Using the SMIF in XMC7000 MCU family

About this document

Scope and purpose

This application note describes the procedure to handle the Serial Memory Interface (SMIF) for XMC7000 family MCUs.

Intended audience

This document is intended for anyone using XMC7000 family XMC7100/7200 series MCUs.

Associated part family

XMC7000 family XMC7100/XMC7200 series of **XMC™ industrial microcontrollers**.

Table of contents

Table of contents

1 Introduction

This application note describes how to provide a low-pin-count connection to off-chip SPI devices for XMC7000 family MCUs. This application note uses the S25FL256S SPI flash memory as an example and describes the use cases for single data rate (SDR) and dual data rate (DDR) in the Dual-Quad SPI mode. See the **architecture technical reference manual (TRM)** for details of single/Dual/Quad/Octal SPI protocols.

To know more about the terminology used in this application note, see **Glossary**.

2 General description

2.1 Features

SMIF has the following features:

- SPI or HYPERBUS™ master functionality
- HYPERBUS™ protocol
  - HYPERFLASH™
  - HYPERRAM™
- SPI protocol
  - SPI mode 0 only, with configurable MISO sampling timing
  - Supports single, Dual, Quad, and Octal SPI
  - Supports Dual-Quad SPI mode
  - Supports single data rate (SDR) and dual data rate (DDR) transfers
- Memory device
  - Supports overall device capacity in the range of 64 KB to 4 GB in power of two multiples
  - Supports configurable external device capacities
  - Supports two external memory devices
- Memory-mapped I/O (MMIO) operation mode
- eXecute-In-Place (XIP) mode
  - XIP operation mode for both read and write accesses
  - XIP mode supports on-the-fly encryption and decryption
  - XIP operation mode via the AHB interface for Arm® Cortex®-M0 and AXI interface for Cortex® M7 core
  - Supports up to four outstanding transactions
- Memory interface logic
  - Supports stalling of SPI and HYPERBUS™ transfers to address the back pressure on FIFOs
  - Supports an asynchronous SPI/HYPERBUS™ transmit and receive interface clock
  - Supports Read-Write Data Strobe (RWDS)
  - Supports multiple interface receive clocks
  - Supports flexible data signal connections with external SPI memory devices such as single, Dual, Quad, or Octal mode
  - Independent SPI interface transmitter clock from PLL/FLL

2.2 Block diagram

SMIF allows a low-pin-count connection to external devices. **Figure 1** gives a high-level overview of SMIF in XMC7200 series.

The bottom part of **Figure 1** shows the SPI interface signal connections to the I/O subsystem (IOSS). The section of **Figure 1** highlighted in yellow shows the AXI slave interface and the AHB-Lite slave interface. The XIP AHB-Lite interface has a dedicated cache. AHB-Lite transfers to the XIP address space either access the cache or are translated on-the-fly into SPI transfers to the external device. The SMIF supports an address space located at the XMC7000 address 0x6000:0000. The location of the external devices in the XIP address space is programmable.

For Dual-Quad SPI mode, it is required to program the same MMIO device register values for the two external devices that are connected in parallel to the SMIF I/O signal interface.

AHB-Lite transfers data to the MMIO address space by accessing the MMIO registers. The MMIO registers include registers to access the FIFOs. While the XIP address space supports highly efficient read and write access to external devices (through on-the fly translation of AHB-Lite transfers into SPI transfers), the MMIO address space provides flexibility in the construction of SPI transfers.

SMIF has two Tx FIFOs and one Rx FIFO. These FIFOs provide an asynchronous clock domain transfer between CLK_mem logic and CLK_if_tx/CLK_if_rx memory interface logic. The memory interface logic is completely controlled through the Tx and Rx FIFOs. Additionally, SMIF has an Rx data MMIO FIFO, which is used only in MMIO mode and is logically an extension of the Rx data FIFO enabling an easy-to-use Rx data handling in software.

SMIF has a single interrupt line.

In XIP mode, the cryptography component supports on-the-fly encryption for write data and on-the-fly decryption for read data. The use of on-the-fly cryptography is determined by a device's CRYPTO_EN bit field in the MMIO CTL register. In MMIO mode, the cryptography component is accessible through a MMIO register interface to support offline encryption and decryption.

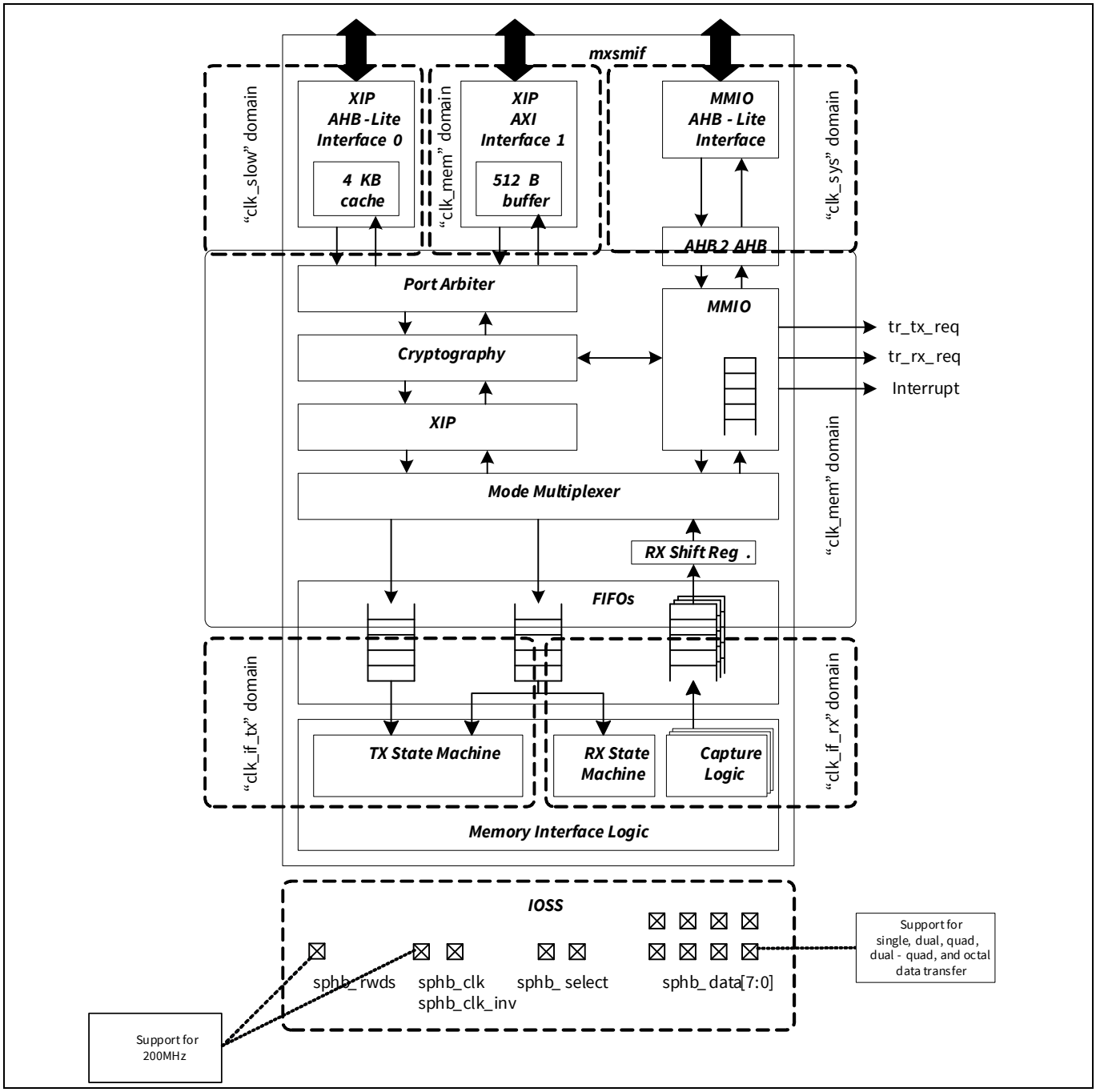

Figure 1 XMC7200 series high-level block diagram

For the interrupt architecture of other series, see the **architecture TRM**.

2.3 Commands for Tx command FIFO

In this application note, various commands are used. The following FIFOs are used to transmit and receive contents of these commands in SMIF:

- Tx command FIFO: Transmits the commands for Quad SPI and HYPERBUS™ to the memory interface logic. The FIFO consists of eight 27-bit entries. Each entry holds a command. The FIFO is controlled by the SMIF_TX_CMD_FIFO_WR.DATA27[26:0] register. DATA27[26:24] specifies the command and DATA [23:0] sets the command specification depending on command type.
- Tx data FIFO: Transmits the write data to the memory interface logic
- Rx data FIFO: Receives the read data from the memory interface logic

The sequence for commands of Quad SPI is classified by phase. For the phase, it is for 1 byte of instruction, 4 bytes of address, 1 byte of mode, dummy cycle decided in the Quad memory, and receive data. A command in the Tx command FIFO specifies the phase of the sequence. **Table 1** explains five types of commands that the Tx command FIFO supports. See the **architecture TRM** and **registers TRM** for details of five types of commands.

Table 1 Five types of commands for the Tx command FIFO

Command	DATA27[26:24]	Specification
TX	0	This command specifies the phase, such as Instruction, Address, and Mode for commands of Quad SPI.
TX_COUNT	1	This command is used when data is transmitted from the Tx data FIFO to external memories. This command specifies the number of memory data units to be transmitted.
RX_COUNT	2	This command is used when data is received from external memories to the Rx data FIFO. This command specifies the number of memory data units to be received.
DUMMY_COUNT	3	This command specifies the number of dummy cycles.
DESELECT	4	This command causes the memory interface transmit logic to finish a transfer and deselect the memory device.

General description

Figure 2 explains how to use the Tx command FIFO for the Quad SPI read command.

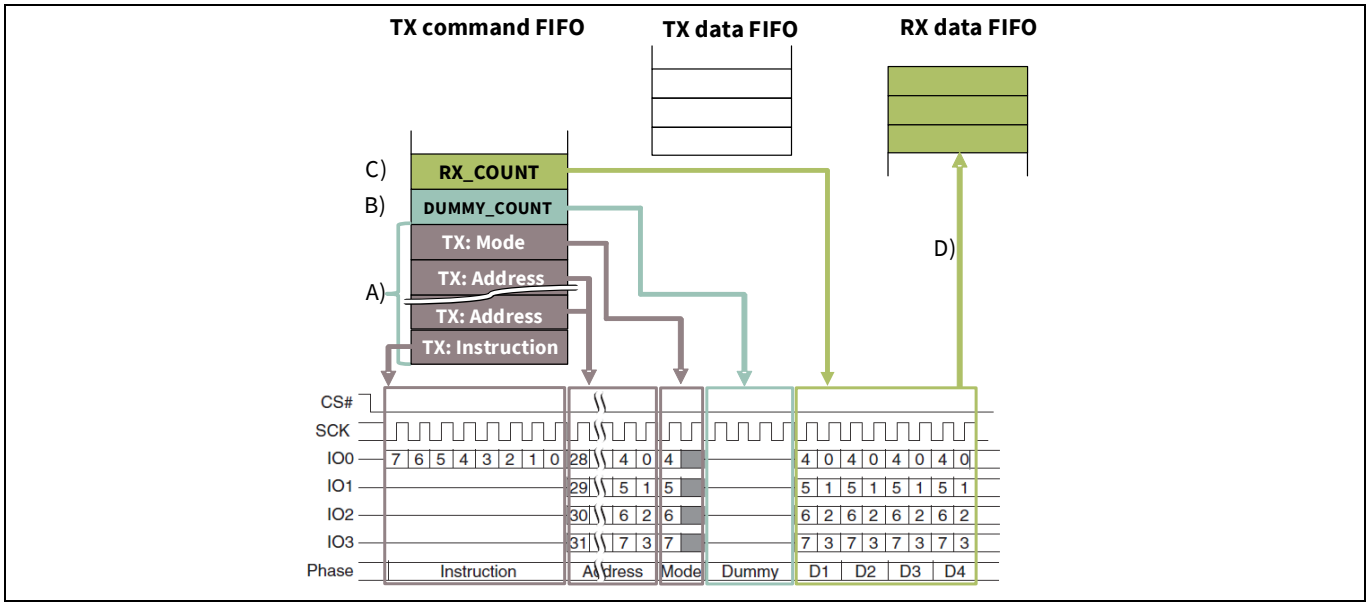

Figure 2 Constructing a Quad SPI read command

The Tx command FIFO setting of the Quad SPI read command is explained as follows:

A) Set entries to the Tx command FIFO in the following order: Instruction, Address, Mode using the 'Tx' command.

B) Set the dummy cycles to the Tx command FIFO using the 'DUMMY_COUNT' command.

C) Set the number of receive data to the Tx command FIFO using the 'RX_COUNT' command.

D) Rx data FIFO receives the number of data sent.

Figure 3 explains how to use the Tx command FIFO for the Quad SPI program command.

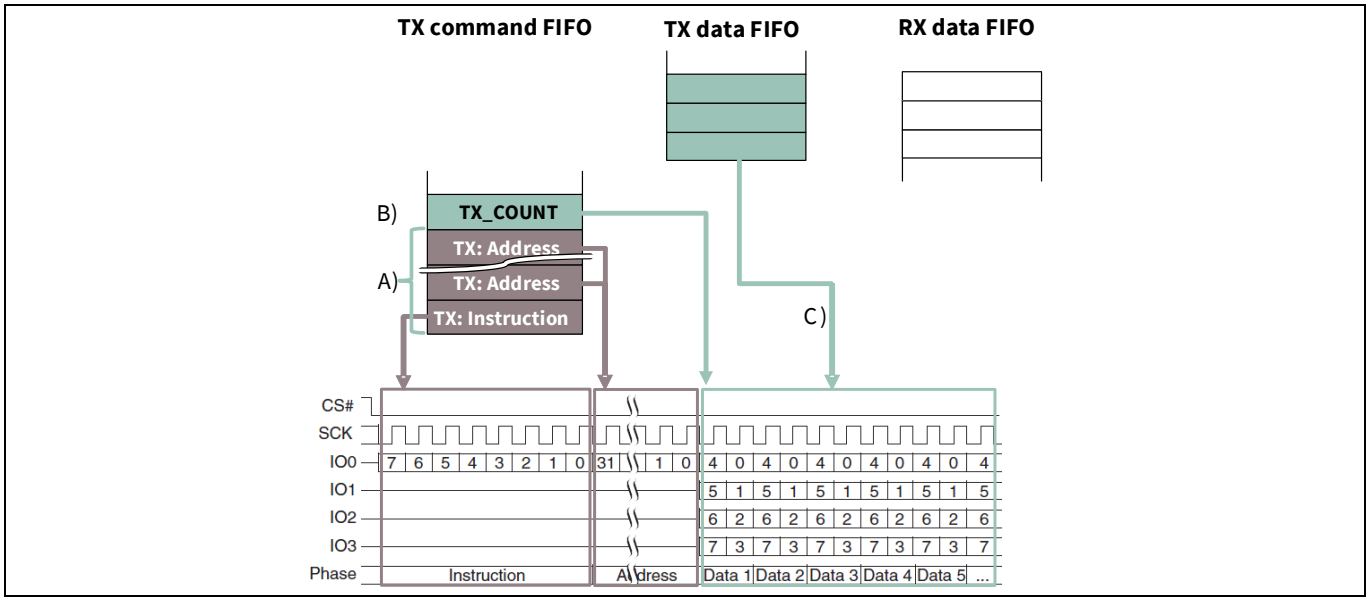

Figure 3 Constructing a program command of Quad SPI

General description

The Tx command FIFO setting of program command of Quad SPI is explained as follows:

A) Set entries to the Tx command FIFO in the order Instruction and Address using the 'Tx' command.

B) Set the number of transmission data to the Tx command FIFO using the 'TX_COUNT' command.

C) SMIF transmits the number of data set from the Tx data FIFO.

Table 6 lists the details of the Tx command FIFO entry for the Quad Page Program command (4QPP 34h).

2.4 Data capture scheme

SMIF supports the following data captures:

- Output/feedback clock-based capture
- Internal clock-based capture
- RWDS-based capture
- Delay line and data learning pattern-based capture

The following is an overview of each capture scheme. See the **architecture TRM** for more details.

2.4.1 Output/feedback clock-based capture

This capture scheme captures data with the SMIF output or output feedback clock for SDR and DDR timing. It uses the memory output clock (spihb_clk_out), inverted memory output clock, memory output feedback clock (spihb_clk_in), or inverted memory output feedback clock as the capture clock. This scheme has the delay line for delaying the output or feedback to adjust the sample time with a finer granularity. The clock can be selected by CLOCK_IF_RX_SEL[3:0] in the SMIFx_CTL register.

2.4.2 Internal clock-based capture

This capture scheme uses the interface clock (clk_if) or the inverted interface clock as the capture clock. This scheme is always available in the source of the capture clock in the internal clock compared to the output/feedback clock-based capture scheme. The clock can be selected by CLOCK_IF_RX_SEL[3:0] in the SMIFx_CTL register.

2.4.3 RWDS-based capture

In this capture scheme, the RWDS signal is used as a clock to capture the input data. It has the delay line for delaying the output or feedback to adjust the sample time with a finer granularity. The clock can be selected by CLOCK_IF_RX_SEL[3:0] in the SMIFx_CTL register.

2.4.4 Delay line and data learning pattern (DLP)-based capture

This capture scheme uses the internal clock (clk_if or inverted clk_if) as a clock to capture the input data. This capture scheme provides a delay line to adjust the capture timing precisely.

The selection of the delay line tap can be done by software or automatically in hardware via data learning. The data learning scheme finds the best delay line tap in hardware for each data input line by comparing the captured data learning pattern with the expected one. The input data is received after the data learning pattern.

In this scheme, the software should initiate a memory read transaction in MMIO mode.

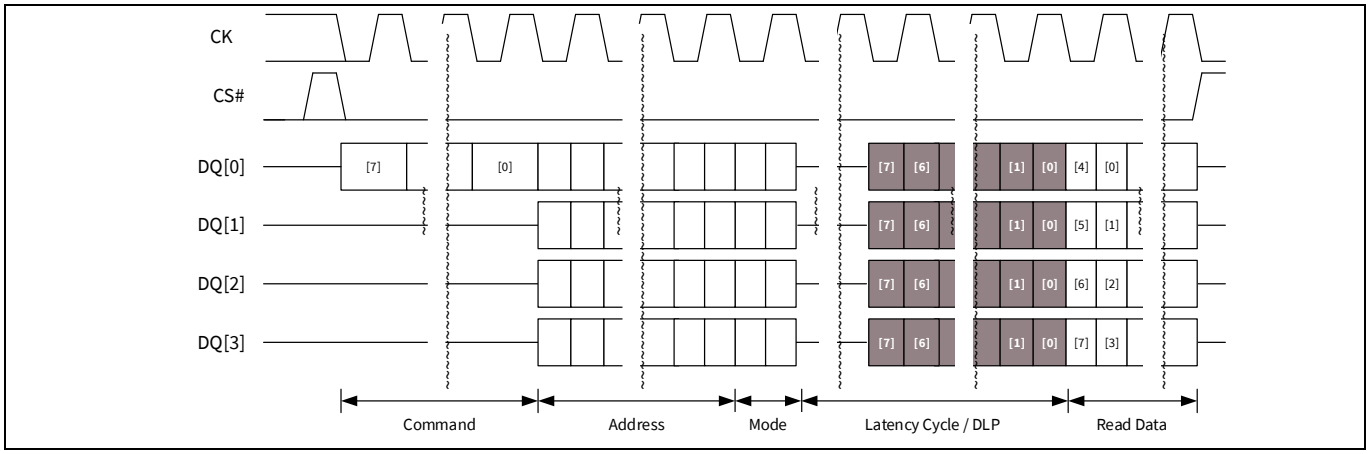

Figure 4 DLP output image

The memory device provides a known data pattern (the data learning pattern) on every data I/O pin within the read latency cycles before the requested read data is provided.

3 Example of Dual-Quad SPI DDR mode

This section explains an application to write the data (SDR) and read the data (DDR). Cryptography with the MMIO mode is used to write the data. XIP mode is used to read the data. Dual-Quad SPI mode is also discussed for the connection method of the application.

3.1 Connecting memory devices

In this diagram, Dual-Quad SPI mode is used as an example. **Figure 5** illustrates memory devices 0 and 1; both are Quad SPI memories. Each device uses dedicated data signal connections. The devices' address regions in the XMC7000 address space are the same to ensure that the activation of spi_select[0] and spi_select[1] are the same. This is known as a Dual-Quad configuration: during SPI read and write transfers, each device provides a nibble of a byte.

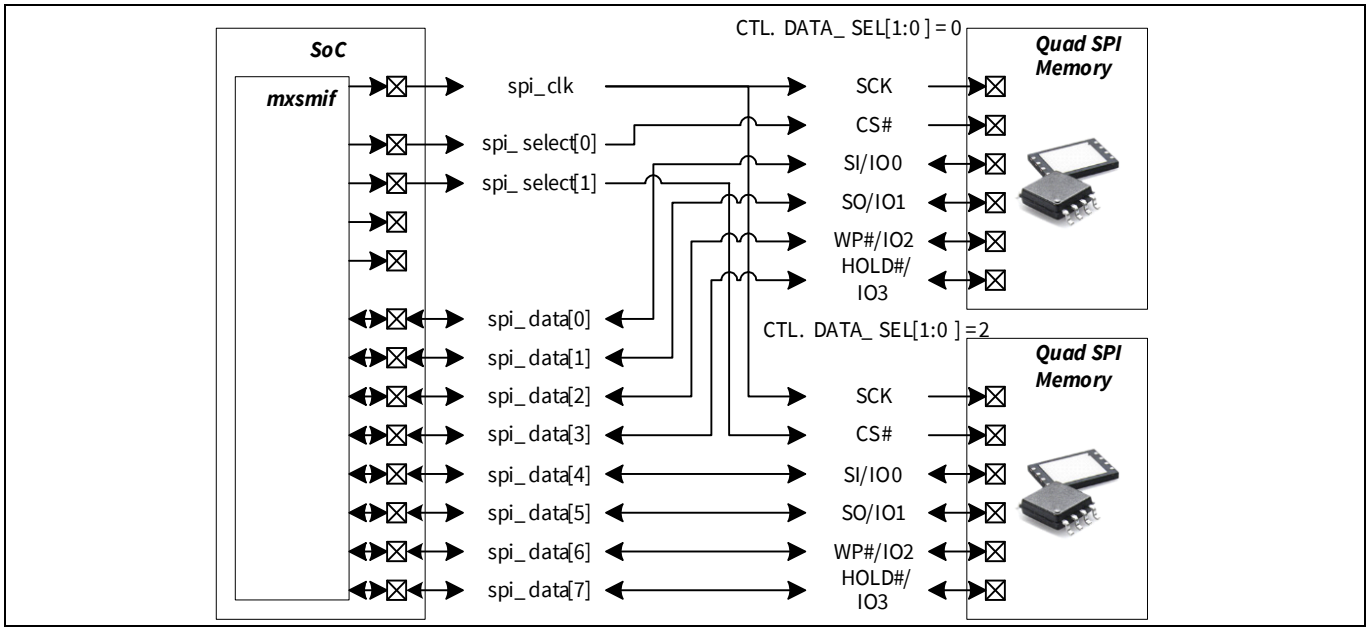

Figure 5 Dual-Quad configuration

3.2 Using S25FL256 device in Dual-Quad mode

In this use case, erase, write (SDR), and read (DDR) are checked with the Quad SPI flash memory S25FL256S. **Figure 6** shows the flow of commands of the memory. This flow writes 16 bytes of data into the devices using Dual-Quad SPI mode in MMIO mode with cryptography. 16 bytes of data being written is random data. In the case of read, this flow sets the mode in the devices for Quad I/O read (4QIOR ECh) with MMIO mode and read transfer with decrypting on-the-fly in XIP mode. All commands of Quad SPI in **Figure 6** are Dual-Quad mode.

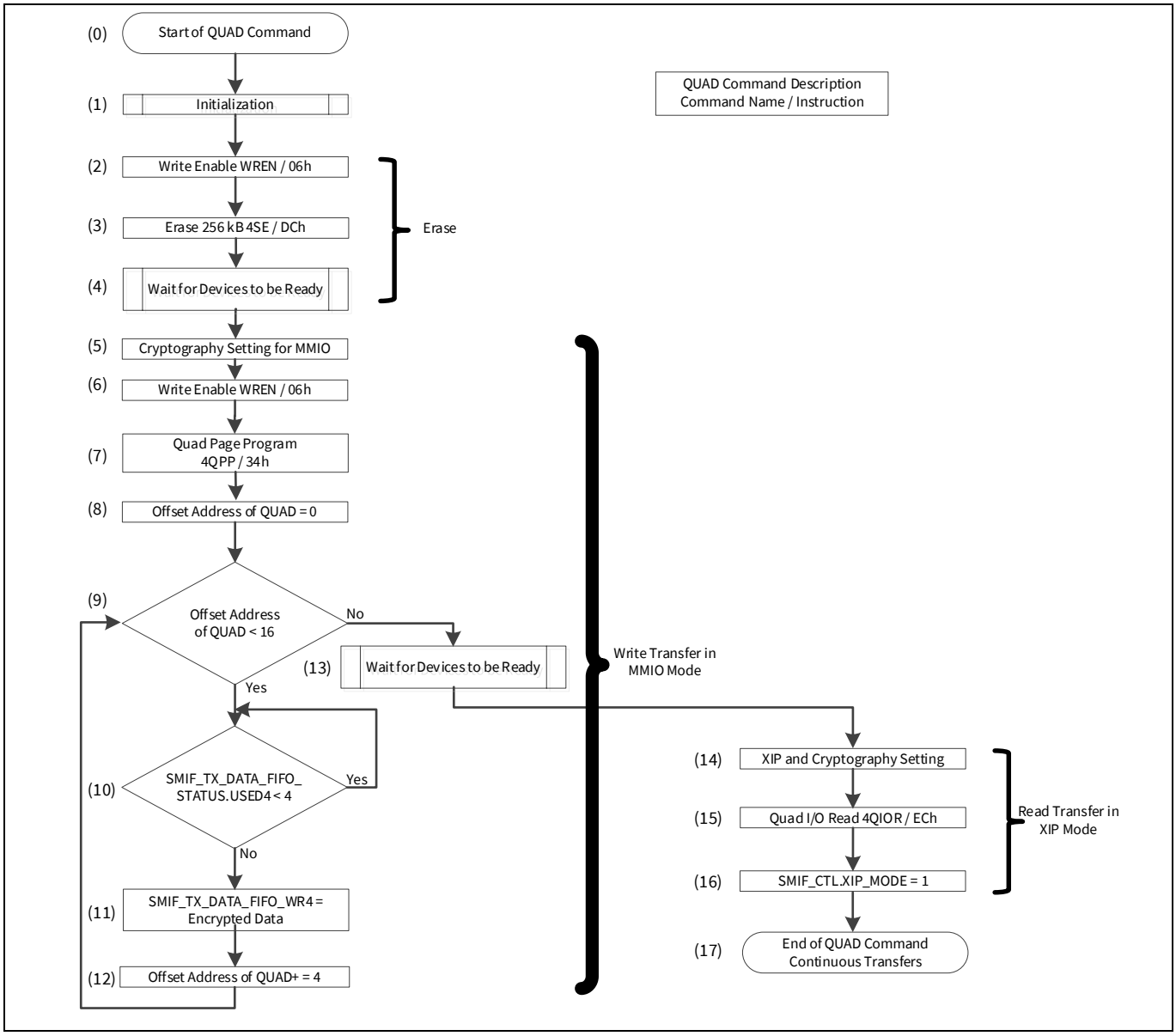

Figure 6 Example of software flowchart using Quad SPI commands

Example of Dual-Quad SPI DDR mode

The example of software flowchart is explained as follows:

(0) Dual-Quad SPI mode is set for the example.

(1) Initialize for SMIF and Dual-Quad devices. See **3.3 Initialization**.
From here to (4) is part of the erase operation using the Quad 4SE command.

(2) Transmit the WREN command for the 4SE command. See **3.4 Erase**.

(3) Transmit the 4SE command to set all bits in the addressed sector to 1. See **Table 5**.

(4) Wait for devices to be ready. See **Figure 9**.
From here to (12) is part of the program operation using Quad 4QPP command.

(5) Set the cryptography setting for MMIO. See **Figure 13**.

(6) Transmit the WREN command for the 4QPP command.

(7) Transmit the 4QPP command to program. See **Table 6**.
Address of a sector of each Quad SPI device used in the example: 0x01240000
Size: 16 bytes

(8) Initialize the offset address of the 'for' loop.

(9) Check the quantity of encrypted data with the offset address.
If Offset address of Quad <16, go to (10).
Otherwise, go to (13).

(10) Wait until the SMIF_TX_DATA_FIFO_STATUS.USED4 bits are greater than or equal to 5.

(11) Set SMIF_TX_DATA_FIFO_WR4 to encrypted data.

(12) Add 4 to the offset address and goes to (9).

(13) Wait for devices to be ready.

(14) Set the XIP setting and cryptography setting for XIP. See **3.6 Read transfer in XIP mode**.

(15) Transmit the 4QIOR command to set Quad I/O high-performance read mode.

(16) Set SMIF_CTL.XIP_MODE to '1' (XIP mode).

(17) The memory remains in Quad I/O high-performance read mode and the next address can be entered (after CS# is raised HIGH and then asserted LOW) without requiring the instruction. After this, hardware automatically generates memory read transfers for AHB-Lite or AXI read transfers.

3.3 Initialization

In this scenario, SMIF and the devices are initialized as indicated by (1) in **Figure 6**. **Figure 7** illustrates the flow of initialization. In the flowchart, SMIF is initialized and reset is sent to devices for Dual-Quad.

The configuration registers of devices are confirmed with the RDCR (Read Configuration Register) command as (5) and (6) in **Figure 7**. If 4-bit-wide Quad I/O is not set in the devices, QUAD bit will be set as '1'.

Quad SPI commands without the device specification (for example: (2)) are Dual-Quad mode. When multiple bits are set for 'Device select' bits of 'SMIF_TX_CMD_FIFO_WR', for example, 0x3 (**Table 2**, **Table 5**), the command is in Dual-Quad mode. Quad SPI commands with the device specification (for example, (5)) are Quad mode. When only one bit is set for 'Device select', for example, 0x1 or 0x2 (**Table 3**, **Table 4**), the command is in Quad mode.

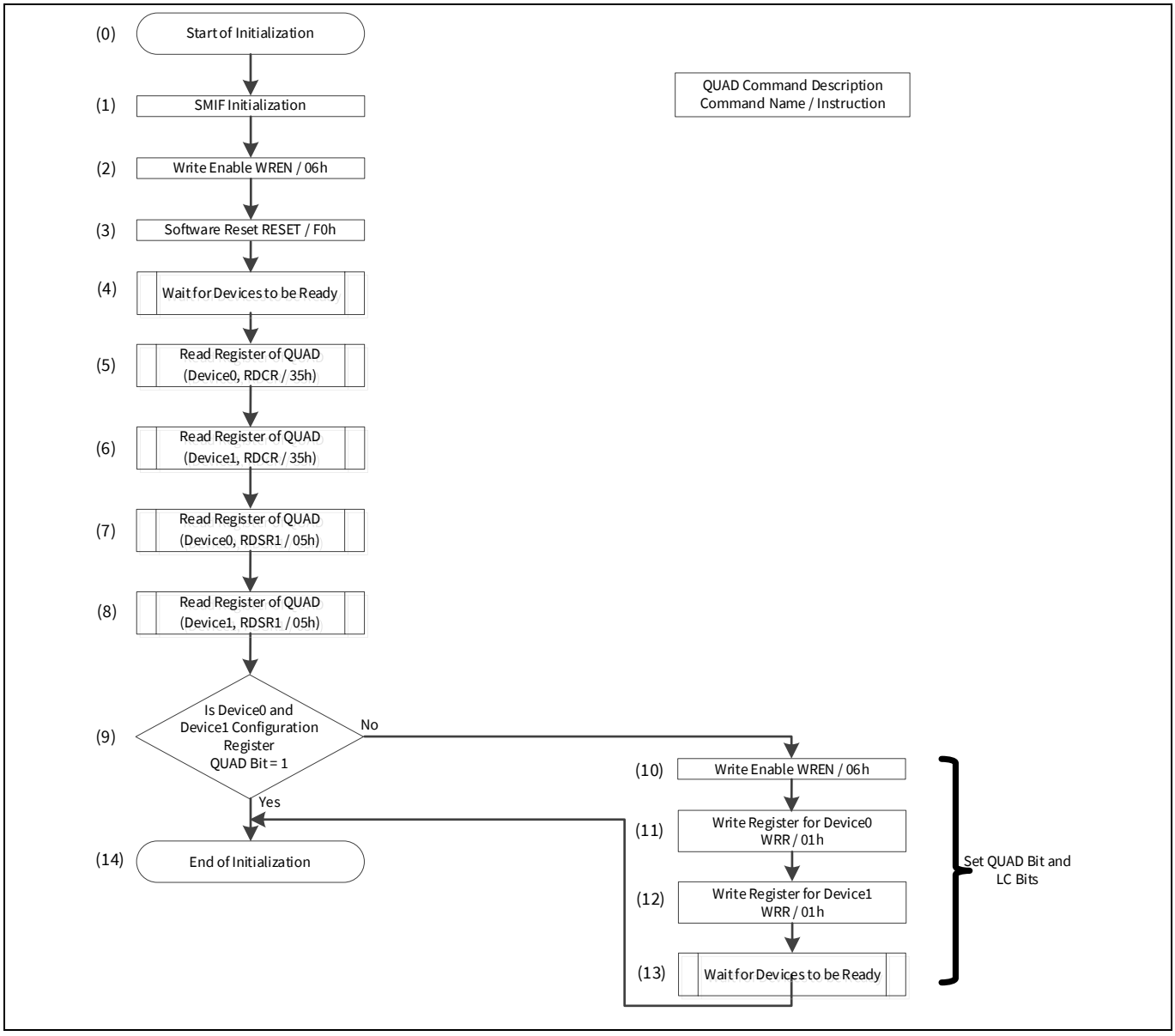

Figure 7 Software flowchart for initialization

Example of Dual-Quad SPI DDR mode

The software flow for initialization is explained as follows:

(0) Set the initialization for SMIF and the Dual-Quad SPI devices.

(1) Initialize SMIF to send a Quad command. See **3.3.1 Initialization for SMIF**.

(2) Transmit the WREN command to allow the WRR command to execute afterwards. See **Table 2**.

(3) Transmit the RESET command to reset the memory device to await any new command. See **Table 2**.

(4) Wait for devices to be ready. See **Figure 9**.

(5) Transmit the RDCR (Read Configuration Register) command to check the Quad bit in device0. See **Table 3**.

(6) Transmit the RDCR (Read Configuration Register) command to check the Quad bit in device1.

(7) Transmit the RDSR1 (Read Status Register-1) command to check the BP bits in device0. See **Table 3**.

(8) Transmit the RDSR1 (Read Status Register-1) command to check the BP bits in device1.

(9) When the Quad bit is set to '1', this bit switches the data width of the device to 4 bits - Quad mode.
If both QUAD bit are not '1', go to (10).
If both QUAD bit are '1', go to (14).

(10) Transmit the WREN command to allow the WRR command to execute afterwards. See **Table 2**.

(11) Transmit the WRR command to set the Quad bit and LC bits in device0. See **Table 4**.

(12) Transmit the WRR command to set the Quad bit and LC bits in device1.

(13) Wait for devices to be ready.

(14) Initialization of SMIF and devices of Dual-Quad is completed.

3.3.1 Initialization for SMIF

SMIF is set for initialization as indicated by (1) in **Figure 7**.

Devices 0 and 1 are used to implement Dual-Quad SPI mode. In the SMIF0_DEVICE chapter of the **registers TRM**, devices 0 and 1 are defined as SMIF0_DEVICE0 and SMIF0_DEVICE1. SMIF0_DEVICE0 and SMIF0_DEVICE1 have a SMIF_DEVICE_CTL register each. In this application note, registers are shown as SMIF_DEVICE0_CTL and SMIF_DEVICE1_CTL. If both SMIF0_DEVICE0 and SMIF0_DEVICE1 are set, the register is shown like SMIF_DEVICEx_CTL.

The procedure to configure SMIF is as follows:

1. Configure the clocking system.[1]
2. Configure the port.
3. Initialize the SMIF block as a communication block:
   - Set SMIF_INTR_MASK.TR_TX_REQ to '0' (Disabled: Configure the initial interrupt mask)
   - Set SMIF_INTR_MASK.TR_RX_REQ to '0' (Disabled: Configure the initial interrupt mask)
   - Set SMIF_CTL.CLOCK_IF_TX_SEL to '1' (DDR)
   - Set SMIF_CTL.CLOCK_IF_RX_SEL to '1' (SMIF output inverted clock for DDR or SDR capturing)
   - Set SMIF_CTL.DELAY_TAP_ENABLED to '0' (Registers DELAY_TAP_SEL or INT_CLOCK_DELAY_TAP_SEL0/1 are not used)

[1] See the related chapter of **architecture TRM** for details for setting.

Example of Dual-Quad SPI DDR mode

- Set SMIF_CTL.INT_CLOCK_DL_ENABLED to '1' (Enabled: The delay line tap selections are modified by HW based on the data learning pattern)
- Set SMIF_CTL.XIP_MODE to '0' (MMIO mode)

4. Configure SMIF Device 0/1:
   - Set SMIF_DEVICE0_CTL.DATA_SEL to '0' (spi_data[0] = IO0, spi_data[1] = IO1, …, spi_data[7] = IO7.
   - This value is allowed for single, Dual, Quad, Dual-Quad and Octal SPI modes. This value must be used for the first device in Dual-Quad SPI mode.)
   - Set SMIF_DEVICE0_CTL.WR_EN to '1' (Write transfers are allowed to this device.)
   - Set SMIF_DEVICE1_CTL.DATA_SEL to '2' (spi_data[4] = IO0, spi_data[5] = IO1, …, spi_data[7] = IO3. This value is only allowed for single, Dual, Quad and Dual-Quad SPI modes. In Dual-Quad SPI mode, this value must be used for the second device.)
   - Set SMIF_DEVICE1_CTL.WR_EN to '1' (Write transfers are allowed to this device.)
5. Enable SMIF:
   - Set SMIF_CTL.ENABLED to '1' (Enabled).
   - Read SMIF_CTL (Read the register to flush the buffer).

3.3.2 Reset for QUAD

The RESET command is sent to reset devices for Dual-Quad as indicated by (3) in **Figure 7**. Before sending the RESET command, it is necessary to send the Quad WREN command. After sending the WREN command, the device will set the Write Enable Latch (WEL) in the Status Register to enable any write operations. RESET and WREN are stand-alone instruction commands that consist of only the instruction as shown in **Figure 8**. See the **datasheet of S25FL256S** for details of the WRR command.

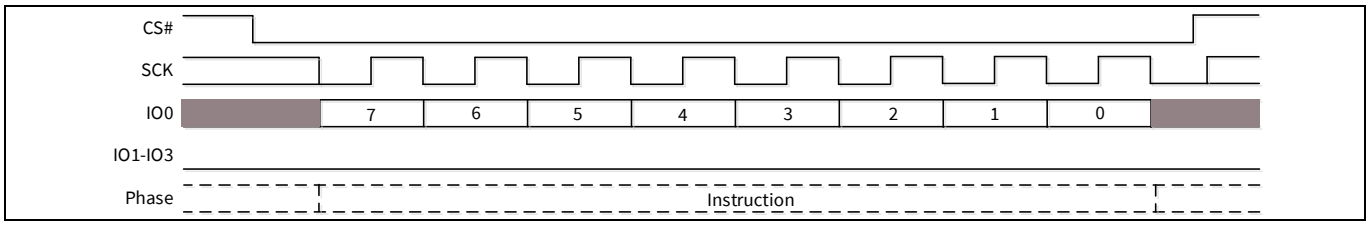

Figure 8 Standalone instruction command

Table 2 explains the Tx command FIFO entry for the standalone instruction command. Both spi_select[0] and spi_select[1] are selected because of Dual-Quad SPI mode.

Table 2 SMIF_TX_CMD_FIFO_WR setting for standalone instruction command

bits	26:24	23	22	21	20	19	18	17:16	15:18 [2]	7:0	SMIF_TX_CMD_FIFO_WR
Entry	Command	Device Select				Last TX	SDR/DDR	Width of the data transfer		Transmitted Byte	
1	0 (Instruction)	3				1	0	0	0	8-bit instruction	0x038_00xx

[2] In case of "Tx" command, DATA[15:8] specifies the second transmitted byte. This is used only (and must be specified) for Octal data transfer with DDR mode, i.e., when DATA[17:16] = "3" and DATA[18] = "1".

Example of Dual-Quad SPI DDR mode

In the case of the RESET command, set the 8-bit instruction to ‘0xF0’. In the case of the WREN command, set the 8-bit instruction specified in **Table 2** to ‘0x01’.

Step (4) in **Figure 7** checks the WIP bit in the devices to confirm the completion of the RESET command.

Figure 9 illustrates the flow of ‘Wait for Devices to be Ready’.

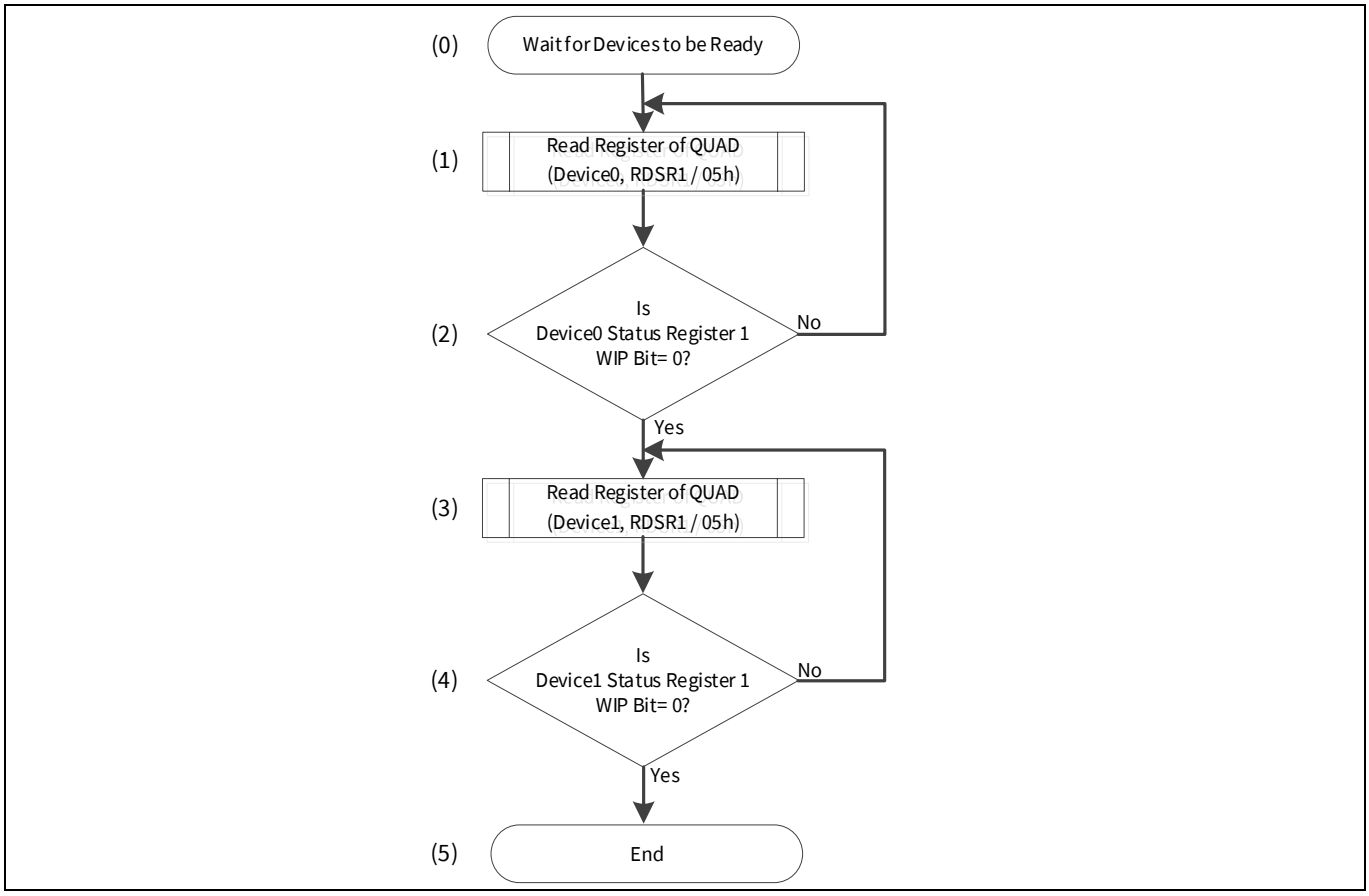

Figure 9 Software flowchart for wait for devices to be ready

The software flow for Wait for Devices to be Ready is explained as follows:

(0) Start of flowchart for ‘Wait for Devices to be Ready’.

(1) Transmit the RDSR1 (Read Status Register-1) command to check the WIP bit in device0. See **Table 3**.

(2) When the WIP bit is set to ‘1’, the device is busy.
If WIP bit is not ‘0’, go to (1).
If WIP bit is ‘0’, go to (3).

(3) Transmit the RDSR1 (Read Status Register-1) command to check the WIP bit in device1.

(4) When the WIP bit is set to ‘1’, the device is busy.
If WIP bit is not ‘0;’, go to (3).
If WIP bit is ‘0’, go to (5).

(5) End of flowchart for ‘Wait for Devices to be Ready’.

3.3.3 Register reading for QUAD

The ‘Read the Quad register’ function is used in both **Figure 7** and **Figure 9**. This function gets the value of the Quad register set with an argument. **Figure 10** is flow chart for ‘Read Quad register’.

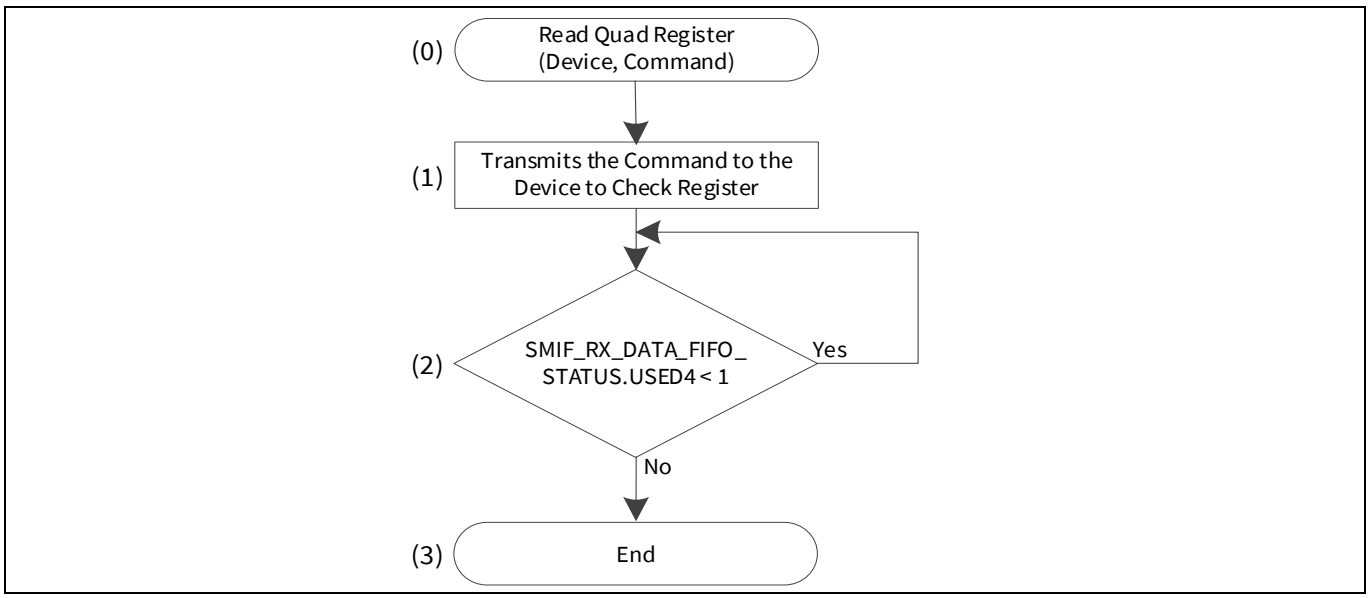

Figure 10 Software flowchart to read the Quad register

(0) Read the Quad register. The two arguments include Device and Command.

(1) Transmit the Command argument to the set device.

(2) Waits until the SMIF_RX_DATA_MMIO_FIFO_STATUS.USED4 bits are greater than or equal to 1.

(3) End. When it is necessary to read only 1 byte from FIFO, check SMIF_RX_DATA_MMIO_FIFO_RD1 register.

RDCR and RDSR1 commands are read register command sequences as shown in **Figure 11**.

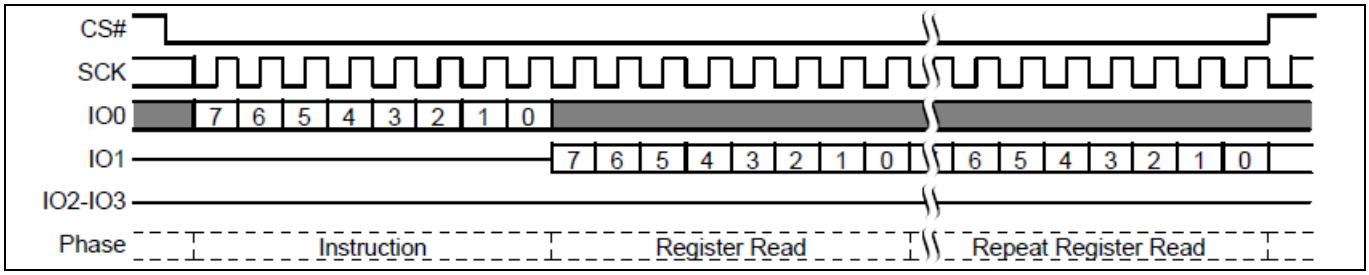

Figure 11 Read register command sequence

Example of Dual-Quad SPI DDR mode

Table 3 explains Tx command FIFO entry for read register command sequence. In this application note, SMIF_TX_CMD_FIFO_WR setting of spi_select[0] and spi_select[1] are set separately for reading the Quad flash register.

Table 3 SMIF_TX_CMD_FIFO_WR setting for read register command sequence

bits	26:24	23	22	21	20	19	18	17:16	15:18 [3]	7:0	SMIF_TX_CMD_FIFO_WR
Entry	Command	Device select				Last TX	SDR/DDR	Width of the data transfer		Transmitted byte	
For device 0											
1	0 (Instruction)	1 (spi_select_out[0])				0	0	0	0	8-bit instruction	0x010_00xx
2	2 (RX_COUNT)	1				1	0	0	0x0000 (received memory data units [4])		0x218_0000
For device 1											
1	0 (Instruction)	2 (spi_select_out[1])				0	0	0	0	8-bit instruction	0x020_00xx
2	2 (RX_COUNT)	2				1	0	0	0x0000 (received memory data units [4])		0x228_0000

In the case of the RDCR command, set the 8-bit instruction to ‘0x35’. In the case of the RDSR1 command, set the 8-bit instruction to ‘0x05’.

[3] In case of “TX” command, DATA[15:8] specifies the second transmitted byte. This is used only (and must be specified) for Octal data transfer with DDR mode, that is, when DATA[17:16] = “3” and DATA[18] = “1”.

[4] In case of “RX_COUNT” command, DATA[15:0] specifies the number of received memory data units (minus 1); “0”: 1 unit, “1”: 2 units. For SPI (except Octal SPI with DDR) ,one memory data unit is a byte; for Octal SPI with DDR, one memory data unit is a 2-byte word. The number of used Rx data FIFO entries (in RX_DATA_FIFO_STATUS) is equal to the number of memory data units to be received * 8/data width (1, 2, 4, 8). The number of used Rx data MMIO FIFO entries (in RX_DATA_MMIO_FIFO_STATUS) is equal to the number of bytes.

3.3.4 Register setting for Quad

Steps (10) to (13) in **Figure 7** sets the register of Quad. The WRR command is sent to set the device register as indicated by (11) and (12) in **Figure 7**. Before sending the WRR command, it is necessary to send the WREN command of Quad like the RESET command.

Table 4 explains the Tx command FIFO entry for the WRR command. In this application note, SMIF_TX_CMD_FIFO_WR Setting of spi_select[0] and spi_select[1] are set separately for the WRR command. **Figure 9** checks the WIP bit in devices to confirm the completion of the WRR command.

Table 4 SMIF_TX_CMD_FIFO_WR setting for write registers (WRR 01h)

bits	26:24	23	22	21	20	19	18	17:16	15:18 [5]	7:0	SMIF_TX_CMD_FIFO_WR
Entry	Command	Device select				Last TX	SDR/DDR	Width of the data transfer		Transmitted byte	
For device 0											
1	0 (Instruction)	1 (spi_select_out[0])				0	0	0	0	0x01	0x010_0001
2	0 (Input Data)	1				0	0	0	Set BP bits = 0 in status register 1 [6]		0x010_00xx
3	0 (Input Data)	1				1	0	0	Set QUAD bit = 1 LC bits = 1 in configuration register [6]		0x018_00xx
For device 1											
1	0 (Instruction)	2 (spi_select_out[1])				0	0	0	0	0x01	0x020_0001
2	0 (Input Data)	2				0	0	0	Set BP bits = 0 in Status Register 1 [6]		0x020_00xx
3	0 (Input Data)	2				1	0	0	Set QUAD bit = 1 LC bits = 1 in configuration register [6]		0x028_00xx

[5] In case of “Tx” command, DATA[15:8] specifies the second transmitted byte. This is used only (and must be specified) for Octal data transfer with DDR mode, i.e., when DATA[17:16] = “3” and DATA[18] = “1”.

[6] As shown in **Table 3**, obtain the register value of each Quad in advance. Then, set the necessary bit for the register value.

3.4 Erase

Steps (2) to (4) in **Figure 6** indicate the erase operation for Quad. The 4SE command is sent to erase sector of devices as indicated by (3) in **Figure 6**. Before sending the 4SE command, it is necessary to send the WREN command of Quad.

Table 5 explains the Tx command FIFO entry for the 4SE command.

Table 5 SMIF_TX_CMD_FIFO_WR setting for Sector Erase (4SE DCh)

bits	26:24	23	22	21	20	19	18	17:16	15:18 [7]	7:0	SMIF_TX_CMD_FIFO_WR
Entry	Command	Device select				Last TX	SDR/DDR	Width of the data transfer		Transmitted byte	
1	0 (Instruction)	3				0	0	0	0	0xDC	0x030_00DC
2	0 (Address)	3				0	0	0	0	0x01	0x030_0001
3	0 (Address)	3				0	0	0	0	0x24	0x030_0024
4	0 (Address)	3				0	0	0	0	0x00	0x030_0000
5	0 (Address)	3				1	0	0	0	0x00	0x030_0000

Figure 8 checks the WIP bit in devices to confirm the completion of the 4SE command.

3.5 Write transfer in MMIO mode

Steps (5) to (13) in **Figure 6** programs the QUAD. MMIO mode is used in steps (5) to (13). Before sending the commands of QUAD, it is necessary to set cryptography.

In XIP mode, a cryptography component supports on-the-fly encryption for the write data and on-the-fly decryption for the read data. The use of on-the-fly cryptography is determined by a device's SMIF_DEVICEx_CTL.CRYPTO_EN. **Figure 12** illustrates the complete XIP mode functionality.

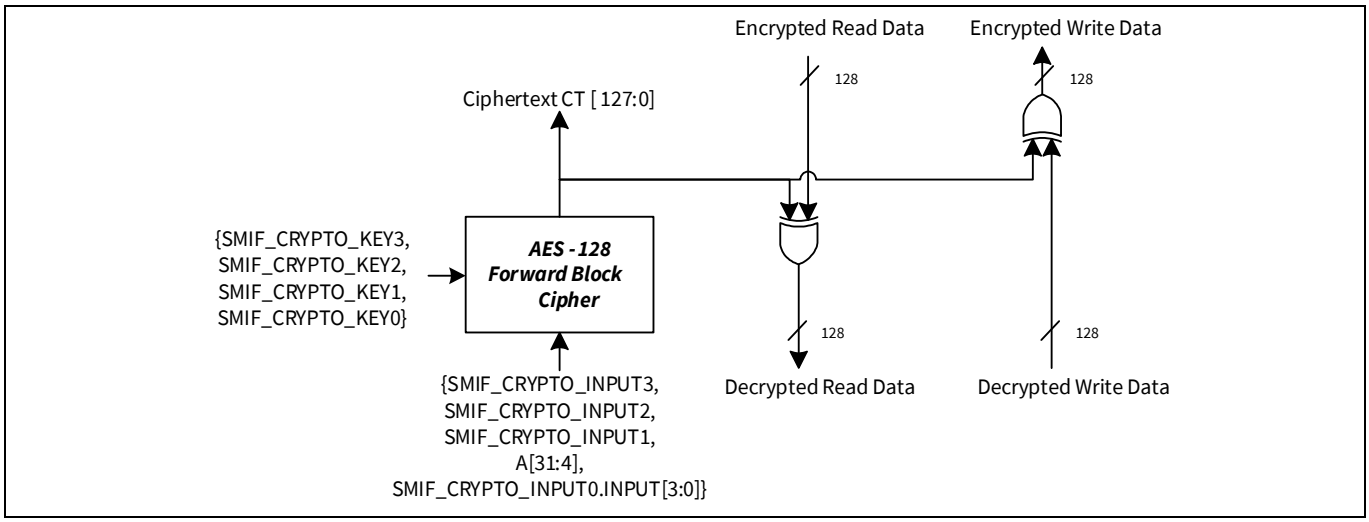

Figure 12 Cryptography in XIP mode

[7] In case of "Tx" command, DATA[15:8] specifies the second transmitted byte. This is used only (and must be specified) for Octal data transfer with DDR mode, i.e., when DATA[17:16] = "3" and DATA[18] = "1".

Example of Dual-Quad SPI DDR mode

In MMIO mode, the cryptography component is accessible through a MMIO register interface to support offline encryption and decryption. In this example, before programming using the 4QPP command in the memory, encryption is set for program data with MMIO mode as indicated by (5) in **Figure 6**.

Figure 13 shows the encryption for the program data with MMIO mode. In cryptography of XIP mode, the flow is realized by hardware.

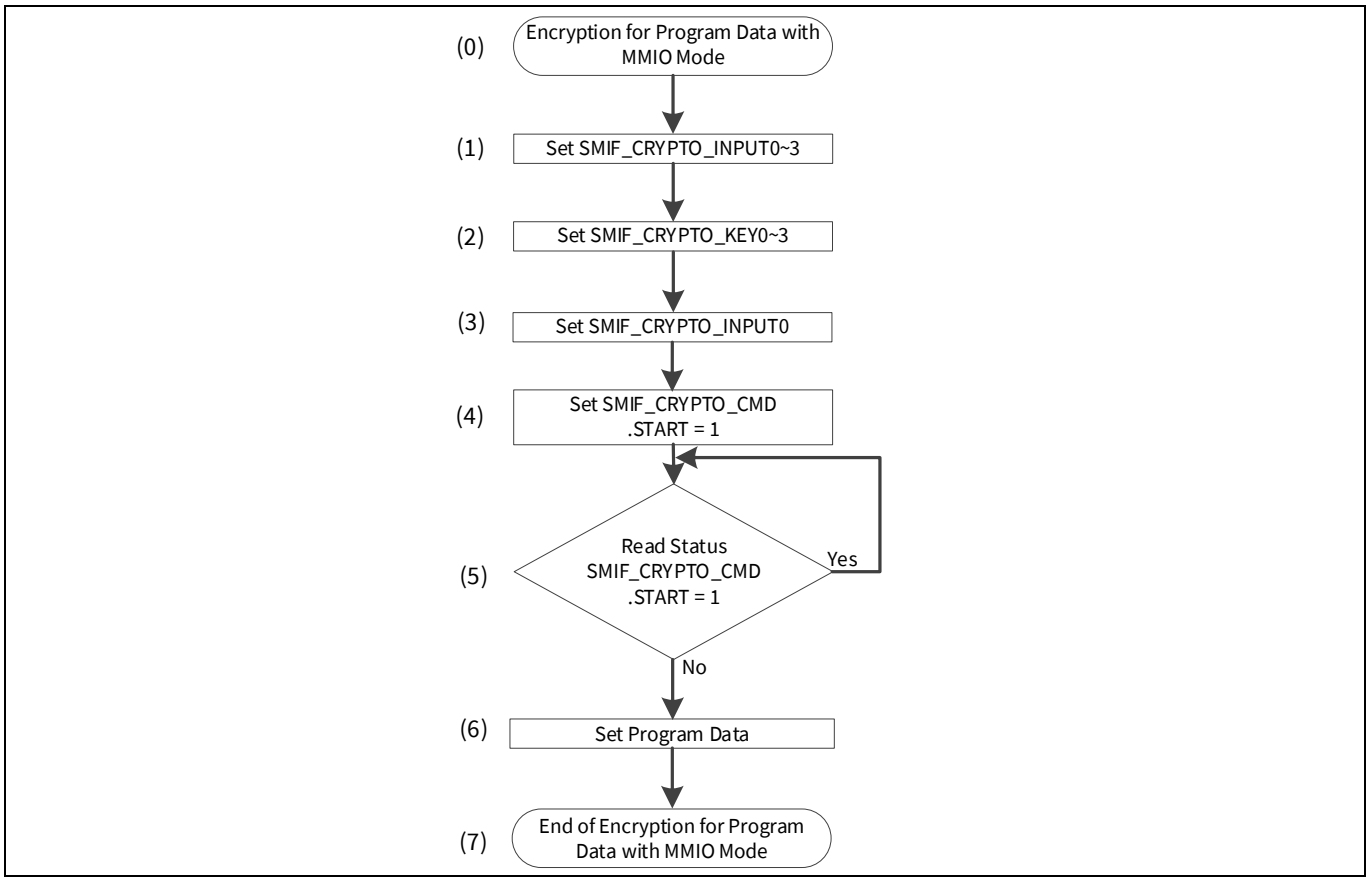

Figure 13 Example of software flowchart of encryption for program data with MMIO mode

The example software flow is explained as follows:

(0) Encryption with MMIO mode is set for the example.

(1) Set SMIF_CRYPTO_INPUT0~3 to a plaintext PT[127:0] (The plaintext PT[127:0] is the input to the AES-128 forward block cipher.) SMIF_CRYPTO_INPUT0[0:3], SMIF_CRYPTO_INPUT1[0:31], SMIF_CRYPTO_INPUT2[0:31], SMIF_CRYPTO_INPUT3[0:31] are static values. It means that they correspond to 'nonce'. SMIF_CRYPTO_INPUT0[4:31] is set dynamically according to read/write address. This corresponds to the counter. So, this is like a CTR mode of the block cipher. The nonce in the cryptographic context is an arbitrary number that is usually only used once for a cryptographic operation.

(2) Set SMIF_CRYPTO_KEY0~3 to a secret key KEY[127:0] (The key KEY[127:0] is the key of the AES-128 forward block cipher).

(3) Set SMIF_CRYPTO_INPUT0 to the following value:
Value = ((0x6000_0000 + 0x0124_0000 x 2) & 0xFFFF_FFF0) + (SMIF_CRYPTO_INPUT0 & 0x0000_000F)
Start address range of SMIF in the address map for Cortex®-M7 and Cortex®-M0+: 0x6000_0000
Address of a sector of each Quad device used in the example: 0x0124_0000

Example of Dual-Quad SPI DDR mode

(4) Set SMIF_CRYPTO_CMD.START to '1' (SW sets this field to '1' to start a AES-128 forward block cipher operation. Hardware sets this field to '0' to indicate that the operation has completed).

(5) Check until SMIF_CRYPTO_CMD.START bit is set to '0'.

(6) Set the Program data as follows:
Program data = (Input data) XOR (SMIF_CRYPTO_OUTPUT0~3)

(7) End of encryption with MMIO mode is set for the example.

The 4QPP command is sent to program for devices as (7) in **Figure 6**. Before sending it, it is necessary to send the WREN command of Quad. **Table 6** explains the Tx command FIFO entry for Quad Page Program (4QPP 34h). **Figure 9** checks the WIP bit in devices to confirm the completion of the 4QPP command completion.

Table 6 SMIF_TX_CMD_FIFO_WR setting for Quad Page Program (4QPP 34h)

bits	26:24	23	22	21	20	19	18	17:16	15:18 [8]	7:0	SMIF_TX_CMD_FIFO_WR
Entry	Command	Device select				Last TX	SDR/DDR	Width of the data transfer		Transmitted byte	
1	0 (Instruction)	3				0	0	0	0	0x34	0x030_0034
2	0 (Address)	3				0	0	0	0	0x01	0x030_0001
3	0 (Address)	3				0	0	0	0	0x24	0x030_0024
4	0 (Address)	3				0	0	0	0	0x00	0x030_0000
5	0 (Address)	3				0	0	0	0	0x00	0x030_0000
6	1 (TX_COUNT)	3				1	0	3	0x000F (transmitted memory data units [9])		0x030_0000

3.6 Read transfer in XIP mode

The SMIF in XIP mode automatically (without software intervention) generates memory transfers by accessing the Tx FIFOs and Rx FIFO. It generates memory read/write transfers for AHB-Lite or AXI read/write transfers.

This is done in the XIP block which:

- Translates read or write transfer requests from the AHB-Lite or AXI interfaces to commands in the Tx command FIFO
- Sends and receives data to and from the Tx/Rx data FIFOs

In this example, Quad I/O High Performance Read Mode, a function of Quad flash, is used. In case of Dual-Quad configuration as indicated by (14) in **Figure 6**, the setting is different between device0 connected to spi_data[3:0] and device1 connected to spi_data[7:4] and both device0 and device1 must set each register. The setting by XIP mode explained as follows:

[8] In case of the "Tx" command, DATA[15:8] specifies the second transmitted byte. This is used only (and must be specified) for Octal data transfer with DDR mode, i.e., when DATA[17:16] = "3" and DATA[18] = "1".

[9] In case of the "TX_COUNT" command, DATA[15:0] specifies the number of transmitted memory data units (minus 1); "0": 1 unit, "1": 2 units. For SPI (except Octal SPI with DDR), one memory data unit is a byte; for Octal SPI with DDR and HYPERBUS™, one memory data unit is a 2-byte word. The number of used Tx data FIFO entries (in TX_DATA_FIFO_STATUS) is equal to the number of memory data units to be transmitted.

Example of Dual-Quad SPI DDR mode

6. Set SMIF_DEVICE0_CTL.DATA_SEL to '0' (spi_data[0] = IO0, spi_data[1] = IO1, …, spi_data[7] = IO7.)
   Set SMIF_DEVICE1_CTL.DATA_SEL to '2' (spi_data[4] = IO0, spi_data[5] = IO1, …, spi_data[7] = IO3.)
7. Set SMIF_DEVICEx_CTL.WR_EN to '1' (Write transfers are allowed to this device).
8. Set SMIF_DEVICEx_CTL.CRYPTO_EN to '1' (Cryptography on read/write accesses is enabled)
9. Set SMIF_DEVICEx_CTL.MERGE_TIMEOUT to '0'.
   Set SMIF_DEVICEx_CTL.MERGE_EN to '1'.
10. Set SMIF_DEVICEx_CTL.TOTAL_TIMEOUT to '1000'.
    Set SMIF_DEVICEx_CTL.TOTAL_TIMEOUT_EN to '1'.
11. Set SMIF_DEVICEx_ADDR to '0x6000_0000' (Specifies the base address of the device region)
12. Set SMIF_DEVICEx_MASK to '0xFC00_0000' (Specifies the size of the device region. 32 Mbytes 2 devices)
13. Set SMIF_DEVICEx_ADDR_CTL.SIZE2 to '3' (Specifies the size of the XIP device address in bytes:"3": 4 byte address. ReadCmd: 4QOR, WriteCmd: 4QPP)
14. Set SMIF_DEVICEx_ADDR_CTL.DIV2 to '1' ('1': Divide by 2)
15. From here to 16, set the read for XIP mode:
    Set SMIF_DEVICEx_RD_CMD_CTL.CODE to '0x00' (Command byte code. It is setting-free for Continuous Transfer)
    Set SMIF_DEVICEx_RD_CMD_CTL.DDR_MODE to '0' (Mode of transfer rate: '0': SDR mode)
    Set SMIF_DEVICEx_RD_CMD_CTL.WIDTH to '0' (Width of data transfer: '0': 1 bit/cycle single data transfer)
    Set SMIF_DEVICEx_RD_CMD_CTL.PRESENT2 to '0' (Presence of command field: '0': not present because of Continuous Transfer)
16. Set SMIF_DEVICEx_RD_ADDR_CTL.DDR_MODE to '0' (Mode of transfer rate: '0': SDR mode)
    Set SMIF_DEVICEx_RD_ADDR_CTL.WIDTH to '2' (Width of data transfer: '2': 4 bits/cycle Quad data transfer.)
17. Set SMIF_DEVICEx_RD_MODE_CTL.CODE to '0xA5' (Mode byte code.) See the section of Quad I/O Read in the **datasheet of S25FL256S**.
    Set SMIF_DEVICEx_RD_MODE_CTL.DDR_MODE to '0' (Mode of transfer rate: '0': SDR mode)
    Set SMIF_DEVICEx_RD_MODE_CTL.WIDTH to '2' (Width of data transfer: '2': 4 bits/cycle Quad data transfer.)
    Set SMIF_DEVICEx_RD_MODE_CTL.PRESENT2 to '1' (Presence of command field: '1': present (1 Byte))
18. Set SMIF_DEVICEx_RD_DUMMY_CTL.SIZE5 to '3' (Number of dummy cycles (minus 1)). See the section of Configuration Register 1 (CR1) in the **datasheet of S25FL256S**.
    Set SMIF_DEVICEx_RD_DUMMY_CTL.PRESENT2 to '1' (Presence of command field: '1': present (1 Byte))
19. Set SMIF_DEVICEx_RD_DATA_CTL.DDR_MODE to '0' (Mode of transfer rate: '0': SDR mode)
    Set SMIF_DEVICEx_RD_DATA_CTL.WIDTH to '3' (Width of data transfer: '3': 8 bits/cycle octal data transfer.)
20. Set SMIF_DEVICEx_RD_CRC_CTL.CODE to '0x0000' (Read Bus CRC control is not used.)
21. Set SMIF_DEVICEx_RD_BOUND_CTL.CODE to '0x0000' (Read boundary control is used for HYPERBUS™.)
22. Set SMIF_DEVICEx_WR_CRC_CTL.CODE to '0x0000' (Read Bus CRC control is not used.)
23. Set SMIF_DEVICEx_CTL.ENABLED to '1' (Device enable: '1': Enabled.)

Example of Dual-Quad SPI DDR mode

In the case of the Quad SPI read command, address jumps can be done without the need for additional Quad I/O Read instructions. This is controlled through the setting of the Mode bits after the address sequence, as shown in **Figure 2**. **Table 7** explains the Tx command FIFO entry for the Mode setting of Quad I/O Read (4QIOR ECh). See the **datasheet of S25FL256S** for details of Quad I/O High Performance Read Mode, which is the function of Quad Flash In **Figure 6**, the Mode setting with the MMIO mode and the reading with the XIP mode are done separately unlike shown in **Figure 2**.

Table 7 SMIF_TX_CMD_FIFO_WR setting for Quad I/O Read (4QIOR ECh)

bits	26:24	23	22	21	20	19	18	17:16	15:18 [10]	7:0	SMIF_TX_CMD_FIFO_WR
Entry	Command	Device select				Last TX	SDR/DDR	Width of the data transfer		Transmitted byte	
1	0 (Instruction)	3				0	0	0	0	0xEC	0x030_00EC
2	0 (Address)	3				0	0	2	0	0x01	0x032_0001
3	0 (Address)	3				0	0	2	0	0x24	0x032_0024
4	0 (Address)	3				0	0	2	0	0x00	0x032_0000
5	0 (Address)	3				0	0	2	0	0x00	0x032_0000
6	0 (Mode)	3				1	0	2	0	0xA5	0x032_00A5

3.7 Data learning pattern

The data learning pattern (DLP) setting method is shown below. See **datasheet of S25FL256S** for DLP details.

3.7.1 Setting of DLP

DLP is enabled by default.

3.7.2 DLP calibration

The setting method of DLP calibration is described.

Quad SPI devices with DLP calibration hardware can perform DLP calibration by hardware.

Figure 14 shows the DLP calibration.

[10] In case of the "Tx" command, DATA[15:8] specifies the second transmitted byte. This is used only (and must be specified) for Octal data transfer with DDR mode, i.e., when DATA[17:16] = "3" and DATA[18] = "1".

Example of Dual-Quad SPI DDR mode

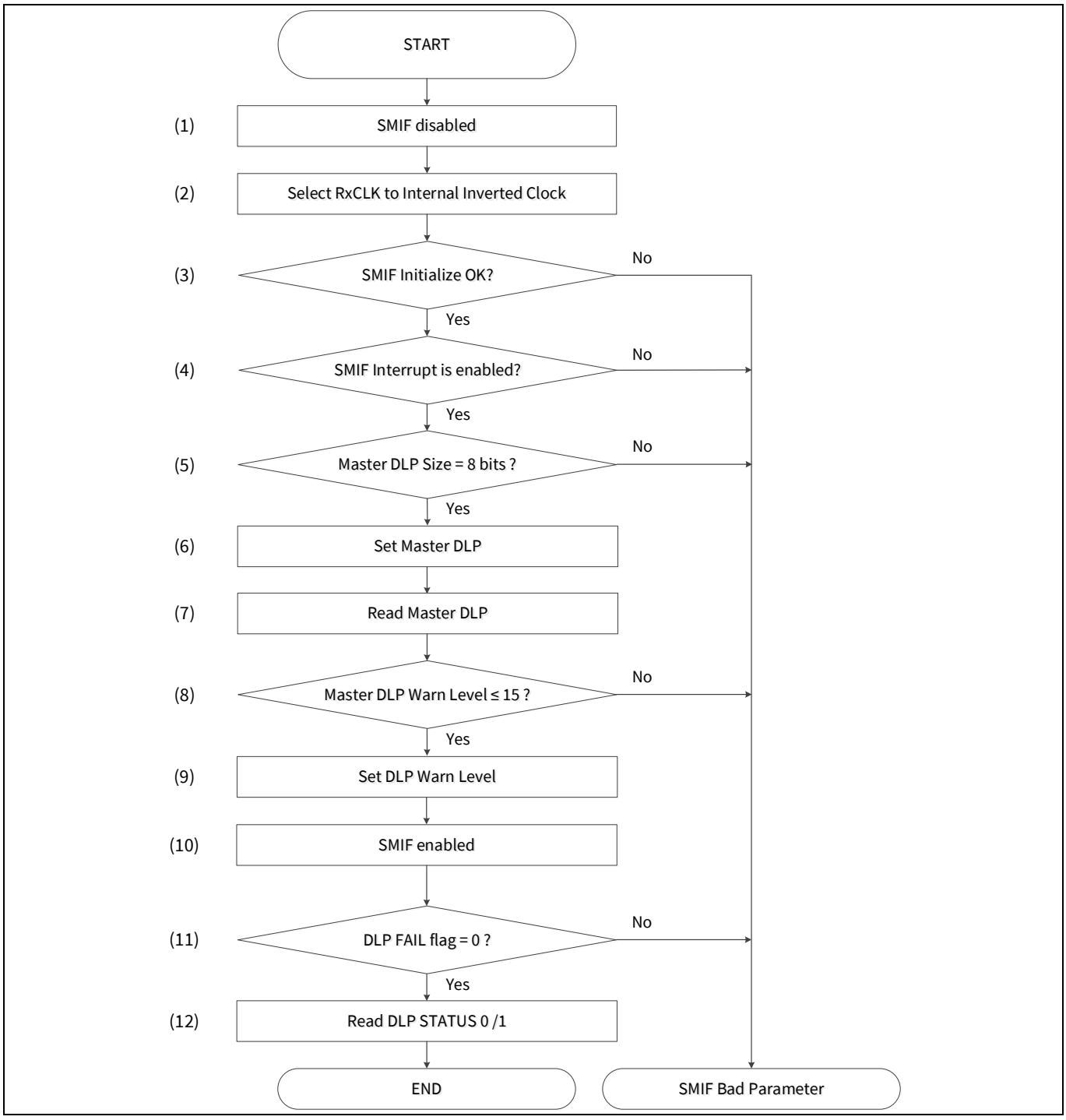

Figure 14 DLP calibration

(1) Disable SMIF function.
Set SMIF_CTL.ENABLED to '0' (SMIF disabled)

(2) Select the RxCLK to Internal Inverted Clock.
Set SMIF_CTL.CLOCK_IF_RX_SEL = '5' (Inverted Internal Clock)

(3) Initialize SMIF.
Check the SMIF interrupt is disabled during the initializing SMIF.
Set SMIF_INTR_MASK.TR_TX_REQ = '0' (Default)
Set SMIF_INTR_MASK.TR_RX_REQ = '0' (Default)

Example of Dual-Quad SPI DDR mode

(4) End SMIF initializing and check the SMIF interrupt enable.

(5) Check the size of Master DLP.

(6) Set Master DLP to compare.
DLP width is constant 8 bits.
Although the input value is 16-bit integer, upper 8 bits will be ignored.
Set SMIF_DL_CTL.DLP = dlp, (Example: dlp = 0xAA)

(7) Read the Master DLP.
Read SMIF_DL_CTL.DLP.

(8) Check Master Warn Level.[11]
If the Warning Level (warnlevel) is greater than 15, the SMIF BAD parameter is returned. Otherwise, SMIF Success is returned.
Read SMIF_DL_CTL.DLP_WARNING_LEVEL

(9) Set DLP Warn Level
Set SMIF_DL_CTL. DLP_WARNING_LEVEL = warnlevel, (Example: warnlevel = '0x3' (Recommend: greater or equal '0x3')).

(10) Enable the SMIF function.
Set SMIF_CTL.ENABLED to "1" (SMIF enabled).

(11) Check the interrupt flag.
Read SMIF_INTR.DL_FAIL.
If the value is '0', calibration succeeded; else failed.

(12) Check that the DLP status (Default Value: All '0').
Read SMIF_DL_STATUS0.
Read SMIF_DL_STATUS1.
Check it was rewritten by hardware.

[11] See the descriptions of the "DL_WARNING" in "Serial Memory Interface" section of the XMC7000 architecture technical reference manual regarding the DLP warning, see **References**.

4 Glossary

Table 8 Glossary

Terms	Description
AES	Advanced Encryption Standard
AHB	AMBA High-performance Bus
DDR	Dual Data Rate
DESELECT command	One of five command types in Tx command FIFO. This command causes the memory interface transmit logic to finish a transfer and deselect the memory device.
DLP	Data Learning Pattern
Dual-Quad	Both devices are Quad SPI memories. During SPI read and write transfers, each device uses dedicated data signal connections and provides a nibble of a byte.
DUMMY_COUNT command	One of five command types in the Tx command FIFO. This command specifies the number of dummy cycles.
HYPERBUS™	Low signal count DDR interface
MMIO	Memory Mapped Input/Output
RX_COUNT command	One of five command types in Tx command FIFO. This command is used when data is received from external memories to Rx data FIFO.
SDR	Single Data Rate
SMIF	Serial Memory Interface
SPI	Serial Peripheral Interface
Tx command	One of five command types in the Tx command FIFO. This command specifies the phase, such as Instruction, Address, and Mode for commands of Quad SPI.
TX_COUNT command	One of five command types in the Tx command FIFO. This command is used when data is transmitted from the Tx data FIFO to external memories.
XIP	eXecute-In-Place

References

The following are the XMC7000 family series datasheets and technical reference manuals. Contact **Technical support** to obtain these documents.

[1] Device datasheets:

- 32-bit Arm® Cortex®-M7 microcontroller XMC7100 family (Doc No. 002-33896)
- 32-bit Arm® Cortex®-M7 microcontroller XMC7200 family (Doc No. 002-33522)
- S25FL128S/S25FL256S, 128 Mb (16 MB)/256 Mb (32 MB) 3.0V SPI flash memory (Doc No. 001-98283)

[2] Technical reference manuals:

- XMC7000 MCU family architecture technical reference manual (TRM) (Doc No. 002-33816)
- XMC7100 registers technical reference manual (TRM) (Doc No. 002-33817)
- XMC7200 registers technical reference manual (TRM) (Doc No. 002-33812)

[3] Application notes:

- AN234118 - GPIO usage setup in XMC7000 family
- AN224119 – How to use timer, counter, and PWM (TCPWM) in XMC7000 family

Revision history

Document version	Date of release	Description of changes
**	2021-12-14	Initial release

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2021-12-14
Published by
Infineon Technologies AG
81726 Munich, Germany

Do you have a question about this document?
Go to www.cypress.com/support

Document reference
002-34227 Rev. **

IMPORTANT NOTICE
The information contained in this application note is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this application note must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this application note.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office (**www.infineon.com**).

WARNINGS
Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.