Freescale Semiconductor
Application Note

Document Number:AN4512
Rev. 0, May 2012

Quad Serial Peripheral Interface (QuadSPI) Module Updates

Implemented on Vybrid MCU

by: David Paterson

Contents

1 Introduction

The QuadSPI module has had some significant changes made for implementation on the Vybrid MCU. These changes will be discussed in this application note in comparison to the QuadSPI versions implemented on MPC5606S and MPC5645S MCUs, and some code examples will be shown.

2 Summary of updates

Figure 1 shows the block diagram of the QuadSPI module implemented on Vybrid.

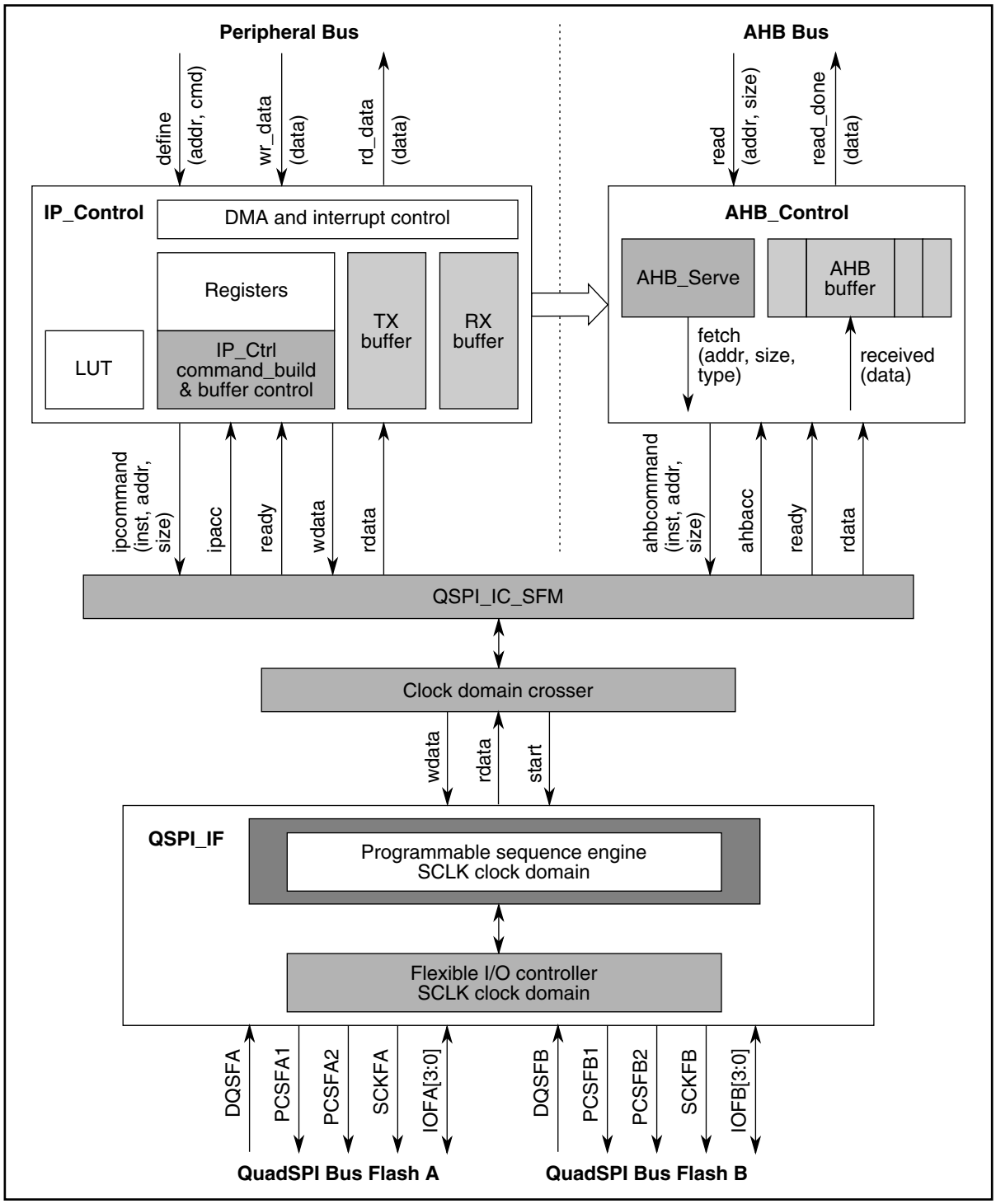

Figure 1. QuadSPI block diagram

The new and key features will be discussed in detail in the next section.

This QuadSPI module still allows for some key functionality similar to previous versions, but with minor implementation updates which will be discussed:

- Parallel mode
- Single, dual and quad mode
- 24-bit (16 MB maximum memory access) and 32-bit (16 MB+ memory access) address mode

- Interrupt and DMA support
- Power saving features (Stop and Disable/Doze mode)

There are some new signals implemented:

- 2 × chip select signals per flash bus (PCSFA1/2 and PCSFB1/2) to allow two serial flash memory devices to be connected and accessed, or one dual-die package which consists of two devices (dies) stacked within the same package to increase the memory capacity of a single package. These two devices would share the same data I/O pins and clock, so a single flash bus could not operate in parallel mode.
- DQS (Data Strobe) signal per flash bus. Some flash vendors provide the DQS signal to which the read data is aligned in DDR mode. For example, Spansion provide this functionality.

DDR (double data rate) / DTR (double transfer rate) mode where the data is generated on every edge of the serial flash clock, if the serial flash device supports this.

A programmable sequence engine is flexible enough to cater for future command/protocol changes and is able to support all existing vendor commands and operations. A look-up table (LUT) stores sequences of instructions and the programmable sequence engine executes the instructions in these sequences to generate a valid serial flash transaction.

Flexible AHB buffers allow multi-master accesses which can be prioritized to allow a flexible and configurable buffer for each master, or a generic buffer for all.

There is an additional ORed interrupt source which is a logical OR from all the QSPI_FR flags.

There are multiple new registers (IPCR, FLSHCR, BUFxCR, BFGENCR, SOCCR, BUFxIND, SPNDST, SPTRCLR, SFAxAD, SFBxAD, LUTKEY, LCKCR, LUT[0…3]-[60…63]).

3 Detail of updates

This section provides more detail about the updates listed in Summary of updates.

3.1 Key functionality

3.1.1 Parallel mode

This mode allows two identical serial flash memory devices to be connected and accessed in parallel, forming one (virtual) flash memory with doubled readout bandwidth.

Parallel Flash mode is valid only for commands related to data read from the serial flash. Any IP command other than data read in Parallel Flash mode will result in the assertion of the QSPI_FR[IUEF] flag, and any AHB command other than data read in Parallel Flash mode will result in the assertion of the QSPI_FR[ABSEF] flag.

The first device of flash memory A has to be paired with the first device of flash memory B and the second device of flash memory A has to be paired with the second device of flash memory B in parallel mode, as shown in Figure 2.

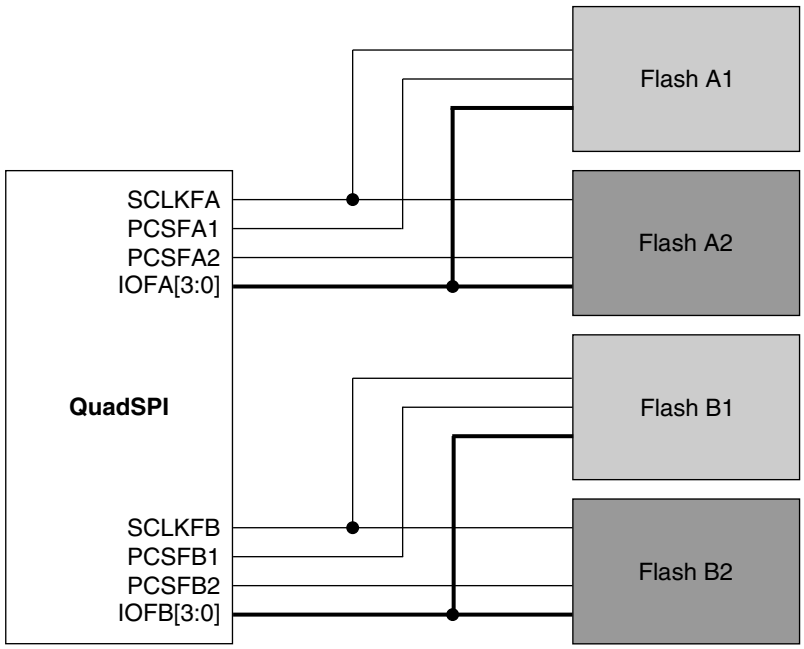

Figure 2. Parallel mode diagram

Parallel mode is selected via the QSPI_BUFxCR[PAR_EN] bit for different masters in AHB-driven mode and via the QSPI_IPCR[PAR_EN] in IP-driven mode.

In parallel mode, the incoming address (SFAR address for IP-initiated transactions and the incoming AHB address for AHB-initiated transactions) is divided by two and sent to the two flashes connected in parallel.

An example access programming for parallel mode access is shown below:

```
QSPI_AMBA_BASE - 0x10000000
QSPI_SFA1AD[TPADA1] - 0x20000000
QSPI_SFA2AD[TPADA2] - 0x30000000
QSPI_SFB1AD[TPADB1] - 0x40000000
QSPI_SFB2AD[TPADB2] - 0x50000000
```

In order to access the first location of A1/B1 pair, the Serial Flash Address Register (QSPI_SFAR) must be programmed with 0x20000000.

This address is divided by two and QSPI_AMBA_BASE is subtracted. Therefore, the address provided to flash A1 and B1 is: Flash Address = ((0x20000000/2) – QSPI_AMBA_BASE) = 0x00000000.

Similarly, in order to access the first location of the A2/B2 pair, the Serial Flash Address Register (QSPI_SFAR) must be programmed with 0x60000000.

So, Flash Address = ((0x60000000/2) – QSPI_AMBA_BASE) = 0x20000000.

3.1.2 Single, dual, and quad I/O mode

The QuadSPI module supports single, dual, and quad I/O mode. The programmer must program the number of pads for the particular instruction in the PAD0/1 field of the LUT register:

Table 1. Pad information for INSTR0/1

Value	Number of pads
00	1

Table continues on the next page...

Table 1. Pad information for INSTR0/1 (continued)

Value	Number of pads
01	2
10	4
11	Reserved

Some example commands and corresponding waveforms are shown below:

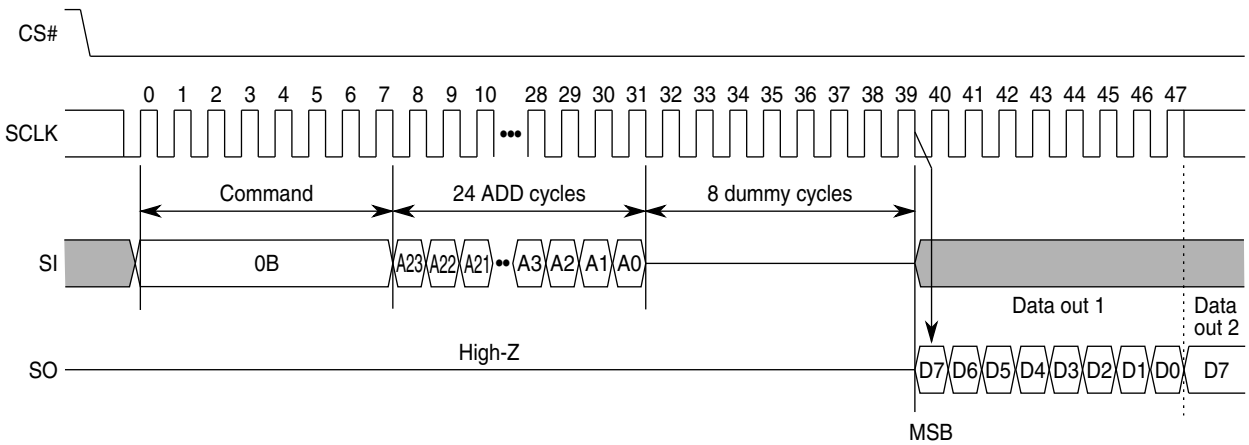

Figure 3. Fast read (0x0B) single IO read

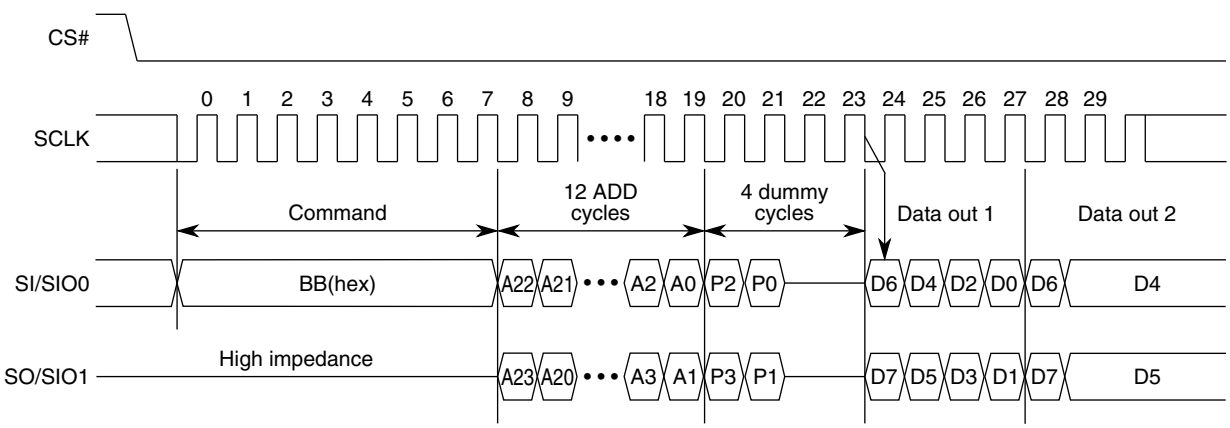

Figure 4. 2READ (0xBB) 2 × IO read

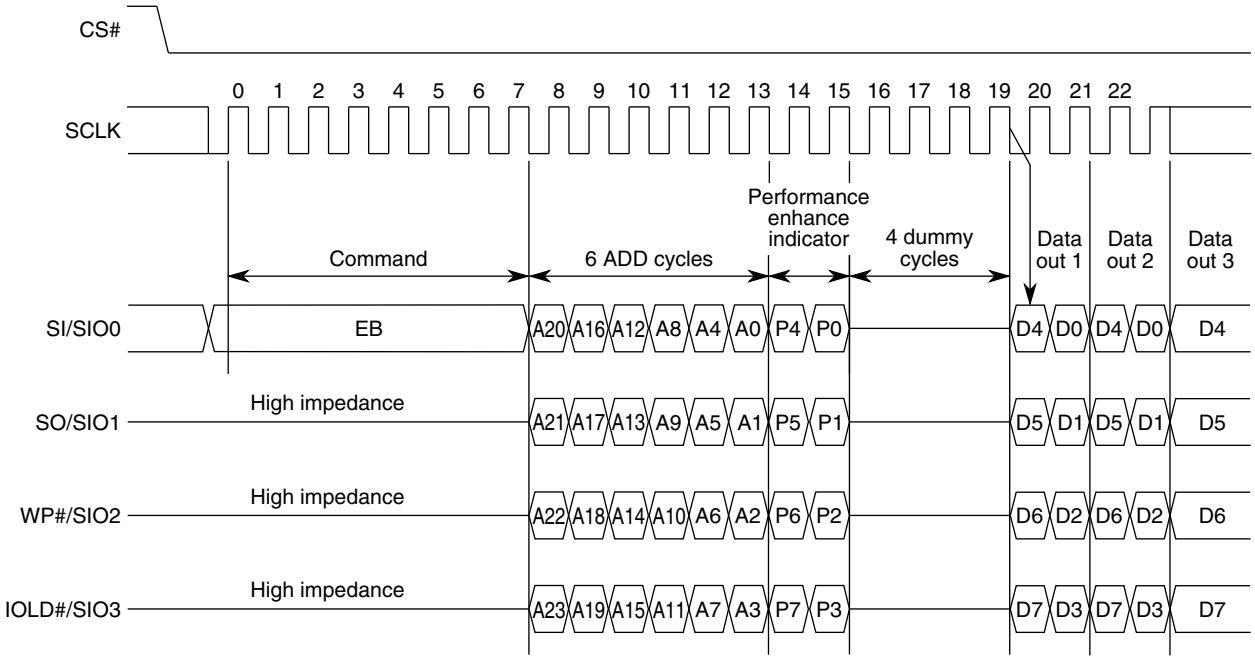

Figure 5. 4READ (0xEB) 4 × IO read

3.1.3 24-bit and 32-bit address modes

The address mode has been made flexible so that the QuadSPI can access 4 GB memory space (32 address bits).

The programmer must specify the number of address bits in the OPRND0/1 field of the LUT register for the ADDR instruction.

Also, the Serial Flash Device may need to be enabled for 32-bit address mode. For example, Macronix command EN4B (0xB7) Enable 4-byte mode.

3.1.4 Interrupt and DMA support

In total there are thirteen interrupt conditions mapped to six different interrupt lines.

The QSPI_FR register provides all available flags which may serve as source for the generation of interrupt service requests.

The QSPI_RSER register provides enables and selectors for the interrupts and DMA in the QuadSPI module. However, there is only one flag that can generate interrupt as well as DMA requests: RX buffer drain (RBDF). There are separate enable bits for generating these IRQ and DMA requests.

Table 2. Interrupt and DMA mapping

IRQ/DMA line	Associated QSPI_FR flags	Description
ipi_int_tfff	TBFF	TX buffer fill
ipi_int_tcf	TFF	Peripheral command transaction finished
ipi_int_rfdf	RBDF	RX buffer drain
ipi_int_overrun	RBOF, TBUF, ABOF	Buffer overflow/underrun error
ipi_int_cerr	IPAEF, IPIEF, IPGEF, IUEF	Serial flash command error

Table continues on the next page...

Table 2. Interrupt and DMA mapping (continued)

IRQ/DMA line	Associated QSPI_FR flags	Description
ipi_int_ored	All flags above plus DLPFF, ILLINE, ABSEF	Logical OR from all the QSPI_FR flags
ipd_req_rfdf	RBDF	DMA RX buffer drain

3.1.5 Power saving features

It is possible to enter stop or module disable mode when:

- QSPI_SR[BUSY] = 0 and
- QSPI_SR[AHBTRN] = 0 and
- QSPI_RBSR[RDBFL] = 0 and
- QSPI_SR[RXDMA] = 0 and
- None of the flags in the QSPI_FR register enabled as interrupts is set

The conditions given above ensure that:

- There is no SFM command currently being executed.
- All the data read into the RX buffer from the serial flash have been fetched by the application.
- No current AHB access is pending.
- No active DMA is pending.
- No enabled interrupt is pending.

Certain read or write operations have a different effect when the QuadSPI is in Stop mode. In Stop mode not all of the status and flag bits of the QuadSPI module are updated and writing to them will have no effect. Interrupt and DMA request signals cannot be cleared while in Stop mode.

3.1.5.1 Stop mode

Stop mode ultimately shuts off the QuadSPI clocks and memory-mapped logic is not accessible, therefore reducing power. When a request is made to enter Stop mode, the QuadSPI completes the action currently being processed.

Stop mode is initially configured by setting the QSPI_MCR[MDIS] bit.

Module Disable	The MDIS bit allows the clock to the non-memory mapped logic in the QuadSPI to be stopped, putting the QuadSPI in a software controlled power-saving state. 0 = Enable QuadSPI clocks. 1 = Allow external logic to disable QuadSPI clocks.

3.1.5.2 Module Disabled mode (Doze)

The clock to the non-memory mapped logic in the QuadSPI can be stopped while in Module Disable mode.

The module enters the mode while QSPI_MCR[DOZE] is set, and by setting QSPI_MCR[MDIS] or when a request is asserted by an external controller. Note that the memory mapped registers are still accessible.

The DOZE bit provides support for externally controlled Doze Mode power-saving mechanism.

- 0 A doze request will be ignored by the QuadSPI module
- 1 A doze request will be processed by the QuadSPI module

Doze Enable	The DOZE bit provides support for externally controlled Doze mode power-saving mechanism. 0 = A doze request will be ignored by the QuadSPI module 1 = A doze request will be processed by the QuadSPI module

3.1.6 New signals

3.1.6.1 Dual die chip selects

There are now four chip select signals, two per flash bus instance. This allows dual-die package devices to be connected. Since these two devices share the same data I/O pins and clock within a package, they cannot function in parallel mode.

Flash Bus A

- PCSFA1 — Peripheral Chip Select Flash A1
- PCSFA2 — Peripheral Chip Select Flash A2

Flash Bus B

- PCSFB1 — Peripheral Chip Select Flash B1
- PCSFB2 — Peripheral Chip Select Flash B2

3.1.6.2 Data strobe

Some flash vendors provide the DQS signal to which the read data is aligned in DDR mode. The QuadSPI module provides two input signals for each flash bus (DQSFA and DQSFB).

3.2 DDR/DTR mode

When using DDR (Dual Data Rate), also known as DTR (Dual Transfer Rate) mode, the data is generated on every edge of the serial flash clock, hence doubling the bandwidth.

Note that some flash vendors provide the DQS signal to which the read data is aligned in DDR mode (Spansion for example).

DDR Mode is enabled by setting the QSPI_MCR[DDR_EN]:

DDR_EN	DDR mode enable: 0 = 2× and 4× clocks are disabled. 1 = 2× and 4× clocks are enabled.

The sampling register (QSPI_SMPR) should also be configured. When the serial flashes function in DDR mode, the data is valid for only half a clock cycle. This, along with the fact that the time for which the data is actually valid is smaller than half a clock cycle, requires that we provide closely spaced sampling points. The QuadSPI samples the incoming data at multiple sampling points provided by a 4x serial flash clock in DDR mode. Figure 6 shows the different sampling points as configured by QSPI_SMPR[DDRSMP].

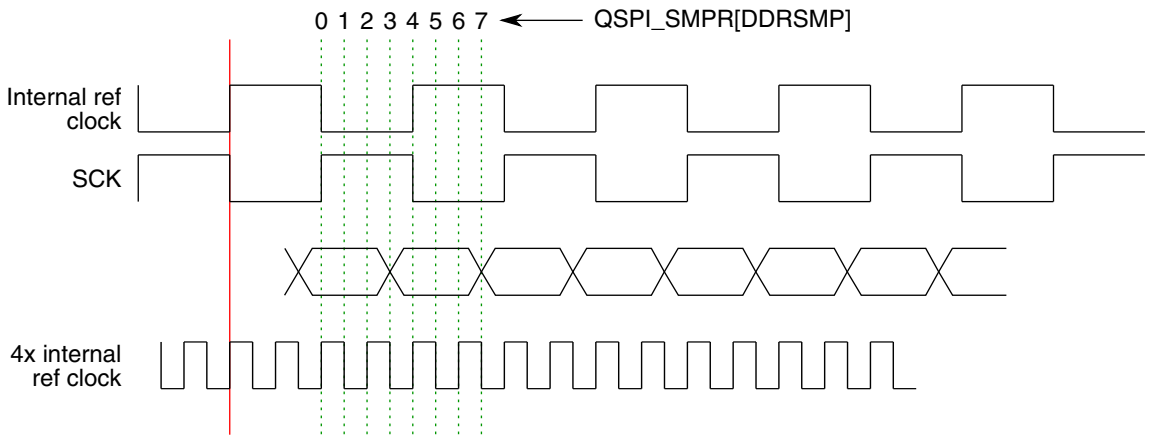

Figure 6. DDR sampling edges

When the DQS signal is used, the software must set QSPI_MCR[DQS_EN]. This bitfield is valid only when the DDR_EN bitfield is set. When enabled, the incoming data is sampled on both the edges of DQS input. The QSPI_SMPR[DDR_SMP] values are ignored.

- 0 DQS disabled
- 1 DQS enabled

DQS_EN	DQS enable: 0 = DQS disabled 1 = DQS enabled

3.3 Programmable sequence engine

The core of the QuadSPI module is a programmable sequence engine that works on instruction-operand pairs. The core controller executes each programmed instruction sequentially. It caters for future command/protocol changes and is able to support all existing vendor commands and operations. The engine is flexible enough to support both 24- and 32-bit address modes as well as any future variations in addressing.

A sequence of such instruction-operand pairs may be prepopulated in the LUT according to the device connected on board. Each instruction-operand pair is 16 bits (2 bytes) long, and every sequence preprogrammed in the LUT is referred to by its index.

The look-up table (LUT) consists of a number of preprogrammed sequences. Each sequence consists of instruction-operand pairs which when executed sequentially generate a valid serial flash transaction. Each sequence can have a maximum of eight instruction-operand pairs. The LUT can hold a maximum of up to sixteen sequences. Figure 7 shows the basic structure of the sequence in the LUT.

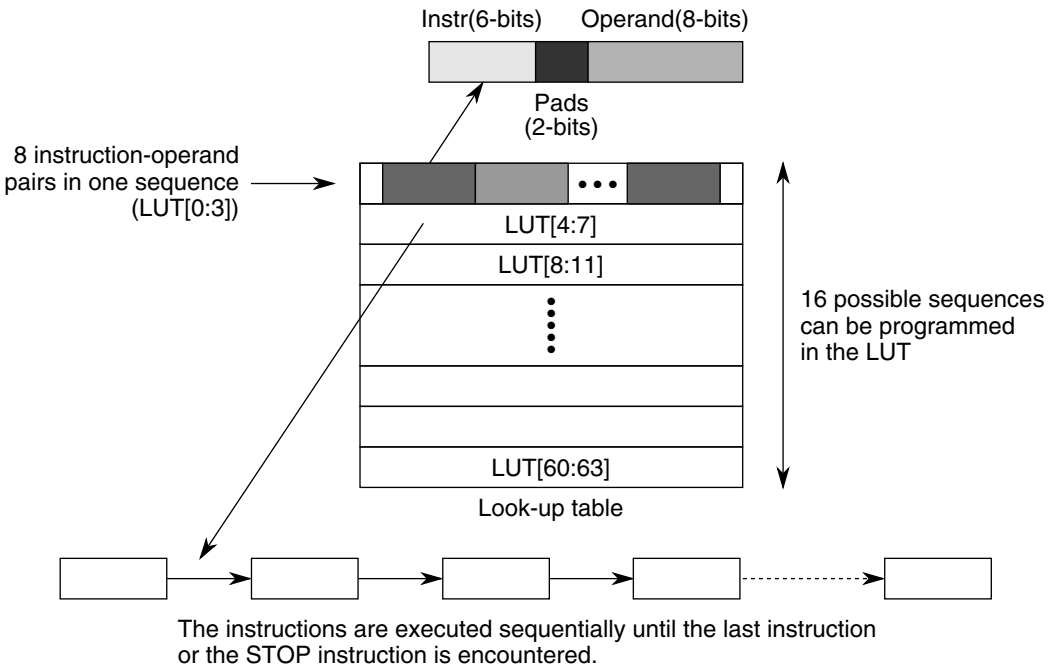

Figure 7. LUT and sequence structure

At reset, there is one sequence programmed that reads 32 bytes of data from the serial flash using a 24-bit address and on a single I/O, shown in Table 3 :

Table 3. Reset sequence

Instruction	Pad	Operand	Comment
CMD	0x0	0x03	Read data byte command on one pad
ADDR	0x0	0x18	24 address bits to be sent on one pad
READ	0x0	0x04	Read 32 bytes
JMP_ON_CS	0x0	0x00	Jump to instruction 0 (CMD)

The user must prepopulate the LUT with the required serial flash command sequences. The LUT may be locked to protect its contents from being changed during a code runover.

The process for locking the LUT is:

1. Write the key (0x5AF05AF0) to the LUT Key Register (QSPI_LUTKEY).
2. Write 0b01 to the LUT Lock Configuration Register (QSPI_LCKCR)).

The process for unlocking the LUT is:

1. Write the key (0x5AF05AF0) to the LUT Key Register (QSPI_LUTKEY).
2. Write 0b10 to the LUT Lock Configuration Register (QSPI_LCKCR)).

Note that the transactions should immediately follow each other (no other transactions can be issued in between).

Some example sequences can be found in Code examples.

3.4 Flexible multi-master access

The QuadSPI implements flexible and configurable buffers for multiple masters with priority.

Each flexible AHB buffer is associated with:

- An AHB master. Optionally, buffer3 may be configured as an all-master buffer by setting the QSPI_BUF3_CR[ALLMST] bit. When buffer3 is configured in this way, any access from a master not associated with any other buffer is routed to buffer3.
- A data size field representing the amount of data to be fetched from the flash on every missed access.

Figure 8 shows the flexible AHB buffers.

The QSPI_BFGENCR[SEQID] field points to an index of the LUT.

Buffer0 may optionally be configured to be associated with a high-priority master. An access by a high-priority master suspends any ongoing prefetch to any of the other buffers. The ongoing prefetch is suspended and the high-priority master is serviced first. Once the high-priority master access is complete, the suspended transaction is resumed (before any other AHB access is entertained). The status of the suspended buffer can be read from Sequence Suspend Status Register (QSPI_SPNDST).

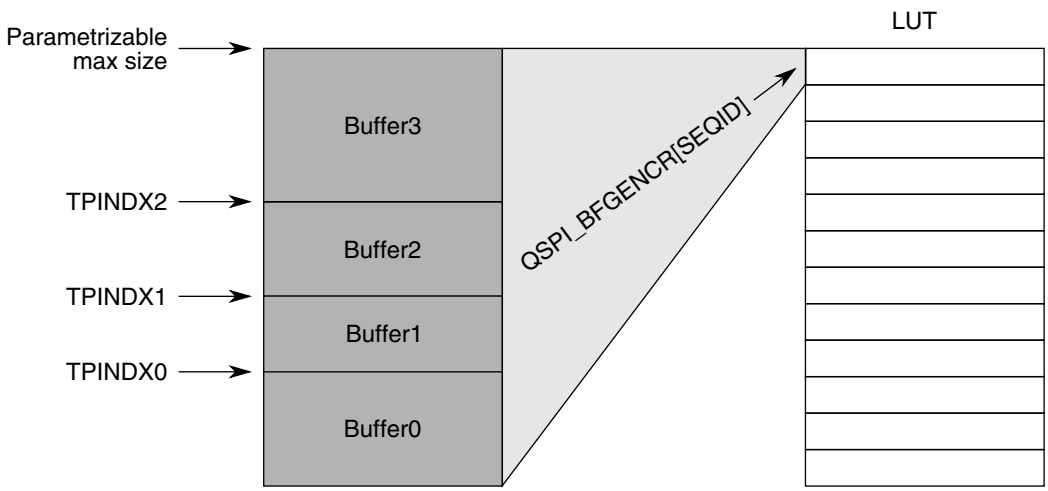

Figure 8. Flexible AHB buffers

3.5 Other new features

There are a few more useful features detailed in this section.

3.5.1 Idle signal drive

The QSPI_MCR register contains some bits for configuring the idle signal drive for I/O pins 2 and 3. This is particularly useful when using single or dual I/O mode where these pins are not used but may be connected to the serial flash device. Table 3 shows the settings for each I/O for Flash A and Flash B.

Table 4. Idle signal drive

Field	Description
ISD2FA	Idle Signal Drive IOFA[2] Flash A. This bit determines the logic level that the IOFA[2] output of the QuadSPI module is driven to in the inactive state. 0 IOFA[2] is driven to logic L 1 IOFA[2] is driven to logic H
ISD3FA	Idle Signal Drive IOFA[3] Flash A. This bit determines the logic level the IOFA[3] output of the QuadSPI module is driven to in the inactive state. 0 IOFA[3] is driven to logic L 1 IOFA[3] is driven to logic H
ISD2FB	Idle Signal Drive IOFB[2] Flash B. This bit determines the logic level the IOFB[2] output of the QuadSPI module is driven to in the inactive state. 0 IOFB[2] is driven to logic L 1 IOFB[2] is driven to logic H
ISD3FB	Idle Signal Drive IOFB[3] Flash B. This bit determines the logic level the IOFB[3] output of the QuadSPI module is driven to in the inactive state. 0 IOFB[3] is driven to logic L 1 IOFB[3] is driven to logic H

3.5.2 Software reset

The QSPI_MCR register contains some bits for resetting the host and serial flash domains. It is advisable to reset both the serial flash domain and AHB domain at the same time as resetting only one domain might lead to unwanted side effects.

Table 5. Software reset

Field	Description
SWRSTHD	Software reset for HOST domain 0 = No action 1 = Host domain flops are reset; does not reset configuration registers
SWRSTSD	Software reset for serial flash domain 0 = No action 1 = Serial flash domain flops are reset; does not reset configuration registers

3.5.3 Sequence pointer clear

The sequence pointer clear register provides bits to reset the IP and buffer sequence pointers. The software should reset the sequence pointers whenever the sequence is changed either by updating the SEQ_ID field of Buffer Generic Configuration Register (QSPI_BFGENCR) or Sequence Pointer Clear Register (QSPI_SPTRCLR).

4 Code examples

Here are some C code examples.

4.1 Programmable sequence engine

This code example shows how the LUT can be unlocked, programmed, and locked. Note that only a few example sequences are shown.

```
//Unlock the LUT
QUADSPI.LUTKEY.R = 0x5AF05AF0;
QUADSPI.LCKCR.R = 0x2;
while(!QUADSPI.LCKCR.B.UNLOCK);

// SEQID 0 -Programmed by default (Read 32 bytes)
QUADSPI.LUT[0].B.INSTR0 =  CMD;//CMD = 0x1
QUADSPI.LUT[0].B.PAD0 = 0;//Single IO
QUADSPI.LUT[0].B.OPRND0 = MXIC_READ;//MCIC_READ = 0x3
QUADSPI.LUT[0].B.INSTR1 = FL_ADDR;//FL_ADDR = 0x2
QUADSPI.LUT[0].B.PAD1= 0;
QUADSPI.LUT[0].B.OPRND1 = MXIC_ADDR;//MXIC_ADDR = 0x18 (24bit)

QUADSPI.LUT[1].B.INSTR0 =  READ;//READ = 0x7
QUADSPI.LUT[1].B.PAD0 = 0;
QUADSPI.LUT[1].B.OPRND0 = 4;//32bytes
QUADSPI.LUT[1].B.INSTR1 = JMP_ON_CS;// JMP_ON_CS = 0x9
QUADSPI.LUT[1].B.PAD1= 0;
QUADSPI.LUT[1].B.OPRND1 = 0;

// SEQID 1 -Set Write Enable on Serial Flash Device
QUADSPI.LUT[4].B.INSTR0 =  CMD; //CMD = 0x1
QUADSPI.LUT[4].B.PAD0 = 0; //Single IO
QUADSPI.LUT[4].B.OPRND0 = MXIC_WREN ;//MXIC_WREN = 0x6
QUADSPI.LUT[4].B.INSTR1 = 0;
QUADSPI.LUT[4].B.PAD1= 0;
QUADSPI.LUT[4].B.OPRND1 = 0 ;

// SEQID 2 -Read from flash  in Single SDR Mode
QUADSPI.LUT[8].B.INSTR0 =  CMD; //CMD = 0x1
QUADSPI.LUT[8].B.PAD0 = 0;
QUADSPI.LUT[8].B.OPRND0 = MXIC_SIODRD;// MXIC_SIODRD = 0x0B
QUADSPI.LUT[8].B.INSTR1 = FL_ADDR;//FL_ADDR = 0x2
QUADSPI.LUT[8].B.PAD1= 0;
QUADSPI.LUT[8].B.OPRND1 = MXIC_ADDR; //MXIC_ADDR = 0x18 (24bit)

QUADSPI.LUT[9].B.INSTR0 =  DUMMY;//DUMMY = 0x3
QUADSPI.LUT[9].B.PAD0 = 0;
QUADSPI.LUT[9].B.OPRND0 = MXIC_DUMMY8; // MXIC_DUMMY8 = 0x8
QUADSPI.LUT[9].B.INSTR1 = READ; //READ = 0x7
QUADSPI.LUT[9].B.PAD1= 0;
QUADSPI.LUT[9].B.OPRND1 = MXIC_READ; // MXIC_READ = 0x03

QUADSPI.LUT[10].B.INSTR0 =  JMP_ON_CS; // JMP_ON_CS = 0x9
QUADSPI.LUT[10].B.PAD0 = 0;
QUADSPI.LUT[10].B.OPRND0 = 0;
```

```
QUADSPI.LUT[10].B.INSTR1 = 0;
QUADSPI.LUT[10].B.PAD1= 0;
QUADSPI.LUT[10].B.OPRND1 = 0;

//Lock the LUT
QUADSPI.LUTKEY.R = 0x5AF05AF0;
QUADSPI.LCKCR.R = 0x1;
while(!QUADSPI.LCKCR.B.LOCK);
```

Now point the programmable sequence engine to a particular sequence. Not that writing IDATSZ at the same time as writing the SEQID in the IPCR, or passing a variable nbytes in this example, will override the number of bytes programmed in the LUT:

```
QUADSPI.IPCR.R = ((RESET_SEQUENCE<<24)+nbytes);     //RESET_SEQUENCE = 0x0 (Seq ID0)
```

4.2 Parallel mode

Initially the software needs to set the following bits, depending on the type of access:

```
QSPI_BUFxCR[PAR_EN] //AHB driven mode
QSPI_IPCR[PAR_EN] //IP driven mode.
```

Next the address registers for each of the flash memory devices need to be set:

```
QSPI_AMBA_BASE - 0x10000000
QSPI_SFA1AD[TPADA1] - 0x20000000
QSPI_SFA2AD[TPADA2] - 0x30000000
QSPI_SFB1AD[TPADB1] - 0x40000000
QSPI_SFB2AD[TPADB2] - 0x50000000
```

In order to access the first location of A1/B1 pair, the Serial Flash Address Register (QSPI_SFAR) should be programmed with 0x20000000.

This address is divided by two and QSPI_AMBA_BASE is subtracted. Therefore, address provided to flash A1 and B1 Flash Address = ((0x20000000/2) – QSPI_AMBA_BASE) = 0x00000000.

Similarly, in order to access the first location of A2/B2 pair, the Serial Flash Address Register (QSPI_SFAR) should be programmed with 0x60000000.

So, Flash Address = ((0x60000000/2) – QSPI_AMBA_BASE) = 0x20000000.

One this has been configured, the application software should perform a single, dual, or quad IO read:

```
//Single IO Read (parallel)
QUADSPI.IPCR.R = ((READ_1IO_SDR<<24)+ (par_en<<16) + nbytes);
//Dual IO Read (parallel)
QUADSPI.IPCR.R = ((READ_2IO_SDR<<24)+ (par_en<<16) + nbytes);
//Quad IO Read (parallel)
QUADSPI.IPCR.R = ((READ_4IO_SDR<<24)+ (par_en<<16) + nbytes);
```

Where the LUT is programmed as following for each of these commands. Note that key entry descriptions are shown in the proceeding comments:

```
// Readfrom flash in Single SDR Mode
// SEQID 2
QUADSPI.LUT[8].B.INSTR0 =  CMD;
QUADSPI.LUT[8].B.PAD0 = 0;
QUADSPI.LUT[8].B.OPRND0 = MXIC_SIODRD;// MXIC_SIODRD = 0x0B
QUADSPI.LUT[8].B.INSTR1 = FL_ADDR;
QUADSPI.LUT[8].B.PAD1= 0;//1 pad (single IO)
QUADSPI.LUT[8].B.OPRND1 = MXIC_ADDR;

QUADSPI.LUT[9].B.INSTR0 =  DUMMY;
QUADSPI.LUT[9].B.PAD0 = 0;
```

```
QUADSPI.LUT[9].B.OPRND0 = MXIC_DUMMY8; //8 dummy cycles
QUADSPI.LUT[9].B.INSTR1 = READ;
QUADSPI.LUT[9].B.PAD1= 0;
QUADSPI.LUT[9].B.OPRND1 = nbytes;//Number of Bytes

QUADSPI.LUT[10].B.INSTR0 =  JMP_ON_CS;
QUADSPI.LUT[10].B.PAD0 = 0;
QUADSPI.LUT[10].B.OPRND0 = 0;
QUADSPI.LUT[10].B.INSTR1 = 0;
QUADSPI.LUT[10].B.PAD1= 0;
QUADSPI.LUT[10].B.OPRND1 = 0;

// Read from flash in Dual SDR Mode
// SEQID 3
QUADSPI.LUT[12].B.INSTR0 =  CMD;
QUADSPI.LUT[12].B.PAD0 = 0x0;
QUADSPI.LUT[12].B.OPRND0 = MXIC_DIODRD;// MXIC_ DIODRD = 0xBB
QUADSPI.LUT[12].B.INSTR1 = FL_ADDR;
QUADSPI.LUT[12].B.PAD1= 0x1;//2 pads (dual IO)
QUADSPI.LUT[12].B.OPRND1 = MXIC_ADDR;

QUADSPI.LUT[13].B.INSTR0 =  DUMMY;
QUADSPI.LUT[13].B.PAD0 = 0x0;
QUADSPI.LUT[13].B.OPRND0 = MXIC_DUMMY4; //4 dummy cycles
QUADSPI.LUT[13].B.INSTR1 = READ;
QUADSPI.LUT[13].B.PAD1= 0x1;
QUADSPI.LUT[13].B.OPRND1 = nbytes;//Number of Bytes

QUADSPI.LUT[14].B.INSTR0 = JMP_ON_CS;
QUADSPI.LUT[14].B.PAD0 = 0;
QUADSPI.LUT[14].B.OPRND0 = 0;
QUADSPI.LUT[14].B.INSTR1 = 0;
QUADSPI.LUT[14].B.PAD1= 0;
QUADSPI.LUT[14].B.OPRND1 = 0;

// Read from Flash in Quad SDR mode
// SEQID 4
QUADSPI.LUT[16].B.INSTR0 =  CMD;
QUADSPI.LUT[16].B.PAD0 = 0;
QUADSPI.LUT[16].B.OPRND0 = MXIC_QIODRD; // MXIC_ DIODRD = 0xEB
QUADSPI.LUT[16].B.INSTR1 = FL_ADDR;
QUADSPI.LUT[16].B.PAD1= 0x2; //4 pads (quad IO)
QUADSPI.LUT[16].B.OPRND1 = MXIC_ADDR;

QUADSPI.LUT[17].B.INSTR0 = FL_MODE;
QUADSPI.LUT[17].B.PAD0 = 0x2;
QUADSPI.LUT[17].B.OPRND0 = 0x00;
QUADSPI.LUT[17].B.INSTR1 = DUMMY;
QUADSPI.LUT[17].B.PAD1= 0;
QUADSPI.LUT[17].B.OPRND1 = MXIC_DUMMY4;//4 dummy cycles

QUADSPI.LUT[18].B.INSTR0 = READ ;
QUADSPI.LUT[18].B.PAD0 = 0x2;
QUADSPI.LUT[18].B.OPRND0 = nbytes;//Number of Bytes
QUADSPI.LUT[18].B.INSTR1 = JMP_ON_CS;
QUADSPI.LUT[18].B.PAD1= 0;
QUADSPI.LUT[18].B.OPRND1 = 0;
```

5 References

- QuadSPI chapter of relevant Freescale chip reference manual
- Freescale document VybridRM, *Vybrid Reference Manual*
- Freescale document AN4186, "Using the QuadSPI Module on MPC56xxS"
- Macronix document PM1557: *MX25L6465E/MX25L12865E High Performance Serial Flash Specification,* Rev. 1.4

6 Glossary

Here is a list of terms and definitions used in this document:

SFM	Serial Flash Module
LUT	Look-Up Table
MCU	Microcontroller Unit
MB	MegaByte = 1024 Bytes
GB	GigaByte = 1024 MB
DDR / DTR	Dual Data Rate / Double Transfer Rate
DQS	Data Strobe
I/O	Input / Output
AHB	AMBA High-performance Bus
IP	Intellectual Property
ADDR	Address
DMA	Direct Memory Access
RX	Receive
TX	Transmit
IRQ	Interrupt Request

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

Document Number: AN4512
Rev. 0, May 2012