Freescale Semiconductor
Application Note

Document Number: AN4375
Rev. 1, 06/2012

QorIQ eSPI Controller Register Setting Considerations and Programming Examples

About this document

This document describes how to calculate the maximum frequency, transfer formats or different timings with various configuration settings. It also includes programming examples.

Contents

1 eSPI background information

The enhanced serial peripheral interface (eSPI) is a full-duplex, synchronous, character-oriented channel that supports a simple interface. The eSPI:

- Can only be used as an SPI master
- Is different from a regular SPI interface, because both its transmitter and receiver have a FIFO of 32 bytes
- Is optimized to send data in a frame
- Can support different operation modes
- Can achieve better performance when it interfaces with SPI-based flash memories or EEPROMs

To obtain a comprehensive understanding of the eSPI functionality and the basic operation of the SPI-based flash memory or other SPI based devices, see the following documentation, most of which are available on the Freescale website listed on the last page of this document:

- Applicable chip reference manual
- Applicable chip hardware specifications
- Applicable chip errata
- Manufacturer data sheet on the SPI-based devices selected

2 Calculating the maximum SPI frequency

Some chip reference manuals state that the eSPI can transfer a single character at very high rates—a maximum (up to system clock / 2), which is not true. When the SPMODEx[DIV16x] = 0, and SPMODEx[PMx] = 0000, the eSPI clock is configured to be the system clock divided by 2. In most cases, the system clock is defined to be the platform clock divided by 2. However, the maximum SPI clock frequency is also limited by the eSPI AC timing specification and its connected SPI slave device. In most cases, the SPI clock is lower than the platform clock divided by 4.

The maximum eSPI clock (Fmax) is calculated from the minimum eSPI cycle period. For the output side of eSPI, the formula for the minimum eSPI cycle period is:

Tout = eSPI output Master output delay + board_skew + external spi device input setup time. ***Eqn. 1***

For the input side of eSPI, the formula for the minimum eSPI cycle period is:

Tin = eSPI inputs Master input setup time + board_skew + external spi device output delay. ***Eqn. 2***

The maximum eSPI clock is calculated from:

Fmax = 1/(max(Tin, Tout)) ***Eqn. 3***

Let's choose the P1022 and Atmel® AT25DF641 flash memory as an example to show how to calculate the maximum SPI frequency. This table shows the AC timing specifications of the P1022.

Table 1. P1022 SPI AC timing specifications

Parameter	Symbol[1]	Min	Max	Unit	Note
SPI_MOSI output—Master data (internal clock) hold time	t_{NIKHOX}	0.5 + ($t_{PLATFORM\_CLK}$ *SPMODE[HO_ADJ])	—	ns	2, 3
SPI_MOSI output—Master data (internal clock) delay	t_{NIKHOV}	—	5.5 + ($t_{PLATFORM\_CLK}$ *SPMODE[HO_ADJ])	ns	2,3
SPI_CS outputs—Master data (internal clock) hold time	$t_{NIKHOX2}$	0	—	ns	2
SPI_CS outputs—Master data (internal clock) delay	$t_{NIKHOV2}$	—	5.5	ns	2
SPI inputs—Master data (internal clock) input setup time	t_{NIIVKH}	5	—	ns	—
SPI inputs—Master data (internal clock) input hold time	t_{NIIXKH}	0	—	ns	—

Notes:

1. The symbols used for timing specifications follow the pattern of $t_{\text{(first two letters of functional block)(signal)(state) (reference)(state)}}$ for inputs and $t_{\text{(first two letters of functional block)(reference)(state)(signal)(state)}}$ for outputs. For example, t_{NIKHOV} symbolizes the NMSI outputs internal timing (NI) for the time t_{SPI} memory clock reference (K) goes from the high state (H) until outputs (O) are valid (V).
2. Output specifications are measured from the 50% level of the rising edge of CLKIN to the 50% level of the signal. Timings are measured at the pin.
3. See QorIQ P1022 Integrated Processor Family Reference Manual for detail about the register SPMODE.

This table shows the AC timing specifications of the AT25DF641 from Atmel®.

Table 2. Atmel® AT25DF641 AC Timing Specifications

Parameter	Symbol[1]	Min	Max	Unit
Max frequency	—	—	100	MHz
Output hold time	t_{OH}	2	—	ns
Output valid time	t_V		5	ns
Input setup time	t_{DS}	2	—	ns
Input hold time	t_{DH}	1	—	ns

Ignore the board skew for now and choose the maximum platform clock of 533.33 MHz, which $t_{PLATFORM\_CLK}$ is 1.875 ns. To meet the SPI flash of 1 ns hold time, the SPMODE[HO_ADJ] must be set to 1. From Equation 1, the T_{out} can be calculated by:

$$T_{out} = 5.5 + (t_{PLATFORM\_CLK} * SPMODE[HO\_ADJ]) + 2 = 5.5 + (1.875*1) + 2 = 9.375(ns) \quad \text{Eqn. 4}$$

Based on Equation 2, the T_{in} can be calculated by:

$$T_{in} = 5 + 5 = 10\ (ns) \qquad \text{Eqn. 5}$$

The maximum SPI frequency is calculated from Equation 3:

$$Fmax = 1/Tin = 1/10 \qquad \text{Eqn. 6}$$

Therefore, the maximum SPI frequency value is 100 MHz based on the AC timing specifications.

However, the final maximum SPI frequency is also limited by the software settings. It is determined by the selection of SPMODEx[PMx] and SPMODEx[DIV16x]. The formula to calculate the maximum frequency is:

$$Fmax = System\_Clk / [(SPMODEx[PMx]+1)*2] \text{ when } SPMODE[DIV16x] = 0 \qquad \text{Eqn. 7}$$

$$Fmax = System\_Clk / [(SPMODEx[PMx]+1)*2*16] \text{ when } SPMODE[DIV16x] = 1 \qquad \text{Eqn. 8}$$

where: System_clk is platform clock divided by 2

For this example, the eSPI maximum frequency can be calculated by selecting SPMODEx[PMx] = 1 and SPMODEx[DIV16x] = 0.

$$Fmax = 533.33/2 / [(1+1)*2] = 66.66\ (MHz) \qquad \text{Eqn. 9}$$

If the platform clock is 400 MHz, the eSPI maximum SPI clock can be 100 MHz by selecting SPMOEx[PMx] = 0 and SPMODEx[DIV16x]=0.

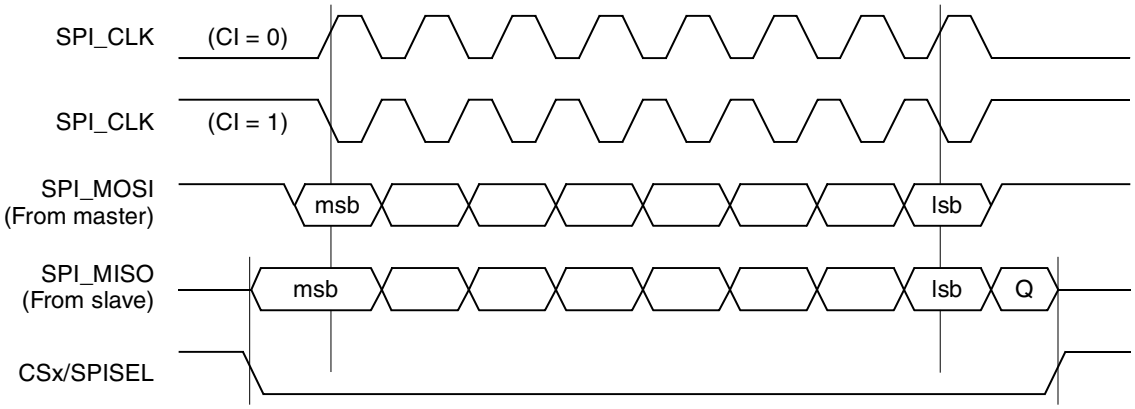
$$Fmax = 400/2 / [(0+1)*2] = 100.00\ (MHz) \qquad \text{Eqn. 10}$$

As you can see, a higher platform frequency may not be necessary to get a higher SPI clock.

3 eSPI transfer formats

Depending on what the value of SPMODEx[Cpx] is set to, the SPI_CLOCK starts toggling at a different time. The eSPI transfer format in which SPI_CLK starts toggling is in the middle of the transfer when SPMODEx[CPx] = 0. While the eSPI transfer format in which SPI_CLK starts toggling is at the beginning of the transfer when SPMODEx[CPx] = 1.

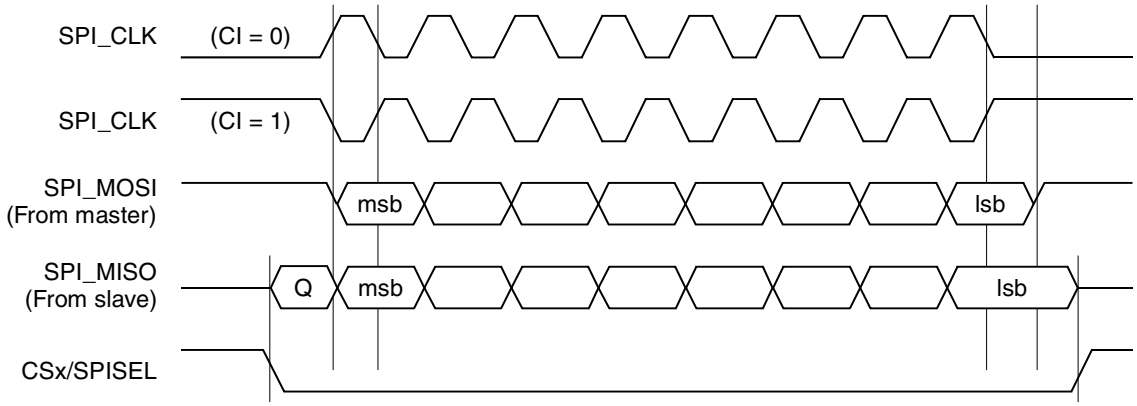
This figure shows the eSPI transfer format with SPMODEx[CPx] = 0.



NOTE: Q = Undefined signal

Figure 1. eSPI transfer format with SPMODEx[CPx] = 0

This figure shows the eSPI transfer format with SPMODEx[CPx] = 1.



NOTE: Q = Undefined signal

Figure 2. eSPI transfer format with SPMODEx[CPx] = 1

This figure shows how to program RxDelay and HLD to work with RapidS mode on AT25DF641 SPI serial flash memory. The SPCOM[RxDelay] register should be set to 1. The SPCOM[HLD] register should be set to 1. The SPMODEx[CIx] = 0; SPMODEx[CPx] = 1.

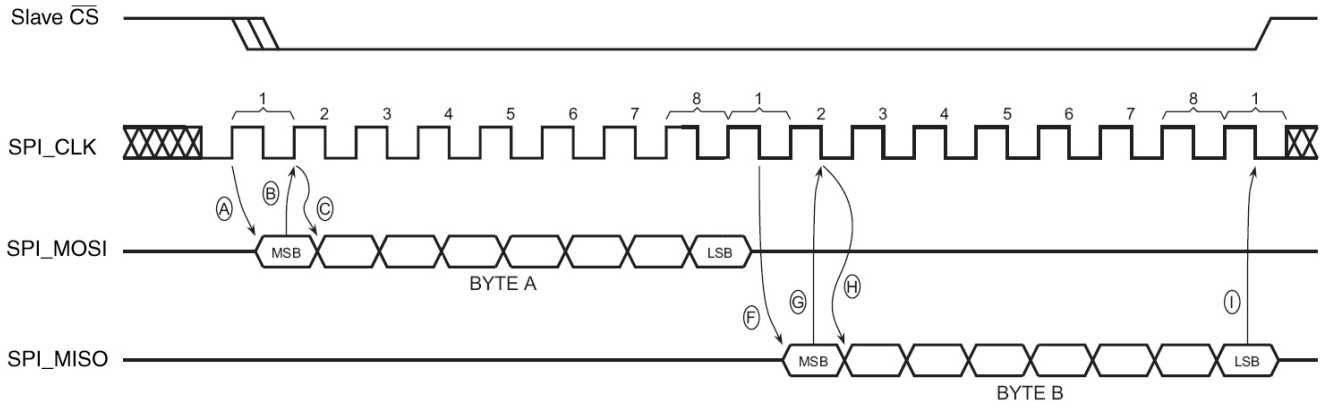


Figure 3. eSPI Transfer Format with SPCOM[RxDelay] = 1

This figure shows the read-data bytes at a higher speed (FAST_READ) command sequence with a Spansion S25FLxxx flash chip. The SPCOM[RxDelay] register should be set to 1. The SPCOM[HLD] register should be set to 0. SPMODEx[CIx] = 0; SPMODEx[CPx] = 0. Note that an extra dummy-byte writes to SPITF before writing to SPCOM.

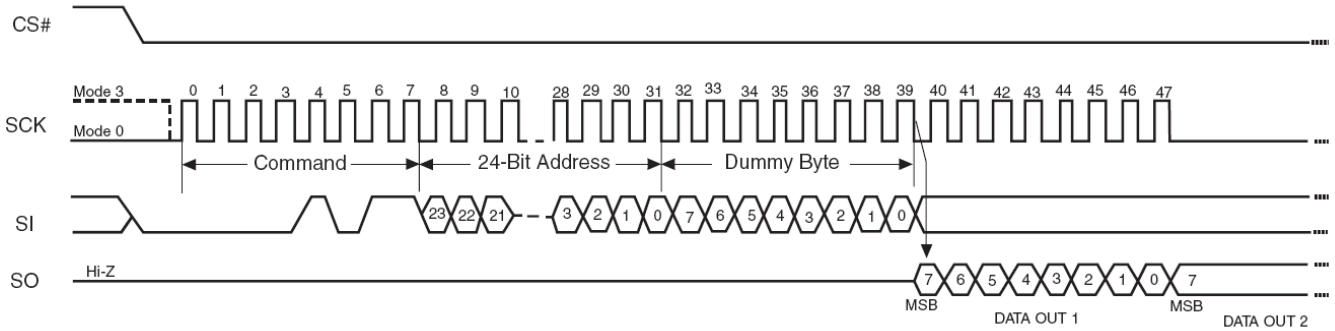


Figure 4. Read data bytes at higher speed (FAST_READ) command sequence

4 eSPI programming examples

NOTE

The applicable chip reference manual includes good examples regarding the content of SPITF and SPIRF with various parameters set (specifically, see sections “eSPI transmit FIFO access register (SPITF),” and “eSPI receive FIFO access register (SPIRF)”. Section “eSPI Programming Examples” includes 24-bit and 16-bit address examples, but the examples in this document are more detailed.

NOTE

Note that in these examples, hex is denoted by an “h” suffix.

4.1 16-bit address example

The following sequence initializes the eSPI to read 36 bytes from 16-bit address memory, start address = 0040h. In this example, chip-select 1 is used.

1. Configure a parallel I/O signal to operate as the eSPI CS1 output signal.
2. Write FFFF FFFFh to SPIE to clear any previous events. Configure SPIM to enable all desired eSPI interrupts.
3. Configure SPMODE = 8000 100Fh to enable normal operation, eSPI enabled.
4. Configure SPMODE1 = 2417 1108h—REV1 = 1, PM1 = 4 (divide eSPI input clock by 10), LEN1 = 7, POL1 = 1, CS1BEF = CS1AFT = CS1CG= 1.
5. Configure SPITF = 0300 40*xx*h (*xx* is don't care)—03h is read opcode while 0040h is the 16-bit start address. This should be done by two writes to SPITF:
   - 1 half-word write with 0300h, then
   - 1-byte write with 40h
6. Configure SPCOM = 4003 0026h so 3 bytes are skipped (1 for opcode and 2 for 16-bit address), TRANLEN = 36 + 3 – 1 = 38 = 26h.

4.2 8-bit address example

The following sequence initializes the eSPI to read 6 bytes from 8-bit address memory from start address = 5Eh and then to read 2 bytes from start address = 40h. In this example, chip-select 0 is used.

1. Configure a parallel I/O signal to operate as the eSPI CS0 output signal.
2. Write FFFF FFFFh to SPIE to clear any previous events. Configure SPIM to enable all desired eSPI interrupts.
3. Configure SPMODE = 8000 100Fh to enable normal operation, eSPI enabled.
4. Configure SPMODE0 = 2417 1108h—REV0 = 1, PM0 = 4 (divide eSPI input clock by 10), LEN0 = 7, POL0 = 1, CS0BEF = CS0AFT = CS0CG = 1.
5. Configure SPITF = 035E 0340h—03h is read opcode while 5Eh is the first 8-bit start address.The second 03h is for the second read opcode while 40h is the second 8-bit start address.
6. Write SPCOM = 0002 0007h so that 2 bytes are skipped (1 for opcode and 1 for 8-bit address), TRANLEN = 6 + 2 – 1.
7. Wait for SPIE[DON] to be set.
8. Write SPCOM = 0002 0003h so that it skips 2 bytes (1 for opcode and 1 for 8-bit address), TRANLEN= 2 + 2 – 1.

4.3 RapidS example

The following sequence initializes the eSPI to read 36 bytes from 16-bit address memory, start address = 0040h using the RapidS mode (mode 0). It corresponds to Figure 3. In this example, chip-select 1 is used.

1. Configure a parallel I/O signal to operate as the eSPI CS1 output signal.
2. Write FFFF FFFFh to SPIE to clear any previous events. Configure SPIM to enable all desired eSPI interrupts.
3. Configure SPMODE = 8000 100Fh to enable normal operation, eSPI enabled.
4. Configure SPMODE1 = 6117 1108h—CP1 = 1, REV1 = 1, PM1 = 1 (divide eSPI input clock by 4), LEN1 = 7, POL1 = 1, CS1BEF = CS1AFT = CS1CG = 1. (Ci = 1, cp = 0 if mode 3 is used)
5. Configure SPITF = 0300 40*xx*h (*xx* is don't care)—03h is read opcode while 0040h is the 16-bit start address. This should be done by two writes to SPITF:
   - 1 half-word write with 0300h, then
   - 1-byte write with 40h
6. Configure SPCOM = 6403 0026h—RxDelay = 1, HLD=1 (it should set to 0 if mode 3 is used); 3 bytes are skipped (1 for opcode and 2 for 16-bit address), TRANLEN = 36 + 3 – 1 = 38 = 26h.

4.4 32-bit address example

The following sequence initializes the eSPI to read 36 bytes from 32-bit address memory, start address = 0000 0040h:

1. Configure a parallel I/O signal to operate as the eSPI CS1 output signal.
2. Write FFFF FFFFh to SPIE to clear any previous events. Configure SPIM to enable all desired eSPI interrupts.
3. Configure SPMODE = 8000 100Fh to enable normal operation, eSPI enabled.
4. Configure SPMODE1 = 2417 1108h—REV1 = 1, PM1 = 4 (divide eSPI input clock by 10), LEN1 = 7, POL1 = 1, CS1BEF = CS1AFT = CS1CG = 1.
5. Configure SPITF = 0300 0000h (32-bit write), 40*xx xxxx*h (only one byte write)—03h is read opcode while 0000 0040h is the 32-bit start address. This should be done by two writes to SPITF:
   - 1 word write with 0300 0000h, then
   - 1-byte write with 40h
6. Configure SPCOM = 4005 0028h so 5 bytes are skipped (1 for opcode and 4 for 32-bit address), TRANLEN = 36 + 5 – 1.

5 Revision history

This table provides a revision history for this document.

Table 3. Document revision history

Rev. number	Date	Substantive change(s)
1	06/2012	• Updated the steps in Section 4.2, “8-bit address example.” • Modified Equation 4. •
0	02/2012	Initial public release

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number: AN4375
Rev. 1
06/2012

freescale™