Freescale Semiconductor
Application Note

Document Number: AN4777
Rev 1, 7/2013

Using the QuadSPI Module on MPC564xS

by: Luis Olea and José Cisneros

Contents

1 Introduction

This application note discusses how to take advantage of the new, special features of the MPC564xS QuadSPI module. It is not intended to be a complete guide to the module; for further information, consult the references section.

This document is accompanied by a ZIP file containing all the code discussed in it, as well as drivers to implement QuadSPI support for various debuggers.

2 Overview

Quad Serial Peripheral Interface (QuadSPI) is a communications protocol used for communications between a microcontroller and an external flash memory. It is based on the popular Serial Peripheral Interface (SPI). Whereas a SPI makes use of up to four connections—Data In, Data Out, Clock, and Chip Select (used to signify that a transmit or receive is active)—QuadSPI uses Clock, up to six Chip Select channels, and up to four bidirectional data channels. This extra connectivity allows data to be read from the flash memory in a prompt manner, making QuadSPI an excellent choice for using additional off-chip memory.

Due to the smaller number of pins, requests for reads/writes/erases are carried out by sending commands across the bus. For example, to read data from flash memory the “Read Data (0x03)” command is sent, followed by the 24- or 32-bit address to be read. The data is then sent to the microcontroller.

There are several suppliers of QuadSPI-compatible memory, such as Winbond™, Spansion®, Macronix™, and Numonyx™. Like SPI before it, QuadSPI does not adhere to a set standard, but as a rule different manufacturers’ devices interface via a similar command set.

3 Special MPC564xS QuadSPI Features

Compared to the QuadSPI module in the MPC560xS devices, there are some powerful new features in the module that enhance performance and functionality. These are the special QuadSPI features available in the MPC564xS.

3.1 Dual mode

Dual mode consists in interfacing to QuadSPI devices using dual line I/O with two bidirectional I/O lines, driven alternatively by the serial flash device or the QuadSPI module. The connection to the external memory would be as shown in the following figure:

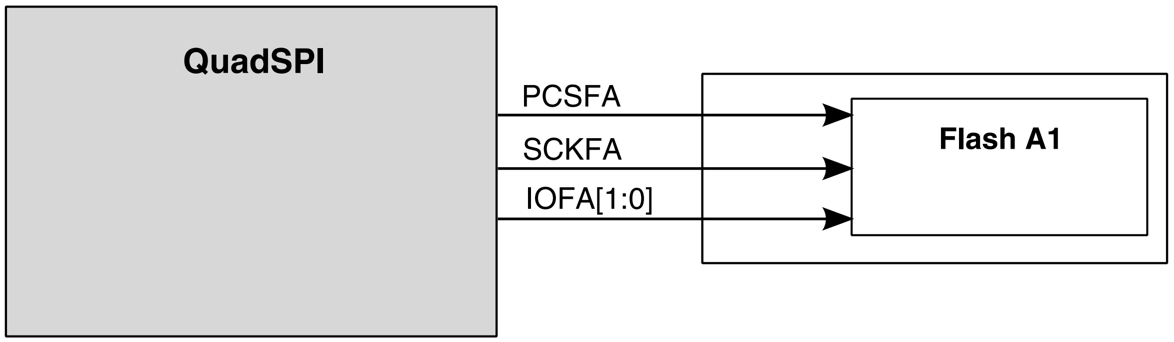

Figure 1. QuadSPI connections in dual mode

3.2 Quad mode

Quad mode consists in interfacing to external serial flash using four I/O lines with four bidirectional I/O lines, driven alternatively by the serial flash device or the QuadSPI module.

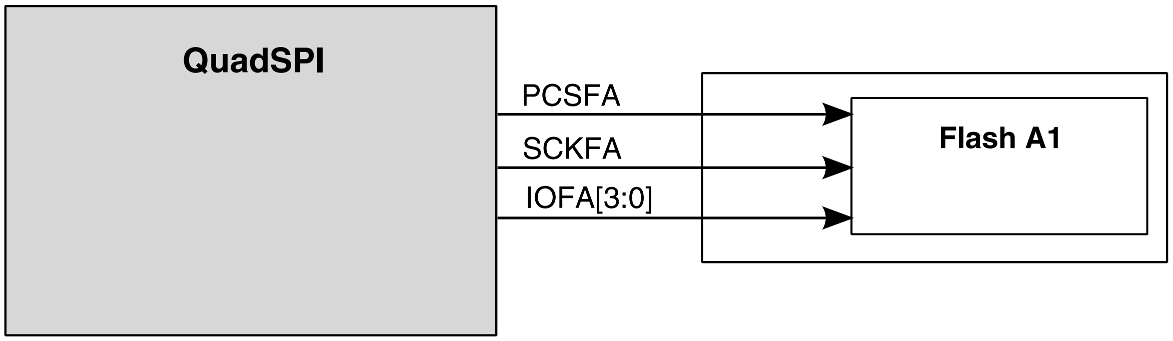

Figure 2. QuadSPI connections in quad mode

3.3 32-bit addresses

In addition to the 24-bit address mode, there is a 32-bit address mode that allows the user to address up to 128 MB of flash memory in external serial flash devices. To enable this mode, simply write 1 to the EXT_ADD field of the QSPI_MCR register, taking into account that the respective external serial flash memory should also be independently enabled for accepting 32-bit addresses.

There are two different ways of 32-bit addressing mode:

- *Extended address mode*: In this mode, the legacy 24-bit commands are converted to accept 32-bit address commands. The flash memory needs to be configured for 32-bit address mode. Also,while programming the LUT sequence in QuadSPI for 32-bit mode, the ADDR and ADDR_DDR command should be programmed with 8'd32 as the operand value. By default, the QuadSPI is in 24-bit legacy address mode. Each of the memory vendors have different way of enabling this mode (Refer to the memory specification from memory vendors). For example, the command B7h sent to Macronix flash will enable it for 32-bit address mode.
- *Extended address register*: In this mode, the upper 8 bits of the 32-bit address is provided by the Extended Address Register in the memory itself. The memory provides a specific register which is updated according to the address to be accessed. This effectively converts the legacy 24-bit address command into 32-bit address commands. The memories greater in size than 16 MB consist of banks of 16 MB. The 8 bits written in the extended address register effectively enables a bank. For example, in Spansion memory, when the extended address register is updated with a value of 0x01 with the help of the command 17h, it will open Bank1 of the memory. The consequent 24-bit address commands will lead to Bank1. The extended address register needs to be updated with the respective value for access to other banks. This way effectively converts the legacy 24-bit address command into 32-bit address commands.

3.4 Parallel simultaneous access to dual-flash devices

In parallel flash mode, the QuadSPI module can communicate with two external serial flash devices, each with four bidirectional data lines. A simple way to view this configuration is that now the user has access to a “virtual” single external flash device that has double storage size and also double bandwidth, since now there are eight I/O lines working at the same time.

There are some limitations in this mode:

- The two flash devices must be accessed with identical control signals.
- Special alignment per flash device is not possible.
- Only identical flash devices can be operated in parallel flash mode; different flash devices only work on individual flash mode.
- Only data read commands are supported in parallel flash mode, other commands will result in an error condition.

The physical connections for a dual flash device setup are shown in the following figure:

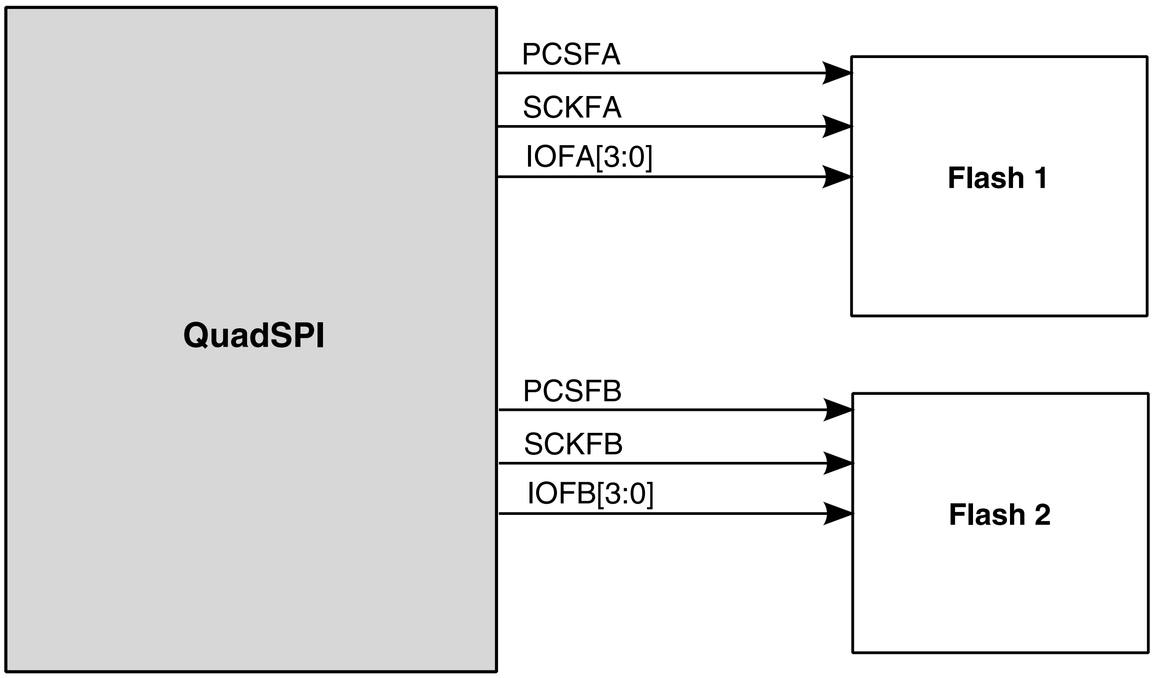

Figure 3. Parallel simultaneous access to dual flash devices

3.5 14 interrupt conditions

There are 14 different interrupt conditions, which are mapped to one of five different interrupt flags. The user has to check which flag triggered the interrupt to know which condition occurred. The possible conditions are:

- TX buffer fill
- IP command transaction finished
- RX buffer drain
- Buffer overflow/underrun error
  - RX buffer overrun
  - TX buffer underrun
  - AHB buffer overflow
- Serial flash command error
  - IP access while AHB busy error
  - IP command could not be triggered error
  - IP access while AHB grant error
  - IP command usage error
- QSPI_FR flags
  - AHB command error flag and buffer overflow/underrun flags
  - IP command usage error flag, IP command error flag, IP command mode error flag, IP command trigger during AHB access error flag, IP command trigger could not be executed, IP command trigger during AHB grant error flag
  - IP command transaction finished flag

4 Setting Up the QuadSPI Module

This section describes the basic configuration needed to interface the QuadSPI with external flash memory. These steps must be completed before a read/write/erase operation can be carried out.

- **Step 1:** Set up the ports.

  Since the QuadSPI on MPC564xS devices is assigned to PortF[10] through PortF[15], the Port Configuration Registers (PCRs) must be set up to designate these pins for use by the QuadSPI module. The number of ports required depends on how many data I/O lines are to be used: one, two, or four.

  All I/O ports used should have both their input and output buffers enabled. Depending on hardware setup, it may also be necessary to vary the slew rate of the output signals. An oscilloscope can be used to find which settings present the most suitable signals.

- **Step 2:** Set up the clock.

  The speed at which the QuadSPI interface operates is restricted by a number of factors, including board layout and pad speeds. With a good layout it should be possible to communicate at speeds of up to approximately 80 MHz on the MPC564xS.

  On the MPC560xS the QuadSPI module is clocked through Auxiliary Clock 3, which is configured in the Clock Generation Module (CGM). The clocking options are:
  - System clock
  - System clock ÷ 2
  - Secondary FMPLL
  - Secondary FMPLL ÷ 2

- **Step 3:** Change module to desired mode.
  The following bitfields need to be configured in the MCR register to set the desired mode:
  - MDIS should be set to 0 to turn on the module clocking.
  - VMID should be set to the appropriate flash device vendor. The options from 1 to 4 are Winbond, Spansion, Macronix, and Numonyx.
  - EXT_ADD needs to be set to 1 if 32-bit addressing will be used.

After completing these steps, the serial flash memory should be accessible. This can be tested by opening a memory window in the debug environment and accessing address 0x80000000. The debugger should be able to perform reads on the flash across the crossbar. In a new device, all memory will be set to 0xFF.

The code to set up the QuadSPI module is included in the companion software, AN4777SW.

5 Using the QuadSPI Module

The following sections refer to the code examples in software ZIP file, AN4777SW, associated with this application note.

6 Reading data

The QuadSPI module has two different methods for reading from the external flash: IP commands and AHB commands. Each of these methods has its own internal buffers.

IP commands (so named because they are written to the QuadSPI module over the I/O system via the IPS interface) are created by writing individual parameters (instruction code, address, size) to the correct registers. These are then passed to Serial Flash Mode (SFM) Command Processing where the full command is built up. This command is then passed across the Clock Domain Crosser and is output on the correct pins. The received data is stored in the Rx Buffer FIFOs (RBDR[0..14]).

Everything above the Clock Domain Crosser is clocked by the system clock, whereas everything below it is clocked by the QuadSPI clock (see Setting Up the QuadSPI Module for further details).

AHB bus accesses have their instruction code and size configured in advance. When the crossbar attempts to access an address that corresponds to the external flash memory, the code, size, and address are sent to the SFM Mode Command Processing as before.

The received data is stored in the AHB Rx buffer. This is not memory-mapped—the data is fed directly to the master that made the access.

The code to set up a read is included in the associated software, AN4777SW.

7 Erasing data

As is typical with large flash memories, erase instructions affect a larger block size than write operations. The size of an erasable sector differs between parts, and the chip's data sheet should be consulted to find sector and block sizes. Also note that the erase size may not be uniform over the full memory map.

An erase is carried out in this way:

1. The start address of the sector/block to be erased should be written to the SFAR register. If the address is not aligned with a sector or block, then the erase will still take place, and all data within the sector/block that contains the address will be erased.
2. All erase and write instructions require a Write Enable instruction (0x06) to be sent prior to the operation. The code should then pause until the module busy flag (SFMSR[BUSY]) is cleared.
3. The instruction code for the required erase command should then be written to the ICR register.
4. The module should now wait for the module busy flag to clear (SFMSR[BUSY]) and for the busy flag on the external flash to clear, indicating that the erase is complete.

The code to carry out a 4 Kbyte sector erase at address 0x000000 of the external flash memory is included in the associated software, AN4777SW.

8 Programming data

External flash memory is programmed in pages of 256 bytes. Two commands are available: Page Program (0x02), which uses a single line to transmit data, and Quad Page Program (0x32) which, as the name suggests, uses all four lines. The write speed of the external flash memory will be much slower than its read speed.

Because of this constraint there is no great benefit from using Quad Writes at higher bus speeds, because the bottleneck is not the transmission time but the flash programming time.

To program a page of flash, the following routine should be used:

1. The start address of the page to be programmed must be written to the SFAR register. The address does not have to be aligned with the beginning of a page; however, if the end of the page is reached and there is still more data to be sent, then this will wrap around and continue writing at the first address of the page.
2. Send a Write Enable command, and wait until the module busy flag is cleared.
3. Write the first 16 bytes (or all the data if size < 16 bytes) to be sent to the Transmit Buffer (TBDR) register.
4. Send the write command and size (in bytes) using the ICR register. The maximum size is 256 bytes (one full page).

5. The transmit buffer will now begin emptying as the data is sent. The SFMSR[TXFULL] field should be polled until it is cleared, at which point further data should then be put into the buffer. This should be repeated until all data has been transmitted.
6. Wait until both the module and the external flash are no longer busy.

Code to carry out the above steps is in the annexed software AN4777SW.zip.

9 References

Freescale Semiconductor, "Quad Serial Peripheral Interface (QuadSPI) Module Updates" (AN4512)

———, "TFT Panel Support in the Vybrid Microcontroller Family" (AN4651)

———, "Using the QuadSPI Module on MPC56xxS" (AN4186)

Document Number AN4777
Revision 1, 7/2013