RZ/A2M Group

Example of Booting from Serial Flash Memory

Introduction

This application note describes an example of booting from the serial flash memory via the SPI multi-I/O bus controller (hereinafter called "SPIBSC") of RZ/A2M by using the boot mode 3 (serial flash boot 3.3V) function.

Target Device

RZ/A2M

When using this application note with other Renesas MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.

Contents

1. Specifications

1.1 Booting from Serial Flash Memory

In boot mode 3, the RZ/A2M boots from the serial flash memory allocated to the SPI multi-I/O bus space (hereinafter called "serial flash boot"). Figure 1.1 shows the Conceptual diagram of serial flash boot operation.

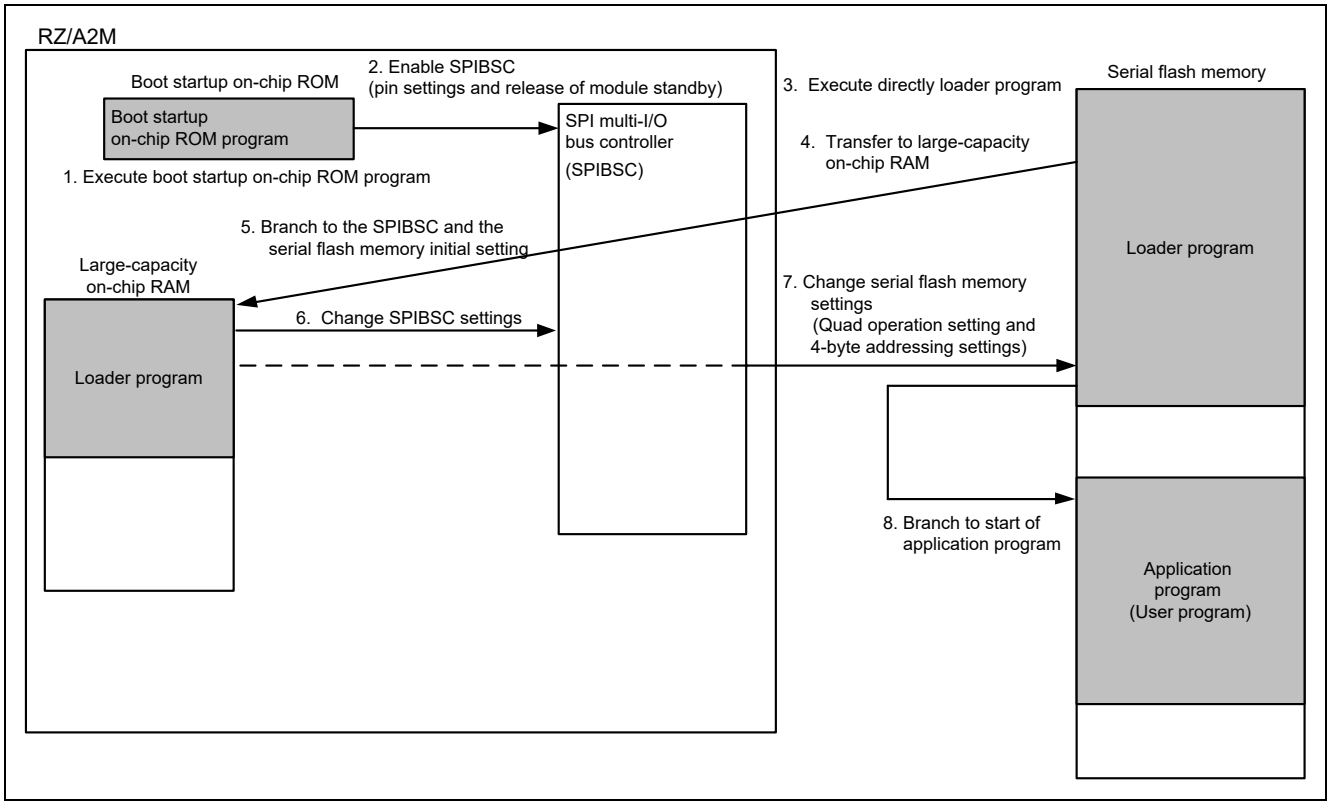


Figure 1.1 Conceptual diagram of serial flash boot operation

The conceptual diagram of serial flash boot operation is described below.

1 When the RZ/A2M starts up by serial flash boot, the boot startup on-chip ROM program runs after power-on reset is canceled.
2 The boot startup on-chip ROM program sets the SPIBSC to external address space read mode to enable to directly run programs allocated to the SPI multi-I/O bus space.
3 Execute directly the loader program stored in the serial flash memory.
4 The loader program is transferred from the serial flash memory to the large-capacity on-chip RAM by section initialization of the loader program.
5 Branch to the SPIBSC and the serial flash memory initial setting transferred to the large-capacity on-chip RAM.
6 The loader program changes the SPIBSC settings.
7 The loader program changes the serial flash memory settings.
8 Execution branches to the start address of the application program.

The boot startup on-chip ROM program makes settings to allow common access to typical serial flash memory devices, so it is necessary to provide the optimal settings to the serial flash memory used by the customer. This application note describes how to allocate the loader program to the start address (H'2000_0000) of the SPI multi-I/O bus space branched by the boot startup on-chip ROM program, and then branch to the customer-created application program (user program) after the loader program are provided optimal settings to the serial flash memory used by the customer

1.2 Peripheral Functions Used

This sample code not only configures the SPIBSC but also initializes the clock pulse oscillator, interrupt controller, general-purpose input/output ports, memory management unit, primary cache (L1 cache), and secondary cache (L2 cache).

In this application note, the SPI multi-I/O bus controller is referred to as the SPIBSC, the Clock pulse generator as the CPG, the Interrupt controller as the INTC, the OS timer as the OSTM, the Serial communication interface with FIFO as the SCIFA, the General I/O ports as the GPIO, the Power-down modes as the STB, and the Memory management unit as the MMU.

Table 1.1 summarizes Peripheral functions and their applications, and Figure 1.2 shows Operating environment for the sample code.

Table 1.1 Peripheral functions and their applications

Peripheral Function	Application
SPI multi-I/O bus controller (SPIBSC)	When set to external address space read mode, it generates signals that enable the CPU to directly read from serial flash memory connected to the SPI multi-I/O bus space.
Clock pulse generator (CPG)	Generate the operating frequency of the RZ/A2M.
Interrupt controller (INTC)	Control OSTM channel 0, OSTM channel 2 and SCIFA channel 4 interrupts.
OS timer (OSTM)	Use OSTM channel 0 and channel 2 • Channel 0 Control the cycle for blinking LED • Channel 2 Use for time management by OS Abstraction Layer
Serial communication interface with FIFO (SCIFA)	Communicate between SCIFA channel 4 and the host PC.
General I/O ports (GPIO)	Switch multiplexed pin functions for SCIFA channel 4. Control pin for LED on/off.
Power-down modes (STB)	Cancel the module standby state of the RZ/A2M's peripheral I/O. Enable writing to the on-chip data retention RAM.
Memory management unit (MMU), L1 cache, and L2 cache	Generate conversion tables such as specifying valid area of L1 cache or specifying memory type in the RZ/A2M external address area. Enable the L1 and L2 caches.

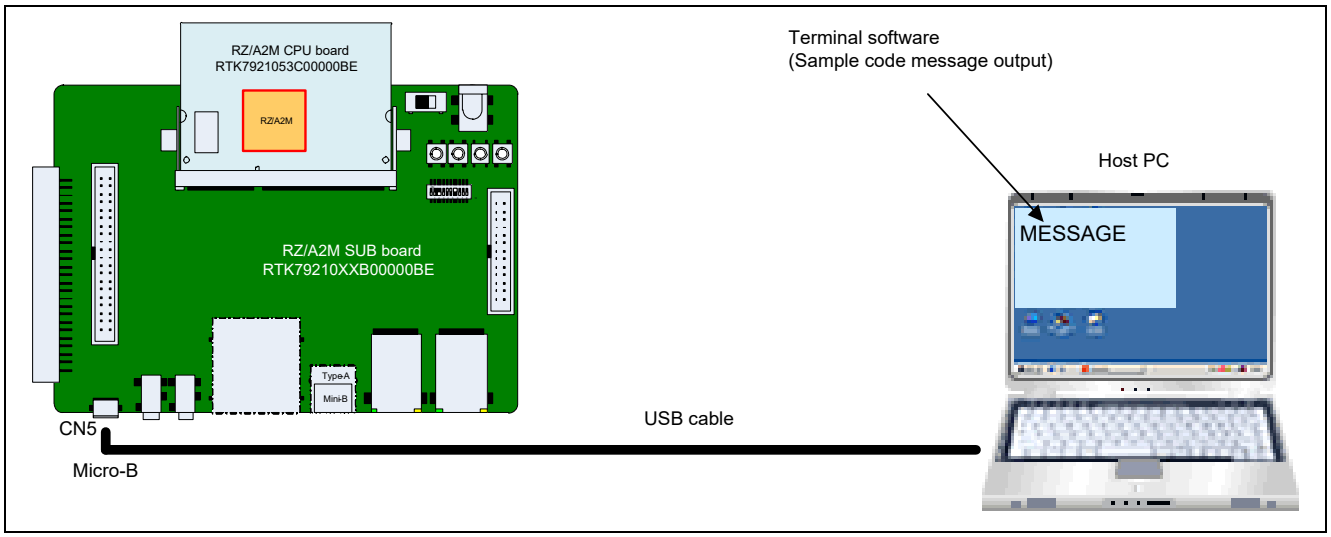


Figure 1.2 Operating environment

2. Operation Confirmation Conditions

The sample code accompanying this application note has been run and confirmed under the conditions below.

Table 2.1 Operation confirmation conditions (1/2)

Item	Contents
MCU used	RZ/A2M
Operating frequency (Note)	CPU clock (Iϕ): 528MHz Image processing clock (Gϕ): 264MHz Internal bus clock (Bϕ): 132MHz Peripheral clock 1 (P1ϕ): 66MHz Peripheral clock 0 (P0ϕ): 33MHz QSPI0_SPCLK: 66MHz CKIO: 132MHz
Operating voltage	Power supply voltage (I/O): 3.3V Power supply voltage (1.8/3.3V switch I/O (PVcc_SPI)): 3.3V Power supply voltage (internal): 1.2V
Integrated development environment	e2 studio Version 2020-07
C compiler	GNU Arm Embedded Toolchain 6-2017-q2-update
	Compiler option (addition of directory path excluded) Release configuration: -mcpu=cortex-a9 -march=armv7-a -marm -mlittle-endian -mfloat-abi=hard -mfpu=neon -mno-unaligned-access -Os -ffunction-sections -fdata-sections -Wunused -Wuninitialized -Wall -Wextra -Wmissing-declarations -Wconversion -Wpointer-arith -Wpadded -Wshadow -Wlogical-op -Waggregate-return -Wfloat-equal -Wnull-dereference -Wmaybe-uninitialized -Wstack-usage=100 -fabi-version=0 Hardware Debug configuration: -mcpu=cortex-a9 -march=armv7-a -marm -mlittle-endian -mfloat-abi=hard -mfpu=neon -mno-unaligned-access -Og -ffunction-sections -fdata-sections -Wunused -Wuninitialized -Wall -Wextra -Wmissing-declarations -Wconversion -Wpointer-arith -Wpadded -Wshadow -Wlogical-op -Waggregate-return -Wfloat-equal -Wnull-dereference -Wmaybe-uninitialized -g3 -Wstack-usage=100 -fabi-version=0

Note: The operating frequency used in clock mode 1 (Clock input of 24MHz from EXTAL pin)

Table 2.2 Operation confirmation conditions (2/2)

Item	Contents
Operating mode	Boot mode 3 (Serial flash boot 3.3V)
Communications settings of the terminal software	• Baud rate: 115200bps • Data length: 8 bits • Parity: None • Stop bits: 1 bit • Flow control: None
Boards used	RZ/A2M CPU board RTK7921053C00000BE RZ/A2M SUB board RTK79210XXB00000BE
Devices used (functions used on the board)	• Serial flash memory allocated to SPI multi-I/O bus space Manufacturer: Macronix, Product No.: MX25L51245GXD • RL78/G1C (Convert between USB communication and serial communication to communicate with the host PC.) • LED1

3. Reference Application Notes

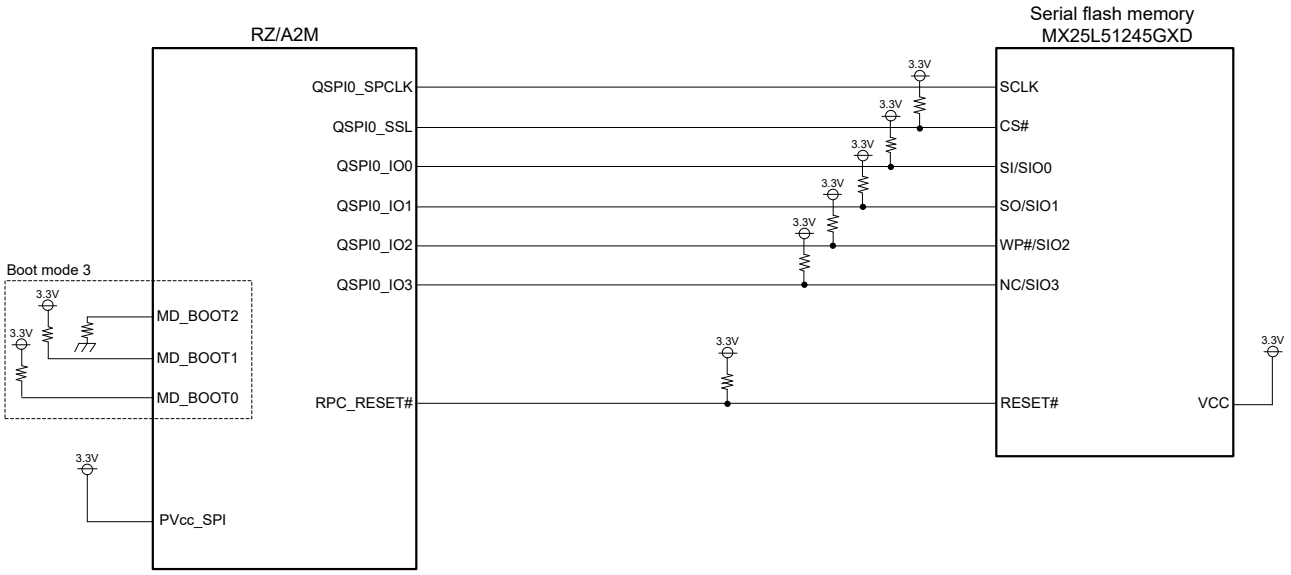
For additional information associated with this document, refer to the following application notes.

- RZ/A2M Group: Example of Initialization (R01AN4321EJ)

4. Hardware

4.1 Hardware Configuration

In the serial flash boot example introduced in this application note, processing is performed by the programs stored in the serial flash memory connected to the SPI multi-I/O bus space using boot mode 3. Figure 4.1 shows the Connection example for booting from serial flash memory in boot mode 3.



Note: # is a symbol that indicates negative logic (or active low).

Figure 4.1 Connection example for booting from serial flash memory in boot mode 3

4.2 Pins Used

Table 4.1 lists the Pins used and their functions.

Table 4.1 Pins used and their functions

Pin name	I/O	Function
MD_BOOT2	Input	Select boot mode (set to boot mode 3) MD_BOOT2: "L", MD_BOOT1: "H", MD_BOOT0: "H"
MD_BOOT1	Input	
MD_BOOT0	Input	
QSPI0_SPCLK	Output	Serial flash memory clock
QSPI0_SSL	Output	Serial flash memory slave select
QSPI0_IO0	I/O	Serial flash memory data 0
QSPI0_IO1	I/O	Serial flash memory data 1
QSPI0_IO2	I/O	Serial flash memory data 2
QSPI0_IO3	I/O	Serial flash memory data 3
RPC_RESET#	Output	Serial flash memory reset
P6_0	Output	Turns on and off LED
RxD4(P9_1)	Input	Serial receive data signal
TxD4(P9_0)	Output	Serial transmit data signal

Note: # is a symbol that indicates negative logic (or active low).

5. Software

5.1 Operation Overview

This section provides an overview of the sample code operation presented in this application note.

5.1.1 Terms Related to Serial Flash Boot

Table 5.1 lists the Terms related to serial flash boot operation described in this application note.

Table 5.1 Terms related to serial flash boot operation

Term	Description
Boot startup on-chip ROM program	This program provides settings to directly execute the programs stored in the serial flash memory connected to the SPI multi-I/O bus space when started up in boot mode 3 (serial flash boot 3.3V). The RZ/A2M branches to the address of H'2000_0000 which is the start address of the SPI multi-I/O bus space after the boot startup on-chip ROM program has been executed. Note that the boot startup on-chip ROM program makes settings to enable common access to typical serial flash memory devices. Since this program is stored in the on-chip ROM of the RZ/A2M, it does not need to be created by the customer.
Loader program	This program is executed after the boot startup on-chip ROM program process has completed. The loader program makes settings to the SPIBSC and to the registers in the serial flash memory corresponding to the serial flash memory used by the customer, and then branches to the start address of the application program. The loader program should be created by the customer according to the specifications of the serial flash memory to be used while referring to this application note. In the sample code, the initial settings are optimized for use with the Macronix serial flash memory (product No.: MX25L51245GXD).
Application program (User program)	This program should be created by customers depending on their system to be used.

5.1.2 Operation Overview of Sample Code Overall

The sample code comprises the loader program executed after boot startup on-chip ROM program process completion and the application program.

1 Loader program (Project name: rza2m_sflash_boot_loader_gcc)
The loader program provides optimal settings to the serial flash memory used (Macronix serial flash memory (product No.: MX25L51245GXD)). The loader program is located at the start address (H'2000_0000) of the SPI multi-I/O bus space, which is branched from the boot startup on-chip ROM program. After the loader program runs, it branches to the start address of the application program. The start address of the application program is specified by the linker_script.ld symbol definition "__application_base_address".

2 Application program (Project name: rza2m_sflash_boot_sample_osless_gcc)
This is an application program to be executed after optimal settings for the serial flash memory are provided in the loader program. Change the location address so the "VECTOR_TABLE" section in the application program is consistent with the address specified by "__application_base_address".
In the sample code, the application program is located at address H'2001_0000.

Figure 5.1 shows the Operation overview of sample code presented in this application note.

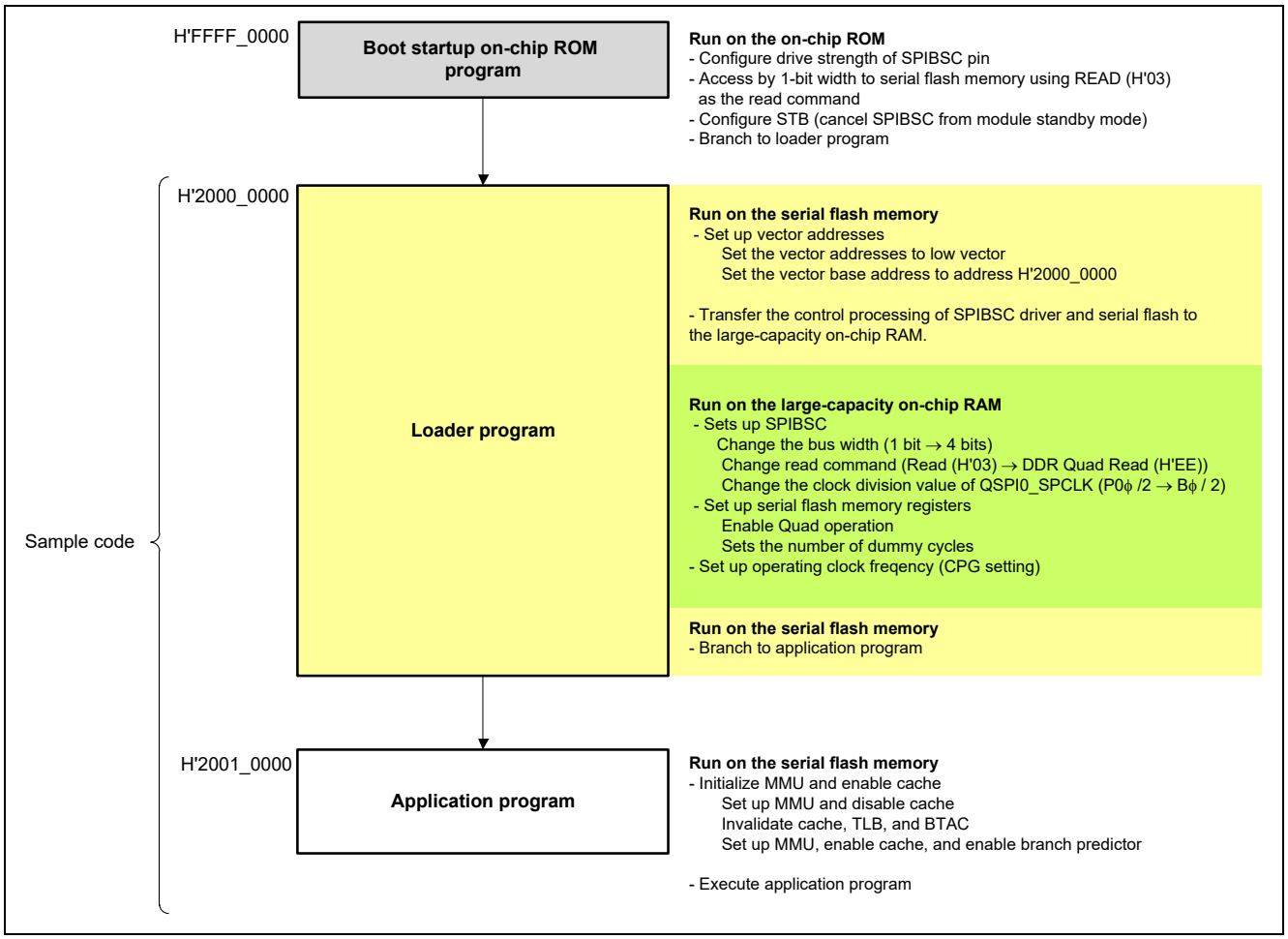


Figure 5.1 Operation overview of sample code

5.1.3 Operation Overview of Loader Program

The loader program is executed after the boot startup on-chip ROM program process has completed. The loader program should be located at the start address (H'2000_0000) of the SPI multi-I/O bus space branched from the boot startup on-chip ROM program.

The boot startup on-chip ROM program makes settings to enable the SPIBSC to operate in external address space read mode. These settings cause the RZ/A2M to convert read operations targeting the SPI multi-I/O bus space to SPI communication so that the direct read operations are enabled to the connected serial flash memory. This makes it possible for the RZ/A2M to directly run programs allocated to the SPI multi-I/O bus space. The settings for commands targeting the serial flash memory used in SPI communication conversion allow common access to typical serial flash memory devices, so it is necessary to provide the optimal settings to the serial flash memory used by the customer.

The optimal settings for the serial flash memory are provided in two places: the registers in the SPIBSC module (hereinafter called "SPIBSC settings") and the registers in the serial flash memory (hereinafter called "serial flash memory settings").

In the loader program of the sample code, the data bus width is set to 4 bits, the transfer bit rate is optimized to the used read command, and the SPIBSC register is set to output a 4-byte address. Also, a Macronix serial flash memory (product No.: MX25L51245GXD) register is set to establish the number of serial flash memory dummy cycles, enable quad operations, and change to 4-byte addressing, and optimal settings are established to access MX25L51245GXD.

The process that establishes the loader program's SPIBSC settings and serial flash memory settings cannot be set by the program allocated to the SPI multi-I/O bus space, so these should be executed from the large-capacity on-chip RAM. In the sample code, the SPIBSC driver process and serial flash memory control process are transferred to the large-capacity on-chip RAM to be executed.

Refer to Table 5.2 to Table 5.4 for the settings after execution of the boot startup on-chip ROM program and loader program.

Table 5.2 to Table 5.4 list the settings made by the boot startup on-chip ROM program and the loader program.

After the settings listed in Table 5.2 to Table 5.4 are made, the loader program branches to the start address of the application program. In the sample code, the application program is allocated to the area starting at H'2001_0000, which is to be the branch target.

Table 5.2 Settings for the Boot Startup On-Chip ROM Program and Loader Program (1/3)

	Item	After execution of boot startup on-chip ROM program	After execution of loader program
SPIBSC settings	**Delay settings**		
	Next access delay setting: SSLDR.SPNDL[2:0]	B'000 (1 QSPIn_SPCLK)	B'000 (1 QSPIn_SPCLK)
	QSPIn_SSL negate delay setting: SSLDR.SLNDL[2:0]	B'000 (1 QSPIn_SPCLK)	B'000 (1 QSPIn_SPCLK)
	Clock delay setting: SSLDR.SCKDL[2:0]	B'000 (1.5 QSPIn_SPCLK)	B'000 (1.5 QSPIn_SPCLK)
	Serial clock (QSPI0_SPCLK):	Pϕ / 2=16.5 [MHz] (Note 1)	Bϕ / 2=66 [MHz] (Note 1)
	QSPIn_SSL output idle value fix:	Sets output values in QSPIn_SSL negation period to the last bit value of the previous transfer	Sets output values in QSPIn_SSL negation period to Hi-Z
	QSPIn_IO3 setting	CMNCR.MOIIO3[1:0]=B'10	CMNCR.MOIIO3[1:0]=B'11
	QSPIn_IO2 setting	CMNCR.MOIIO2[1:0]=B'10	CMNCR.MOIIO2[1:0]=B'11
	QSPIn_IO1 setting	CMNCR.MOIIO1[1:0]=B'10	CMNCR.MOIIO1[1:0]=B'11
	QSPIn_IO0 setting	CMNCR.MOIIO0[1:0]=B'10	CMNCR.MOIIO0[1:0]=B'11
	Fixes the output value of the pin for 1-bit size:	Sets the pin output value for 1-bit size to the last bit value of the previous transfer	Sets the pin output value for 1-bit size to Hi-Z
	QSPIn_IO3 setting	CMNCR.IO3FV[1:0]=B'10	CMNCR.IO3FV[1:0]=B'11
	QSPIn_IO2 setting	CMNCR.IO2FV[1:0]=B'10	CMNCR.IO2FV[1:0]=B'11
	QSPIn_IO0 setting	CMNCR.IO0FV[1:0]=B'10	CMNCR.IO0FV[1:0]=B'11 (Note 2)
	Data bus size: CMNCR.BSZ[1:0]	One serial flash memory connected B'00	One serial flash memory connected B'00
	Read cache: DRCR.RBE	1 (Enabled)	1 (Enabled)
	QSPIn_SSL Negation: DRCR.SSLE	QSPIn_SSL negate every transfer end 0	Address holds assertion of QSPIn_SSL as continuously as possible. If discontinuous from previous transfer address, QSPIn_SSL is negated. 1
	Read data burst length: DRCR.RBURST[4:0]	4 data units (32 bytes) B'00011	4 data units (32 bytes) B'00011
	Data read bit size: DRENR.DRDB[1:0]	1 [bit] B'00	4 [bits] B'10
	Read command: DRCMR.CMD[7:0]	Read H'03	Quad I/O DT Read (4B address) H'EE
	Command enable: DRENR.CDE	Output enabled 1	Output enabled 1
	Command bit size: DRENR.CDB[1:0]	1 [bit] B'00	1 [bit] B'00
	Optional command enable: DRENR.OCDE	Output disabled 0	Output disabled 0

Note: 1 Refer to "Operating clock settings" and "SPIBSC clock selection" in Table 5.4.
2 IO0FV bit for CMNCR register must be set to B'11 (QSPIn_IO0 pin is placed in the Hi-Z) when data read transfer size is not 1-bit.

Table 5.3 Settings for the boot startup on-chip ROM program and loader program (2/3)

	Item	After execution of boot startup on-chip ROM program	After execution of loader program
SPIBSC settings	**Address enable:** **DRENR.ADE[3:0]**	Output address [23:0] B'0111	Output address [31:0] B'1111
	Address bit size: **DRENR.ADB[1:0]**	1 [bit] B'00	4 [bits] B'10
	Option data enable: **DRENR.OPDE[3:0]**	Output disabled B'0000	Output OPD3 (Note) B'1000
	Option data bit size: **DRENR.OPDB[1:0]**	—	4 [bits] B'10
	Option data: **DROPR.OPD3[7:0]** DROPR.OPD2[7:0] DROPR.OPD1[7:0] **DROPR.OPD0[7:0]**	 — — — —	 H'00 — — —
	Dummy cycle enable: **DRENR.DME**	Insertion disabled 0	Insertion enabled 1
	Number of dummy cycles: **DRDMCR.DMCYC[4:0]**	—	7 cycles B'00110
	Extended upper address: DREAR.EAC[2:0] **DREAR.EAV[7:0]**	External address bits [24:0] enabled Directly accessible 32MB spaces B'000 H'00	External address bits [27:0] enabled Directly accessible 256MB spaces B'011 H'00
	Transfer format: DRDRENR.HYPE[2:0] DRDRENR.ADDRE DRDRENR.OPDRE DRDRENR.DRDRE PHYOFFSET1.DDRTMG[1:0] **PHYOFFSET2.OCTTMG[2:0]**	Address, option data, and data are transferred in SDR mode, SPI flash mode B'000 0 0 0 B'11 (SDR) B'100 (Serial flash)	Address, option data, and data are transferred in DDR mode, SPI flash mode B'000 1 1 1 B'10 (DDR) B'100 (Serial flash)
	Octal-SPI flash memory alternative alignment: PHYCNT.ALT_ALIGN **PHYCNT.OCTA[1:0]**	Alternative alignment during Octal-SPI flash memory connection is not supported. 0 B'00	Alternative alignment during Octal-SPI flash memory connection is not supported. 0 B'00
	Octal-SPI flash memory protocol mode: **PHYCNT.OCT**	Octal-SPI flash memory protocol mode not used 0	Octal-SPI flash memory protocol mode not used 0
	External data strobe: **PHYCNT.EXDS**	External data strobe signal not used 0	External data strobe signal not used 0
	Device selection: **PHYCNT.PHYMEM[1:0]**	SDR mode serial flash B'00	DDR mode serial flash B'01
	High-Speed response mode: **PHYCNT.HS**	High-Speed response mode not used 0	High-Speed response mode not used 0

Note: The MX25L51245GXD transits to the Performance Enhance Mode when data (e.g., H'A5, H'5A, H'F0, H'0F, etc.) that toggles between bits 7-4 and bits 3-0 is input during the performance enhance indicator cycle that follows the address cycle. Since the RZ/A2M's external address space read mode does not support the data transfer in Performance Enhance Mode, the sample code makes configuration so that the MX25L51245GXD will not switch into the Performance Enhance Mode by making configuration so that H'00 is output from the OPD3 when a QuadIO DT Read command is issued.

Table 5.4 Settings for the boot startup on-chip ROM program and loader program (3/3)

	Item	After execution of boot startup on-chip ROM program	After execution of loader program
Pin settings	**Pin voltage** **PPOC.POCSEL0** **PPOC.POC0**	Setting in which SPIBSC pin operates at 3.3V 1 1	Setting in which SPIBSC pin operates at 3.3V 1 1
	Drive strength **PSPIBSC[31:0]**	H'0555_5555 (Drive strength of SPIBSC-related pin is 8mA)	H'0555_5555 (Drive strength of SPIBSC-related pin is 8mA)
Serial flash memory settings	**Status Register**	No change (Note 1)	Quad operation enabled QE=1
	Configuration Register	No change (Note 1)	DC[1:0] = B'10 ODS[2:0] = B'110
Other	**Operating clock settings** **Clock input of 24MHz from EXTAL pin in clock mode 1**	Iϕ =132[MHz] Gϕ =264[MHz] Bϕ =132[MHz] P1ϕ =66[MHz] P0ϕ =33[MHz]	Iϕ =528[MHz] Gϕ =264[MHz] Bϕ =132[MHz] P1ϕ =66[MHz] P0ϕ =33[MHz]
	SPIBSC clock selection **SCLKSEL.SPICR[1:0]**	Select P0ϕ B'00	Select Bϕ B'10
	CPU exception vector position	High vector (from H'FFFF_0000)	Low vector (from H'0000_0000)
	CP15 vector-based address register (VBAR)	—	H'2000_0000

Note: 1. In boot mode 3 (serial flash booting 3.3V) of RZ/A2M, the boot program sets the SPIBSC register to issue a read command (opcode: H'03, address: 24 bits, dummy cycle: none) as the command to the serial flash memory. Therefore, if the serial flash memory cannot receive the above read command by the register value in serial flash booting, normal booting may not be possible.

2. If serial flash memory is connected for data transfer, the timing must be adjusted so the input data is loaded. This is done with the loader program. Set the PHYCNT, PHYADJ1, and PHYADJ2 registers for timing adjustment. These registers will change after execution of the boot startup on-chip ROM program and loader program. For details on timing adjustment, refer to "5.10.8 SPIBSC Read Timing Calibration ".

5.1.4 Application Program

(1) Operation of the application program

After a reset is cancelled, the boot startup on-chip ROM program and loader program are executed in that order. Then the execution transfers to the application program that is allocated to address H'2001_0000.

In the startup process, the settings for the stack pointer, MMU, and FPU are executed. The section initialization is performed, and it branches to the resetprg function.

In the resetprg function, after RTC and USB unused channel initialization processing is executed, L1 cache and L2 cache are enabled and INTC initialization is performed. The large-capacity on-chip RAM address is set in VBAR to enable high-speed interrupt processing, IRQ interrupt and FIQ interrupt are enabled, and the main function is called.

In the main function, CPG, OSTM channel 0, SCIFA channel 4, and GPIO initial setting processing is performed. As a result of this initialization processing, the main function outputs the character strings (startup message) to the terminal on the host PC connected with the serial interface and sets the OSTM channel 0 timer to interval timer mode to activate the timer. It generates the OSTM channel 0 interrupt with a cycle of 500ms and repeats turning on/off the LED on the CPU board every 500ms using such interrupt.

For details on the initialization executed by the application program, refer to the application note "RZ/A2M Group Example of Initialization".

(2) Notes to be observed when creating an application program

The application program should be allocated to the address branched from the loader program. Note that the application program should be allocated to the different sector in the serial flash memory from the one in the loader program.

The sector size of the Macronix serial flash memory (MX25L51245GXD) mounted on the RZ/A2M CPU board is 4KB. In the sample code, the application program is allocated to the address of H'2001_0000 (Sector no. 16).

Figure 5.2 shows the Sample code program allocation.

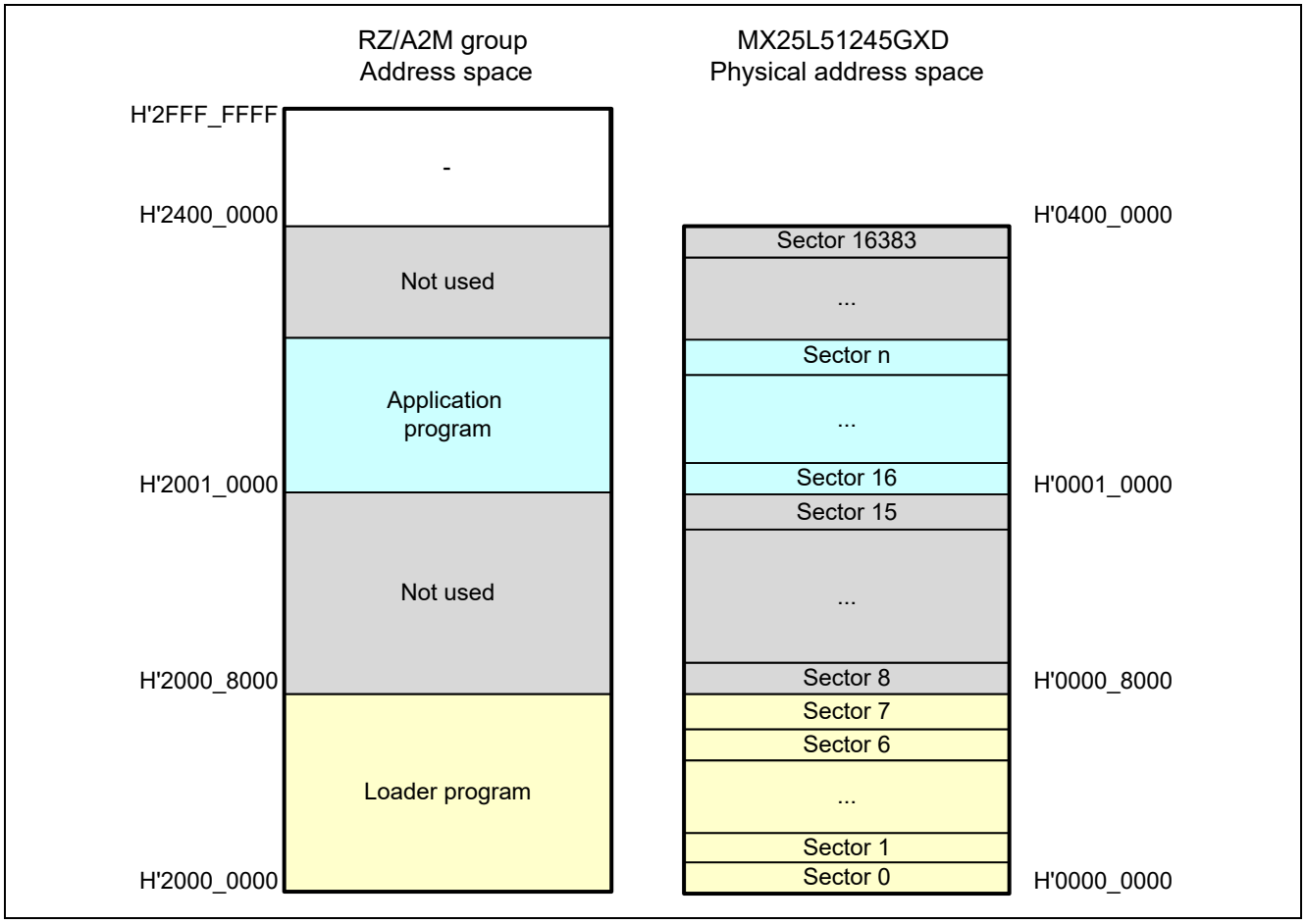


Figure 5.2 Sample code program allocation

The start address of the application program can be changed by making the following changes:

- Project for the loader program
  The branch to the starting address of the application program is executed by the loader program (reset_handler.asm). Specify the destination of branch with the linker script symbol definition "__application_base_address" in linker_script.ld.
- Project for the application program
  Change the allocation address so that the VECTOR_TABLE section of the application program matches the address that is specified in "__application_base_address."

5.2 Peripheral Functions and Memory Allocation in Sample Code

5.2.1 Setting for Peripheral Functions

Table 5.5 lists the Setting for Peripheral Functions during execution of the sample code.

Table 5.5 Setting for Peripheral Functions

Module	Settings
CPG	CPU clock: Set to 1/2 the PLL circuit frequency Internal bus clock: Set to 1/8 the PLL circuit frequency Peripheral clock 1 (P1ϕ): Set to 1/16 the PLL circuit frequency If the input clock is 24MHz in clock mode 1 (divider 1: ×1/2, PLL circuit: ×88), set to the following frequencies • CPU clock (Iϕ): 528MHz • Image processing clock (Gϕ): 264MHz • Internal bus clock (Bϕ): 132MHz • Peripheral clock 1 (P1ϕ): 66MHz • Peripheral clock 0 (P0ϕ): 33MHz • QSPI0_SPCLK: 66MHz (when Bϕ is selected) • CKIO clock: 132MHz (when Bϕ is selected)
SPIBSC	When set to the external address space read mode, it generates the signals which enable the CPU to read directly from the serial flash memory connected to the SPI multi-I/O bus space.
STB	Write permission to on-chip data retention RAM and provision of clock to peripheral functions Clock is supplied to OSTM0, OSTM2, SCIFA4, and SPIBSC with STBCR3, STBCR4, and STBCR8.
GPIO	PORT6 and PORT9 shared pin functions are set. • P6_0: Turns on and off LED • P9_1: RxD4, P9_0: TxD4
OSTM	Sets the channel 0 and the channel 2 in interval timer mode. • Channel 0 Sets the timer counter to have interrupt request generated every 500ms when P1ϕ=66MHz. Generate the intervals at which the LEDs are turned on and off. • Channel 2 Sets the timer counter to have interrupt request generated every 1ms when P1ϕ=66MHz. Used for time management via OS Abstraction Layer.
INTC	Initializes INTC, and registers and executes OSTM channel 0 interrupt (interrupt ID is 88) handler, OSTM channel 2 interrupt (interrupt ID is 90) handler and SCIFA channel 4 interrupt (interrupt ID is 322, 323) handler
SCIFA	Sets the channel 4 in asynchronous communication mode • Data length: 8 bits • Stop bits: 1 bit • Parity: None • Data transfer direction: LSB first transfer Sets the clock source without frequency dividing, the baud rate generator to double speed mode, and the basic clock at 8 times the bit rate when P1ϕ is 66MHz. Sets the bit rate to 71 so that the bit rate is 115200bps. (The bit rate error is -0.53%)

5.2.2 Memory Mapping

Figure 5.3 shows the RZ/A2M Group Address Space and RZ/A2M CPU board memory map.

In the sample code, the code and data that use the ROM area are assigned to the serial flash memory connected to the SPI multi-I/O bus, and the code and data that use the RAM area are assigned to the large-capacity on-chip RAM.

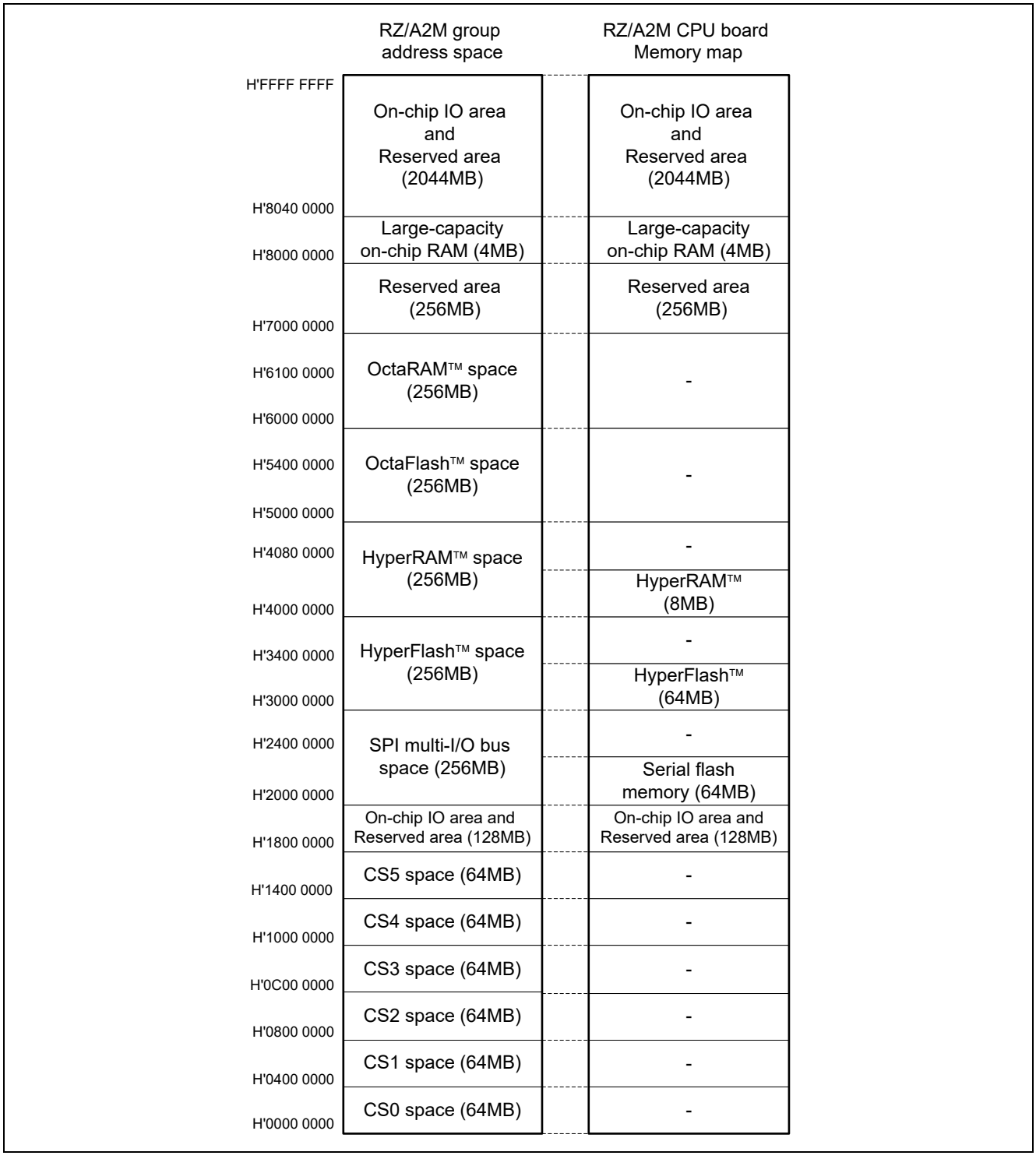


Figure 5.3 Memory mapping

5.2.3 Section Assignment in Sample Code

Table 5.6 shows the Sections and Objects to Be Used in the Loader Program.

For section assignments used in the application program, refer to the application note "RZ/A2M Group Example of Initialization".

Table 5.6 Sections and Objects to Be Used in the Loader Program

Output section name	Input section name Input object name	Description	Loading area	Execution area
LOAD_MODULE1	VECTOR_TABLE	Exception processing vector table	FLASH	FLASH
LOAD_MODULE2	*/r_cpg/*.o (.text .rodata)	CPG settings processing	FLASH	LRAM
	*/rza_io_regrw.o (.text .rodata)	I/O register access processing		
	/r_spibsc/.o (.text .rodata)	SPIBSC settings processing (Except constants for calibrating)		
	*/hwsetup.o (.text .rodata)	Hardware Setup settings processing		
	* (.data .data.*)	Data area with default initial values		
LOAD_MODULE3	RESET_HANDLER	Reset processing	FLASH	FLASH
	INIT_SECTION */sections.o	Section initialization processing		
	* (.text .text.*)	Code area for defaults		
	* (.rodata .rodata.*)	Constant data area for defaults		
.data.memclk_setup	*/r_memclk_setup.o (.text .rodata .data)	Memory clock setting processing	FLASH	LRAM
	*/r_spibsc_setup.o (.text .rodata .data)	Memory clock setting processing for SPIBSC		
	*/r_*_memclk_setup.o (.text .rodata .data)	Memory clock setting processing for each driver		
.bss.memclk_setup	*/r_memclk_setup.o (.bss COMMON)	Data area without default initial values for memory clock settings processing	—	LRAM
	*/r_spibsc_setup.o (.bss COMMON)	Data area without default initial values for memory clock settings processing for SPIBSC		
	*/r_*_memclk_setup.o (.bss COMMON)	Data area without default initial values for memory clock settings processing for each driver		
.stack	None	Stack area for SVC mode	—	LRAM
.bss	* (.bss .bss.*) * (COMMON)	Data area without default initial values	—	LRAM
.heap	None	Heap area	—	LRAM

Note: "FLASH" and "LRAM" shown in Loading Area and Execution Area indicate the serial flash memory area and the large-capacity on-chip RAM area respectively.

5.3 Interrupt Used

Interrupt is not used in the loader program.

For interrupt used in the application program, refer to the application note "RZ/A2M Group Example of Initialization".

5.4 Data Types

Table 5.7 shows the Data Types Used in the Sample Code.

Table 5.7 Data Types Used in the Sample Code

Symbol	Description
char_t	8-bit character
bool_t	Boolean type. Value is true (1) or false (0).
int_t	Fast integer, signed, 32-bit integer in this sample code
int8_t	8-bit integer, signed (defined in standard library stdint.h)
int16_t	16-bit integer, signed (defined in standard library stdint.h)
int32_t	32-bit integer, signed (defined in standard library stdint.h)
int64_t	64-bit integer, signed (defined in standard library stdint.h)
uint8_t	8-bit integer, unsigned (defined in standard library stdint.h)
uint16_t	16-bit integer, unsigned (defined in standard library stdint.h)
uint32_t	32-bit integer, unsigned (defined in standard library stdint.h)
uint64_t	64-bit integer, unsigned (defined in standard library stdint.h)
float32_t	32-bit float
float64_t	64-bit float
float128_t	128-bit float

5.5 Constants Used by the Loader Program

Table 5.8 and Table 5.9 list the constants used by the loader program.

For the constants used in the application program, refer to the application note "RZ/A2M Group Example of Initialization".

Table 5.8 Constants Used in the Loader Program (1/2)

Constant	Setting value	Description
SPIBSC_SUCCESS	(0)	Normal end
SPIBSC_ERR_INVALID	(-1)	Error end
SPIBSC_PORT_VOLTAGE_3_3V	(0)	Sets the operating voltage of the dedicated pin used by the SPIBSC to 3.3V
SPIBSC_PORT_VOLTAGE_1_8V	(1)	Sets the operating voltage of the dedicated pin used by the SPIBSC to 1.8V
SPIBSC_FLASH_SPI	(0)	Sets the flash memory type to serial flash memory
SPIBSC_MODE_MANUAL	(0)	Sets the SPIBSC operating mode to manual mode
SPIBSC_MODE_XIP	(1)	Sets the SPIBSC operating mode to external address space read mode
SPIBSC_CMNCR_BSZ_SINGLE	(0)	Sets the number of connected serial flash memories to 1
SPIBSC_CMNCR_BSZ_DUAL	(1)	Sets the number of connected serial flash memories to 2 (Unsupported)
SPIBSC_QE_DISABLE	(0)	Sets the serial flash memory (MX25L51245GXD) Status Register QE bit to 0.
SPIBSC_QE_ENABLE	(1)	Sets the serial flash memory (MX25L51245GXD) Status Register QE bit to 1.
SPIBSC_1BIT_WIDTH	(0)	Sets the command, optional command, address, option data, and transfer data bit width to 1 bit
SPIBSC_4BIT_WIDTH	(2)	Sets the command, optional command, address, option data, and transfer data bit width to 4 bits
SPIBSC_OUTPUT_DISABLE	(0)	Disables the output of command, optional command, address, option data, and dummy cycle
SPIBSC_OUTPUT_ENABLE	(1)	Enables the output of command, optional command, and dummy cycle
SPIBSC_OUTPUT_ADDR_24	(0x07)	Outputs a 24-bit address
SPIBSC_OUTPUT_ADDR_32	(0x0f)	Outputs a 32-bit address
SPIBSC_OUTPUT_OPD_3	(0x08)	Outputs option data OPD3
SPIBSC_OUTPUT_OPD_32	(0x0c)	Outputs option data OPD3 and OPD2
SPIBSC_OUTPUT_OPD_321	(0x0e)	Outputs option data OPD3, OPD2, and OPD1
SPIBSC_OUTPUT_OPD_3210	(0x0f)	Outputs option data OPD3, OPD2, OPD1, and OPD0

Table 5.9 Constants Used in the Loader Program (2/2)

Constant	Setting value	Description
SPIBSC_DUMMY_02CYC	(1)	Sets the number of dummy cycles to 2
SPIBSC_DUMMY_03CYC	(2)	Sets the number of dummy cycles to 3
SPIBSC_DUMMY_04CYC	(3)	Sets the number of dummy cycles to 4
SPIBSC_DUMMY_05CYC	(4)	Sets the number of dummy cycles to 5
SPIBSC_DUMMY_06CYC	(5)	Sets the number of dummy cycles to 6
SPIBSC_DUMMY_07CYC	(6)	Sets the number of dummy cycles to 7
SPIBSC_DUMMY_08CYC	(7)	Sets the number of dummy cycles to 8
SPIBSC_DUMMY_09CYC	(8)	Sets the number of dummy cycles to 9
SPIBSC_DUMMY_10CYC	(9)	Sets the number of dummy cycles to 10
SPIBSC_DUMMY_11CYC	(10)	Sets the number of dummy cycles to 11
SPIBSC_DUMMY_12CYC	(11)	Sets the number of dummy cycles to 12
SPIBSC_DUMMY_13CYC	(12)	Sets the number of dummy cycles to 13
SPIBSC_DUMMY_14CYC	(13)	Sets the number of dummy cycles to 14
SPIBSC_DUMMY_15CYC	(14)	Sets the number of dummy cycles to 15
SPIBSC_DUMMY_16CYC	(15)	Sets the number of dummy cycles to 16
SPIBSC_DUMMY_17CYC	(16)	Sets the number of dummy cycles to 17
SPIBSC_DUMMY_18CYC	(17)	Sets the number of dummy cycles to 18
SPIBSC_DUMMY_19CYC	(18)	Sets the number of dummy cycles to 19
SPIBSC_DUMMY_20CYC	(19)	Sets the number of dummy cycles to 20
SPIBSC_DDR_TRANSFER	(1)	Sets the address, option data, and data transfer to DDR transfer
SIPBSC_SDR_TRANSFER	(0)	Sets the address, option data, and data transfer to SDR transfer
SPIBSC_NO_DATA	(0x00)	Specifies that data is not read/written in manual mode
SPIBSC_READ_DATA	(0x01)	Specifies that data is read in manual mode
SPIBSC_WRITE_DATA	(0x02)	Specifies that data is written in manual mode
SPIBSC_TEST_PATTERN_EXPECTED_VALUE	(0xAA00FF55)	The expected value to be used when determining that timing adjustment is OK
SPIBSC_QSPI_IO_OUTPUT_0	(0x00)	Sets the QSPIn_IO output value to 0
SPIBSC_QSPI_IO_OUTPUT_1	(0x01)	Sets the QSPIn_IO output value to 1
SPIBSC_QSPI_IO_OUTPUT_PREVIOUS	(0x02)	Sets the QSPIn_IO output value to previous state
SPIBSC_QSPI_IO_OUTPUT_HI_Z	(0x03)	Sets the QSPIn_IO output value to HI-Z

5.6 List of Structures/Unions Used by the Loader Program

Table 5.10 to Table 5.21 list the structures used by the loader program.

Table 5.10 Structure for Configuring the SPIBSC Register (st_spibsc_config_t)

Member	Description
uint8_t flash_type	Specifies the type of connected flash memory. SPIBSC_FLASH_SPI: SPI FLASH
uint8_t flash_num	Specifies the number of serial flash connections. SPIBSC_CMNCR_BSZ_SINGLE: 1 connection SPIBSC_CMNCR_BSZ_DUAL: 2 connections (unsupported)
uint8_t flash_port_voltage	Specifies the voltage setting of the SPIBSC's dedicated pin. SPIBSC_PORT_VOLTAGE_3_3V: 3.3V SPIBSC_PORT_VOLTAGE_1_8V: 1.8V

Table 5.11 Structure for Configuring the SPIBSC External Address Space Read Mode (st_spibsc_xip_config_t) (1/5)

Member	Description
uint8_t command_name[20]	Character string that identifies the read command • This member does not affect the register settings.
uint8_t cmd	Read command • Specifies the read command output to the serial flash memory when converting read operations targeting the SPI multi-I/O bus space to SPI communication. • The setting value of this member is set in the CMD[7:0] bit field in the data read command setting register (DRCMR).
uint8_t cmd_width	Read command bit width • Specifies the bit width used when issuing read commands. • Settable values: SPIBSC_1BIT_WIDTH: 1-bit width SPIBSC_4BIT_WIDTH: 4-bit width • The setting value of this member is set in the CDB[1:0] bit field in the data read enable setting register (DRENR).
uint8_t cmd_output_enable	Read command enable • Selects whether or not a read command is issued. • Settable values: SPIBSC_OUTPUT_DISABLE: Not issued SPIBSC_OUTPUT_ENABLE: Issued • The setting value of this member is set in the CDE bit field in the data read enable setting register (DRENR).
uint8_t ocmd	Optional command • Specifies the optional command output to the serial flash memory when converting read operations targeting the SPI multi-I/O bus space to SPI communication. • The setting value of this member is set in the OCMD[7:0] bit field in the data read command setting register (DRCMR).
uint8_t ocmd_width	Optional command bit width • Specifies the bit width used when issuing optional commands. • Settable values: SPIBSC_1BIT_WIDTH: 1-bit width SPIBSC_4BIT_WIDTH: 4-bit width • The setting value of this member is set in the OCDB[1:0] bit field in the data read enable setting register (DRENR).
uint8_t ocmd_output_enable	Optional command enable • Selects whether or not an optional command is issued. • Settable values: SPIBSC_OUTPUT_DISABLE: Not issued SPIBSC_OUTPUT_ENABLE: Issued • The setting value of this member is set in the OCDE bit field in the data read enable setting register (DRENR).

Table 5.12 Structure for Configuring the SPIBSC External Address Space Read Mode (st_spibsc_xip_config_t) (2/5)

Member	Description
uint8_t addr_width	Address bit width • Specifies the address bit width output to the serial flash memory when converting read operations targeting the SPI multi-I/O bus space to SPI communication. • Settable values: SPIBSC_1BIT_WIDTH: 1-bit width SPIBSC_4BIT_WIDTH: 4-bit width • The setting value of this member is set in the ADB[1:0] bit field in the data read enable setting register (DRENR).
uint8_t addr_output_enable	Address enable • Specifies the address output to the serial flash memory when converting read operations targeting the SPI multi-I/O bus space to SPI communication. • Settable values: SPIBSC_OUTPUT_DISABLE: Not output SPIBSC_OUTPUT_ADDR_24: 24-bit address output SPIBSC_OUTPUT_ADDR_32: 32-bit address output • The setting value of this member is set in the ADE[3:0] bit field in the data read enable setting register (DRENR).
uint8_t addr_ddr_enable	Address DDR enable • Selects SDR/DDR transfer for the address output in external address space read mode. • Settable values: SPIBSC_SDR_TRANSFER: SDR transfer SPIBSC_DDR_TRANSFER: DDR transfer • The setting value of this member is set in the ADDRE bit field in the data read DDR enable register (DRDRENR).
uint8_t reserve1	Reserve data • This member is not referenced in the sample code.
uint8_t reserve2	Reserve data • This member is not referenced in the sample code.
uint8_t opdata_width	Option data bit width • Specifies the bit width used when issuing option data. • Settable values: SPIBSC_1BIT_WIDTH: 1-bit width SPIBSC_4BIT_WIDTH: 4-bit width • The setting value of this member is set in the OPDB[1:0] bit field in the data read enable setting register (DRENR).
uint8_t opdata_output_enable	Option data enable • Selects whether or not the option data is issued. • Settable values: SPIBSC_OUTPUT_DISABLE: Not output SPIBSC_OUTPUT_OPD_3: OPD3 output SPIBSC_OUTPUT_OPD_32: OPD3, OPD2 output SPIBSC_OUTPUT_OPD_321: OPD3, OPD2, OPD1 output SPIBSC_OUTPUT_OPD_3210: OPD3, OPD2, OPD1, OPD0 output • The setting value of this member is set in the OPDE[3:0] bit field in the data read enable setting register (DRENR).

Table 5.13 Structure for Configuring the SPIBSC External Address Space Read Mode (st_spibsc_xip_config_t) (3/5)

Member	Description
uint8_t opdata_ddr_enable	Option data DDR enable • Selects SDR/DDR transfer for the option data output in external address space read mode. • Settable values: SPIBSC_SDR_TRANSFER: SDR transfer SPIBSC_DDR_TRANSFER: DDR transfer • The setting value of this member is set in the OPDRE bit field in the data read DDR enable register (DRDRENR).
uint8_t opd3 uint8_t opd2 uint8_t opd1 uint8_t opd0	Option data • Specifies the option data output to the serial flash memory when converting read operations targeting the SPI multi-I/O bus space to SPI communication. • The setting value of this member is set in the OPD3[7:0], OPD2[7:0], OPD1[7:0], and OPD0[7:0] bit fields in the data read option setting register (DROPR).
uint8_t reserve3	Reserve data This member is not referenced in the sample code.
uint8_t dummy_cycle_enable	Dummy cycle enable • Selects whether or not dummy cycles are inserted. • Settable values: SPIBSC_OUTPUT_DISABLE: Not inserted SPIBSC_OUTPUT_ENABLE: Inserted • The setting value of this member is set in the DME bit field in the data read enable setting register (DRENR).
uint8_t dummy_cycle_count	Number of dummy cycles • Sets the number of inserted dummy cycles. • Settable values: SPIBSC_DUMMY_02CYC to SPIBSC_DUMMY_20CYC • The setting value of this member is set in the DMCYC[4:0] bit field in the data read dummy cycle setting register (DRDMCR).
uint8_t data_width	Data read bit width • Specifies the data read bit width of the serial flash memory when converting read operations targeting the SPI multi-I/O bus space to SPI communication. • Settable values: SPIBSC_1BIT_WIDTH: 1-bit width SPIBSC_4BIT_WIDTH: 4-bit width • The setting value of this member is set in the DRDB[1:0] bit field in the data read enable setting register (DRENR).
uint8_t data_ddr_enable	Transfer data DDR enable • Selects SDR/DDR transfer for the data transferred in external address space read mode. • Settable values: SPIBSC_SDR_TRANSFER: SDR transfer SPIBSC_DDR_TRANSFER: DDR transfer • The setting value of this member is set in the DRDRE bit field in the data read DDR enable register (DRDRENR).

Table 5.14 Structure for Configuring the SPIBSC External Address Space Read Mode (st_spibsc_xip_config_t) (4/5)

Member	Description
uint8_t cmncr_moiio3	Level for QSPIn_IO3 while SSL negated. • Specify the level after finished the data transfer. • Settable values: SPIBSC_QSPI_IO_OUTPUT_0: The output value is 0. SPIBSC_QSPI_IO_OUTPUT_1: The output value is 1. SPIBSC_QSPI_IO_OUTPUT_PREVIOUS: The output value is that of the last bit in the previous transfer (the pin is placed in the Hi-Z state if that was the case in the previous transfer). SPIBSC_QSPI_IO_OUTPUT_HI_Z: The pin is placed in the Hi-Z state. • The setting value of this member is set in the MOIIO3[1:0] bit field in the Common Control Register (CMNCR).
uint8_t cmncr_moiio2	Level for QSPIn_IO2 while SSL negated. • Specify the level after finished the data transfer. • Settable values: SPIBSC_QSPI_IO_OUTPUT_0: The output value is 0. SPIBSC_QSPI_IO_OUTPUT_1: The output value is 1. SPIBSC_QSPI_IO_OUTPUT_PREVIOUS: The output value is that of the last bit in the previous transfer (the pin is placed in the Hi-Z state if that was the case in the previous transfer). SPIBSC_QSPI_IO_OUTPUT_HI_Z: The pin is placed in the Hi-Z state. • The setting value of this member is set in the MOIIO2[1:0] bit field in the Common Control Register (CMNCR).
uint8_t cmncr_moiio1	Level for QSPIn_IO1 while SSL negated. • Specify the level after finished the data transfer. • Settable values: SPIBSC_QSPI_IO_OUTPUT_0: The output value is 0. SPIBSC_QSPI_IO_OUTPUT_1: The output value is 1. SPIBSC_QSPI_IO_OUTPUT_PREVIOUS: The output value is that of the last bit in the previous transfer (the pin is placed in the Hi-Z state if that was the case in the previous transfer). SPIBSC_QSPI_IO_OUTPUT_HI_Z: The pin is placed in the Hi-Z state. • The setting value of this member is set in the MOIIO1[1:0] bit field in the Common Control Register (CMNCR).
uint8_t cmncr_moiio0	Level for QSPIn_IO0 while SSL negated. • Specify the level after finished the data transfer. • Settable values: SPIBSC_QSPI_IO_OUTPUT_0: The output value is 0. SPIBSC_QSPI_IO_OUTPUT_1: The output value is 1. SPIBSC_QSPI_IO_OUTPUT_PREVIOUS: The output value is that of the last bit in the previous transfer (the pin is placed in the Hi-Z state if that was the case in the previous transfer). SPIBSC_QSPI_IO_OUTPUT_HI_Z: The pin is placed in the Hi-Z state. • The setting value of this member is set in the MOIIO0[1:0] bit field in the Common Control Register (CMNCR).

Table 5.15 Structure for Configuring the SPIBSC External Address Space Read Mode (st_spibsc_xip_config_t) (5/5)

Member	Description
uint8_t cmncr_io3fv	Level for QSPIn_IO3 while 1-bit width transfer. • Specify the level while the data transfer is by 1-bit width. • Settable values: SPIBSC_QSPI_IO_OUTPUT_0: The output value is 0. SPIBSC_QSPI_IO_OUTPUT_1: The output value is 1. SPIBSC_QSPI_IO_OUTPUT_PREVIOUS: The output value is that of the last bit in the previous transfer (the pin is placed in the Hi-Z state if that was the case in the previous transfer). SPIBSC_QSPI_IO_OUTPUT_HI_Z: The pin is placed in the Hi-Z state. • The setting value of this member is set in the IO3FV[1:0] bit field in the Common Control Register (CMNCR).
uint8_t cmncr_io2fv	Level for QSPIn_IO2 while 1-bit width transfer. • Specify the level while the data transfer is by 1-bit width. • Settable values: SPIBSC_QSPI_IO_OUTPUT_0: The output value is 0. SPIBSC_QSPI_IO_OUTPUT_1: The output value is 1. SPIBSC_QSPI_IO_OUTPUT_PREVIOUS: The output value is that of the last bit in the previous transfer (the pin is placed in the Hi-Z state if that was the case in the previous transfer). SPIBSC_QSPI_IO_OUTPUT_HI_Z: The pin is placed in the Hi-Z state. • The setting value of this member is set in the IO2FV[1:0] bit field in the Common Control Register (CMNCR).
uint8_t cmncr_io0fv	Level for QSPIn_IO0 while 1-bit width read transfer. • Specify the level while the data read transfer is by 1-bit width. • Settable values: SPIBSC_QSPI_IO_OUTPUT_0: The output value is 0. SPIBSC_QSPI_IO_OUTPUT_1: The output value is 1. SPIBSC_QSPI_IO_OUTPUT_PREVIOUS: The output value is that of the last bit in the previous transfer (the pin is placed in the Hi-Z state if that was the case in the previous transfer). SPIBSC_QSPI_IO_OUTPUT_HI_Z: The pin is placed in the Hi-Z state. • The setting value of this member is set in the IO0FV[1:0] bit field in the Common Control Register (CMNCR).

Note: cmncr_io0fv must be set "SPIBSC_QSPI_IO_OUTPUT_HI_Z" when data read transfer size is not 1-bit (data_width is not set "SPIBSC_1BIT_WIDTH").

Table 5.16 SPIBSC Manual Mode Settings Structure (st_spibsc_manual_mode_command_config_t) (1/6)

Member	Description
uint8_t command_name[20]	Character string that identifies the SPI command • This member does not affect the register settings.
uint8_t cmd	Command • Specifies the command output in manual mode. • The setting value of this member is set in the CMD[7:0] bit field in the manual mode command setting register (SMCMR).
uint8_t cmd_width	Command bit width • Specifies the bit width used when issuing commands. • Settable values: SPIBSC_1BIT_WIDTH: 1-bit width SPIBSC_4BIT_WIDTH: 4-bit width • The setting value of this member is set in the CDB[1:0] bit field in the manual mode enable setting register (SMENR).
uint8_t cmd_output_enable	Command enable • Selects whether or not a command is issued. • Settable values: SPIBSC_OUTPUT_DISABLE: Not issued SPIBSC_OUTPUT_ENABLE: Issued • The setting value of this member is set in the CDE bit field in the manual mode enable setting register (SMENR).
uint8_t ocmd	Optional command • Specifies the optional command output in manual mode. • The setting value of this member is set in the OCMD[7:0] bit field in the manual mode command setting register (SMCMR).
uint8_t ocmd_width	Optional command bit width • Specifies the optional command bit width in manual mode. • Settable values: SPIBSC_1BIT_WIDTH: 1-bit width SPIBSC_4BIT_WIDTH: 4-bit width • The setting value of this member is set in the OCDB[1:0] bit field in the manual mode enable setting register (SMENR).
uint8_t ocmd_enable	Optional command enable • Specifies whether or not the optional command is output in manual mode. • Settable values: SPIBSC_OUTPUT_DISABLE: Not output SPIBSC_OUTPUT_ENABLE: Output • The setting value of this member is set in the OCDE bit field in the manual mode enable setting register (SMENR).

Table 5.17 SPIBSC Manual Mode Settings Structure (st_spibsc_manual_mode_command_config_t) (2/6)

Member	Description
uint8_t addr_width	Address bit width • Specifies the address bit width in manual mode. • Settable values: SPIBSC_1BIT_WIDTH: 1-bit width SPIBSC_4BIT_WIDTH: 4-bit width • The setting value of this member is set in the ADB[1:0] bit field in the manual mode enable setting register (SMENR).
uint8_t addr_output_enable	Address enable • Specifies whether or not the address is output in manual mode. • Settable values: SPIBSC_OUTPUT_DISABLE: Not output SPIBSC_OUTPUT_ADDR_24: ADR[23:0] output SPIBSC_OUTPUT_ADDR_32: ADR[31:0] output • The setting value of this member is set in the ADE[3:0] bit field in the manual mode enable setting register (SMENR).
uint8_t addr_sdr_ddr	Address DDR enable • Selects SDR/DDR transfer for the address output in manual mode. • Settable values: SPIBSC_SDR_TRANSFER: SDR transfer SPIBSC_DDR_TRANSFER: DDR transfer • The setting value of this member is set in the ADDRE bit field in the manual mode DDR enable register (SMDRENR).
uint8_t opdata_width	Option data bit width • Specifies the option data bit width in manual mode. • Settable values: SPIBSC_1BIT_WIDTH: 1-bit width SPIBSC_4BIT_WIDTH: 4-bit width • The setting value of this member is set in the OPDB[1:0] bit field in the manual mode enable setting register (SMENR).
uint8_t opdata_output_enable	Option data enable • Specifies whether or not the option data is output in manual mode. • Settable values: SPIBSC_OUTPUT_DISABLE : Not output SPIBSC_OUTPUT_OPD_3 : OPD3 output SPIBSC_OUTPUT_OPD_32 : OPD3, OPD2 output SPIBSC_OUTPUT_OPD_321 : OPD3, OPD2, OPD1 output SPIBSC_OUTPUT_OPD_3210 : OPD3, OPD2, OPD1, OPD0 output • The setting value of this member is set in the OPDE[3:0] bit field in the manual mode enable setting register (SMENR).

Table 5.18 SPIBSC Manual Mode Settings Structure (st_spibsc_manual_mode_command_config_t) (3/6)

Member	Description
uint8_t opdata_ddr_enable	Option data DDR enable • Selects SDR/DDR transfer for the option data output in manual mode. • Settable values: SPIBSC_SDR_TRANSFER: SDR transfer SPIBSC_DDR_TRANSFER: DDR transfer • The setting value of this member is set in the OPDRE bit field in the manual mode DDR enable register (SMDRENR).
uint8_t opd3 uint8_t opd2 uint8_t opd1 uint8_t opd0	Option data • Specifies the option data output in manual mode. • The setting value of this member is set in the OPD3[7:0], OPD2[7:0], OPD1[7:0], and OPD0[7:0] bit fields in the manual mode option setting register (SMOPR).
uint8_t reserve3	Reserve data This member is not referenced in the sample code.
uint8_t dummy_cycle_output_enable	Dummy cycle enable • Specifies whether or not dummy cycles are inserted in manual mode. • Settable values: SPIBSC_OUTPUT_DISABLE: Not inserted SPIBSC_OUTPUT_ENABLE: Inserted • The setting value of this member is set in the DME bit field in the manual mode enable setting register (SMENR).
uint8_t dummy_cycle_count	Number of dummy cycles • Sets the number of inserted dummy cycles. • Settable values: SPIBSC_DUMMY_02CYC to SPIBSC_DUMMY_20CYC • The setting value of this member is set in the DMCYC[4:0] bit field in the manual mode dummy cycle setting register (SMDMCR).
uint8_t transfer_data_width	Transfer data bit width • Specifies the transfer data bit width in manual mode. • Settable values: SPIBSC_1BIT_WIDTH: 1-bit width SPIBSC_4BIT_WIDTH: 4-bit width • The setting value of this member is set in the SPIDB[1:0] bit field in the manual mode enable setting register (SMENR).
uint8_t transfer_data_sdr_ddr	Transfer data DDR enable • Selects SDR/DDR transfer for the data transferred in manual mode. • Settable values: SPIBSC_SDR_TRANSFER: SDR transfer SPIBSC_DDR_TRANSFER: DDR transfer • The setting value of this member is set in the SPIDRE bit field in the manual mode DDR enable register (SMDRENR).

Table 5.19 SPIBSC Manual Mode Settings Structure (st_spibsc_manual_mode_command_config_t) (4/6)

Member	Description
uint8_t reserve1	Reserve data This member is not referenced in the sample code.
uint8_t reserve2	Reserve data This member is not referenced in the sample code.
uint8_t cmncr_moiio3	Level for QSPIn_IO3 while SSL negated. • Specify the level after finished the data transfer. • Settable values: SPIBSC_QSPI_IO_OUTPUT_0: The output value is 0. SPIBSC_QSPI_IO_OUTPUT_1: The output value is 1. SPIBSC_QSPI_IO_OUTPUT_PREVIOUS: The output value is that of the last bit in the previous transfer (the pin is placed in the Hi-Z state if that was the case in the previous transfer). SPIBSC_QSPI_IO_OUTPUT_HI_Z: The pin is placed in the Hi-Z state. • The setting value of this member is set in the MOIIO3[1:0] bit field in the Common Control Register (CMNCR).
uint8_t cmncr_moiio2	Level for QSPIn_IO2 while SSL negated. • Specify the level after finished the data transfer. • Settable values: SPIBSC_QSPI_IO_OUTPUT_0: The output value is 0. SPIBSC_QSPI_IO_OUTPUT_1: The output value is 1. SPIBSC_QSPI_IO_OUTPUT_PREVIOUS: The output value is that of the last bit in the previous transfer (the pin is placed in the Hi-Z state if that was the case in the previous transfer). SPIBSC_QSPI_IO_OUTPUT_HI_Z: The pin is placed in the Hi-Z state. • The setting value of this member is set in the MOIIO2[1:0] bit field in the Common Control Register (CMNCR).
uint8_t cmncr_moiio1	Level for QSPIn_IO1 while SSL negated. • Specify the level after finished the data transfer. • Settable values: SPIBSC_QSPI_IO_OUTPUT_0: The output value is 0. SPIBSC_QSPI_IO_OUTPUT_1: The output value is 1. SPIBSC_QSPI_IO_OUTPUT_PREVIOUS: The output value is that of the last bit in the previous transfer (the pin is placed in the Hi-Z state if that was the case in the previous transfer). SPIBSC_QSPI_IO_OUTPUT_HI_Z: The pin is placed in the Hi-Z state. • The setting value of this member is set in the MOIIO1[1:0] bit field in the Common Control Register (CMNCR).

Table 5.20 SPIBSC Manual Mode Settings Structure (st_spibsc_manual_mode_command_config_t) (5/6)

Member	Description
uint8_t cmncr_moiio0	Level for QSPIn_IO0 while SSL negated. • Specify the level after finished the data transfer. • Settable values: SPIBSC_QSPI_IO_OUTPUT_0: The output value is 0. SPIBSC_QSPI_IO_OUTPUT_1: The output value is 1. SPIBSC_QSPI_IO_OUTPUT_PREVIOUS: The output value is that of the last bit in the previous transfer (the pin is placed in the Hi-Z state if that was the case in the previous transfer). SPIBSC_QSPI_IO_OUTPUT_HI_Z: The pin is placed in the Hi-Z state. • The setting value of this member is set in the MOIIO0[1:0] bit field in the Common Control Register (CMNCR).
uint8_t cmncr_io3fv	Level for QSPIn_IO3 while 1-bit width transfer. • Specify the level while the data transfer is by 1-bit width. • Settable values: SPIBSC_QSPI_IO_OUTPUT_0: The output value is 0. SPIBSC_QSPI_IO_OUTPUT_1: The output value is 1. SPIBSC_QSPI_IO_OUTPUT_PREVIOUS: The output value is that of the last bit in the previous transfer (the pin is placed in the Hi-Z state if that was the case in the previous transfer). SPIBSC_QSPI_IO_OUTPUT_HI_Z: The pin is placed in the Hi-Z state. • The setting value of this member is set in the IO3FV[1:0] bit field in the Common Control Register (CMNCR).
uint8_t cmncr_io2fv	Level for QSPIn_IO2 while 1-bit width transfer. • Specify the level while the data transfer is by 1-bit width. • Settable values: SPIBSC_QSPI_IO_OUTPUT_0: The output value is 0. SPIBSC_QSPI_IO_OUTPUT_1: The output value is 1. SPIBSC_QSPI_IO_OUTPUT_PREVIOUS: The output value is that of the last bit in the previous transfer (the pin is placed in the Hi-Z state if that was the case in the previous transfer). SPIBSC_QSPI_IO_OUTPUT_HI_Z: The pin is placed in the Hi-Z state. • The setting value of this member is set in the IO2FV[1:0] bit field in the Common Control Register (CMNCR).

Table 5.21 SPIBSC Manual Mode Settings Structure (st_spibsc_manual_mode_command_config_t) (6/6)

Member	Description
uint8_t cmncr_io0fv	Level for QSPIn_IO0 while 1-bit width read transfer. • Specify the level while the data read transfer is by 1-bit width. • Settable values (Note): SPIBSC_QSPI_IO_OUTPUT_0: The output value is 0. SPIBSC_QSPI_IO_OUTPUT_1: The output value is 1. SPIBSC_QSPI_IO_OUTPUT_PREVIOUS: The output value is that of the last bit in the previous transfer (the pin is placed in the Hi-Z state if that was the case in the previous transfer). SPIBSC_QSPI_IO_OUTPUT_HI_Z: The pin is placed in the Hi-Z state. • The setting value of this member is set in the IO0FV[1:0] bit field in the Common Control Register (CMNCR).

Note: cmncr_io0fv must be set "SPIBSC_QSPI_IO_OUTPUT_HI_Z" when data read transfer size is not 1-bit (data_width is not set "SPIBSC_1BIT_WIDTH").

5.7 List of Variables for Loader Program

Table 5.22 lists the Variables Used in the Loader Program.

Table 5.22 Variables Used in the Loader Program

Variable name	Description	Comments
st_spibsc_xip_config_t gs_read_table[0]	For external address space read mode Settings table data for DDR read command	Refer to Table 6.8
st_spibsc_xip_config_t gs_read_table[1]	For external address space read mode Settings table data for SDR read command	Refer to Table 6.9
st_spibsc_manual_mode_command_config_t gs_command_table[0]	For manual mode Settings table data for READ STATUS command	Refer to Table 6.10
st_spibsc_manual_mode_command_config_t gs_command_table[1]	For manual mode Settings table data for READ CONFIGURATION command	Refer to Table 6.11
st_spibsc_manual_mode_command_config_t gs_command_table[2]	For manual mode Settings table data for WRITE ENABLE command	Refer to Table 6.12
st_spibsc_manual_mode_command_config_t gs_command_table[3]	For manual mode Settings table data for WRITE STATUS command	Refer to Table 6.13

5.8 List of Functions Used in the Loader Program

The sample code comprises interface functions (API functions) for using peripheral functions, user-defined functions (functions called by API functions) which must be prepared by the user for the purpose of the target system, and sample functions which are necessary for the sample code to operate.

For the sample code of the loader program, Table 5.23 lists the Sample Functions, Table 5.24 lists the API Functions, and Table 5.25 lists the User-Defined Functions.

Table 5.23 Sample Functions

Function	Description
reset_handler	Reset handler processing (assembler function)
INITSCT	Program section initialization (assembler function)
R_SC_HardwareSetup	Initial setting of hardware used for booting
r_memclk_setup	Memory clock setting processing

Table 5.24 API Functions

Function	Description
R_SPIBSC_Setup	SPIBSC and serial flash memory initial setting
R_SPIBSC_Init	SPIBSC initial setting
R_SPIBSC_ChangeMode	SPIBSC operating mode setting
R_SPIBSC_SPICMDIssue	Issuance of SPI command to serial flash memory (manual mode)
R_SPIBSC_XipStopAccess	Stop access to serial flash memory
R_SPIBSC_FlushReadCache	Clear SPIBSC read cache
R_SPIBSC_AdjustPhy	SPIBSC data input timing calibration for DDR transfer
R_CPG_InitialiseHwIf	CPG initialization processing

Table 5.25 User-Defined Functions

Function	Description
Userdef_SPIBSC_SFLASH_SetMode	Serial flash memory register setting
Userdef_SPIBSC_SFLASH_ReadStatus	Serial flash memory status register read
Userdef_SPIBSC_SFLASH_ReadConfig	Serial flash memory configuration register read
Userdef_SPIBSC_SFLASH_WriteStatus	Serial flash memory status register and configuration register write
Userdef_SPIBSC_SFLASH_WriteEnable	Serial flash memory write enable
Userdef_SPIBSC_SFLASH_WaitReady	Serial flash memory write completion wait
Userdef_PreHardwareSetup	Necessary hardware initialization processing before the SPIBSC initialization process
Userdef_PostHardwareSetup	Necessary hardware initialization processing after the SPIBSC initialization process

5.9 Function Specification

Specifications of the functions of the sample code loader program are listed below.

reset_handler	
Outline	Reset handler processing
Declaration	reset_handler
Description	The entry function of the loader program.
Arguments	None
Return Value	None

INITSCT		
Outline	Program section initialization	
Declaration	void INITSCT(void)	
Description	The data with initial values which must be transferred to the RAM area (including code and constant data that must be executed in the RAM area) is transferred from the ROM area, and the initialization of the RAM area data with no initial values.	
Arguments	p_dtbl	: Pointer to the area where the section information for data with initial values is stored
	p_btbl	: Pointer to the area where the section information for data with no initial values is stored
Return Value	None	

R_SC_HardwareSetup	
Outline	Initial setting of hardware used for booting
Declaration	void R_SC_HardwareSetup(void)
Description	Makes the optimal settings for the used serial flash memory, sets the SPIBSC to external address space read mode, and accesses the serial flash memory. In the sample code, set the serial flash memory (MX25L51245GXD) registers and initialize SPIBSC registers according to the specifications of MX25L51245GXD by calling the R_SPIBSC_Setup function.
Arguments	None
Return Value	None
Precautions	This function cannot be assigned to execute from the serial flash memory. This function must be assigned to an area other than the serial flash memory.

r_memclk_setup	
Outline	Memory clock setting processing
Declaration	void r_memclk_setup (void)
Description	Before the R_SC_HardwareSetup function is executed, the memory clock setting is performed. In the sample code, the timing adjustment processing in SDR mode is performed before accessing the serial flash memory by SDR transfer, and then in order to accelerate the process to transfer the R_SC_HardwareSetup function to the large-capacity on-chip RAM, SPICLK is switched to P1ϕ.
Argument	None
Return Value	None
Precautions	This function cannot be assigned to execute from the serial flash memory. This function must be assigned to an area other than the serial flash memory.

R_SPIBSC_Setup	
Outline	SPIBSC and serial flash memory initial setting
Declaration	void R_SPIBSC_Setup (void)
Description	Makes the optimal settings for the used serial flash memory, sets the SPIBSC to external address space read mode, and accesses the serial flash memory. The following settings are made in the sample code. • Change to read command: H'03→H'EE • Serial flash memory register setting Status register: QE bit set to 1 Configuration register: DC[1:0] and ODS[2:0] bit settings (The DC[1:0] bit setting differs depending on the read command. Refer to "Table 6.5 List of numbers of dummy cycles necessary for the operating frequency of the MX25L51245GXD".) • Change to QSPIn_SPCLK operating frequency: P1ϕ/2→Bϕ/2 • Execute timing adjustment processing for DDR mode by calling the R_SPIBSC_AdjustPhy function, when accessing by DDR transfer
Argument	ddrsdr : Specifying transfer mode SDR or DDR HWSETUP_SPIBSC_USE_DDR : DDR mode HWSETUP_SPIBSC_USE_SDR : SDR mode
Return Value	None
Precautions	This function cannot be assigned to execute from the serial flash memory. This function must be assigned to an area other than the serial flash memory.

R_SPIBSC_Init	
Outline	SPIBSC initial setting
Declaration	e_spibsc_err_t R_SPIBSC_Init(const spibsc_config_t *p_spibsc_config_tbl)
Description	The SPIBSC-related register initial setting is made by the argument spibsc_config_tbl. When this function has ended, external address space read mode settings are made. • Drive strength setting of SPIBSC-related pin • SPIBSC module standby cancellation • SPIBSC clock setting • SPIBSC-related register setting
Arguments	const spibsc_config_t *p_spibsc_config_tbl : Pointer to SPIBSC initial setting table
Return Value	SPIBSC_SUCCESS : Normal end
Precautions	This function cannot be assigned to execute from the serial flash memory. This function must be assigned to an area other than the serial flash memory.

R_SPIBSC_ChangeMode		
Outline	SPIBSC operating mode setting	
Declaration	void R_SPIBSC_ChangeMode(uint8_t mode, uint8_t sdr_ddr, uint8_t table_no)	
Description	The operating mode specified by the argument mode allows access to the serial flash memory in the transfer format specified by the arguments sdr_ddr and table_no.	
Arguments	uint8_t mode	: Operating mode SPIBSC_MODE_MANUAL : Manual mode SPIBSC_MODE_XIP : External address space read mode
	uint8_t sdr_ddr	: Transfer format SPIBSC_DDR_TRANSFER : DDR transfer SPIBSC_SDR_TRANSFER : SDR transfer
	uint8_t table_no	: External address space read mode command setting table number 0: Command setting table 0 is selected (DDR read command) 1: Command setting table 1 is selected (SDR read command)
Return Value	None	
Precautions	This function cannot be assigned to execute from the serial flash memory. This function must be assigned to an area other than the serial flash memory.	

R_SPIBSC_SPICMDIssue	
Outline	Issuance of SPI command to serial flash memory (for manual mode)
Declaration	spibsc_err_t R_SPIBSC_SPICMDIssue(uint8_t table_no, uint32_t addr, uint8_t *write_buff, uint32_t write_size, uint8_t *read_buff, uint32_t read_size)
Description	Uses the configuration table for issuing SPI commands specified by the argument table_no to issue SPI commands. According to the contents of table_no, when a write command is issued, the data stored in the argument *write_buff is written, at the number of bytes specified by the argument write_size from the address specified by the argument addr. When a read command is issued, the data is read at the number of bytes specified by the argument read_size from the address specified by the argument addr, and stored in the area specified by the argument *read_buff.
Arguments	uint8_t table_no : The table storing the setting information for the command used uint32_t addr : Address uint8_t * write_buff : Write buffer pointer uint32_t write_size : Number of bytes to write uint8_t * read_buff : Read buffer pointer uint32_t read_size : Number of bytes to read
Return Value	SPIBSC_SUCCESS : Normal end
Precautions	This function cannot be assigned to execute from the serial flash memory. This function must be assigned to an area other than the serial flash memory.

R_SPIBSC_XipStopAccess	
Outline	Stop access to serial flash memory
Declaration	void R_SPIBSC_XipStopAccess(void)
Description	Negates QSPIn_SSL with external address space read mode and stops access to the serial flash memory. This function waits for the QSPIn_SSL negation to complete before returning to caller process. When SPIBSC-related register settings are being made, this function is called so that the serial flash memory is not accessed.
Arguments	None
Return Value	None
Precautions	This function cannot be assigned to execute from the serial flash memory. This function must be assigned to an area other than the serial flash memory.

R_SPIBSC_FlushReadCache	
Outline	Clear SPIBSC read cache
Declaration	void R_SPIBSC_FlushReadCache(void)
Description	Clears the SPIBSC read cache.
Arguments	None
Return Value	None
Precautions	This function cannot be assigned to execute from the serial flash memory. This function must be assigned to an area other than the serial flash memory.

R_SPIBSC_AdjustPhy	
Outline	SPIBSC data input timing calibration for DDR transfer
Declaration	e_spibsc_err_t R_SPIBSC_AdjustPhy(void)
Description	Calibrates the data input timing from the serial flash memory. Adjusts the input data input timing so that 4-byte data stored in the flash memory can be correctly read. 4-byte data is stored in the address defined by the g_spibsc_test_pattern symbol. Call this function before accessing to the serial flash memory by DDR transfer.
Arguments	None
Return Value	SPIBSC_SUCCESS : Normal end SPIBSC_ERR_INVALID : Abnormal end
Precautions	• Do not call this function if any devices other than the serial flash memory are connected to the SPIBSC. • The g_spibsc_test_pattern constant data area used in this function must be assigned to the serial flash memory. • This function cannot be assigned to execute from the serial flash memory. This function must be assigned to an area other than the serial flash memory. • Use this function when set to manual mode.

R_CPG_InitialiseHwIf	
Outline	CPC initialization processing
Declaration	int_t R_CPG_InitialiseHwIf(void)
Description	Uses the r_cpg_drv_sc_cfg.h CPG configuration data to make CPG register (FRQCR, CKIOSEL, SCLKSEL) settings. In this sample code, CPG settings are made so that the operating frequency in "Operating clock settings" and "SPIBSC clock selection" in "Table 5.4 Settings for the boot startup on-chip ROM program and loader program (3/3)" is used.
Arguments	None None
Return Value	DRV_SUCCESS : Normal end DRV_ERROR : r_cpg_drv_sc_cfg.h CPG configuration data is wrong
Precautions	This function must be assigned to the large-capacity on-chip RAM.

Userdef_SPIBSC_SFLASH_SetMode	
Outline	Serial flash memory register setting
Declaration	void Userdef_SPIBSC_SFLASH_SetMode(uint8_t mode, uint8_t sdr_ddr)
Description	Implement a processing that sets to the serial flash memory registers so that access is possible in the transfer format specified by the argument sdr_ddr via access at the bit width specified by the argument mode, according to the specifications of the serial flash memory to be used. In the sample code, MX25L51245GXD register settings are made and access is set via DDR transfer at 4-bit width.
Arguments	uint8_t mode : Operating mode SPIBSC_QE_DISABLE: Single mode (bit width: 1 bit) SPIBSC_QE_ENABLE: Quad mode (bit width: 4 bits) uint8_t sdr_ddr : Transfer format SPIBSC_DDR_TRANSFER : DDR transfer SPIBSC_SDR_TRANSFER : SDR transfer
Return Value	None
Precautions	This function cannot be assigned to execute from the serial flash memory. This function must be assigned to an area other than the serial flash memory.

Userdef_SPIBSC_SFLASH_ReadStatus	
Outline	Serial flash memory status register read
Declaration	void Userdef_SPIBSC_SFLASH_ReadStatus(uint8_t *p_status)
Description	Implement a processing to read the serial flash memory status register and to store the data read with the argument *p_status, according to the specifications of the serial flash memory to be used. In the sample code, a processing to read the status register of MX25L51245GXD is performed.
Arguments	uint8_t *p_status : The value read from the status register
Return Value	None
Precautions	This function cannot be assigned to execute from the serial flash memory. This function must be assigned to an area other than the serial flash memory.

Userdef_SPIBSC_SFLASH_ReadConfig	
Outline	Serial flash memory configuration register read
Declaration	void Userdef_SPIBSC_SFLASH_ReadConfig(uint8_t *p_config)
Description	Implement a processing to read the serial flash memory configuration register and to store the data read with the argument *p_config, according to the specifications of the serial flash memory to be used. In the sample code, a processing to read the configuration register of MX25L51245GXD is performed.
Arguments	uint8_t *p_config : The value read from the configuration register
Return Value	None
Precautions	This function cannot be assigned to execute from the serial flash memory. This function must be assigned to an area other than the serial flash memory.

Userdef_SPIBSC_SFLASH_WriteStatus	
Outline	Serial flash memory status register and configuration register write
Declaration	void Userdef_SPIBSC_SFLASH_WriteStatus(uint8_t *p_status, uint8_t *p_config)
Description	Implement a processing that sets the values specified by the arguments *p_status and *p_config to the serial flash memory status register and configuration register respectively, according to the specifications of the serial flash memory to be used. In the sample code, a processing to read the status register and configuration register of MX25L51245GXD is performed.
Arguments	uint8_t *p_status : Setting in status register uint8_t *p_config : Setting in config register
Return Value	None
Precautions	This function cannot be assigned to execute from the serial flash memory. This function must be assigned to an area other than the serial flash memory.

Userdef_SPIBSC_SFLASH_WriteEnable	
Outline	Serial flash memory write enable
Declaration	void Userdef_SPIBSC_SFLASH_WriteEnable(void)
Description	Implement the processing that enables the serial flash memory registers for writes, according to the specifications of the serial flash memory to be used. In the sample code, processing to issue "Write Enable Register (WREN)" commands to MX25L51245GXD is performed.
Arguments	None
Return Value	None
Precautions	This function cannot be assigned to execute from the serial flash memory. This function must be assigned to an area other than the serial flash memory.

Userdef_SPIBSC_SFLASH_WaitReady	
Outline	Serial flash memory write completion wait
Declaration	void Userdef_SPIBSC_SFLASH_WaitReady(void)
Description	Implement the processing that waits for the write to the serial flash memory to be completed, according to the specifications of the serial flash memory to be used. In the sample code, processing to issue "Read Status Register (RDSR)" commands to MX25L51245GXD and to wait for the write to be completed is performed by referencing the status register contents.
Arguments	None
Return Value	None
Precautions	This function cannot be assigned to execute from the serial flash memory. This function must be assigned to an area other than the serial flash memory.

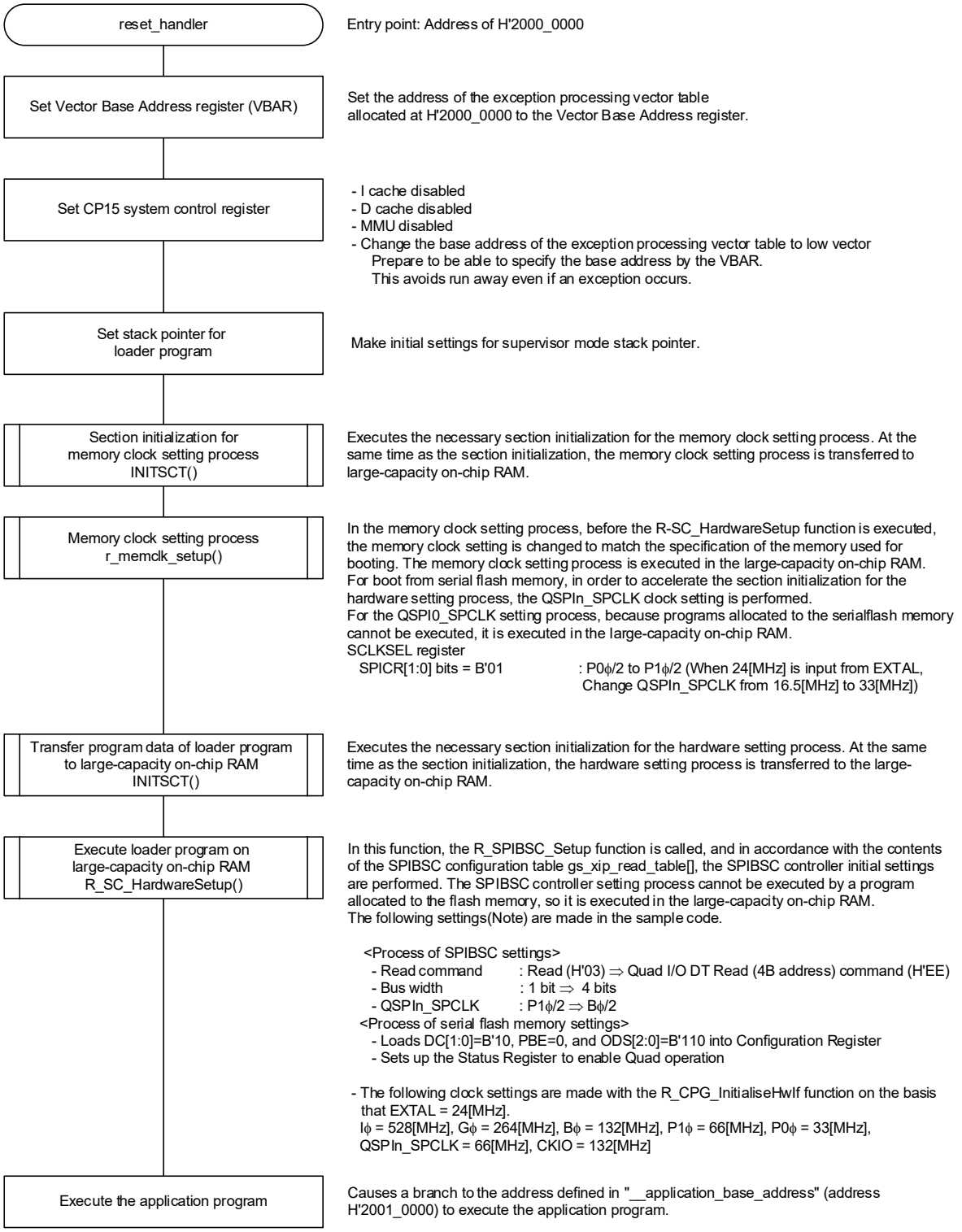
Userdef_PreHardwareSetup	
Outline	Necessary hardware initialization processing before the SPIBSC initialization process
Declaration	void Userdef_PreHardwareSetup (void)
Description	This is the user definable function to describe the hardware initialization process which is required to be executed before the SPIBSC initialization. Nothing is performed in the sample code.
Argument	None
Return Value	None
Precautions	If this function contains processes that cannot be executed from the serial flash memory, this function must be assigned to an area other than the serial flash memory.

Userdef_PostHardwareSetup	
Outline	Necessary hardware initialization processing after the SPIBSC initialization process
Declaration	void Userdef_PostHardwareSetup (void)
Description	This is the user definable function to describe the hardware initialization process which is required to be executed after the SPIBSC initialization. It is called at the end of the R_SC_HardwareSetup function. In the loader program, Make the following clock settings, on the assumption that 24[MHz] is input from EXTAL by calling R_CPG_InitialiseHwIf function. • Iϕ = 528[MHz], Gϕ = 264[MHz], Bϕ = 132[MHz], P1ϕ = 66[MHz], P0ϕ = 33[MHz], QSPI0_SPCLK = 66[MHz]
Argument	None
Return Value	None
Precautions	If this function contains processes that cannot be executed from the serial flash memory, this function must be assigned to an area other than the serial flash memory.

5.10 Loader Program Flowcharts

5.10.1 Loader program (overall)

Figure 5.4 shows the Flowchart of loader program (overall).



Note: The SPIBSC mode is switched from external address space read mode to manual mode when configuring the serial flash memory registers and from manual mode to external address space read mode after the register configuration is completed.

Figure 5.4 Flowchart of loader program (overall)

5.10.2 Memory Clock Setting Processing

In order to accelerate the section initialization for the hardware setting process which includes SPIBSC initial setting and serial flash memory register setting, the setting of clock supplied to the serial flash memory (QSPIn_SPCLK) is performed.

Because a program allocated to the SPI multi-I/O bus space cannot execute the memory clock setting process, the program is loaded into the large-capacity on-chip RAM and executed from there.

Figure 5.5 shows the Memory clock setting process flowchart.

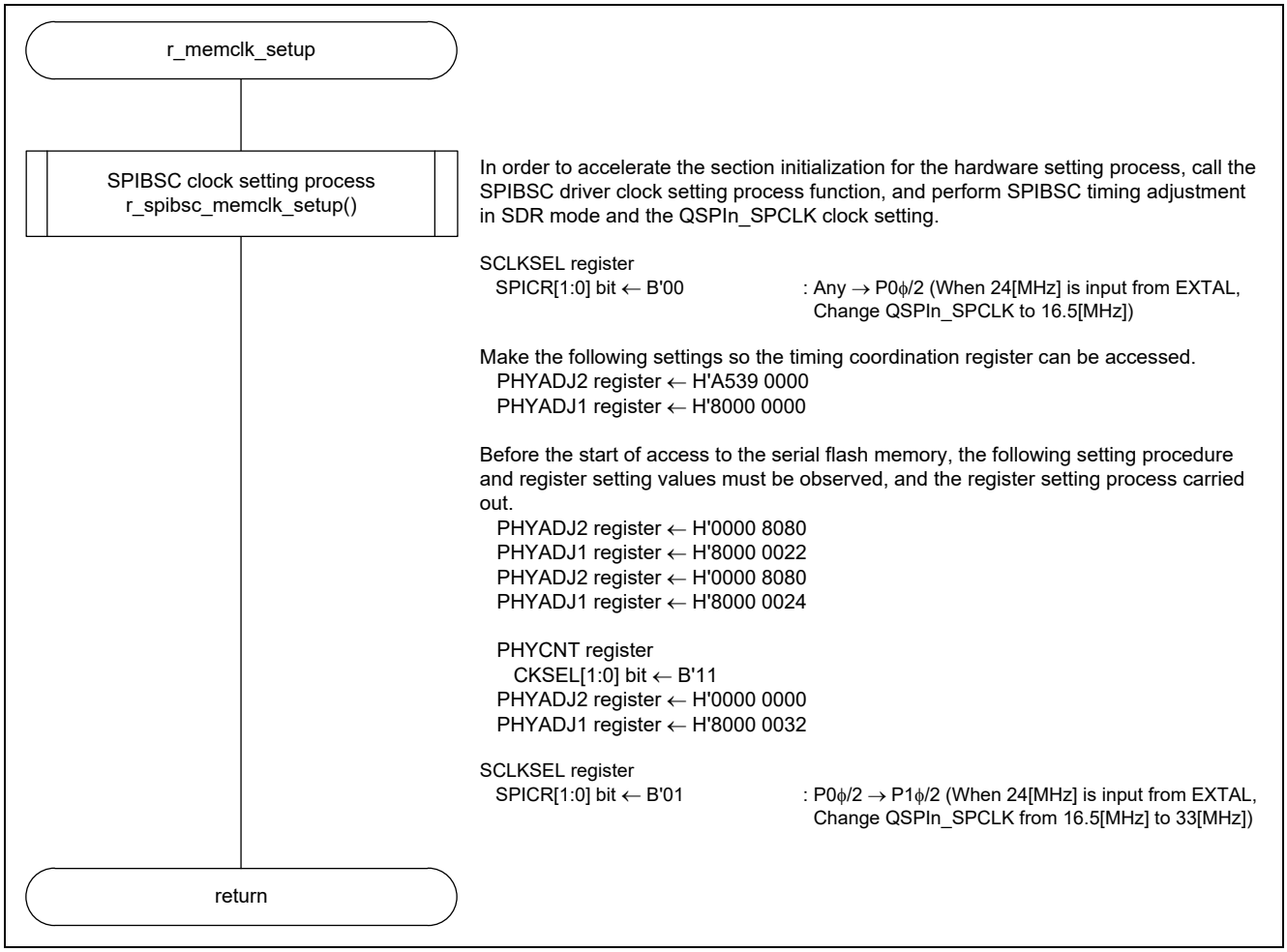


Figure 5.5 Memory clock setting process flowchart

5.10.3 Initial setting of hardware used for booting

This function performs the hardware initial setting. In order to enable high-speed access to the serial flash memory in the loader program, the R_SPIBSC_Setup function is called, and the SPIBSC and serial flash memory settings are performed.

Because a program allocated to the SPI multi-I/O bus space cannot execute the setting process of SPIBSC registers and serial flash memory registers, the program is loaded into the large-capacity on-chip RAM and executed from there.

Figure 5.6 shows the Flowchart of Initial setting any hardware.

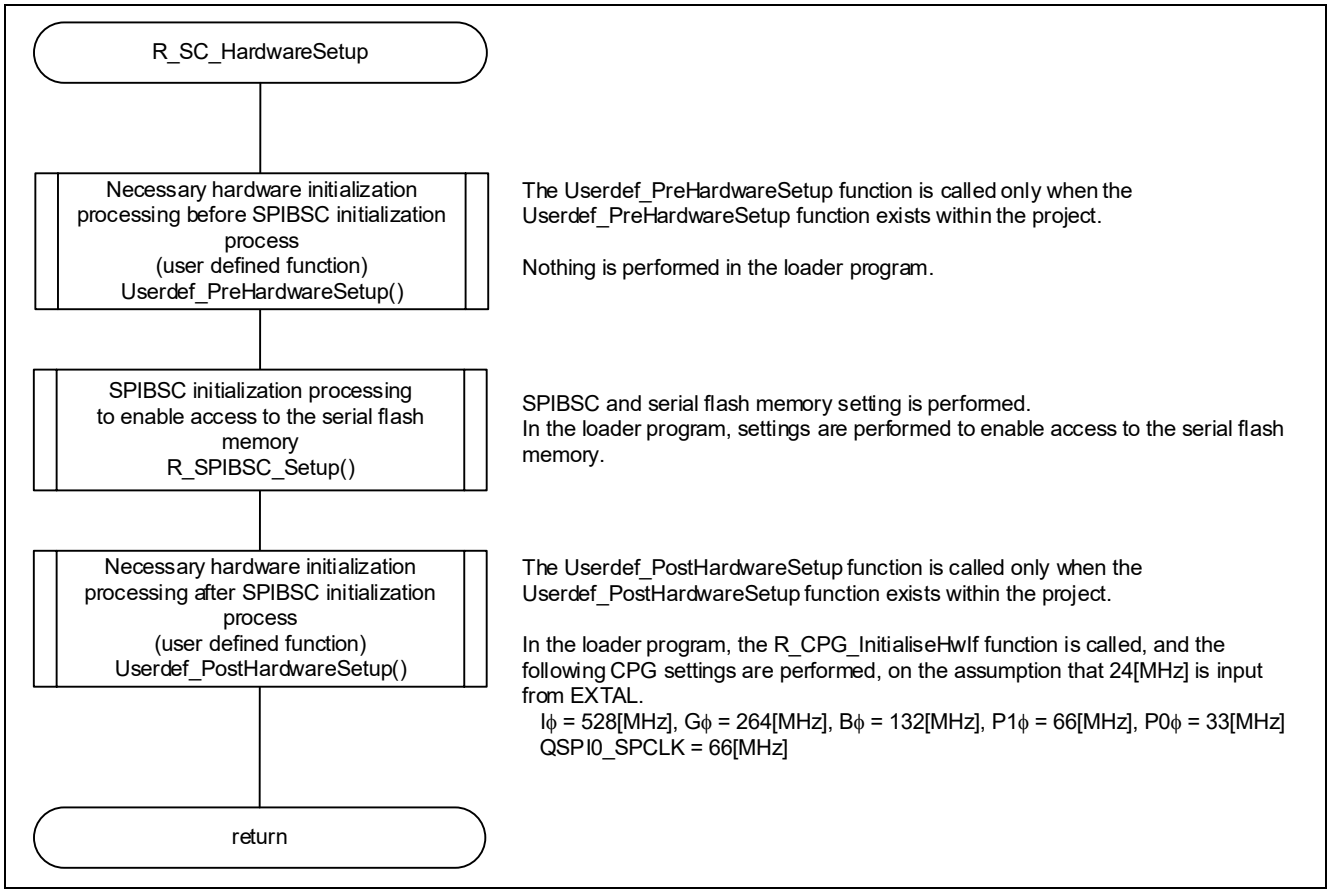


Figure 5.6 Flowchart of Initial setting any hardware

5.10.4 SPIBSC and Serial Flash Memory Initial Setting

The loader program sets up the serial flash memory registers (QE bit of the status register, and DC[1:0] bits, PBE bit, and ODS[2:0] bits of the configuration register) and changes the bus bit width from 1 bit to 4 bits so that the serial flash memory can be accessed at a higher speed. After setting up the serial flash memory registers, the program sets the type of read command to be issued to the serial flash memory when using the SPIBSC in external address space read mode to "Quad I/O DT Read (4B address)" (H'EE) and changes the QSPIn_SPCLK clock frequency to Bϕ/2.

Since the loader program modifies the SPIBSC registers during its processing, it cannot run in the SPI multi-I/O bus space. Accordingly, it is transferred in large-capacity on-chip RAM for execution.

Figure 5.7 shows the flowchart of the loader program, and Figure 5.8 to Figure 5.20 show the flowcharts of the used functions.

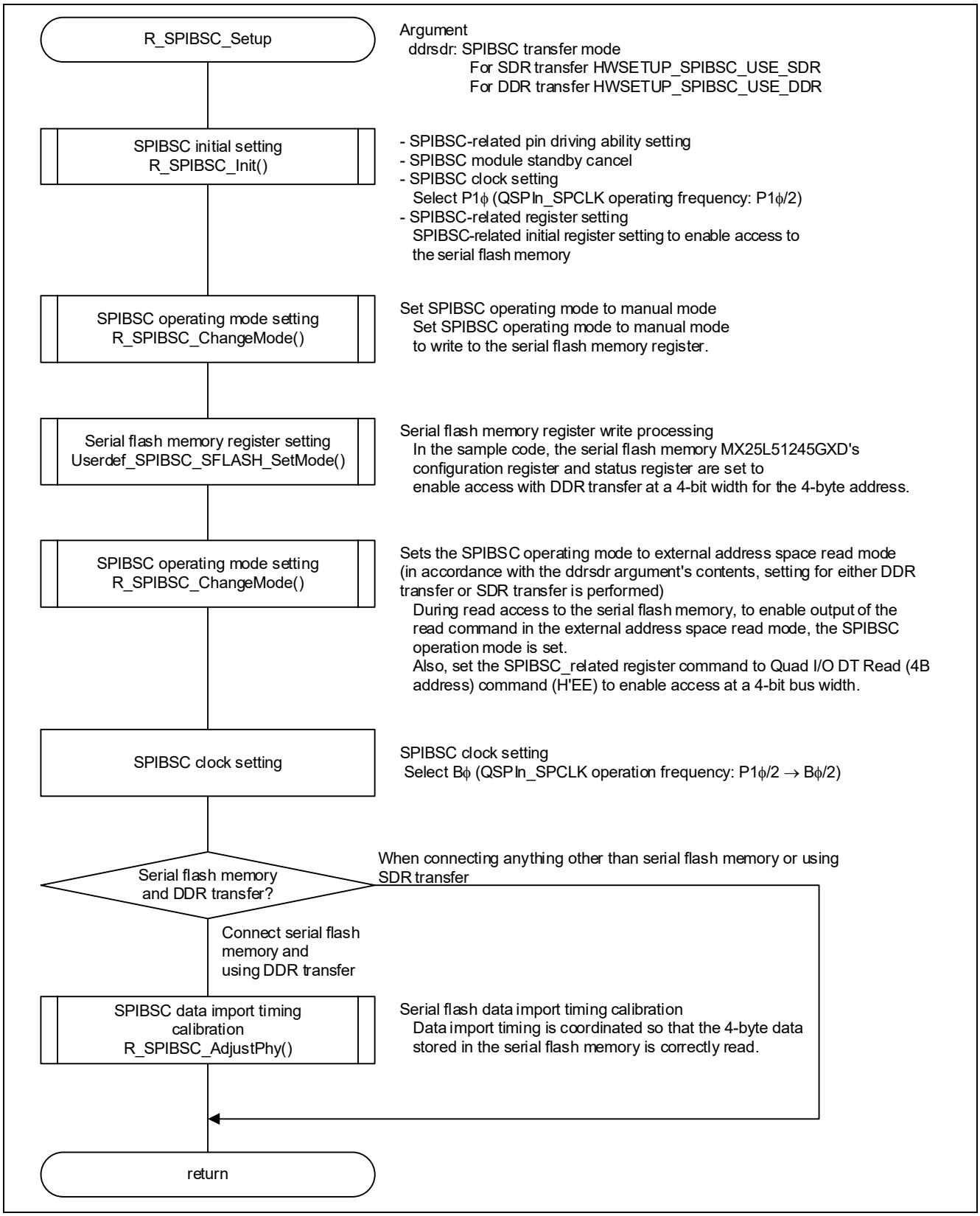


Figure 5.7 Flowchart of SPIBSC and serial flash memory initial setting

5.10.5 SPIBSC Initial Setting

Figure 5.8 and Figure 5.9 show the flowchart of the SPIBSC initial setting.

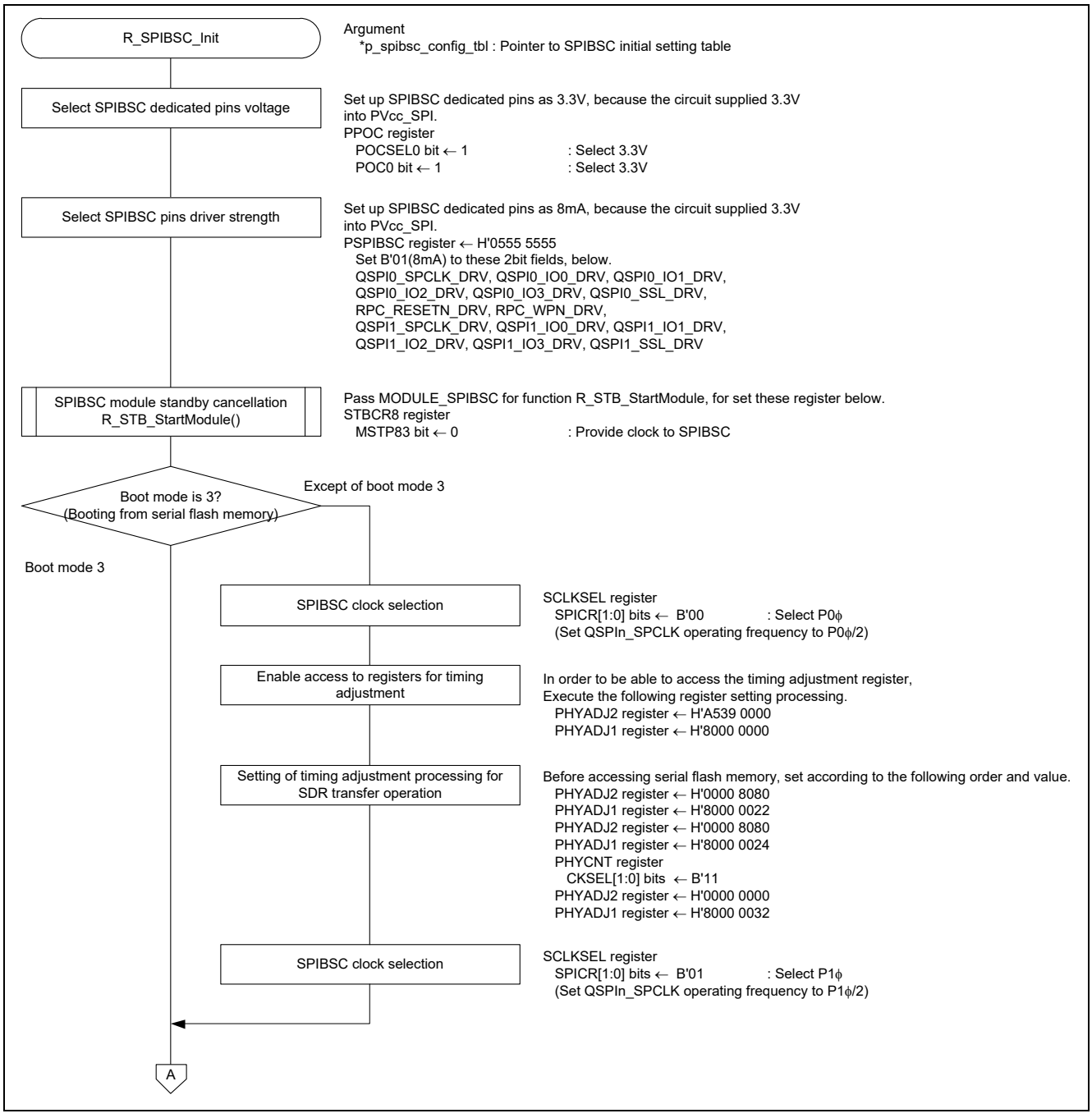


Figure 5.8 Flowchart of SPIBSC initial setting (1/2)

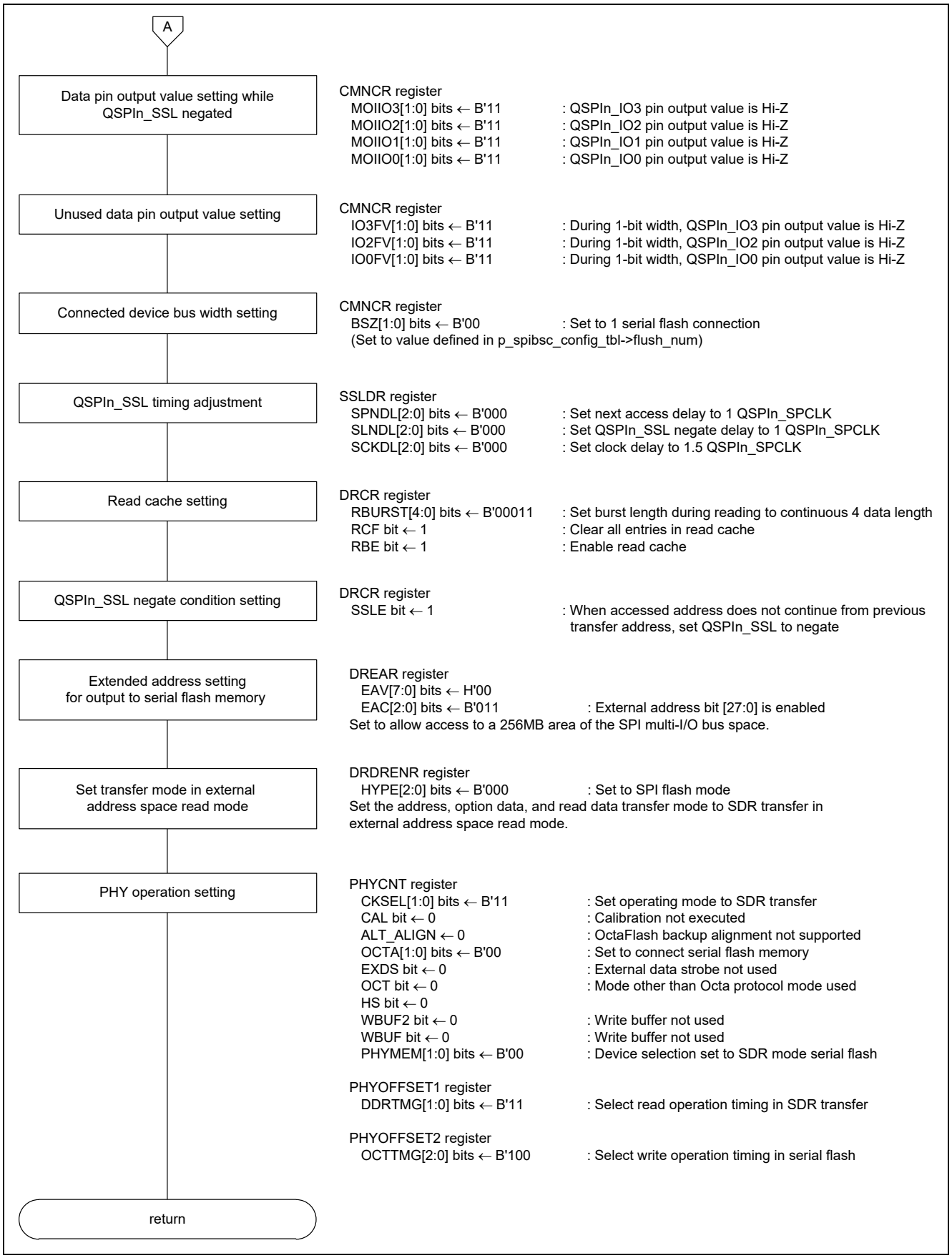


Figure 5.9 Flowchart of SPIBSC initial setting (2/2)

5.10.6 SPIBSC Operating Mode Setting

Figure 5.10 to Figure 5.12 show the SPIBSC operating mode setting.

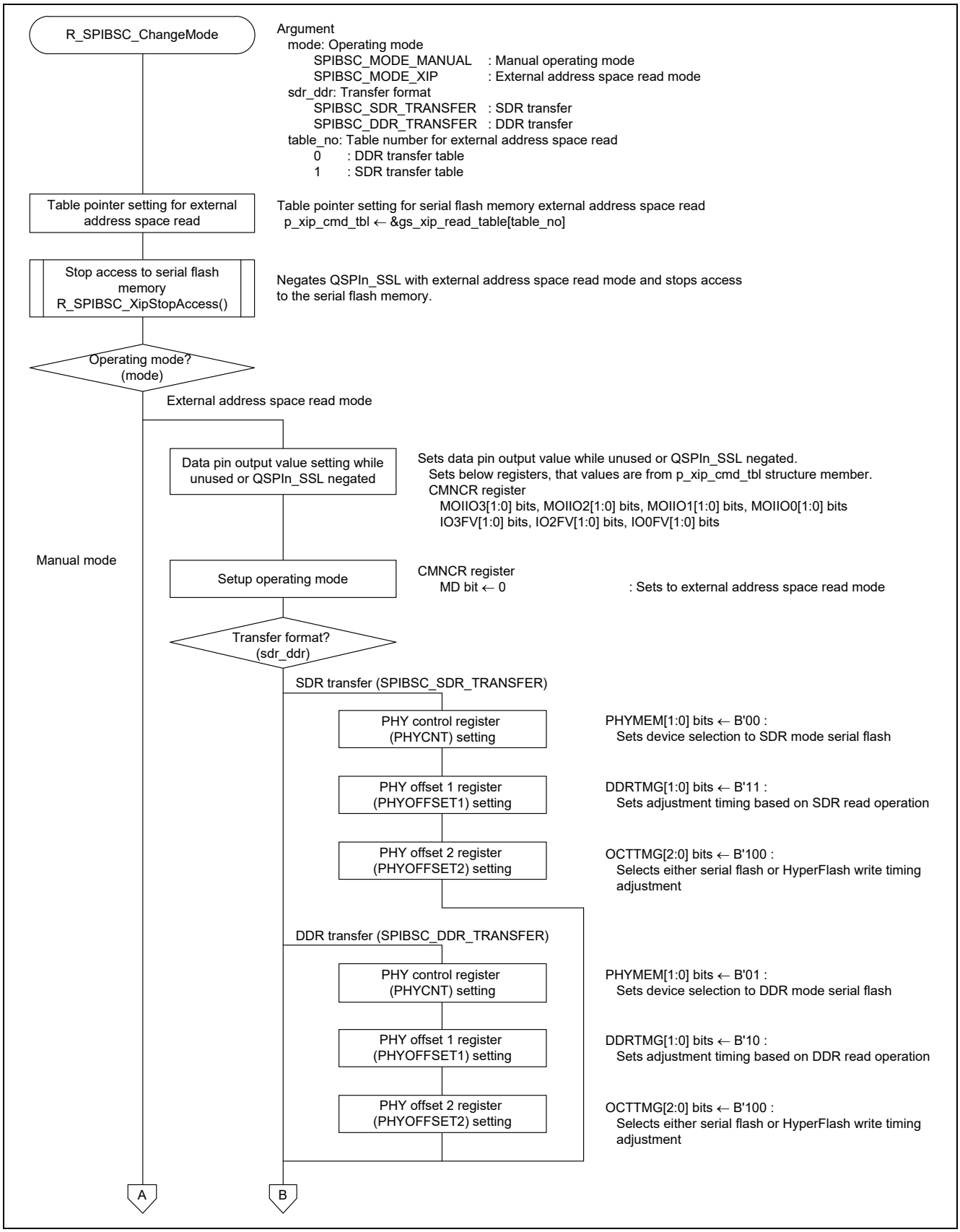


Figure 5.10 Flowchart of SPIBSC operating mode setting (1/3)

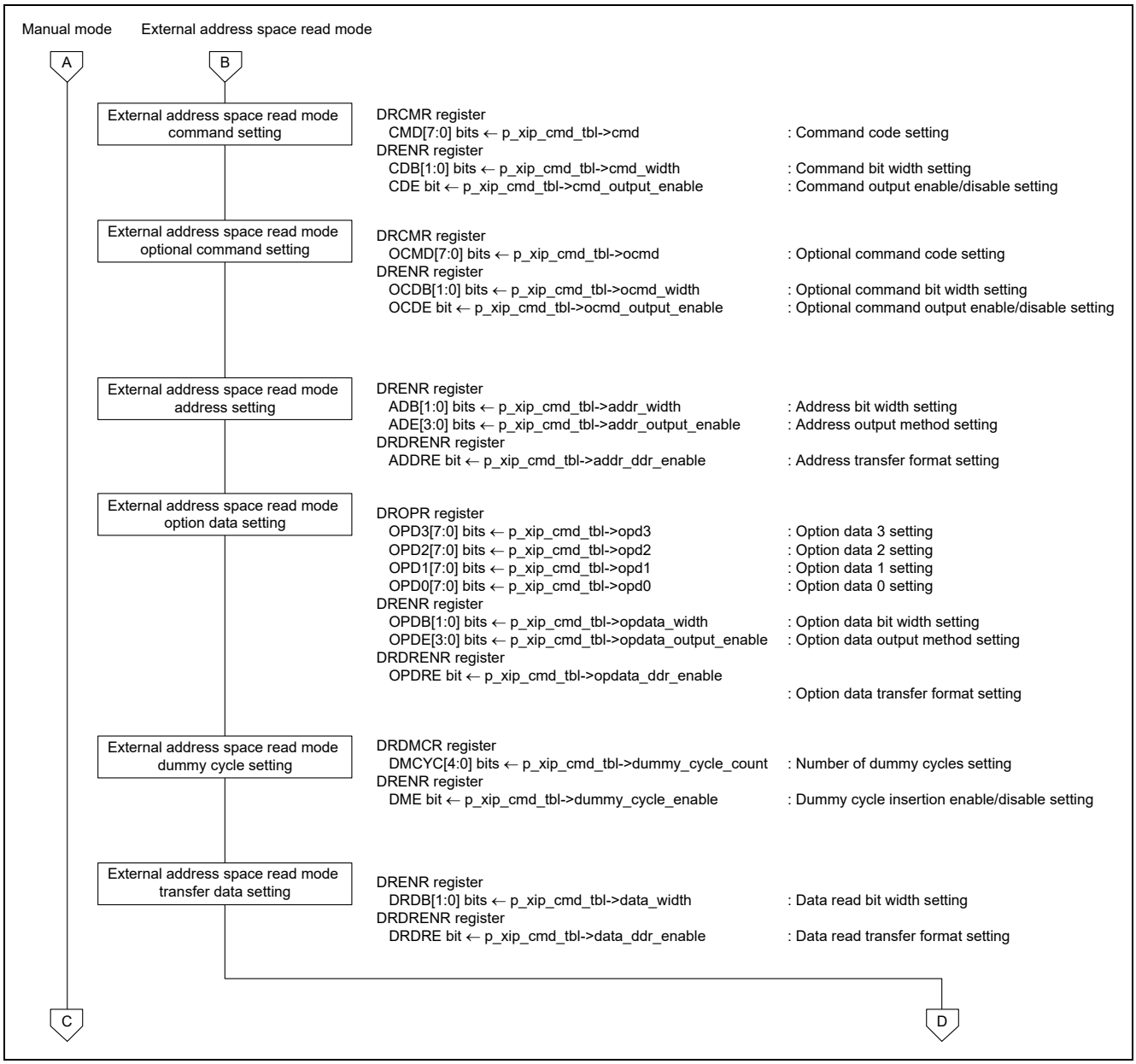


Figure 5.11 Flowchart of SPIBSC operating mode setting (2/3)

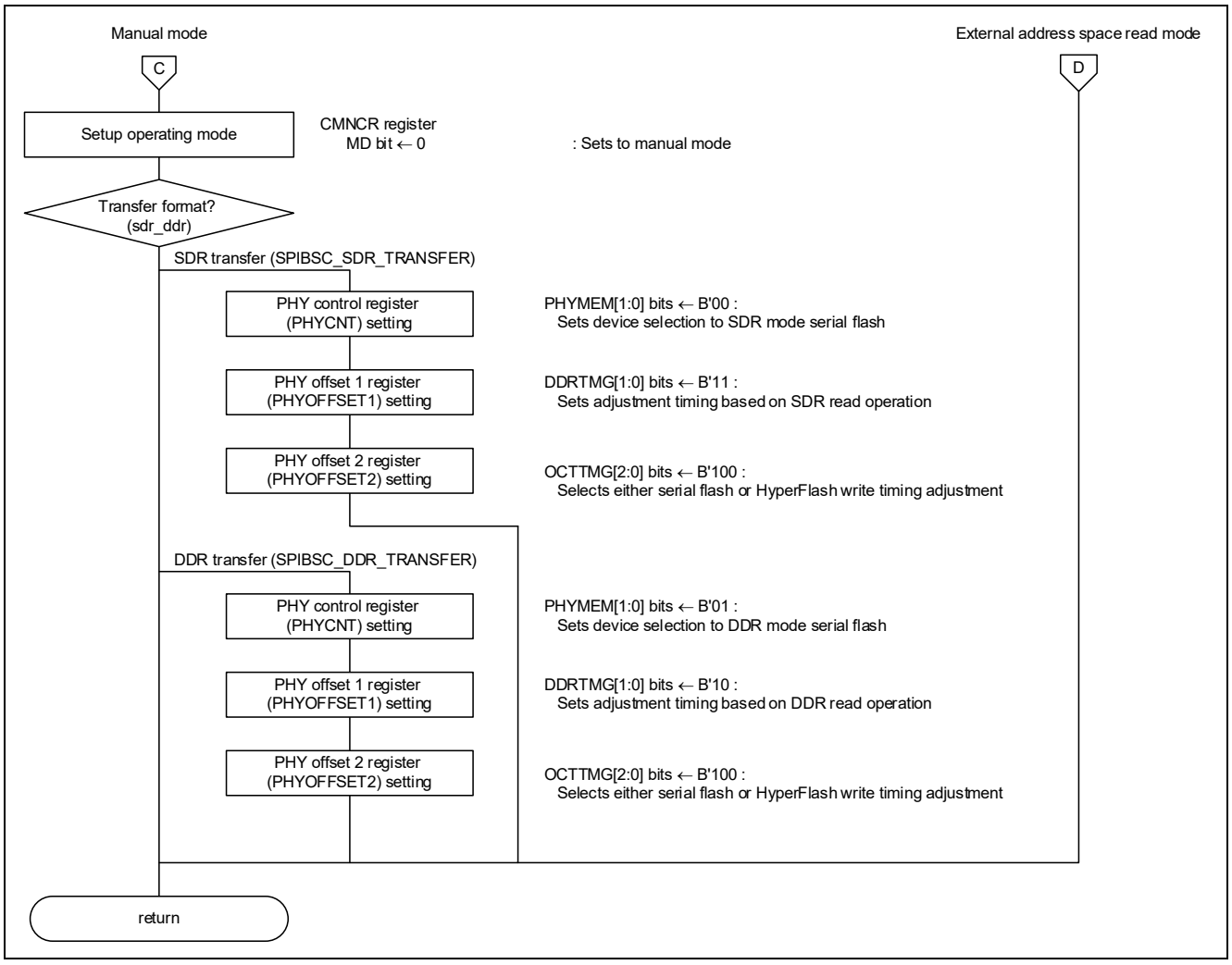


Figure 5.12 Flowchart of SPIBSC operating mode setting (3/3)

5.10.7 Issuance of SPI Command to Serial Flash Memory

Figure 5.13 to Figure 5.16 show the Flowchart for the issuance of an SPI command to serial flash memory. Use this function in manual mode.

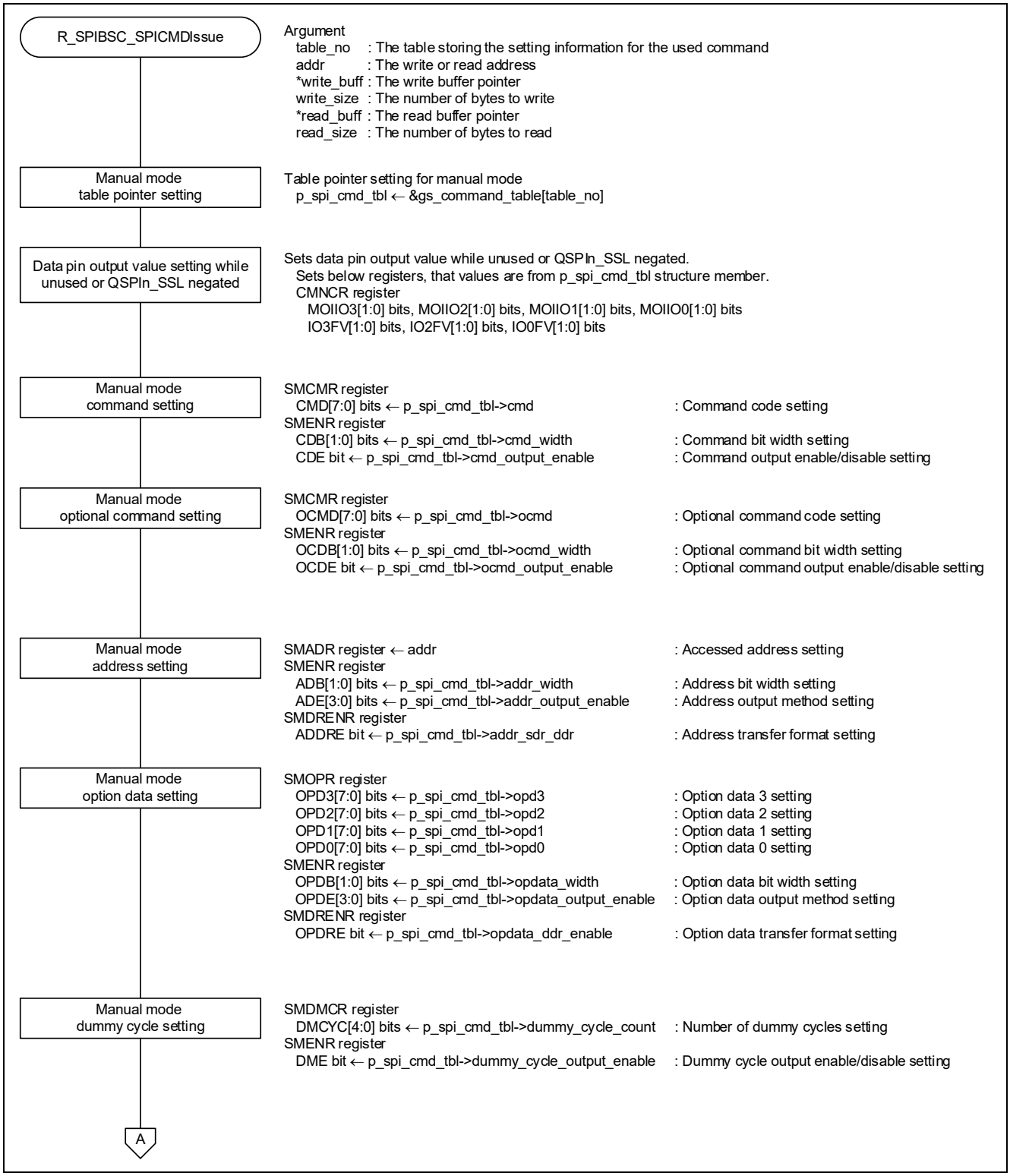


Figure 5.13 Flowchart of issuance of SPI command to serial flash memory (1/4)

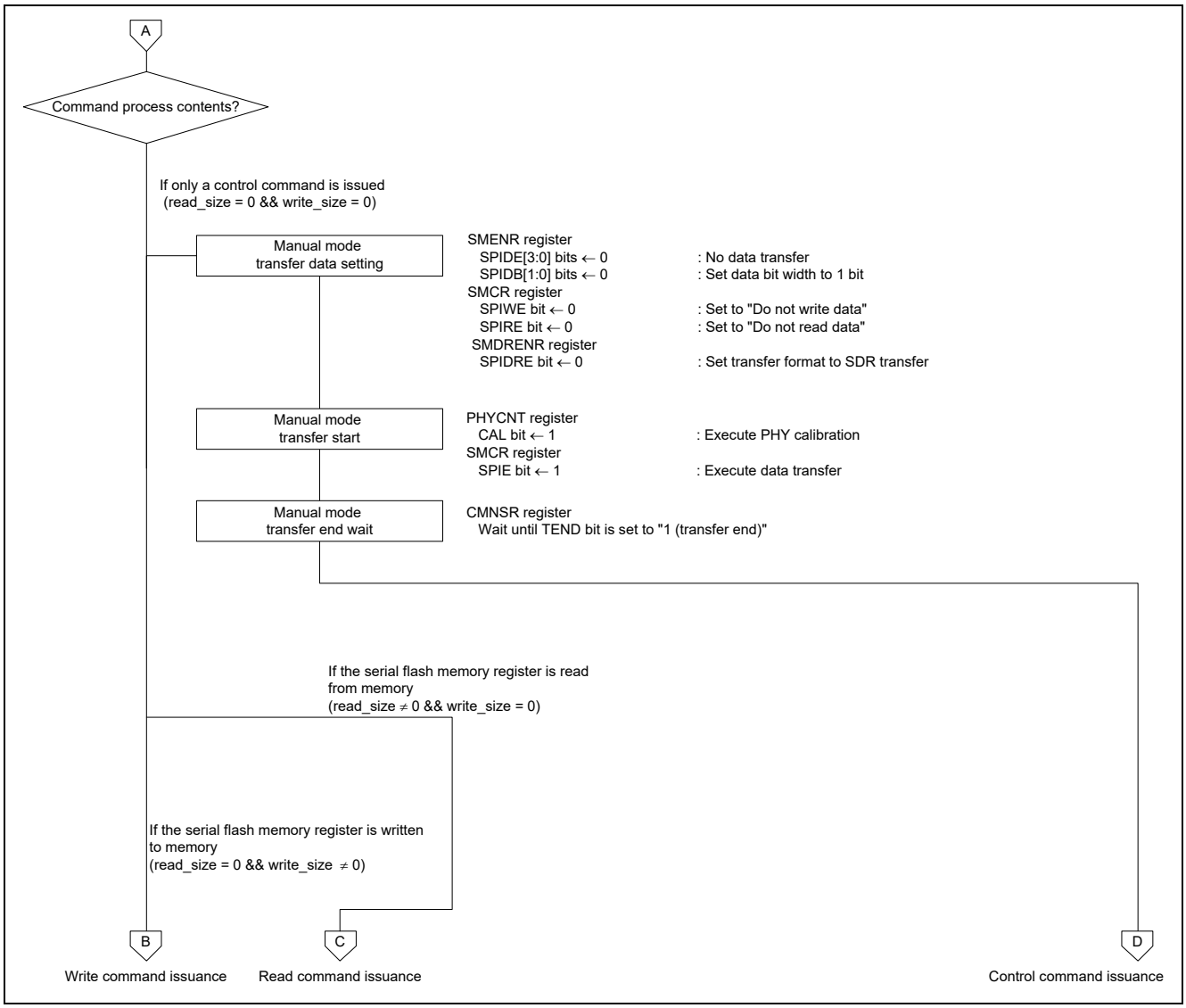


Figure 5.14 Flowchart of issuance of SPI command to serial flash memory (2/4)

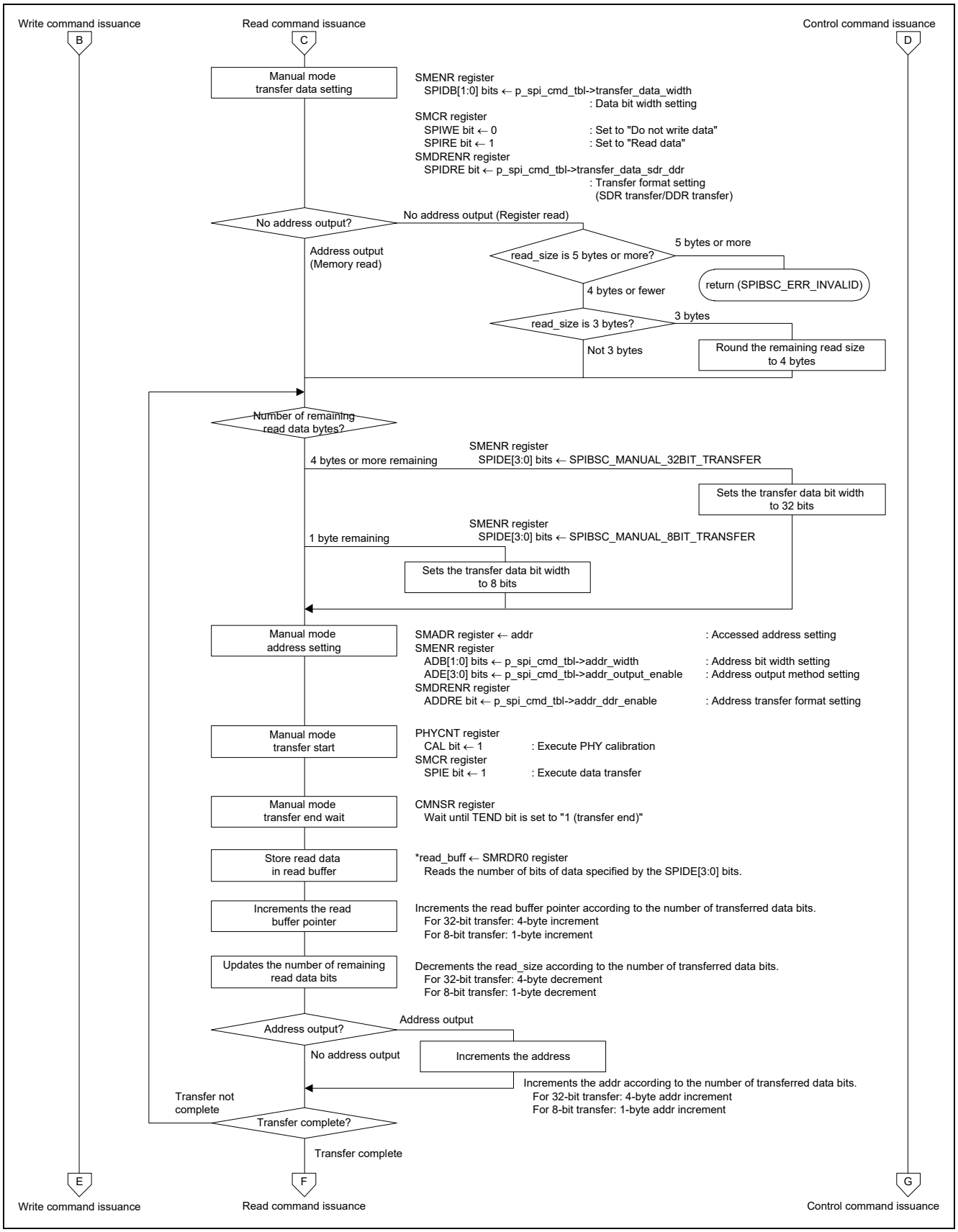


Figure 5.15 Flowchart of issuance of SPI command to serial flash memory (3/4)

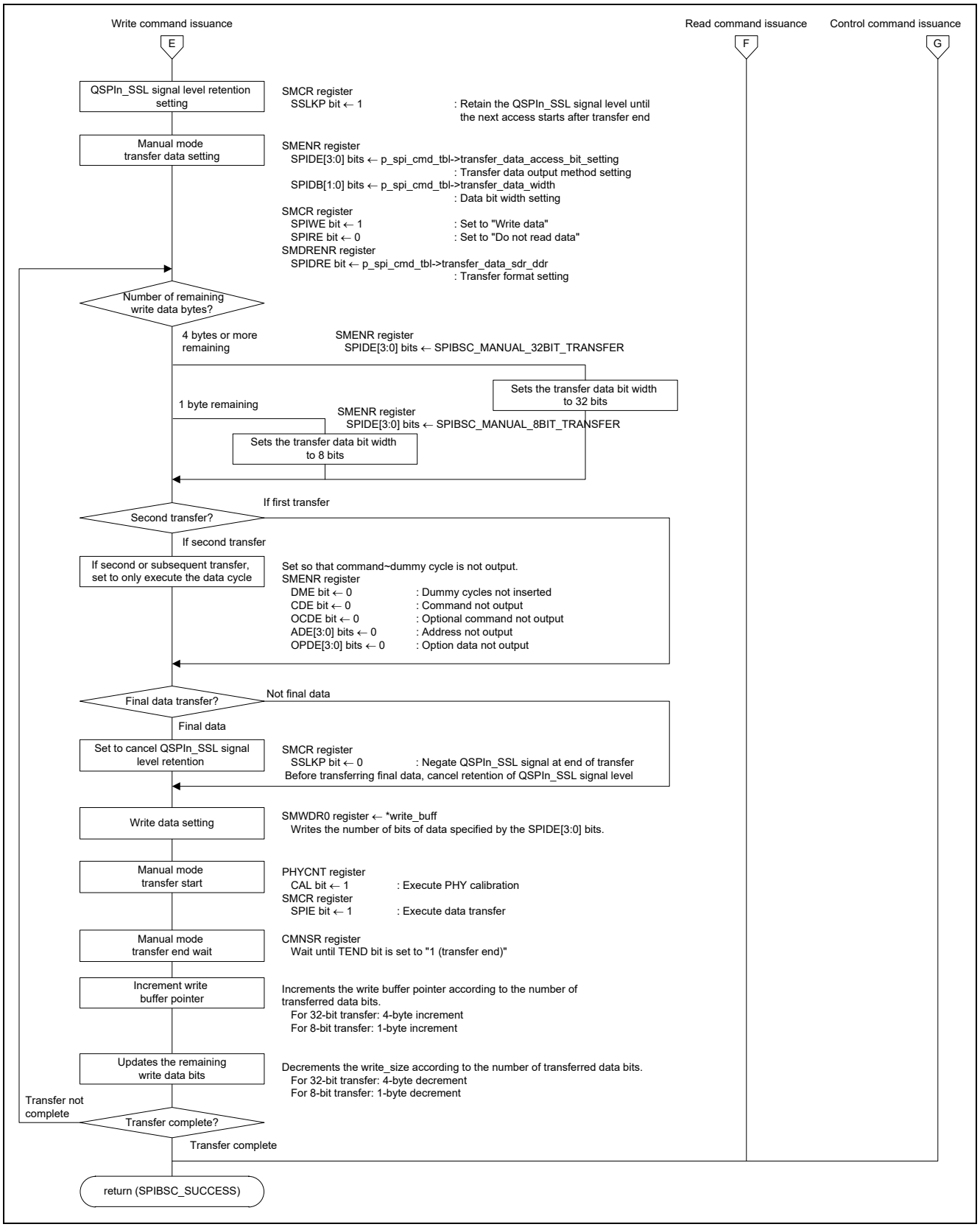


Figure 5.16 Flowchart of issuance of SPI command to serial flash memory (4/4)

5.10.8 SPIBSC Read Timing Calibration for DDR Transfer

When SPIBSC communicates with serial flash memory by DDR transfer, it must need to calibrate data capture delay for valid reading.

Figure 5.17 shows the Allocation of test pattern for calibrate processing. The test pattern is stored in serial flash memory and using for its calibrating operation.

In calibration routine, reads the test pattern by per capture delay that increase gradually, and choose median delay that can read valid pattern. The SPIBSC registers PHYADJ1, PHYADJ2, and PHYCNT.CKSEL are used for applying capture delay.

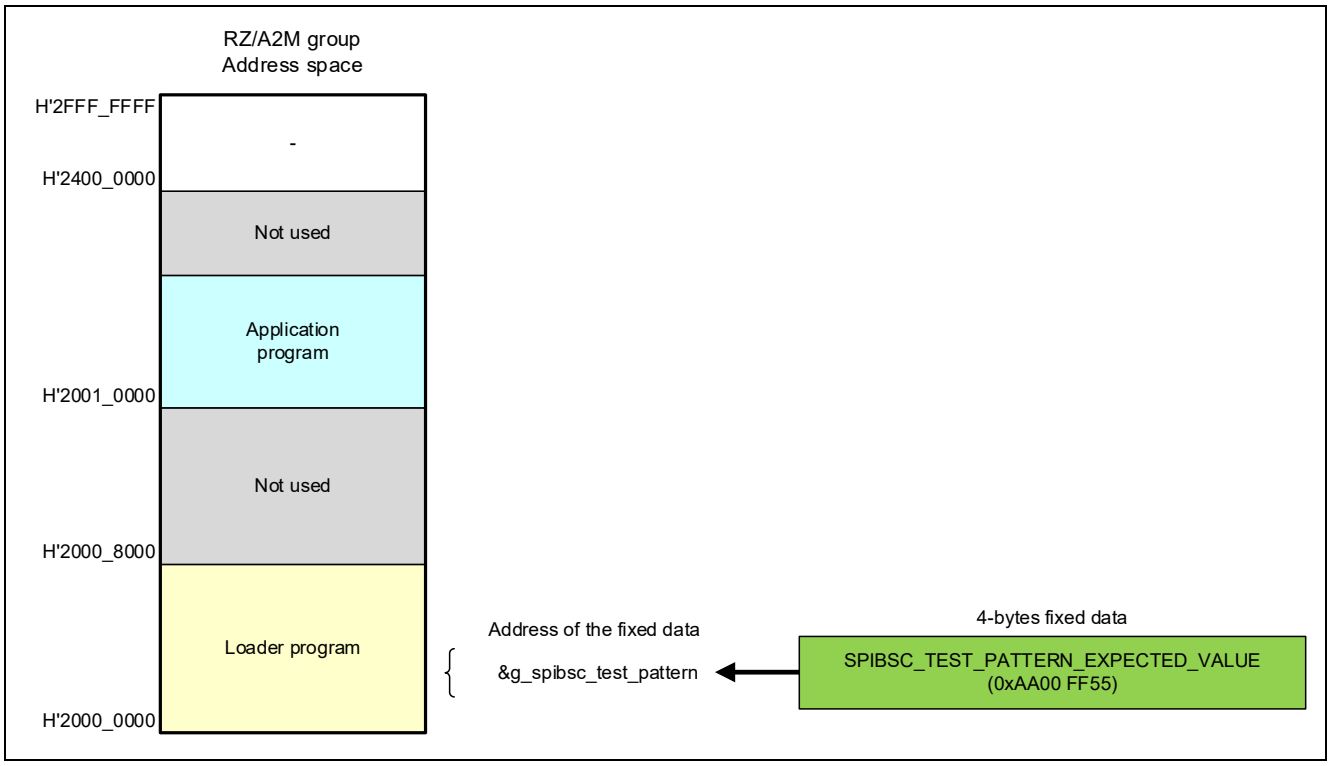


Figure 5.17 Allocation of test pattern for calibrate processing

Figure 5.18 to Figure 5.20 shows Flowchart of Timing Calibration for DDR Transfer. This function must use when the SPIBSC configures as manual transfer mode.

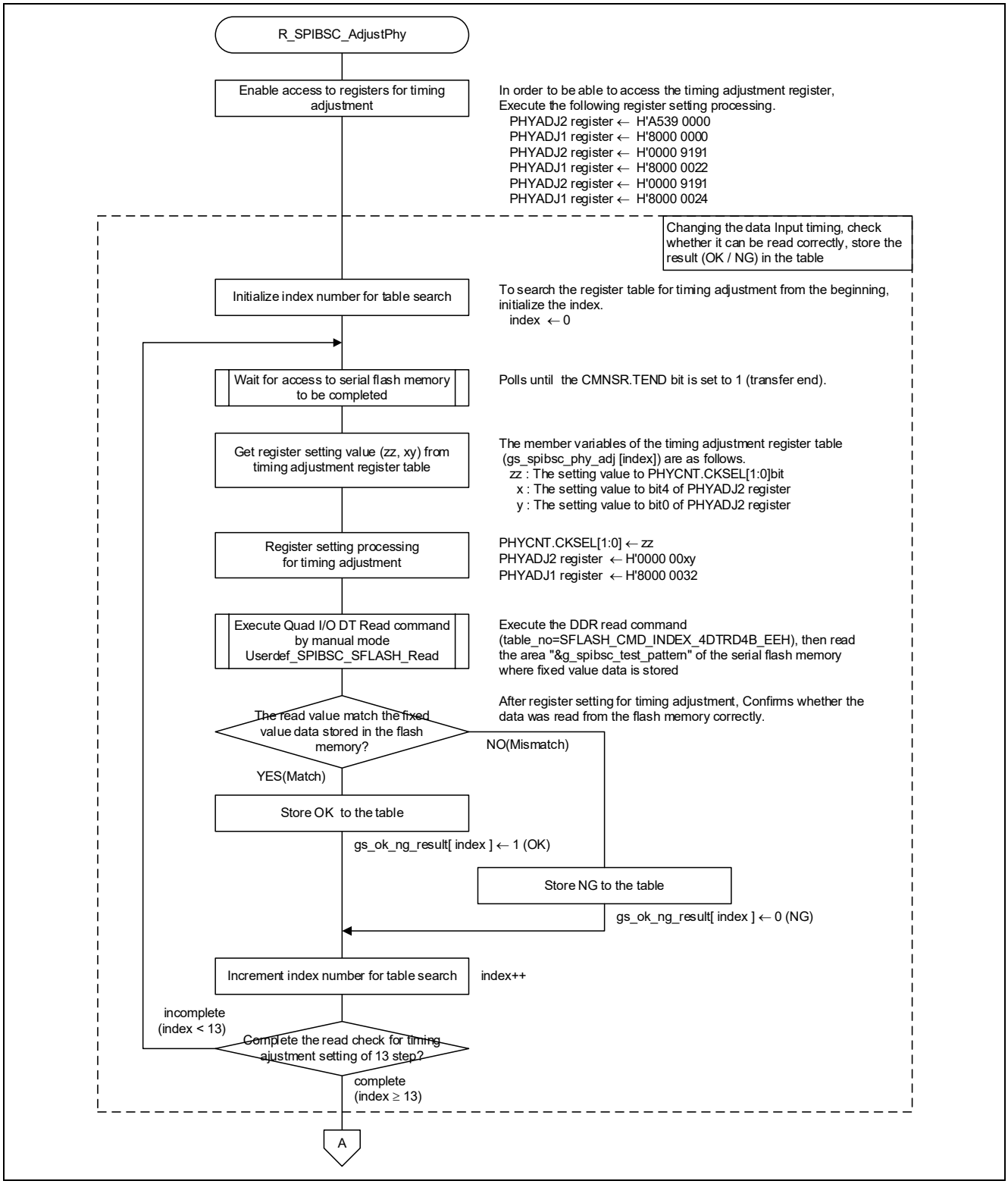


Figure 5.18 Flowchart of Timing Calibration for DDR Transfer (1/3)

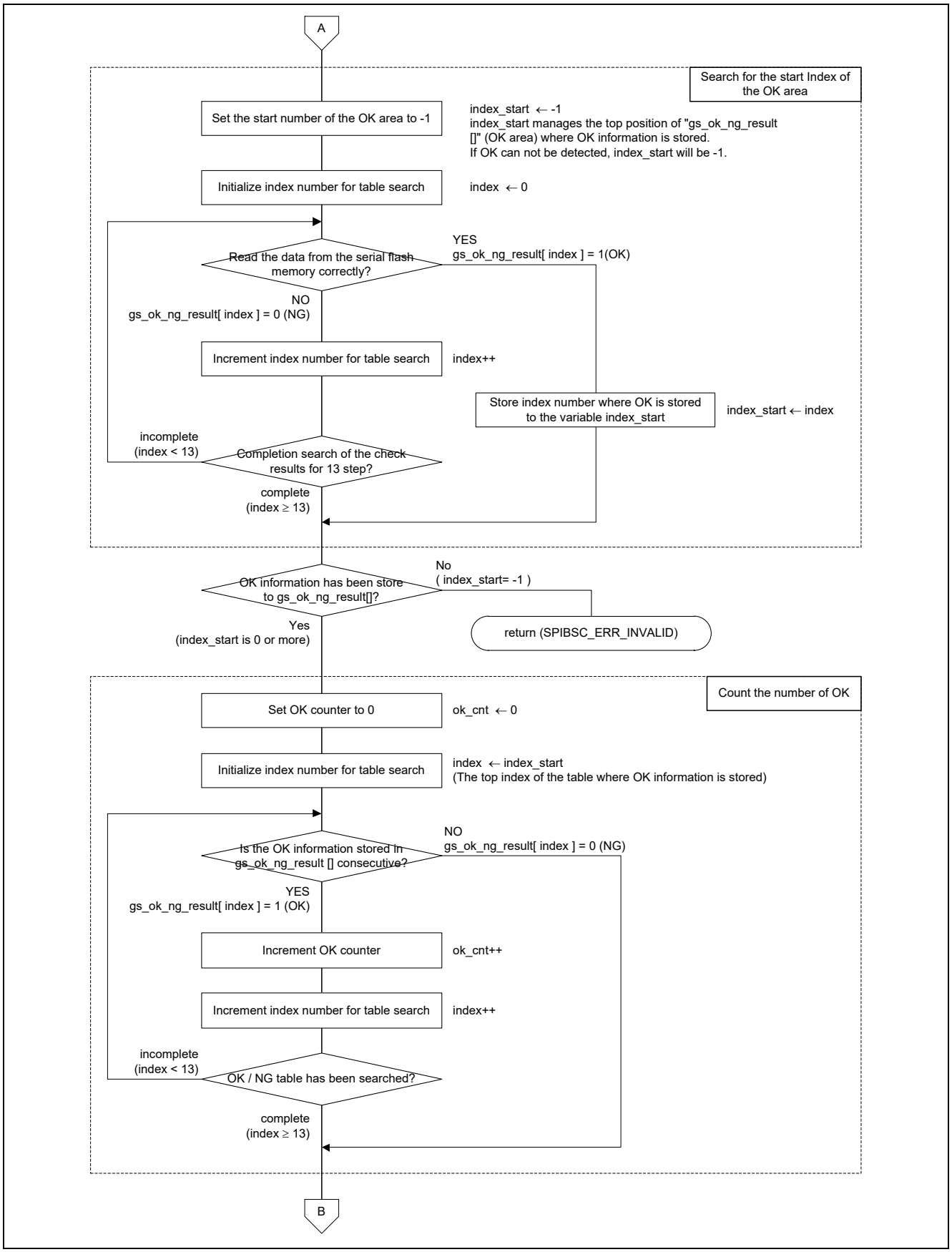


Figure 5.19 Flowchart of Timing Calibration for DDR Transfer (2/3)

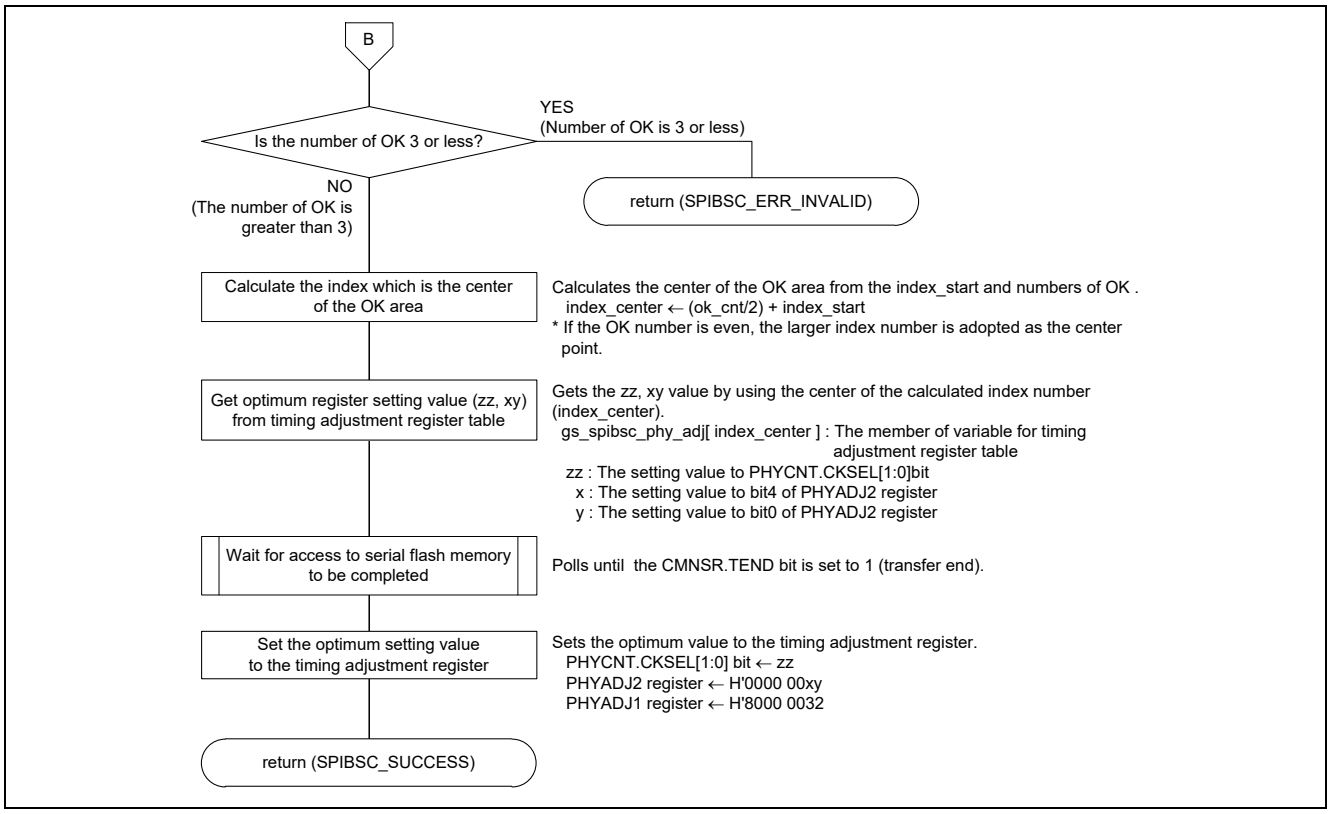


Figure 5.20 Flowchart of timing adjustment when connected to serial flash memory (3/3)

6. Application Example

6.1 Operation of the Sample Code Used in its Initial State

The sample code in its initial state accesses the Macronix serial flash memory (product No.: MX25L51245GXD) according to the settings that are summarized in Table 6.1.

Table 6.1 Sample Code Access Settings

Item	Setting
Serial flash memory	• Macronix serial flash memory • Model name: MX25L51245GXD • Read command used: H'EE (4DTRD4B) 4-bit bus, DDR transfer Number of dummy cycles required: 8 (When running at a maximum operating frequency of 66MHz)
SPIBSC	• Number of serial flash memory devices to be connected: 1 • Data bus width: 4 bits • Number of address bytes: 4 bytes (Number of bytes to be issued when specifying an address) • Transfer format in read mode: DDR transfer

Figure 6.1 shows the Read operation in DDR transfer mode (initial state of the sample code). Table 6.2 shows the Register settings for sample .

In the command cycle, the MX25L51245GXD samples the input data at the rising edge of the clock in SDR transfer mode. The SPIBSC begins data output with respect to the falling edge of the clock and continues data output processing at the rising edge. The MX25L51245GXD samples the MSB of the output data from the SPIBSC at the rising edge of the first QSPIn_SPCLK.

The address cycle and data cycle perform DDR transfer. In the address cycle, the SPIBSC starts address output at the rising edge of the clock after the command cycle has finished, and after that outputs at both the rising and falling edges of the clock. The MX25L51245GXD starts sampling the input of the address cycle at the rising edge, and after that samples at both edges.

The MX25L51245GXD also begins data output at the falling edge of the last clock of the dummy cycle, and after that outputs data at both the rising and falling edges of the clock. The SPIBSC samples the input data at the rising edge of the first clock of the data cycle, and after that samples the input data at both edges.

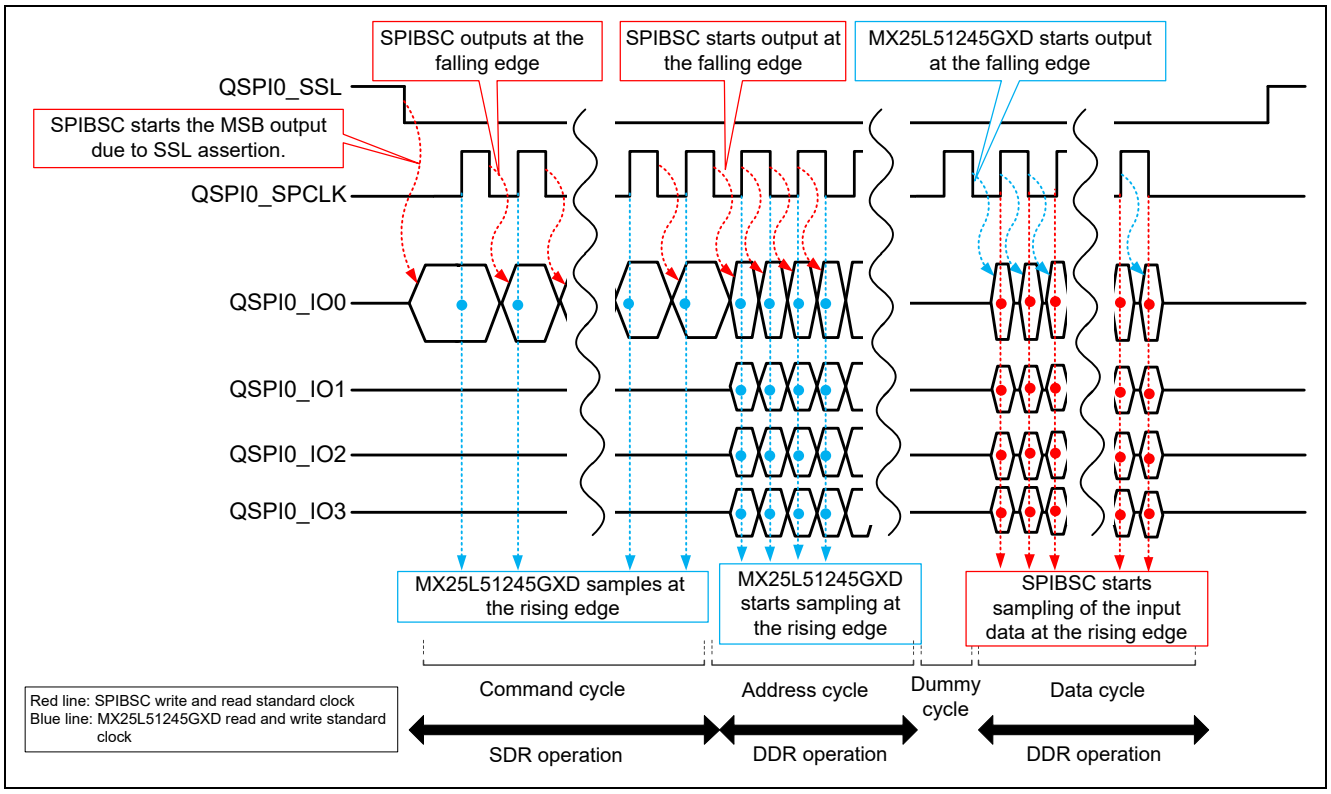


Figure 6.1 Read operation in DDR transfer mode (initial state of the sample code)

In the sample code, the settings shown in Table 6.2 are made in the SPIBSC and serial flash memory registers in the initial state, and DDR transfer read operations are performed.

Table 6.2 Register settings for sample code

Setting	Setting value for sample code
Read command setting	DRCMR.CMD[7:0] = 0xEE DRENR.CDB[1:0] = SPIBSC_1BIT_WIDTH DRENR.CDE = SPIBSC_OUTPUT_ENABLE
Address setting	DRENR.ADB[1:0] = SPIBSC_4BIT_WIDTH DRENR.ADE[3:0] = SPIBSC_OUTPUT_ADDR_32 DRDRENR.ADDRE = SPIBSC_DDR_TRANSFER
Option data setting	DRENR.OPDB[1:0] = SPIBSC_4BIT_WIDTH DRENR.OPDE[3:0] = SPIBSC_OUTPUT_OPD_3 DRDRENR.OPDRE = SPIBSC_DDR_TRANSFER DROPR.OPD3[7:0] = 0x00
Dummy cycle setting	DRENR.DME = SPIBSC_OUTPUT_ENABLE DRDMCR.DMCYC[4:0] = SPIBSC_DUMMY_07CYC
Transfer data setting	DRENR.DRDB[1:0] = SPIBSC_4BIT_WIDTH DRDRENR.DRDRE = SPIBSC_DDR_TRANSFER
Status register setting	QE bit = 1
Configuration register setting	DC[1:0] bits = b'10 (8 cycles) PBE bit = 0 (Disabled) ODS[2:0] bits = b'110

The sample code calls the user-defined function Userdef_SPIBSC_SFLASH_SetMode to set up the serial flash memory registers within the R_SC_HardwareSetup function which is executed during initialization, after switching to manual mode with the R_SPIBSC_ChangeMode function. The Userdef_SPIBSC_SFLASH_SetMode function configures the status register and configuration register of the serial flash memory according to the specifications for the read command to be used.

Table 6.3 shows MX25L51245GXD status register and Table 6.4 shows MX25L51245GXD configuration register. The Userdef_SPIBSC_SFLASH_SetMode function is used to set the bites denoted by " " in the tables and other bits are left to hold their initial value.

Table 6.3 MX25L51245GXD status register

Bit position	Bit name	Attribute (Note)	Description
7	SRWD	NV	Status register write protect 1 = Status register write disabled 0 = Status register write enabled
6	QE	NV	Quad enable 1 = Quad enable 0 = Not Quad enable
5,4,3,2	BP3,BP2,BP1,BP0	NV	Level of protected block
1	WEL	V	Write enable latch 1 = Write enable 0 = Not write enable
0	WIP	V	Write in progress bit 1 = Write operation 0 = Not in write operation

Note: "NV" in the attribute column denotes "Non-volatile bit" and "V" denotes "Volatile bit."

Table 6.4 MX25L51245GXD configuration register

Bit position	Bit name	Attribute (Note 1)	Description
7,6	DC1, DC0	V	Dummy cycle 1, Dummy cycle 0 DC[1:0] = B'10 (Note 2)
5	4BYTE	V	0 = 3-byte address mode 1 = 4-byte address mode
4	PBE	V	Preamble bit enable 0 = Disable 1 = Enable
3	TB	OTP	Top/bottom selected 0 = Top area protect 1 = Bottom area protect
2,1,0	ODS2, ODS1, ODS0	V	Output driver strength (ODS2 = 1, ODS1 = 1, ODS0 = 0) (Note 3)

Note: 1. "NV" in the attribute column denotes "Non-volatile bit" and "OTP" denotes "One-time programmable bit".
2. As seen from Table 6.5, the number of dummy cycles differs depending on the read command to be used. The sample code uses H'EE as the read command and therefore loads DC[1:0] bits with a value of B'10 so that an optimum dummy cycle count can be obtained.
3. The sample code uses the settings ODS2 = 1, ODS1 = 1, and ODS0 = 0 to obtain optimum AC timing characteristics about the data hold time (tCHDX) and data output delay time (tCLQV) of the MX25L51245GXD when connected to the RZ/A2M. (This setting is possible when the load capacity of the device to be connected to the MX25L51245GXD is 15pF or less and 3.0 to 3.6V is applied to the power source (VCC) of the MX25L51245GXD.)

Table 6.5 shows a List of numbers of dummy cycles necessary for the operating frequency of the MX25L51245GXD. The number of necessary dummy cycles differ depending on the read command and operating frequency to be used. Since the sample code uses the H'EE command as the read command and sets QSPIn_SPCLK to 66MHz, its optimum setting is DC[1:0] = B'10 which yields a dummy cycle count of 8 cycles.

When changing the read command to be used, set the number of dummy cycles according to the new read command and the frequency of QSPIn_SPCLK.

Table 6.5 List of numbers of dummy cycles necessary for the operating frequency of the MX25L51245GXD

	Configuration register DC[1:0] bits			
Read command	**B'00**	**B'01**	**B'10**	**B'11**
FAST READ (H'0B) FAST READ4B (H'0C)	8 cycles /133MHz	6 cycles /133MHz	8 cycles /133MHz	10 cycles /66MHz
FASTDTRD (H'0D) FRDTRD4B H'0E)	8 cycles /66MHz	6 cycles /66MHz	8 cycles /66MHz	10 cycles /83MHz
4READ (H'EB) 4READ4B (H'EC)	6 cycles /84MHz	**4 cycles /70MHz (Note 2)**	8 cycles /104MHz	10 cycles /133MHz
4DTRD (H'ED) 4DTRD4B (H'EE)	6 cycles /52MHz	4 cycles /42MHz	**8 cycles /66MHz (Note 1)**	10 cycles /100MHz

Note: 1. In the default state of the sample code, 4DTRD4B (H'EE) is used as the read command in external address space read mode. In the sample code, the DC[1:0] bits are set to B'10 in order to achieve the minimum number of dummy cycles at 66Mhz operation.

2. In the sample code, 4READ4B (H'EC) is used for SDR transfer read access in external address space read mode, and the DC[1:0] bits are set to B'01 in order to achieve the minimum number of dummy cycles at 66Mhz operation.

6.2 Changing the Sample Code When Not Changing the Serial Flash Memory

6.2.1 Changing to SDR Transfer Read Command

In the sample code, the SDR transfer read command can be used as a read command in external address space read mode. By disabling the macro definition SPIBSC_PRV_DDR_SETTING defined in hwsetup.c, the SDR transfer read command is used for direct access to the serial flash memory. Table 6.6 shows the Register settings for changing to SDR transfer read command.

Table 6.6 Register settings for changing to SDR transfer read command

Setting	Value
Read command setting	DRCMR.CMD[7:0] = 0xEC DRENR.CDB[1:0] = SPIBSC_1BIT_WIDTH DRENR.CDE = SPIBSC_OUTPUT_ENABLE
Address setting	DRENR.ADB[1:0] = SPIBSC_4BIT_WIDTH DRENR.ADE[3:0] = SPIBSC_OUTPUT_ADDR_32 DRDRENR.ADDRE = SPIBSC_SDR_TRANSFER
Option data setting	DRENR.OPDB[1:0] = SPIBSC_4BIT_WIDTH DRENR.OPDE[3:0] = SPIBSC_OUTPUT_OPD_3 DRDRENR.OPDRE = SPIBSC_SDR_TRANSFER DROPR.OPD3[7:0] = 0x00
Dummy cycle setting	DRENR.DME = SPIBSC_OUTPUT_ENABLE DRDMCR.DMCYC[4:0] = SPIBSC_DUMMY_02CYC
Transfer data setting	DRENR.DRDB[1:0] = SPIBSC_4BIT_WIDTH DRDRENR.DRDRE = SPIBSC_SDR_TRANSFER
Status register setting	QE bit = 1
Configuration register setting	DC[1:0] bits = b'01 (4 cycles) PBE bit = 0 (Disabled) ODS[2:0] bits = b'110

Figure 6.2 shows the SDR transfer mode read operation. In read operation, the data cycle is started after command output, address output, and dummy cycle output from the SPIBSC, and data is output from the MX25L51245GXD.

The MX25L51245GXD samples the input data at the rising edge of the clock in SDR transfer mode. The SPIBSC begins data output with respect to the falling edge of the clock and holds the data at the rising edge.

The MX25L51245GXD also begins data output at the falling edge of the clock. In the data cycle, data is output at the falling edge of the last clock of the dummy cycle until the rising edge of the next clock. The SPIBSC samples the input data at the rising edge of the clock.

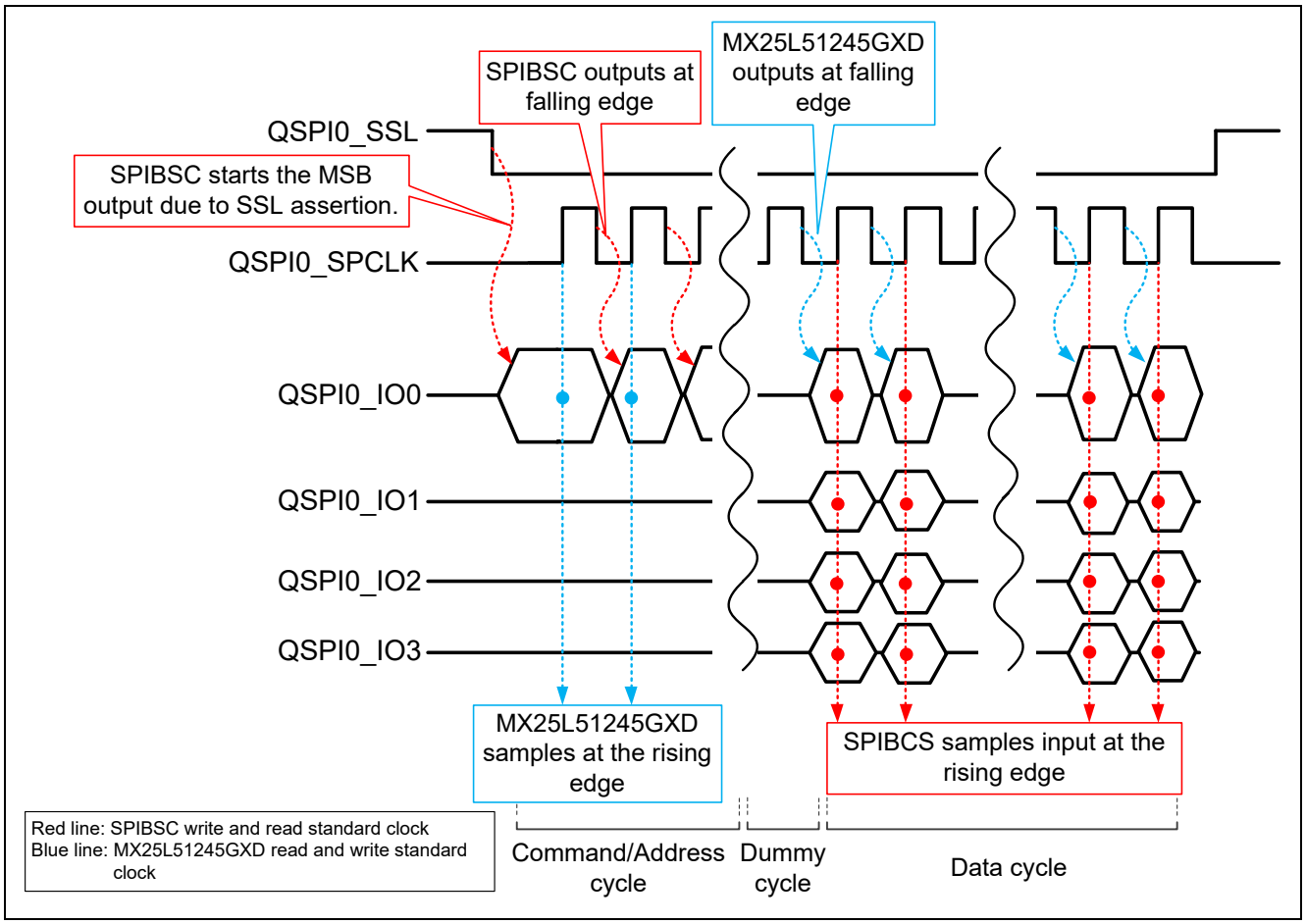


Figure 6.2 SDR transfer mode read operation

6.3 Changing the Sample Code When Changing the Serial Flash Memory

When changing the serial flash memory, the sample code should be changed according to the specifications of the serial flash memory to be used.

Table 6.7 lists the Points for changing sample code.

Table 6.7 Points for changing sample code

Change point	Description	Related section number
Signal Output when a Read Command is Issued	Change the output signal to be sent to the serial flash memory when a read command is issued in external address space read mode according to the specifications for the serial flash memory read command to be used.	6.3.1
Setting up the Serial Flash Memory Registers	The registers in the serial flash memory required to use the SPIBSC in external address read mode are made according to the serial flash memory to be used.	6.3.2
Serial Flash Memory Write Completion Wait	Wait for the write to the serial flash memory to be completed according to the serial flash memory to be used.	6.3.3
Serial Flash Memory Status Register Read	Read the status registers in the serial flash memory according to the serial flash memory to be used.	6.3.4
Serial Flash Memory Configuration Register Read	Read the configuration registers in the serial flash memory according to the serial flash memory to be used.	6.3.5
Serial Flash Memory Write Enable	The settings for the registers in the serial flash memory are made to enable write operations according to the serial flash memory to be used. (Note)	6.3.6
Serial Flash Memory Status/Configuration Register Write	Write the status/configuration registers in the serial flash memory according to the serial flash memory to be used.	6.3.7

Note: In some cases, it is necessary to enable write operations to the serial flash memory in order to make settings to the registers in the serial flash memory.

The settings in Table 6.7 are executed by the SPIBSC loader program (R_SC_HardwareSetup function). It can be handled by changing the processing of the user-defined function in the sample code according to the serial flash memory to be used. Figure 6.3 shows the Hierarchical module diagram of the SPIBSC and serial flash memory settings. Subsections 6.3.1 to 6.3.7 show the outline of the processing executed by the sample program.

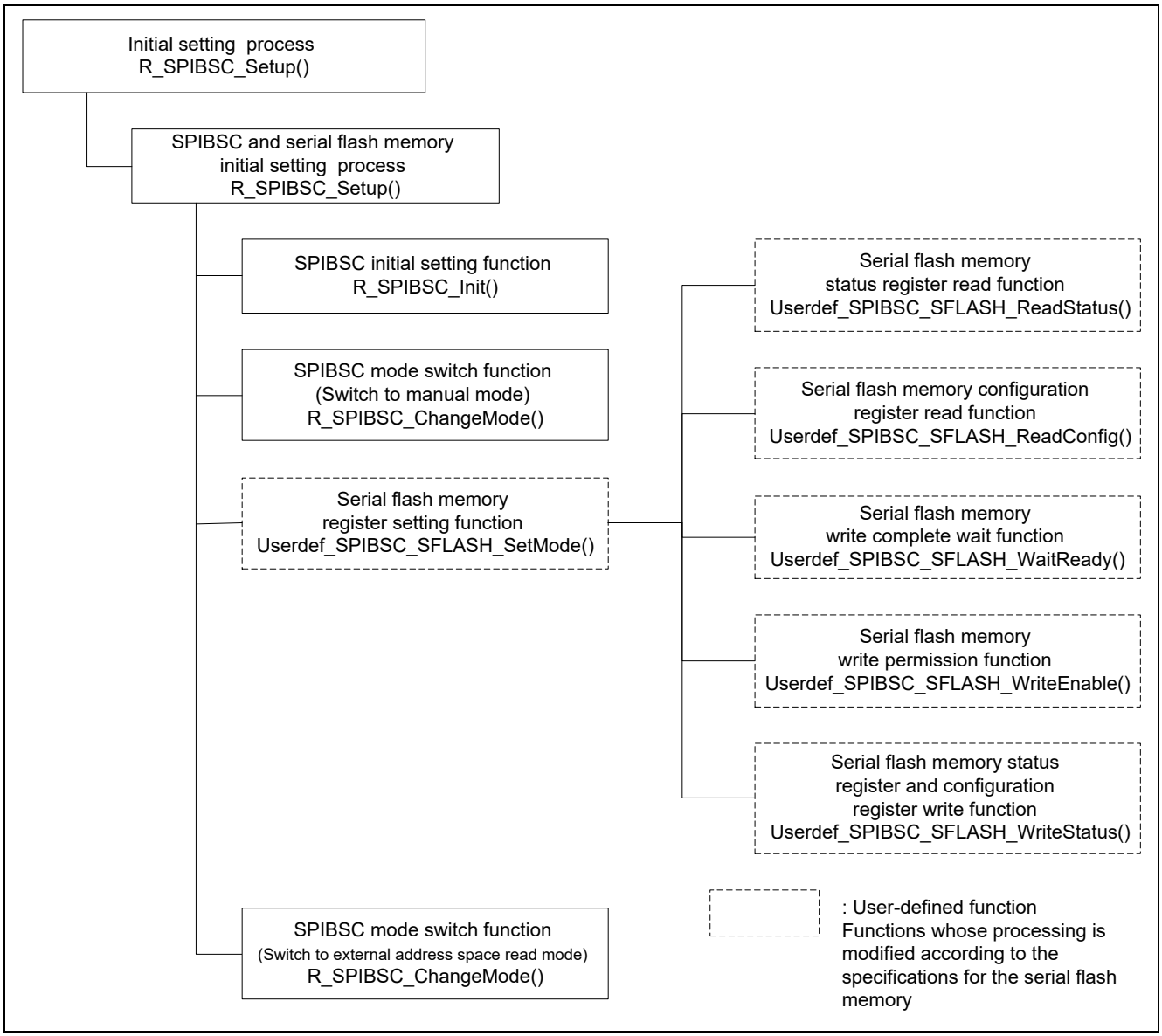


Figure 6.3 Hierarchical module diagram of the SPIBSC and serial flash memory settings

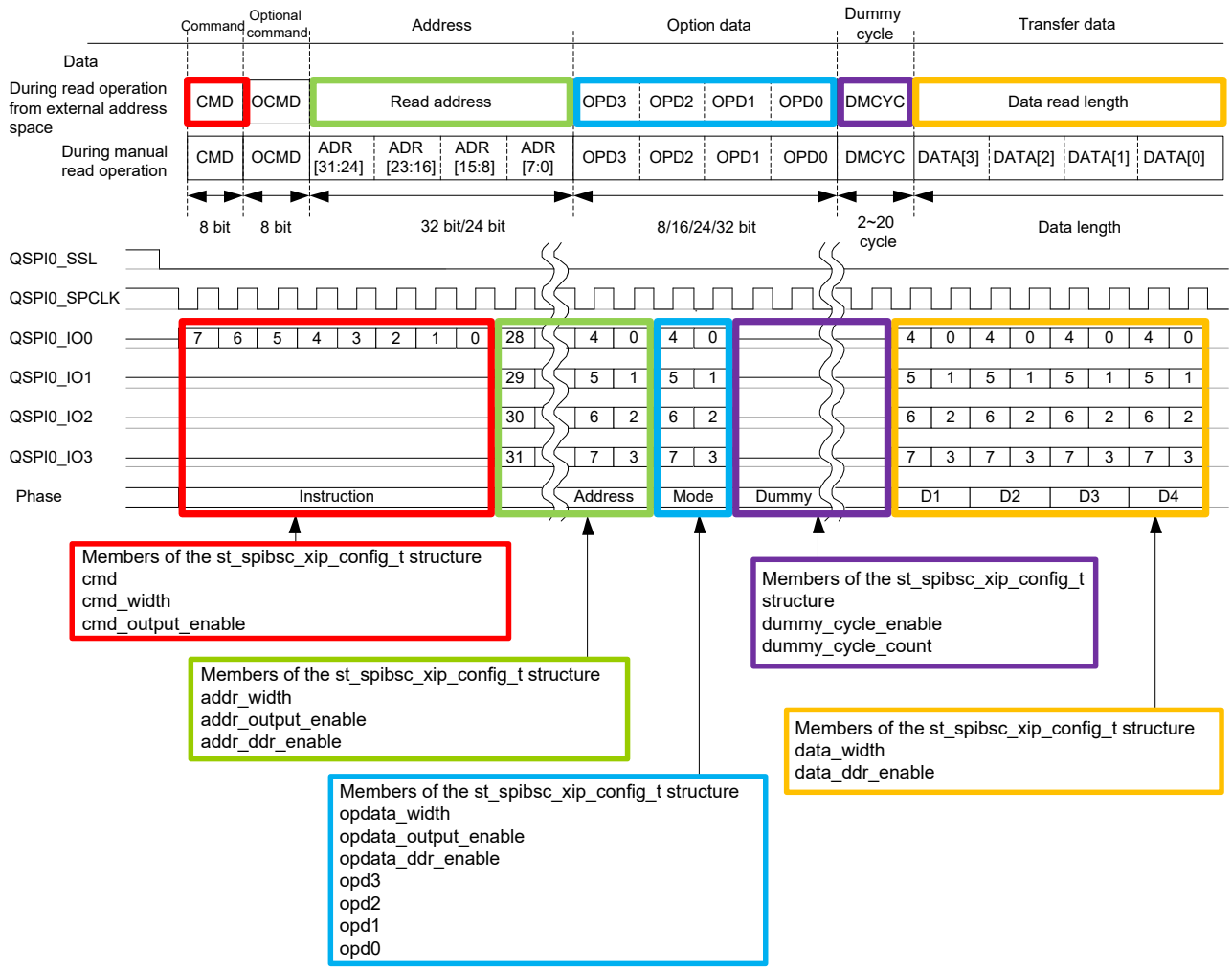
6.3.1 Signal Output when a Read Command is Issued

In external address space read mode, a read access to the SPI multi-I/O bus space is initiated by sending a signal, which is converted for SPI communication, to the serial flash memory when issuing the read command. In the case of changing the serial flash memory to be used, the output signal for issuing a read command needs to be changed according to the specifications for the serial flash memory read command.

The SPIBSC allows the read command signal to be output to the serial flash memory by setting up the SPIBSC register in the external address space read mode.

In the sample code, the SPIBSC register settings can be changed and the output signal when a read command is issued can be changed by changing the contents of the SPIBSC external address space read mode's read command settings tables (Table 6.8, Table 6.9). The SPIBSC register settings specified in the read command settings tables are made by executing the SPIBSC operating mode setting function (R_SPIBSC_ChangeMode). The serial flash memory register settings related to the read command settings (number of dummy cycles, bit width, etc.) are made by executing the serial flash memory register setting function (Userdef_SPIBSC_SFLASH_SetMode). Even if executing register settings in the serial flash memory with the Userdef_SPIBSC_SFLASH_SetMode function, make the changes according to the specifications of the serial flash memory to be used.

Figure 6.4 shows the Correspondence between SPIBSC register settings and waveforms output to serial flash memory during external address read operation. Refer to these example settings when changing the settings in the external address space read mode's read command settings tables (Table 6.8 and Table 6.9) to match the read command of the serial flash memory used.



In the sample code, when Macronix serial flash memory (MX25L51245GXD) is used, each of the structural members is set as shown below.

■When DDR read command (EEh) used: gs_xip_read_table[0]

cmd	: H'EE
cmd_width	: SPIBSC_1BIT_WIDTH
cmd_output_enable	: SPIBSC_OUTPUT_ENABLE
addr_width	: SPIBSC_4BIT_WIDTH
addr_output_enable	: SPIBSC_OUTPUT_ADDR 32
addr_ddr_enable	: SPIBSC_DDR_TRANSFER
opdata_width	: SPIBSC_4BIT_WIDTH
opdata_output_enable	: SPIBSC_OUTPUT_OPD_3
opdata_ddr_enable	: SPIBSC_DDR_TRANSFER
opd3	: H'00
opd2	: H'00 (OPD2 not output.)
opd1	: H'00 (OPD1 not output.)
opd0	: H'00 (OPD0 not output.)
dummy_cycle_enable	: SPIBSC_OUTPUT_ENABLE
dummy_cycle_count	: SPIBSC_DUMMY_7CYC
data_width	: SPIBSC_4BIT_WIDTH
data_ddr_enable	: SPIBSC_DDR_TRANSFER

■When SDR read command (ECh) used: gs_xip_read_table[1]

cmd	: H'EC
cmd_width	: SPIBSC_1BIT WIDTH
cmd_output_enable	: SPIBSC_OUTPUT_ENABLE
addr_width	: SPIBSC_4BIT_WIDTH
addr_output_enable	: SPIBSC_OUTPUT_ADDR 32
addr_ddr_enable	: SPIBSC_SDR_TRANSFER
opdata_width	: SPIBSC_4BIT_WIDTH
opdata_output_enable	: SPIBSCOUTPUT_OPD_3
opdata_ddr_enable	: SPIBSC_SDR_TRANSFER
opd3	: H'00
opd2	: H'00 (OPD2 not output.)
opd1	: H'00 (OPD1 not output.)
opd0	: H'00 (OPD0 not output.)
dummy_cycle_enable	: SPIBSC_OUTPUT_ENABLE
dummy_cycle_count	: SPIBSC_DUMMY_2CYC
data_width	: SPIBSC_4BIT_WIDTH
data_ddr_enable	: SPIBSC_SDR_TRANSFER

Note: Mode cycle in the figure corresponds to the cycle of the performance enhance indicator for the MX25L51245GXD.

Figure 6.4 Correspondence between SPIBSC register settings and waveforms output to serial flash memory during external address read operation

Table 6.8 Command settings table for external address space read mode gs_xip_read_table[0]: 4BIT_DDR_READ

Member	Description	Setting value
uint8_t command_name[20]	Command identification character string	"4BIT_DDR_READ"
uint8_t cmd	Command code	**0xEE**
uint8_t cmd_width	Command bit width	**SPIBSC_1BIT_WIDTH**
uint8_t cmd_output_enable	Command output enable/disable	**SPIBSC_OUTPUT_ENABLE**
uint8_t ocmd	Optional command code	0x00
uint8_t ocmd_width	Optional command bit width	SPIBSC_1BIT_WIDTH
uint8_t ocmd_output_enable	Optional command output enable/disable	SPIBSC_OUTPUT_DISABLE
uint8_t addr_width	Address bit width	**SPIBSC_4BIT_WIDTH**
uint8_t addr_output_enable	Address output setting	**SPIBSC_OUTPUT_ADDR_32**
uint8_t addr_ddr_enable	Address transfer method	**SPIBSC_DDR_TRANSFER**
uint8_t opdata_width	Option data bit width	**SPIBSC_4BIT_WIDTH**
uint8_t opdata_output_enable	Option data output setting	**SPIBSC_OUTPUT_OPD_3 (Note)**
uint8_t opdata_ddr_enable	Option data transfer method	**SPIBSC_DDR_TRANSFER (Note)**
uint8_t opd3	Option Data3 (8bit) setting (outputs for the first)	**0x00 (Note)**
uint8_t opd2	Option Data2 (8bit) setting (outputs for the second)	0x00
uint8_t opd1	Option Data1 (8bit) setting (outputs for the third)	0x00
uint8_t opd0	Option Data0 (8bit) setting (outputs for the fourth)	0x00
uint8_t dummy_cycle_enable	Dummy cycle output enable/disable	**SPIBSC_OUTPUT_ENABLE**
uint8_t dummy_cycle_count	Number of dummy cycles	**SPIBSC_DUMMY_07CYC**
uint8_t data_width	Transfer data bit width	**SPIBSC_4BIT_WIDTH**
uint8_t data_ddr_enable	Transfer data transfer method	**SPIBSC_DDR_TRANSFER**

Note: The MX25L51245GXD transits to the Performance Enhance Mode when data (e.g., H'A5, H'5A, H'F0, H'0F, etc.) that toggles between bits 7-4 and bits 3-0 is input during the performance enhance indicator cycle that follows the address cycle (P[7:0] in Figure 6.5). Since the RZ/A2M's external address space read mode does not support the data transfer in Performance Enhance Mode, the sample code makes configuration so that the MX25L51245GXD will not switch into the Performance Enhance Mode by making configuration so that H'00 is output from the OPD3.

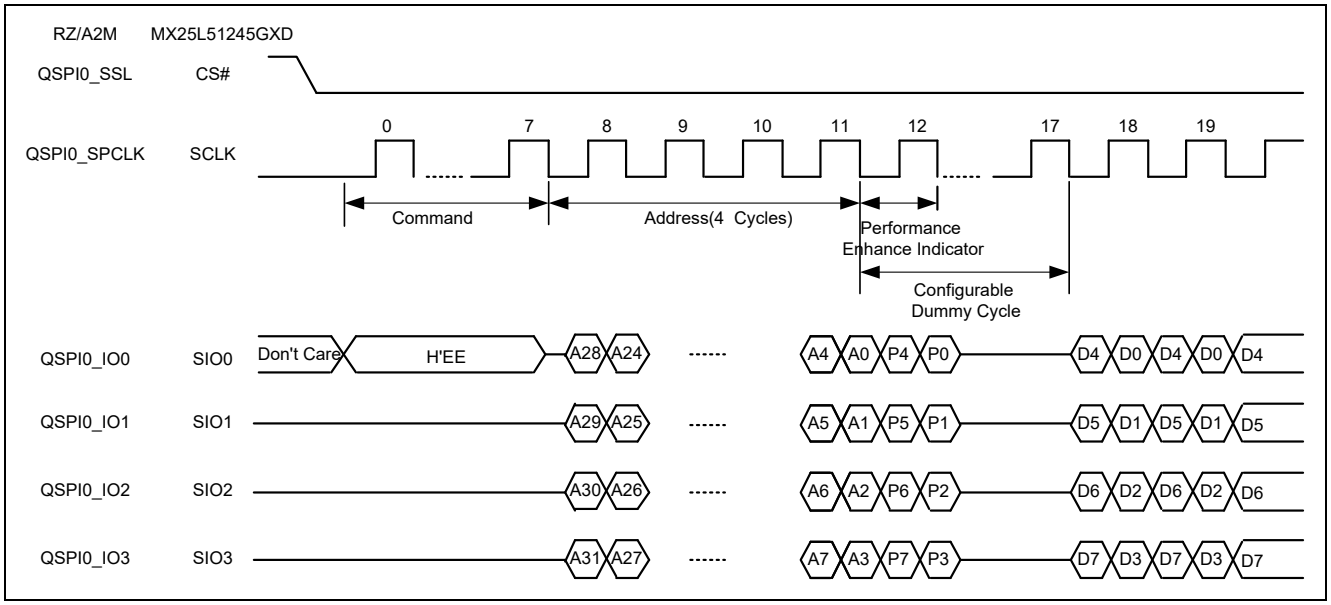


Figure 6.5 Waveform format of EEH read command (reference)

Table 6.9 Command settings table for external address space read mode gs_xip_read_table[1]: 4BIT_SDR_READ

Member	Description	Setting value
uint8_t command_name[20]	Command identification character string	"4BIT_SDR_READ"
uint8_t cmd	Command code	**0xEC**
uint8_t cmd_width	Command bit width	**SPIBSC_1BIT_WIDTH**
uint8_t cmd_output_enable	Command output enable/disable	**SPIBSC_OUTPUT_ENABLE**
uint8_t ocmd	Optional command code	0x00
uint8_t ocmd_width	Optional command bit width	SPIBSC_1BIT_WIDTH
uint8_t ocmd_output_enable	Optional command output enable/disable	SPIBSC_OUTPUT_DISABLE
uint8_t addr_width	Address bit width	**SPIBSC_4BIT_WIDTH**
uint8_t addr_output_enable	Address output setting	**SPIBSC_OUTPUT_ADDR_32**
uint8_t addr_ddr_enable	Address transfer method	**SPIBSC_SDR_TRANSFER**
uint8_t opdata_width	Option data bit width	**SPIBSC_4BIT_WIDTH**
uint8_t opdata_output_enable	Option data output setting	**SPIBSC_OUTPUT_OPD_3 (Note)**
uint8_t opdata_ddr_enable	Option data transfer method	**SPIBSC_SDR_TRANSFER (Note)**
uint8_t opd3	Option Data3 (8bit) setting (outputs for the first)	**0x00 (Note)**
uint8_t opd2	Option Data2 (8bit) setting (outputs for the second)	0x00
uint8_t opd1	Option Data1 (8bit) setting (outputs for the third)	0x00
uint8_t opd0	Option Data0 (8bit) setting (outputs for the fourth)	0x00
uint8_t dummy_cycle_enable	Dummy cycle output enable/disable	**SPIBSC_OUTPUT_ENABLE**
uint8_t dummy_cycle_count	Number of dummy cycles	**SPIBSC_DUMMY_02CYC**
uint8_t data_width	Transfer data bit width	**SPIBSC_4BIT_WIDTH**
uint8_t data_ddr_enable	Transfer data transfer method	**SPIBSC_SDR_TRANSFER**

Note: The MX25L51245GXD transits to the Performance Enhance Mode when data (e.g., H'A5, H'5A, H'F0, H'0F, etc.) that toggles between bits 7-4 and bits 3-0 is input during the performance enhance indicator cycle that follows the address cycle (P[7:0] in Figure 6.6). Since the RZ/A2M's external address space read mode does not support the data transfer in Performance Enhance Mode, the sample code makes configuration so that the MX25L51245GXD will not switch into the Performance Enhance Mode by making configuration so that H'00 is output from the OPD3.

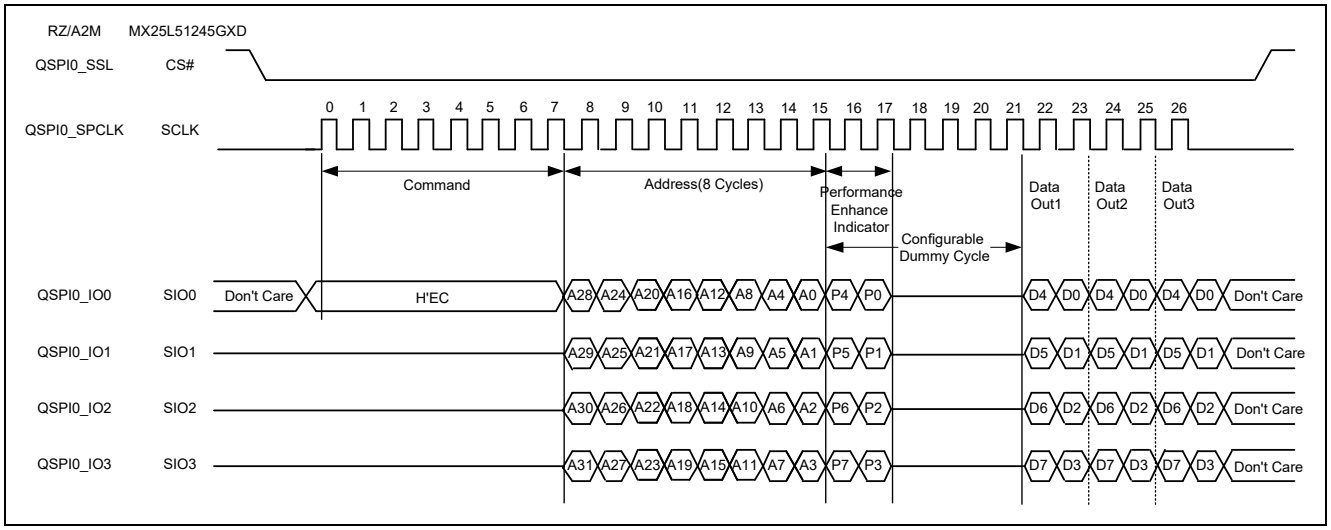


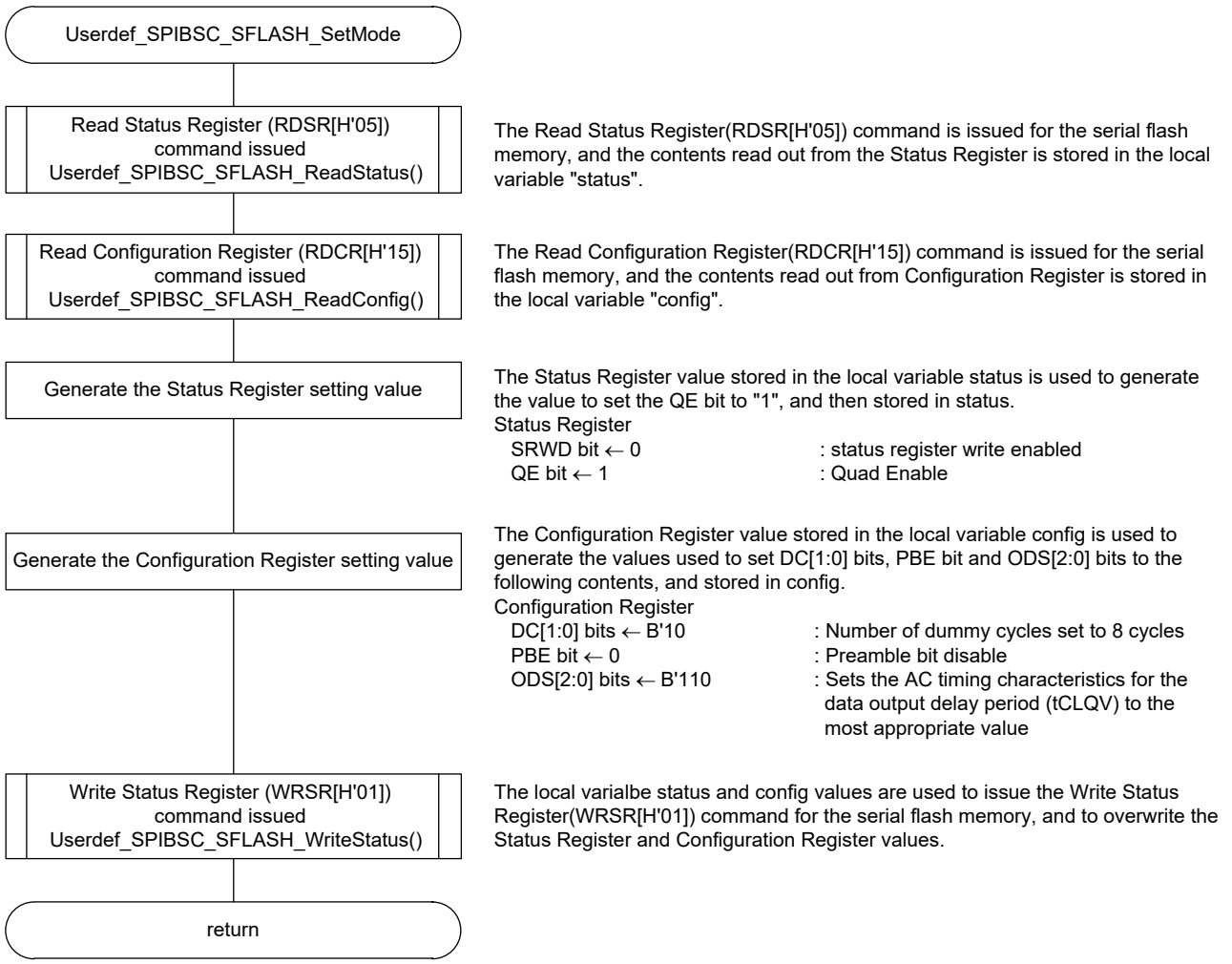
Figure 6.6 Waveform format of ECH read command (reference)

6.3.2 Setting up the Serial Flash Memory Registers

In the sample code, the user-defined function Userdef_SPIBSC_SFLASH_SetMode executes the processing for the serial flash memory MX25L51245GXD register settings (QE bit of the status register, and DC[1:0] bits, PBE bit, and ODS[2:0] bits of the configuration register) according to the specifications of the read command to be used.

In the sample code, because the Quad read command (H'EE or H'EC) is used as the read command, the status register QE (Quad Enable) bit is set to 1 so the bit width is 4 bits. Also, the DC[1:0] bits of the configuration register are set so that the number of inserted dummy cycles is the minimum needed for the read command and operating frequency used (refer to Table 6.5 List of numbers of dummy cycles necessary for the operating frequency of the MX25L51245GXD for details). Further, the ODS[2:0] bits of the configuration register are set to B'110 to achieve the data output delay (tCLQV) needed for the connection of the RZ/A2M and MX25L51245GXD. Change the execution of the Userdef_SPIBSC_SFLASH_SetMode function so that the serial flash memory's control register setting conforms to the read command specifications of the serial flash memory to be used.

Figure 6.7 shows the Userdef_SPIBSC_SFLASH_SetMode function processing flow of the sample code.



Note: The serial flash memory access is changed from 1-bit width to 4-bit width, and the Status Register and Configuration Register are set to achieve the minimum number of dummy cycles for the read command. These register settings require the WRSR command to be issued to the serial flash memory after switching the SPIBSC from external address space read mode to manual mode.

Figure 6.7 Userdef_SPIBSC_SFLASH_SetMode function processing flow

6.3.3 Serial Flash Memory Write Completion Wait

The serial flash memory transits to a busy state when the serial flash memory registers (Status Register and Configuration Register) or memory are written to. It is then necessary to wait until the written data is reflected before accessing the serial flash memory.

In the sample code, this wait processing is executed with the Userdef_SPIBSC_SFLASH_WaitReady function.

Implement the Userdef_SPIBSC_SFLASH_WaitReady function according to the specifications of the serial flash memory to be used so that it can wait until completion of serial flash memory writing.

In the sample code, The Status Register WIP bit is read, and wait processing is performed until writing is complete.

Figure 6.8 shows the Userdef_SPIBSC_SFLASH_WaitReady function processing flow of the sample code.

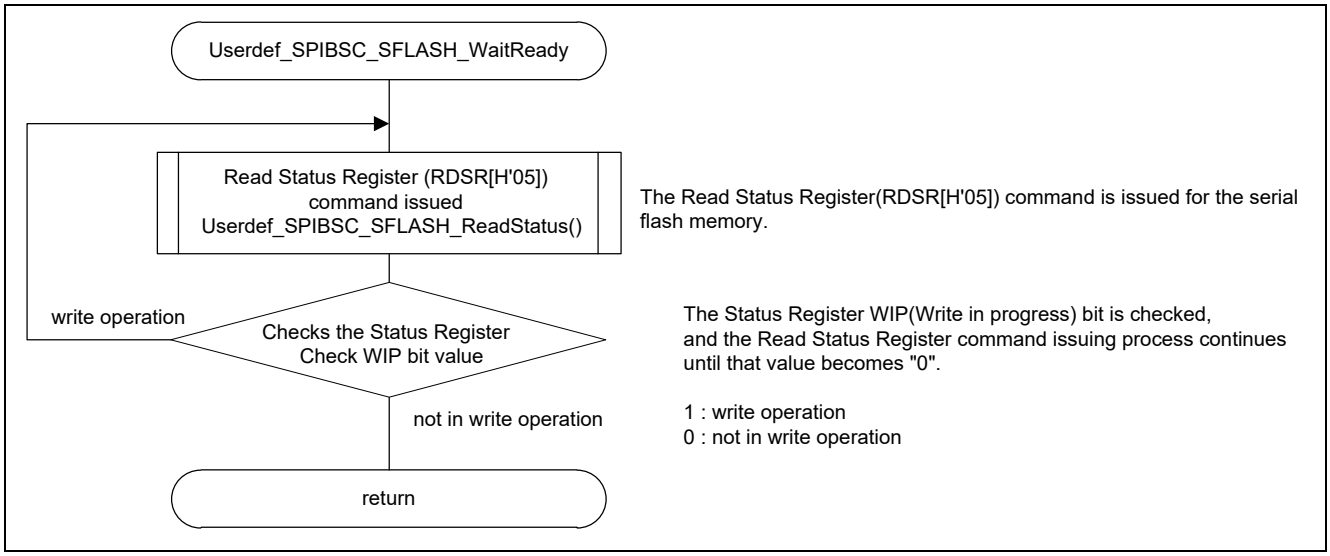


Figure 6.8 Userdef_SPIBSC_SFLASH_WaitReady function processing flow

6.3.4 Serial Flash Memory Status Register Read

In the sample code, serial flash memory Status Register reading is executed with the Userdef_SPIBSC_SFLASH_ReadStatus function.

Implement the serdef_SPIBSC_SFLASH_ReadStatus function according to the specifications of the serial flash memory to be used so that it can read the serial flash memory status register.

Figure 6.9 shows the Userdef_SPIBSC_SFLASH_ReadStatus function processing flow of the sample code.

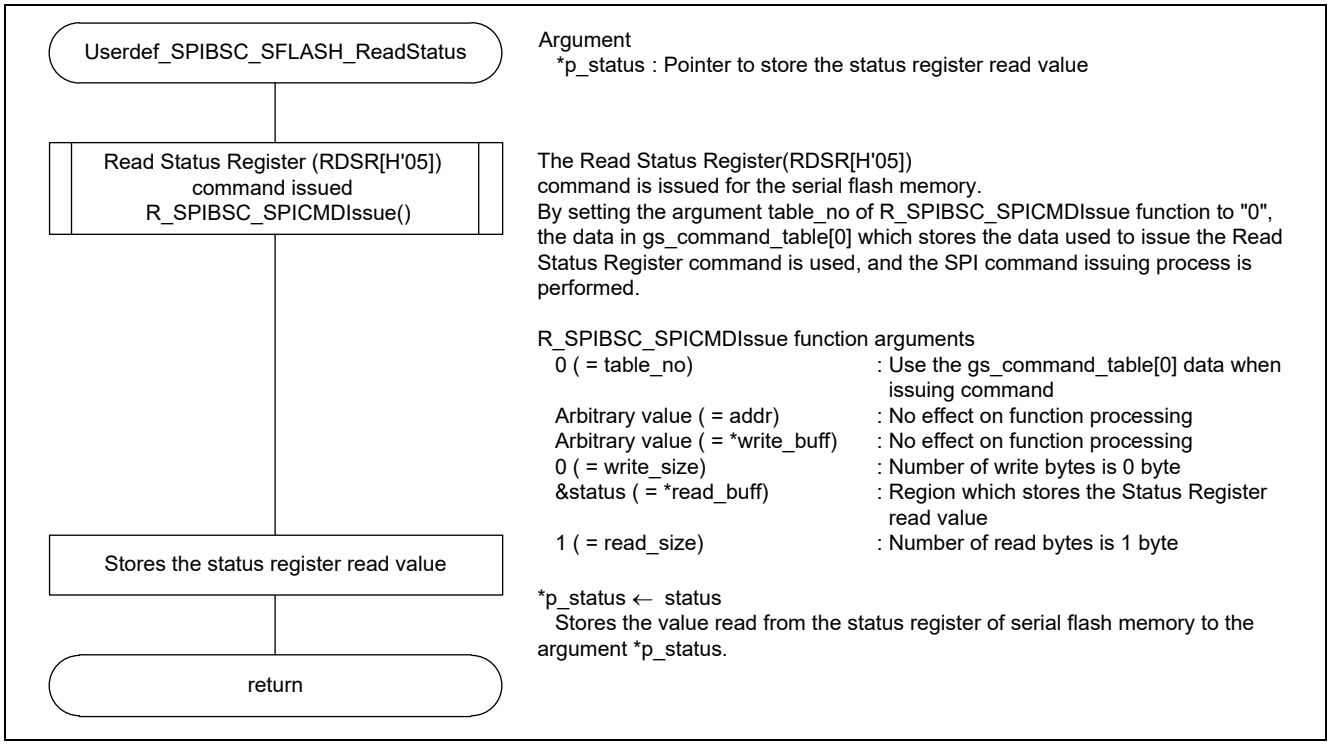


Figure 6.9 Userdef_SPIBSC_SFLASH_ReadStatus function processing flow

Table 6.10 Command settings table for manual mode gs_command_table[0]: READ STATUS command

Member	Description	Setting value
uint8_t command_name[20]	Command identification character string	"READ STATUS"
uint8_t cmd	Command code	**0x05**
uint8_t cmd_width	Command bit width	**SPIBSC_1BIT_WIDTH**
uint8_t cmd_output_enable	Command output enable/disable	**SPIBSC_OUTPUT_ENABLE**
uint8_t ocmd	Optional command code	0x00
uint8_t ocmd_width	Optional command bit width	SPIBSC_1BIT_WIDTH
uint8_t ocmd_enable	Optional command output enable/disable	SPIBSC_OUTPUT_DISABLE
uint8_t addr_width	Address bit width	SPIBSC_1BIT_WIDTH
uint8_t addr_output_enable	Address output setting	SPIBSC_OUTPUT_DISABLE
uint8_t addr_sdr_ddr	Address transfer method	SPIBSC_SDR_TRANSFER
uint8_t opdata_width	Option data bit width	SPIBSC_1BIT_WIDTH
uint8_t opdata_output_enable	Option data output setting	SPIBSC_OUTPUT_DISABLE
uint8_t opdata_ddr_enable	Option data transfer method	SPIBSC_SDR_TRANSFER
uint8_t opd3	Option Data3 (8bit) setting (outputs for the first)	0x00
uint8_t opd2	Option Data2 (8bit) setting (outputs for the second)	0x00
uint8_t opd1	Option Data1 (8bit) setting (outputs for the third)	0x00
uint8_t opd0	Option Data0 (8bit) setting (outputs for the fourth)	0x00
uint8_t dummy_cycle_output_enable	Dummy cycle output enable/disable	SPIBSC_OUTPUT_DISABLE
uint8_t dummy_cycle_count	Number of dummy cycles	0x00
uint8_t transfer_data_width	Transfer data bit width	**SPIBSC_1BIT_WIDTH**
uint8_t transfer_data_sdr_ddr	Transfer data transfer method	**SPIBSC_SDR_TRANSFER**

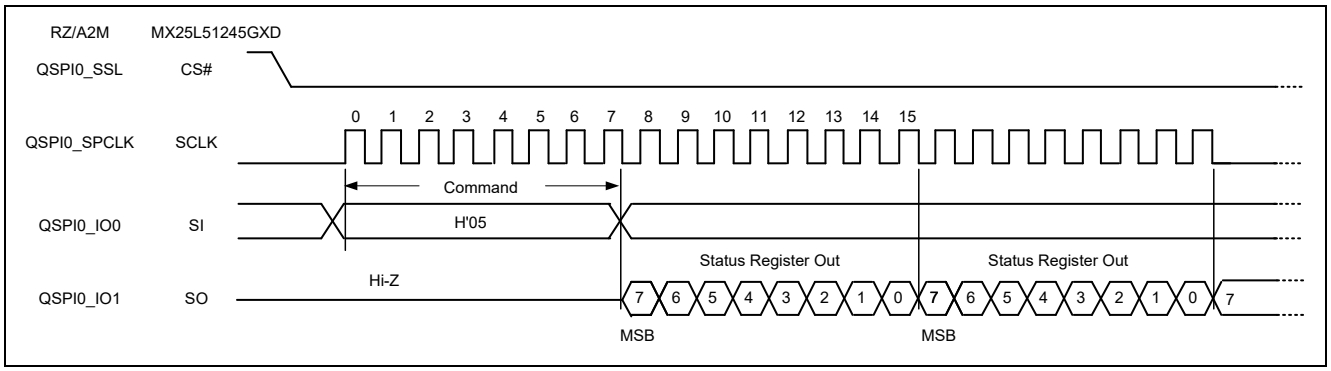


Figure 6.10 Waveform format of READ STATUS command (reference)

6.3.5 Serial Flash Memory Configuration Register Read

In the sample code, serial flash memory configuration register reading is executed with the Userdef_SPIBSC_SFLASH_ReadConfig function.

Implement the Userdef_SPIBSC_SFLASH_ReadConfig function according to the specifications of the serial flash memory to be used so that it can read the serial flash memory configuration register.

Figure 6.11 shows the Userdef_SPIBSC_SFLASH_ReadConfig function processing flow of the sample code.

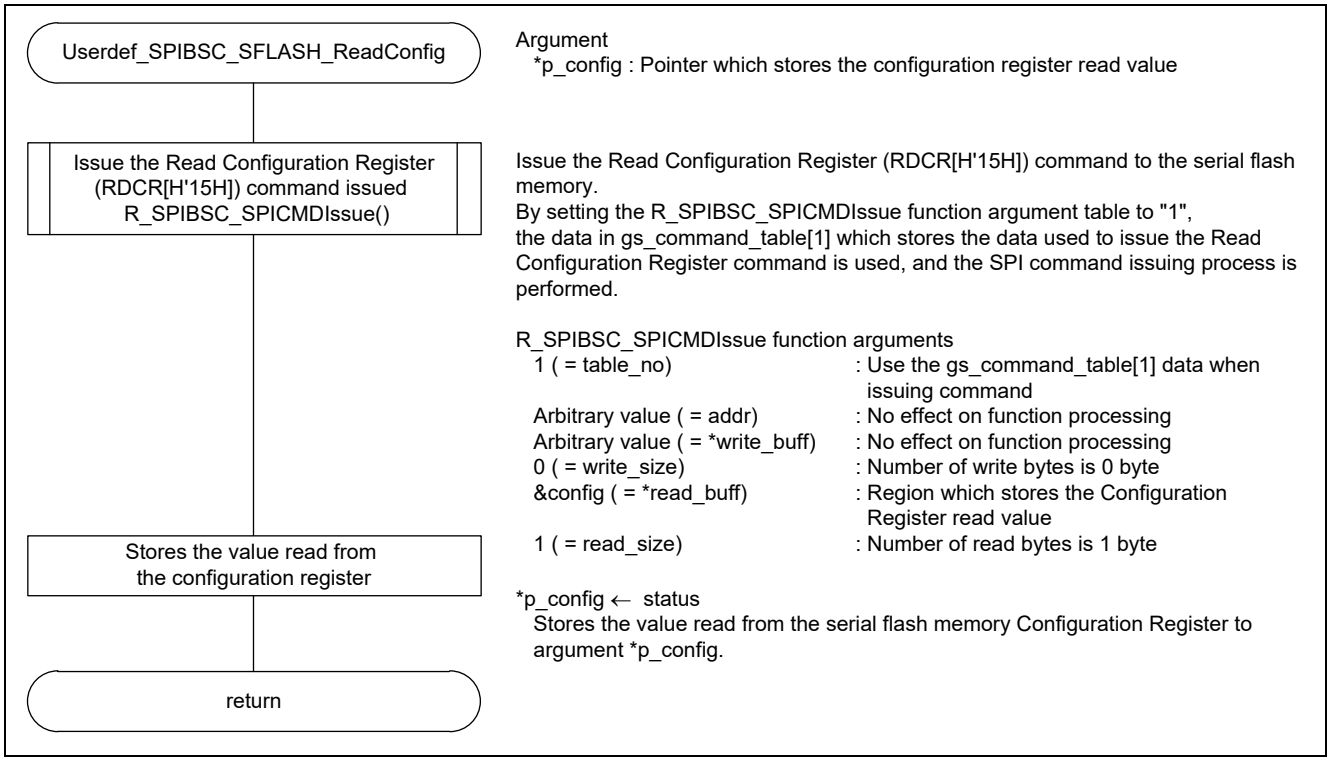


Figure 6.11 Userdef_SPIBSC_SFLASH_ReadConfig function processing flow

Table 6.11 Command settings table for manual mode gs_command_table[1]: READ CONFIGURATION Command

Member	Description	Setting value
uint8_t command_name[20]	Command identification character string	"READ CONFIGURATION"
uint8_t cmd	Command code	**0x15**
uint8_t cmd_width	Command bit width	**SPIBSC_1BIT_WIDTH**
uint8_t cmd_output_enable	Command output enable/disable	**SPIBSC_OUTPUT_ENABLE**
uint8_t ocmd	Optional command code	0x00
uint8_t ocmd_width	Optional command bit width	SPIBSC_1BIT_WIDTH
uint8_t ocmd_enable	Optional command output enable/disable	SPIBSC_OUTPUT_DISABLE
uint8_t addr_width	Address bit width	SPIBSC_1BIT_WIDTH
uint8_t addr_output_enable	Address output setting	SPIBSC_OUTPUT_DISABLE
uint8_t addr_sdr_ddr	Address transfer method	SPIBSC_SDR_TRANSFER
uint8_t opdata_width	Option data bit width	SPIBSC_1BIT_WIDTH
uint8_t opdata_output_enable	Option data output setting	SPIBSC_OUTPUT_DISABLE
uint8_t opdata_ddr_enable	Option data transfer method	SPIBSC_SDR_TRANSFER
uint8_t opd3	Option Data3 (8bit) setting (outputs for the first)	0x00
uint8_t opd2	Option Data2 (8bit) setting (outputs for the second)	0x00
uint8_t opd1	Option Data1 (8bit) setting (outputs for the third)	0x00
uint8_t opd0	Option Data0 (8bit) setting (outputs for the fourth)	0x00
uint8_t dummy_cycle_output_enable	Dummy cycle output enable/disable	SPIBSC_OUTPUT_DISABLE
uint8_t dummy_cycle_count	Number of dummy cycles	0x00
uint8_t transfer_data_width	Transfer data bit width	**SPIBSC_1BIT_WIDTH**
uint8_t transfer_data_sdr_ddr	Transfer data transfer method	**SPIBSC_SDR_TRANSFER**

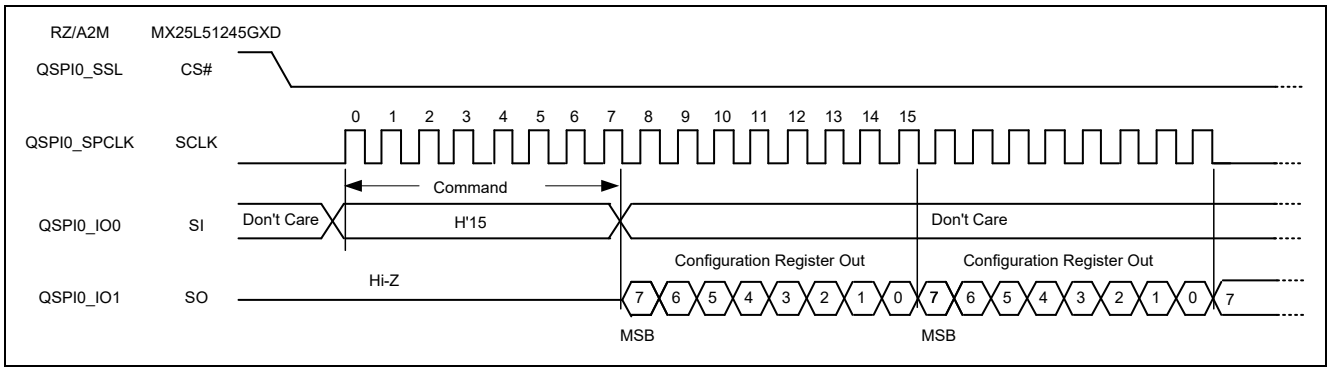


Figure 6.12 Waveform format of READ CONFIGURATION command (reference)

6.3.6 Serial Flash Memory Write Enable

It is necessary to enable the serial flash memory for writes before writing data to the registers (Status Register and Configuration Register) of the serial flash memory. In the sample code, this wait processing is executed with the Userdef_SPIBSC_SFLASH_WriteEnable function.

Implement the Userdef_SPIBSC_SFLASH_WriteEnable function according to the specifications for the serial flash memory to be used so that it can be enabled for writes. In the sample code, a Write Enable command (WREN [H'06]) is issued, thereby enabling writes (setting the WEL bit of the Status Register to 1).

Figure 6.13 shows the Userdef_SPIBSC_SFLASH_WriteEnable function processing flow of the sample code.

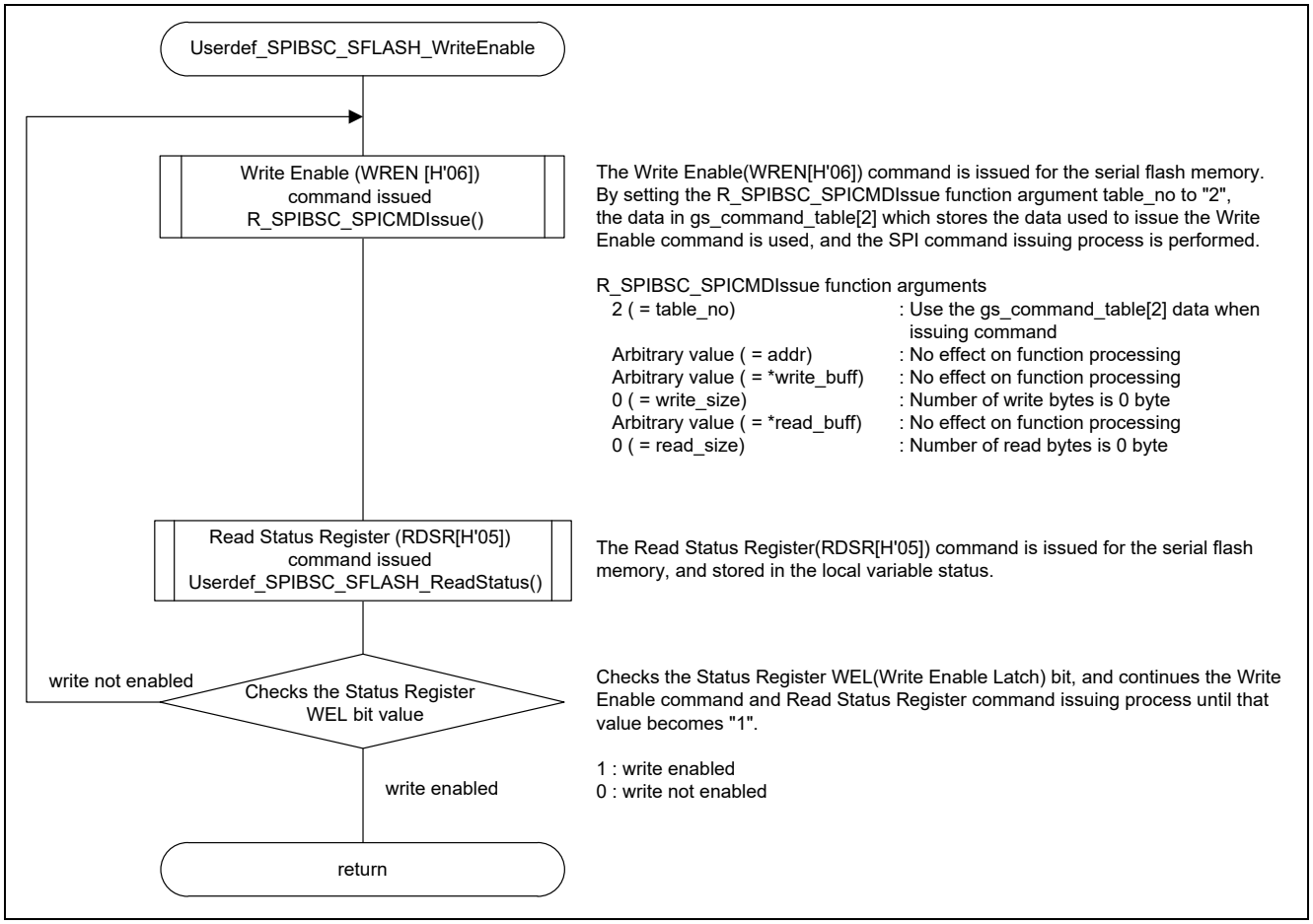


Figure 6.13 Userdef_SPIBSC_SFLASH_WriteEnable function processing flow

Table 6.12 Command settings table for manual mode gs_command_table[2]: WRITE ENABLE command

Member	Description	Setting value
uint8_t command_name[20]	Command identification character string	"WRITE ENABLE"
uint8_t cmd	Command code	**0x06**
uint8_t cmd_width	Command bit width	**SPIBSC_1BIT_WIDTH**
uint8_t cmd_output_enable	Command output enable/disable	**SPIBSC_OUTPUT_ENABLE**
uint8_t ocmd	Optional command code	0x00
uint8_t ocmd_width	Optional command bit width	SPIBSC_1BIT_WIDTH
uint8_t ocmd_enable	Optional command output enable/disable	SPIBSC_OUTPUT_DISABLE
uint8_t addr_width	Address bit width	SPIBSC_1BIT_WIDTH
uint8_t addr_output_enable	Address output setting	SPIBSC_OUTPUT_DISABLE
uint8_t addr_sdr_ddr	Address transfer method	SPIBSC_SDR_TRANSFER
uint8_t opdata_width	Option data bit width	SPIBSC_1BIT_WIDTH
uint8_t opdata_output_enable	Option data output setting	SPIBSC_OUTPUT_DISABLE
uint8_t opdata_ddr_enable	Option data transfer method	SPIBSC_SDR_TRANSFER
uint8_t opd3	Option Data3 (8bit) setting (outputs for the first)	0x00
uint8_t opd2	Option Data2 (8bit) setting (outputs for the second)	0x00
uint8_t opd1	Option Data1 (8bit) setting (outputs for the third)	0x00
uint8_t opd0	Option Data0 (8bit) setting (outputs for the fourth)	0x00
uint8_t dummy_cycle_output_enable	Dummy cycle output enable/disable	SPIBSC_OUTPUT_DISABLE
uint8_t dummy_cycle_count	Number of dummy cycles	0x00
uint8_t transfer_data_width	Transfer data bit width	SPIBSC_1BIT_WIDTH
uint8_t transfer_data_sdr_ddr	Transfer data transfer method	SPIBSC_SDR_TRANSFER

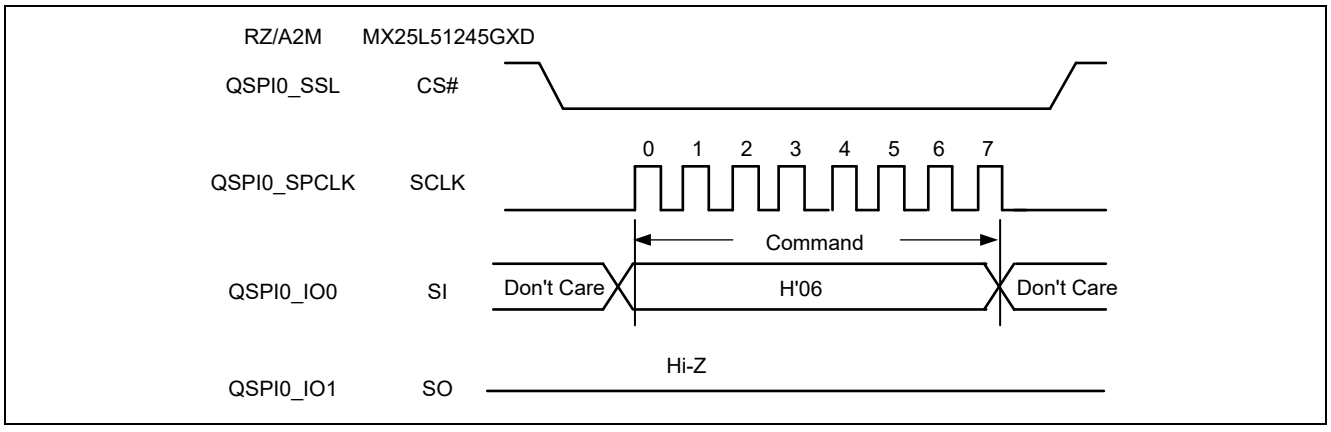


Figure 6.14 Waveform format of WRITE ENABLE command (reference)

6.3.7 Serial Flash Memory Status/Configuration Register Write

In the sample code, writing to the QE bit of the status register and DC[1:0] bits of the configuration register in the serial flash memory is executed with the Userdef_SPIBSC_SFLASH_WriteStatus function.

Implement the Userdef_SPIBSC_SFLASH_WriteStatus function according to the specifications of the serial flash memory to be used so that it can write the serial flash memory status register and configuration register settings.

Figure 6.15 shows the Userdef_SPIBSC_SFLASH_WriteStatus function processing flow of the sample code.

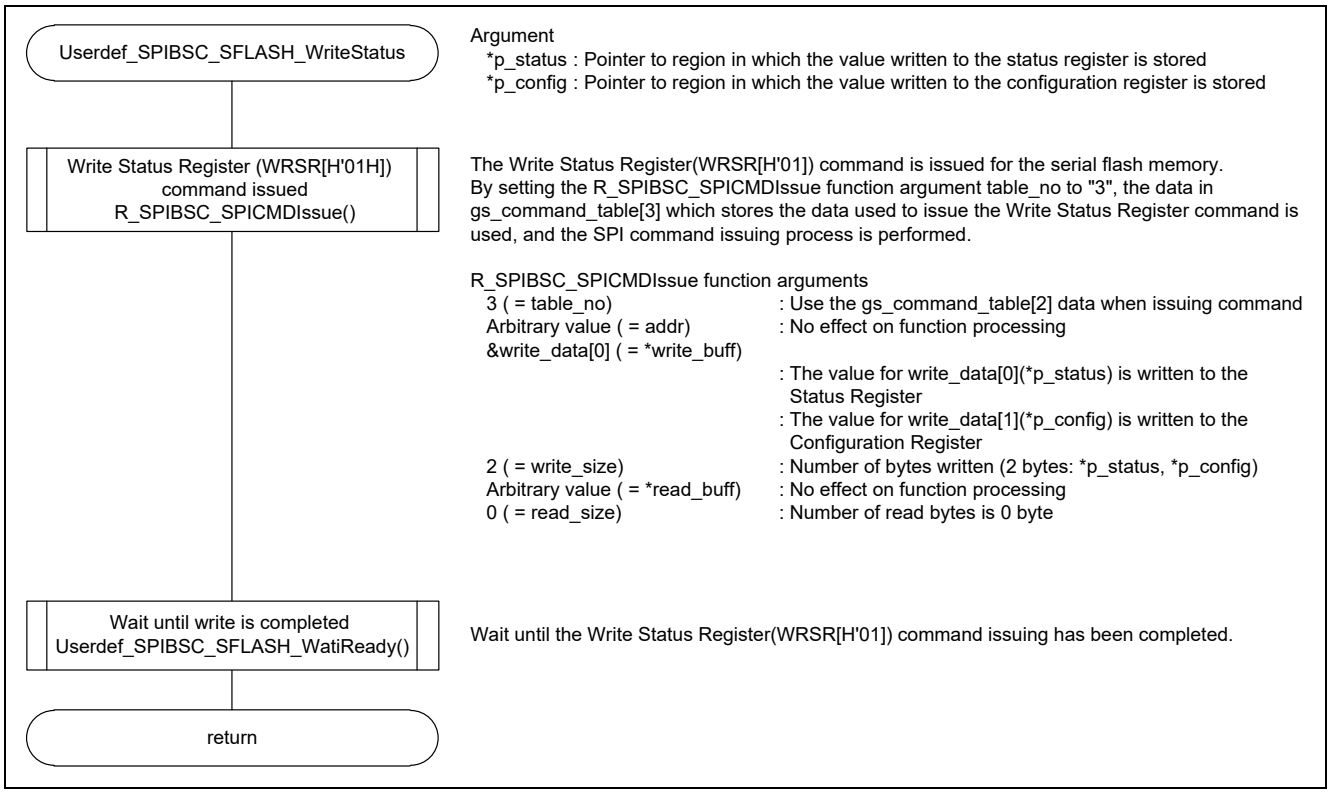


Figure 6.15 Userdef_SPIBSC_SFLASH_WriteStatus function processing flow

Table 6.13 Command settings table for manual mode gs_command_table[3]: WRITE STATUS command

Member	Description	Setting value
uint8_t command_name[20]	Command identification character string	"WRITE STATUS"
uint8_t cmd	Command code	**0x01**
uint8_t cmd_width	Command bit width	**SPIBSC_1BIT_WIDTH**
uint8_t cmd_output_enable	Command output enable/disable	**SPIBSC_OUTPUT_ENABLE**
uint8_t ocmd	Optional command code	0x00
uint8_t ocmd_width	Optional command bit width	SPIBSC_1BIT_WIDTH
uint8_t ocmd_enable	Optional command output enable/disable	SPIBSC_OUTPUT_DISABLE
uint8_t addr_width	Address bit width	SPIBSC_1BIT_WIDTH
uint8_t addr_output_enable	Address output setting	SPIBSC_OUTPUT_DISABLE
uint8_t addr_sdr_ddr	Address transfer method	SPIBSC_SDR_TRANSFER
uint8_t opdata_width	Option data bit width	SPIBSC_1BIT_WIDTH
uint8_t opdata_output_enable	Option data output setting	SPIBSC_OUTPUT_DISABLE
uint8_t opdata_ddr_enable	Option data transfer method	SPIBSC_SDR_TRANSFER
uint8_t opd3	Option Data3 (8bit) setting (outputs for the first)	0x00
uint8_t opd2	Option Data2 (8bit) setting (outputs for the second)	0x00
uint8_t opd1	Option Data1 (8bit) setting (outputs for the third)	0x00
uint8_t opd0	Option Data0 (8bit) setting (outputs for the fourth)	0x00
uint8_t dummy_cycle_output_enable	Dummy cycle output enable/disable	SPIBSC_OUTPUT_DISABLE
uint8_t dummy_cycle_count	Number of dummy cycles	0x00
uint8_t transfer_data_width	Transfer data bit width	**SPIBSC_1BIT_WIDTH**
uint8_t transfer_data_sdr_ddr	Transfer data transfer method	**SPIBSC_SDR_TRANSFER**

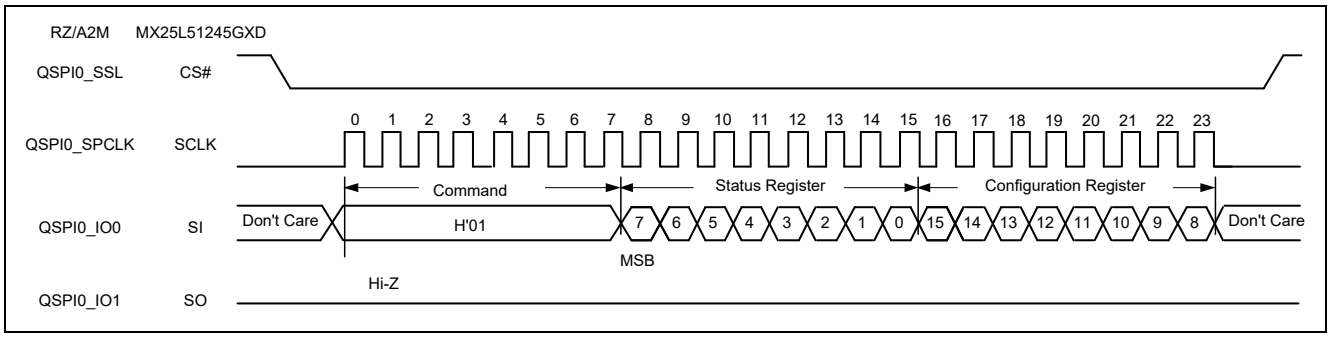


Figure 6.16 Waveform format of WRITE STATUS command (reference)

7. Sample Code Precautions

7.1 Accessible area in external address space read mode

The accessible area in external address space read mode is a 256MB area assigned to the SPI multi-IO bus space (H'2000_0000 to H'2FFF_FFFF). The SPIBSC converts access to this area to H'0000_0000 to H'0FFF_FFFF in the serial flash memory to access it. In the sample code, a 256MB area from H'0000_0000 to H'0FFF_FFFF in the serial flash memory can be accessed. In the RZ/A2M, an area larger than 256MB can be accessed by controlling the SPIBSC data read expansion address setting register (DREAR) and changing the serial flash memory address allocated to the SPI multi-I/O bus space, but since this prevents access to the area before DREAR is controlled, access to an area larger than 256MB is not supported in the sample code. The access specifications are only for H'0000_0000 to H'0FFF_FFFF in the serial flash memory.

In manual mode, access using a 4-byte address is supported, and a 4GB area of the serial flash memory can be accessed.

8. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

9. Reference Documents

User's Manual: Hardware

RZ/A2M Group User's Manual: Hardware
The latest version can be downloaded from the Renesas Electronics website.

RTK7921053C00000BE (RZ/A2M CPU board) User's Manual
The latest version can be downloaded from the Renesas Electronics website.

RTK79210XXB00000BE (RZ/A2M SUB board) User's Manual
The latest version can be downloaded from the Renesas Electronics website.

Arm Architecture Reference Manual ARMv7-A and ARMv7-R edition Issue C
The latest version can be downloaded from the Arm website.

Arm Cortex™-A9 Technical Reference Manual Revision: r4p1
The latest version can be downloaded from the Arm website.

Arm Generic Interrupt Controller Architecture Specification - Architecture version2.0
The latest version can be downloaded from the Arm website.

Arm CoreLink™ Level 2 Cache Controller L2C-310 Technical Reference Manual Revision: r3p3
The latest version can be downloaded from the Arm website.

Technical Update/Technical News

The latest information can be downloaded from the Renesas Electronics website.

User's Manual: Integrated Development

The e^2 studio Integrated Development Environment user's manual can be downloaded from the Renesas Electronics website.
The latest version can be downloaded from the Renesas Electronics website.

Revision History

Rev.	Date	Description	
		Page	Summary
Rev.1.00	Dec.18.18	–	First edition issued
Rev.1.10	Mar.30.20	–	Modified the sample code to be able to output CKIO
		P6	Table 1.1 Peripheral functions and their applications Added OSTM channel 2.
		P7	Table 2.1 Operation confirmation conditions (1/2) Remove compiler option "-mthumb-interwork"
		P19	Table 5.5 Setting for Peripheral Functions Added OSTM channel 2.
		P21	Table 5.6 Sections and Objects to Be Used in the Loader Program Added section for r_memclk_setup function and r_spibsc_setup function because of addition these function
		P38 P41 P48 P49	• Added to functions, function specification and flowchart because of addition the r_memclk_setup functionTable 5.23 Sample Functions • 5.9 Function Specification • Figure 5.4 Flowchart of loader program (overall) • 5.10.2 Memory Clock Setting Processing
		P38 P41 P51	Added to functions, function specification and flowchart because of addition the R_SPIBSC_Setup function • Table 5.24 API Functions • 5.9 Function Specification • 5.10.4 SPIBSC and Serial Flash Memory Initial Setting
		P39 P47	Added to functions and function specification because of addition the Userdef_PreHardwareSetup and the Userdef_PostHardwareSetup function • Table 5.25 User-Defined Functions • 5.9 Function Specification
		P50	5.10.3 Initial setting of hardware used for booting Changed the function specification of R_SC_HardwareSetup function because of addition the R_SPIBSC_Setup function, the Userdef_PreHardwareSetup and the Userdef_PostHardwareSetup function
		P53 P54	Change the fallowing flowchart because of changing the procedure of the timing adjustment processing for SDR mode in the R_SPIBSC_Init function • Figure 5.8 Flowchart of SPIBSC initial setting (1/2) • Figure 5.9 Flowchart of SPIBSC initial setting (2/2)
		P73	Figure 6.3 Hierarchical module diagram of the SPIBSC and serial flash memory settings Added the R_SPIBSC_Setup function in the hierarchical module diagram

Rev.	Date	Description	
		Page	Summary
Rev.1.10	Mar.30.20	P14 P30 P36	Added the note about setting the IO0FV bit in CMNCR register • Table 5.2 Settings for the Boot Startup On-Chip ROM Program and Loader Program (1/3) • Table 5.15 Structure for Configuring the SPIBSC External Address Space Read Mode (st_spibsc_xip_config_t) (5/5) • Table 5.21 SPIBSC Manual Mode Settings Structure (st_spibsc_manual_mode_command_config_t) (6/6)
Rev.1.20	Oct.12.20	P43	Modify procedure of function R_SPIBSC_XipStopAccess that waits for the QSPIn_SSL negation to complete before returning to caller process

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)
   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on
   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state
   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins
   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals
   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin
   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses
   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products
   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.
(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.