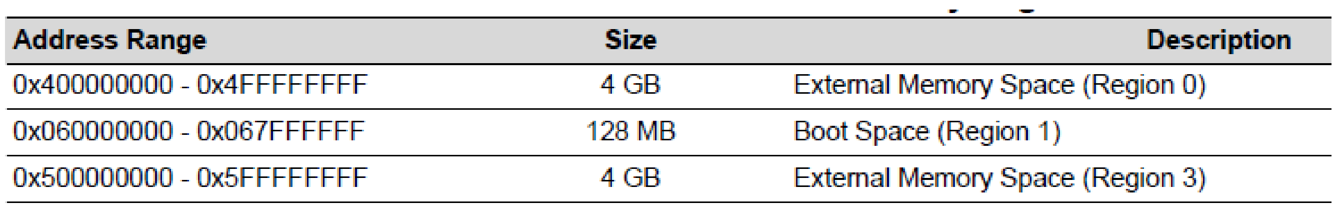
The first being "PHY enabling failed!!! Continuing without PHY..." and second will be failure to compute the OTP (Optimal Tuning Point). The first one occurs, when the attack vector is flashed but not read correctly. So, this in turn points back to sections 5.3.1 and 5.3.2. The second one occurs when the PHY Tuning algorithm returns failure. To debug this, the following steps can be considered:

1. By default, in flash_nor_ospi,c file, OSPI_phyTuneDDR is called. If the protocol used is configured in STR, then the function OSPI_phyTuneSDR must be called.
2. To debug PHY tuning failure, TI recommends generating the PHY tuning graph. Please refer the FAQ here. Upon PHY failure, the graph can be used as a reference to see where the OTP point is marked.

5.4 Flash Read Failure

Flash read failures happen in some cases. They are as follows:

1. A certain address range is used to map the flash contents when the OSPI controller is in DAC state. The available address range is shown in Figure 5-2. By default, 0x60000000 is selected. read failures happen when the MPU Region 0x60000000 is marked as *cached*. This is the region used for Direct Memory Mapping of the Flash. In software, this is achieved by setting the 7th bit, DAC bit, in register 0xFC40000 to 1. As a result, 0x60000000 does not need to be marked as "Cached".
2. Read command set in the SysConfig is incorrect for the configured protocol. Some flashes have different read command for 3 byte and 4 bytes addressing.
3. Suppose the data on the flash is *00h 01h 02h 03h…* and the read values in the buffer are *01h 02h 03h 04h….* This points to section 5.3.1 where the dummy clocks for reading is off by a value of 1.
4. DMA related issues include, data not being read correctly because the read buffer is not cache aligned.



Address Range	Size	Description
0x400000000 - 0x4FFFFFFFF	4 GB	External Memory Space (Region 0)
0x060000000 - 0x067FFFFFF	128 MB	Boot Space (Region 1)
0x500000000 - 0x5FFFFFFFF	4 GB	External Memory Space (Region 3)

Figure 5-2. FSS Memory Regions

5.5 Flash Program Failure

Flash Erase and Write failures happen due to the following reasons:

1. The sector or block to which write operation is issued is not erased prior to the write operation.
2. Write/Erase command set in the SysConfig is incorrect for the configured protocol. Some flashes have different commands for 3-byte and 4-byte addressing.
3. Flash Program and Flash Busy Timeout defined is too low.
4. As mentioned in Figure 5-2 , the default selected address range 0x60000000, must be marked as *Strongly Ordered*, and not *Cached*. Marking the region as *Cached* leads to write failures.

6 Checklist for Requesting OSPI and FLASH Support

Before posting an OSPI/FLASH related question on the TI E2E™ forum, it is helpful if the following queries are addressed:

1. Which TI processor is used?
2. What is the MCU+ SDK version used?
3. Is the testing done on a custom board or a TI EVM?
4. Name the custom flash part and attach the datasheet.
5. Are the values in SysConfig verified with the Flash datasheet?
6. Share the failure logs or the point of failure by following the debugging guide here.
7. Point to a specific section from this application note which resembles close to the failure observed on the test board.

7 Summary

This application note provides a practical debug reference for developers integrating custom xSPI flash devices with TI Sitara processors using the MCU+ SDK. This document outlines systematic methods to diagnose and resolve boot failures, flash read/write/erase issues, and overall device bring-up challenges. Step-by-step checks, configuration guidelines, and validation tips are included to help engineers quickly isolate the root causes and accelerate system bring-up on custom DUTs. By consolidating common failure scenarios and proven debug workflows, this guide aims to reduce integration effort and enable reliable operation of custom NOR or NAND flashes.

8 References

1. Texas Instruments, *AM62Px Sitara Processors*, datasheet.
2. Texas Instruments, *AM62Px MCU+ SDK*.
3. Infineon Technologies, *S28HS512T datasheet*.
4. Texas Instruments, *AM62Px Sitara Processors Technical Reference Mannual*, technical reference manual.
5. Texas Instruments, *AM62x Processor Silicon*, errata.
6. Texas Instruments, *AM62Dx Sitara Processors Silicon Errata*, errata.
7. Texas Instruments, *AM64x/AM243x Processor Silicon*, errata.
8. Texas Instruments, *AM62L Sitara Processors Silicon Errata*, errata.
9. Texas Instruments, *AM62A Processors Silicon Errata*, errata.
10. Texas Instruments, *AM62P Sitara Processors Silicon Errata*, errata.