SH7268/SH7269 Group

SPI Multi I/O Bus Controller

Serial Flash Memory Connection Sample Program

R01AN0671EJ0101
Rev. 1.01
Feb. 16, 2012

Summary

SH7268/SH7269 SPI multi I/O bus controller (SPIBSC) has the function to directly fetch the program data on a serial flash memory and execute them (external address space read mode) besides random serial flash memory reading function(SPI operation mode) . This application note offers explanations about sample for using SPIBSC and the serial flash memory connection.

Target Device

SH7268/SH7269 MCU (hereinafter called "SH7269" collectively.)

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

Contents

1. Introduction

1.1 Specifications

This application note describes about the application sample program of SPI multi I/O buss controller(SPIBSC) and its references. As the explanation of application program, describes the connection sample to the serial flash memory and two control methods: SPI operation mode, external address read mode.

1.2 Functions Used

- SPI multi I/O bus controller (SPIBSC)
- Renesas Serial Peripheral Interface (RSPI)
- Boot mode (serial flash memory boot)
- General input/output port

1.3 Applicable Conditions

MCU	SH7268/SH7269
Operating Frequency	Internal clock (Iϕ) : 266.67 MHz Internal bus clock (Bϕ) : 133.33 MHz Peripheral clock 1 (P1ϕ) : 66.67 MHz Peripheral clock 0 (P0ϕ) : 33.33 MHz
Integrated Development Environment	Renesas Electronics Corporation High-performance Embedded Workshop Ver.4.07.00
C Compiler	Renesas Electronics SuperH RISC engine Family C/C++ compiler package Ver.9.03 Release 02
Compiler Options	Default setting in the High-performance Embedded Workshop (-cpu=sh2afpu -fpu=single -object="$(CONFIGDIR)\$(FILELEAF).obj" -debug -gbr=auto -chgincpath -errorpath -global_volatile=0 -opt_range=all -infinite_loop=0 -del_vacant_loop=0 -struct_alloc=1 –nologo)
Serial Flash Memory	S25FL032P (Spansion) x 1

1.4 Related Application Note

The application note relating to this application note is introduced below. Refer to it along with this application note.

- SH7268/SH7269 Group Boot from the Serial Flash Memory using the SPI Multi I/O Bus Controller

1.5 About Active-low Pins (Signals)

The symbol "#" suffixed to the pin (or signal) names indicates that the pins (or signals) are active-low.

2. Explanation of Application Program

In this application program one serial flash memory is connected to the SPI multi I/O buss controller(SPIBSC), which fetches any read/write accesses and programs. For read/write access, the SPI operation mode is used, and for program fetching, the external address space read mode is used.

In this section, first explained pin connection of the serial flash memory, and later, operation overview in each mode and their control method.

2.1 Features of SPIBSC

The features of SPIBSC are described below.

- Up to two serial flash memories can be connected
- Data bus width is selectable for one serial flash memory from 1-bit, 2-bit and 4-bit
- Possible to fetch the serial flash memory located in the SPI multi I/O bus space directly in external address space read mode
- Possible for read/write operation for the serial flash memory in the SPI operation mode

2.2 Serial Flash Memory Pin Connection

Table 1 describes about the serial flash memory (Spanson's S25FL032P) supporting the SPI multi I/O bus used in this application program.

Table 1 Specification of Serial Flash Memory Used in this Application Program

Term	Description
Bus input/output	Serial input/output(duplex), dual input/output(half-duplex), quad input/output(half-duplex)
SPI mode	available for SPI mode 0 and 3
Clock frequency	at serial input/output: 104MHz(maximum), dual/quad input/output:80MHz(maximum)
Capacity	4MB
Sector size	64KB
Page size	256B
Erase size	In all areas/64KB/8KB/4KB
Program size	Page Program(1 to 256 bytes)
Protect mode	Write enable command by the commands Software/hardware protect mode by the blocks

Figure 1shows the serial flash memory circuit. In this application program, one serial flash memory is connected to access by 4-bit data bus width. SH7269 pin function should be set as described in Table 2 about multiplexed output pin.

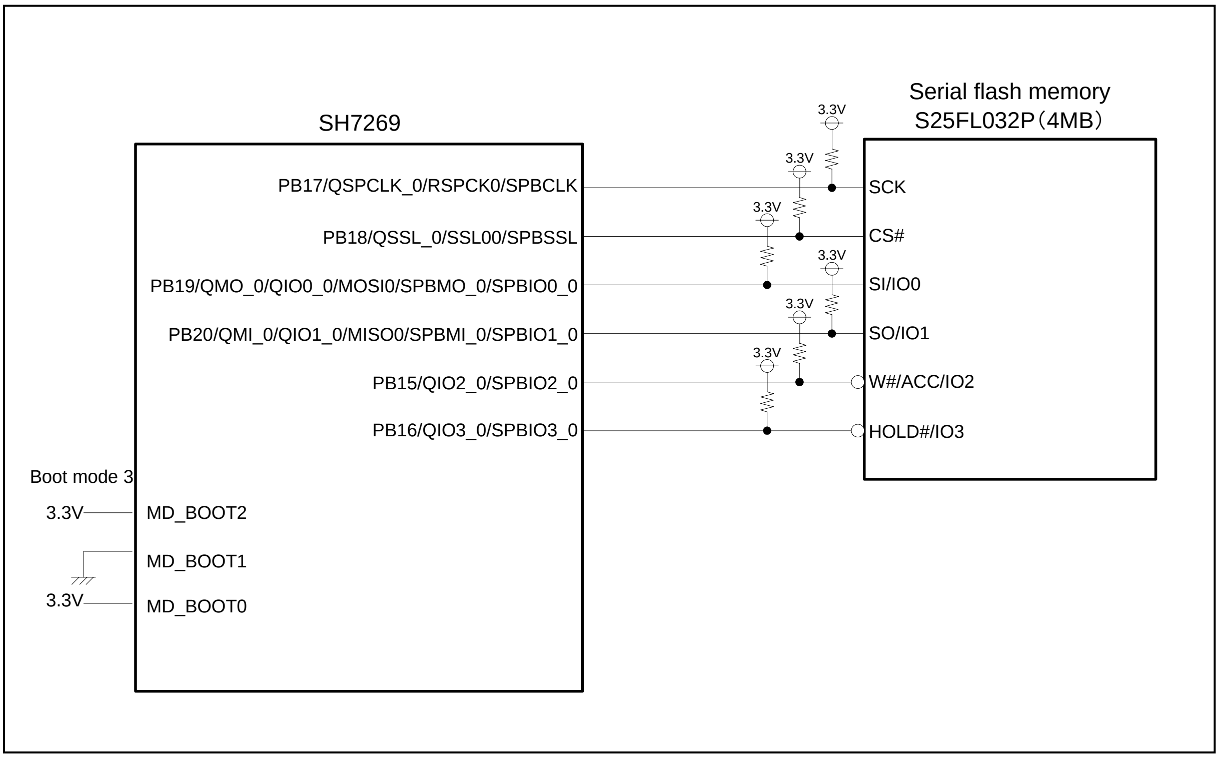


Figure 1 Example of Serial Flash memory Circuit

Note: Treating pull-up/down by the resister attached externally for the control signal pin.

Concerning pull-up/down, the signal line level is determined to avoid producing improper operating signals even when the pin state is high impedance. Pull up the pin by the resister attached externally.

Table 2 Multiplexed Output Pins

Periphera l function	Pin for use	Control register on the SH7269 port		SH7269 multiplexed pin
		Register	Setting value of MD bit	
SPIBSC	SPBCLK	PBCR4	PB17MD[2:0]=B'110	PB17 / A17 / QSPCLK / RSPCK0 / SPBCLK
	SPBSSL	PBCR4	PB18MD[2:0]=B'110	PB18 / A18 / QSSL_0 / SSL00 / SPBSSL
	SPBIO0_0	PBCR4	PB19MD[2:0]=B'110	PB19 / A19 / QMO_0 / QIO0_0 / MOSI0 / SPBMO_0 / SPBIO0_0
	SPBIO1_0	PBCR5	PB20MD[2:0]=B'110	PB20 / A20 / QMI_0 / QIO1_0 / MISO0 / SPBMI_0 / SPBIO1_0
	SPBIO2_0	PBCR3	PB15MD[2:0]=B'110	PB15 / A15 / QIO2_0 / SPBIO2_0
	SPBIO3_0	PBCR4	PB16MD[2:0]=B'110	PB16 / A16 / QIO3_0 / SPBIO3_0

Note: SH7269 multiplex pins

The pins used for SPIBSC are multiplexed. Some are set for general input/output port by default. They should be set to SPIBSC function by the general input/output port control register before accessing to the serial flash memory. Pay attention that when using in boot mode 0 and boot mode 1(boot from the memory connected to the CSO space), they cannot be set to SPIBSC function. Use boot mode 3(serial flash boot).

2.3 Interface Timing

Figure 2, Table 3 and Table 4 show the interface timing and requirement in this application program. SPIBSC setting value should comply to the requirements in the timings described herein.

Table 5 lists the SPIBSC interface setting values in this application program. The serial flash memory used in this application program is set to operate in the SPI mode 0(clock negate level is 'L', receive at rising, transmit at decaying). Therefore the SPIBSC setting is according to the SPI mode 0. Only the timing of reception is set to be delayed by 1/2 cycle to comply with the SH7269 data set up time.

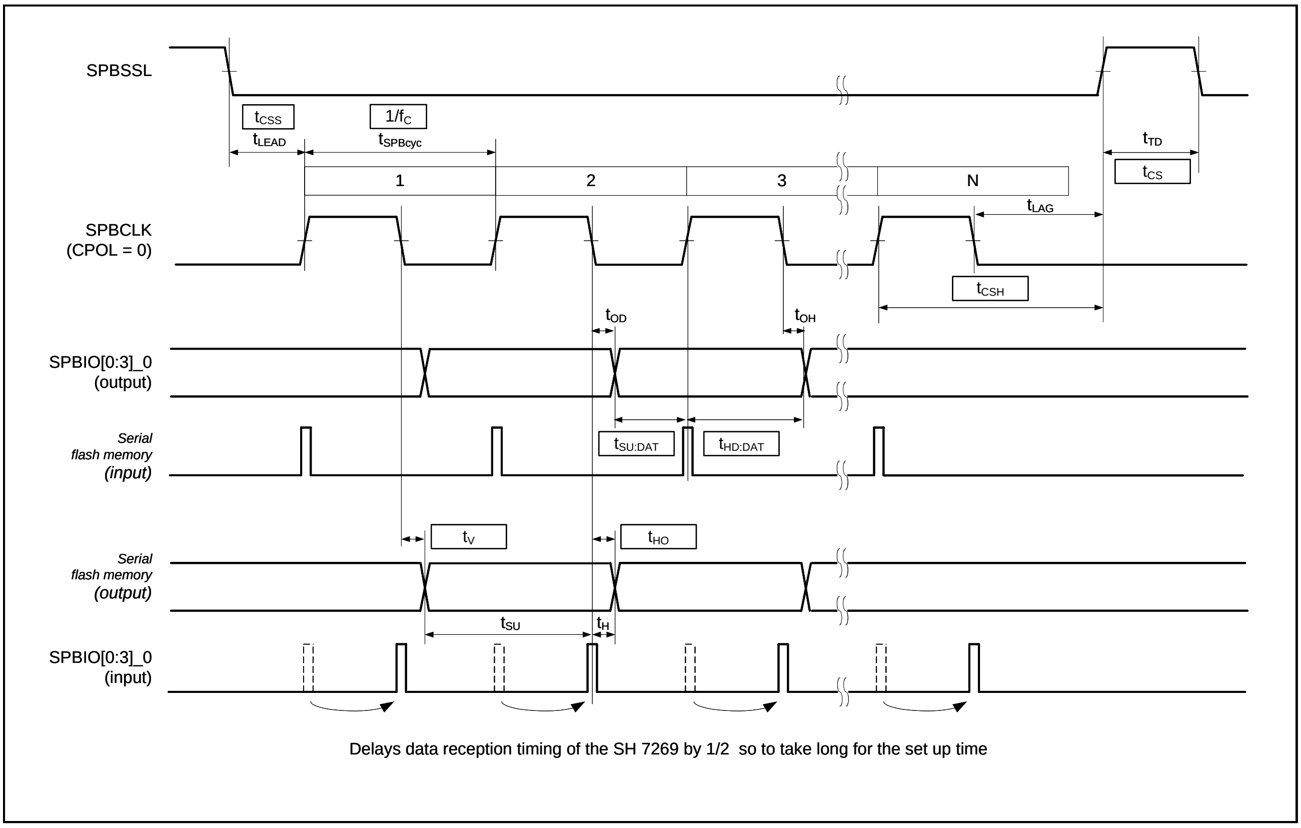


Figure 2 Interface Timing in this Application Program

Table 3 Requirements for Serial Flash Memory Timing

Symbol	Item	Description
t_{CSS}	Chip select 'L' Setup time	Necessary time from SSL asserting to data reception by the serial flash memory. The following requirement should be met. t_{LEAD}(=clock delay) ≧ t_{CSS} (min)
t_{CS}	Chip select 'H'	Necessary as SSL negate period. The following requirement should be met. t_{TD}(=next access delay) ≧ t_{CS} (min)
f_C	Serial clock frequency	The maximum frequency that the serial flash memory can support. The following requirement should be met. f_C(max) ≧ 1 / t_{SPBcyc}
t_{CSH}	Chip select 'L' hold time	Necessary time from SPBCLK rise to SSL negate. The following requirement should be met. t_{LAG}(=SPBSSL negate delay) + t_{SPBcyc}×1/2) ≧ t_{CSH} (min)
$t_{SU:DAT}$	Data input set up time	Necessary set up time for data input. The following requirement should be met. (t_{SPBcyc}×1/2) - t_{OD}(max) ≧ $t_{SU:DAT}$ (min)
$t_{HD:DAT}$	Data input hold time	Necessary hold time for data input. The following requirement should be met. t_{OH}(min) + (t_{SPBcyc}×1/2) ≧ $t_{HD:DAT}$ (min)

Note: t_{SPBcyc} is fixed to 2 $t_{cyc.}$ t_{cyc} represents 1 cycle period of bus clock (B ϕ).

Table 4 Requirements for SPIBSC Timing

Symbol	Item	Description
t_{SU}	Data input set up time	Necessary set up time for data input. The following requirement should be met. t_{SPBcyc} - t_V(maximum) ≧ t_{SU}(minimum)
t_H	Data input hold time	Necessary hold time for data input. The following requirement should be met. t_{HO}(min) ≧ t_H(min)

Note: t_{SPBcyc} is fixed to 2 $t_{cyc.}$ t_{cyc} represents 1 cycle period of bus clock (B ϕ).

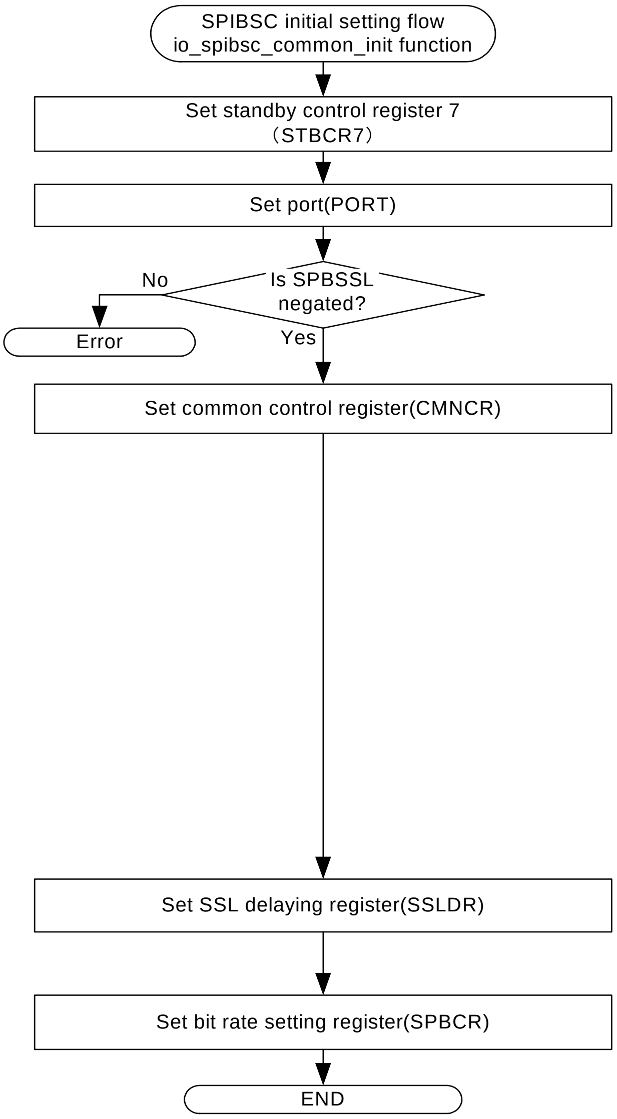
Table 5 Setting Value of Interface Timing in This Application Program

Register	Bit	Set value	Function
Bit rate setting register（SPBCR）	-	H'0000 0100	SPBCLK bit rate is set to half of B ϕ（66.67Mbps）
Common Control register（CMNCR）	CPOL bit	B'0	Set SPBCLK negate level to 'L'
	CPAHT bit	B'0	Data transmission in even edge
	CPAHR bit	B'1	Data reception in even edge
SSL delay register（SSLDR）	SPND[2:0]	B'000	Set the next access delay setting to 1SPBCLK
	SLNDL[2:0]	B'000	Set SPBSSL negate delay setting to 1.5SPBCLK
	SCKDL[2:0]	B'000	Set clock delay setting to 1SPBCLK

2.4 Initial Setting Flow

Figure 3 shows the flow chart of initial setting of SPIBSC in this application program.

.


SPIBSC initial setting flow
io_spibsc_common_init function
Set standby control register 7
（STBCR7）
Set port(PORT)
Is SPBSSL
negated?
No
Error
Yes
Set common control register(CMNCR)
Set SSL delaying register(SSLDR)
Set bit rate setting register(SPBCR)
END

- Release clock supply stop to SPIBSC

- Select multiplexed pins
  Function: SPBIO1_0, SPBIO0_0, SPBSSL, SPBCLK
  SPBIO3_0, SPBIO2_0

- Check SSLF bit in the common status register(CMNSR) to confirm the SPBSSL is negated.

- Set CMNCR
  (MOIIO3 to MOIIO0 bit = B'11)
  Function: The output value from SPBIO3 to SPBIO0 is Hi-Z during SPBSSL negated.
  (IO3FV to IO2FV bit = B'11)
  Function: The output value from SPBIO3 to SPBIO2 is Hi-Z when 1 bit/2bits width.
  （IO0FV bit = B'11)
  Function: The output value from SPBIO0 is Hi-Z when 1 bit width.
  (CPHAT bit = B'0)
  Function: Transmit data at even number edge
  (CPHAR bit = B'0)
  Function: Receive data at even number edge
  (SSLP bit = B'0)
  Function: SPBSSL signal is Low active
  (CPOL bit = B'0)
  Function: 0 output from SPBCLK at SPBSSL negated
  (BSZ bit = B'00)
  Function: One serial flash memory to connect

- Set SSLDR(SSLDR = H'0000 0000)
  Function: Set the next access dlay to 1 SPBCLK(fixed)
  Set SPBSSL negate delay to 1.5 SPBCLK(fixed)
  Set clock delay to 1 SPBCLK

- Set SPBCR(SPBCR = H'0000 0100)
  Function: Set bit rate to double division of Bϕ
  (66.67Mbps)(fixed)

Figure 3 Flow Chart of SPIBSC Initial Setting in this Application Program

2.5 SPI Operation Mode

2.5.1 Operation Overview

The SPI operation mode enables the random read/write operation from/to the serial flash memory. This mode is necessary to write in the serial flash memory using SPIBSC.

For using the SPI operation mode, the settings shown in Table 6, Table 7 and Table 8 are necessary besides the setting shown in Figure 3 Flow Chart of SPIBSC Initial Setting in this Application”.

2.5.2 Data Format and Related Registers

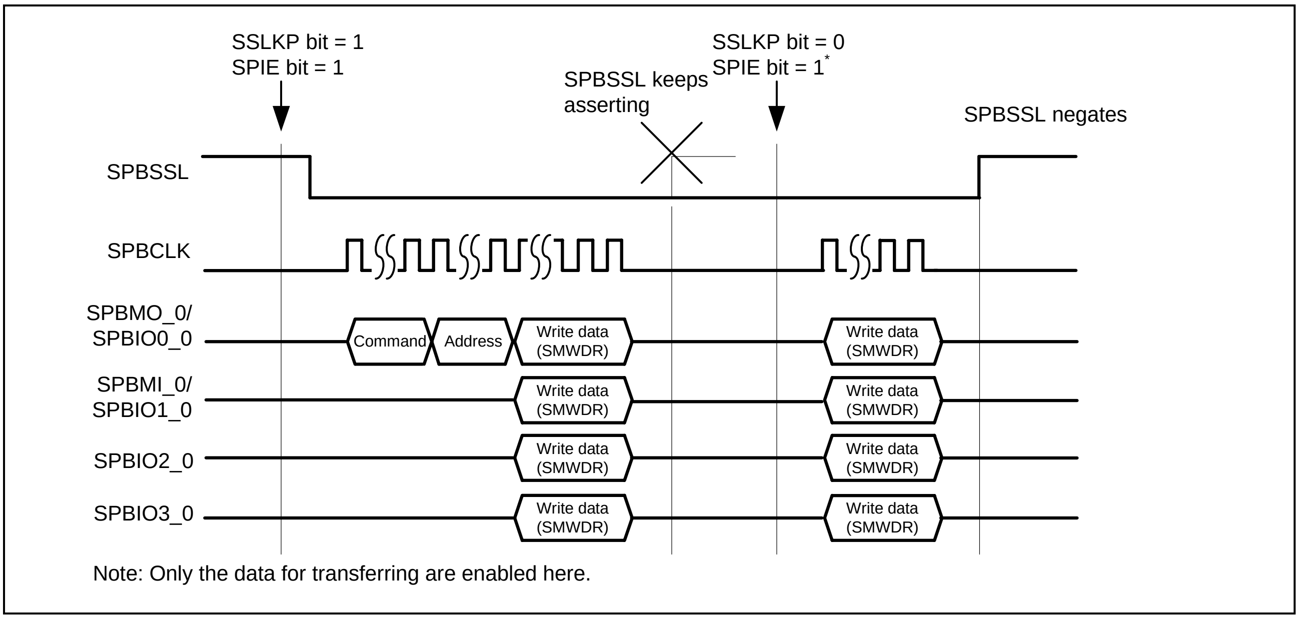
The commands are used for read/write operation from/to the serial flash memory. The command data format is set in the SPIBSC register matching to the commands. Table 6 describes about data format in the SPI operation mode and the related registers.

Table 6 Data Formant in the SPI Operation Mode and Related Registers

Item	Command	Optional command	Address	Optional data	Transfer data
Data	SMCMR. CMD[7:0] bit	SMCMR. OCMD[7:0] bit	32 bit: SMADR.ADR[31:0] bit 24 bit: SMADR.ADR[23:0] bit	SMOPR. OPDn [7:0] bit （n = 0 to 3）	For reading: 32 bit: SMRDR0.RDATA0[31:0] bit 16 bit: SMRDR0.RDATA0[31:16] bit 8 bit: SMRDR0.RDATA0[31:24] bit For writing: 32 bit: SMWDR0.WDATA0[31:0] bit 16 bit: SMWDR0.WDATA0[31:16] bit 8 bit: SMWDR0.WDATA0[31:24] bit
Bit width setting （Single/Dual/Quad）	SMENR. CDB[1:0] bit	SMENR. OCDB[1:0] bit	SMENR.ADB[1:0] bit	SMENR. OPDB[1:0] bit	SMENR.SPIDB[1:0 bit
Enabling data input/output	Always outputting				SMCR.SPIRE bit SMCR.SPIWE bit
Enabling transfer	SMENR. CDE bit	SMENR. OCDE bit	SMENR.ADE[3:0] bit (set bit length as well)	SMENR. OPDE[3:0] bit	SMENR.SPIDE[3:0] bit (set bit length as well)

2.5.3 SPBSSL Pin Assert Retention

Figure 4 shows the SPBSSL assert retention function in the SPI operation mode. In this mode, when setting SSLKP bit in the SPI mode control register(SMCR) to 1, SPBSSL signal is retained from the transfer ending to the next access starting. This function enables continuous transfer except when the bit width of transfer data is set to more than 2 in data read processing.



Note: Only the data for transferring are enabled here.

Figure 4 SPBSSL Assert Retention in the SPI Mode

2.5.4 Data Read Procedure

(1) Read Command

Table 7 describes about S25FL032P read command used in the SPI operation mode. Figure 5 shows its sequence. Only the commands used in the sample program are applied here.

Table 7 S25FL032P Read Command Used in the SPI Operation Mode

Command name	Command code	Number of address byte	Number of dummy byte	Number of data byte	Function
Quad Output Read	H'6B	3	1	One or bigger*	Read data(Quad-SPI)

Note: Read the area incremented from the specified address. Exceeding the final address returns to address 0.

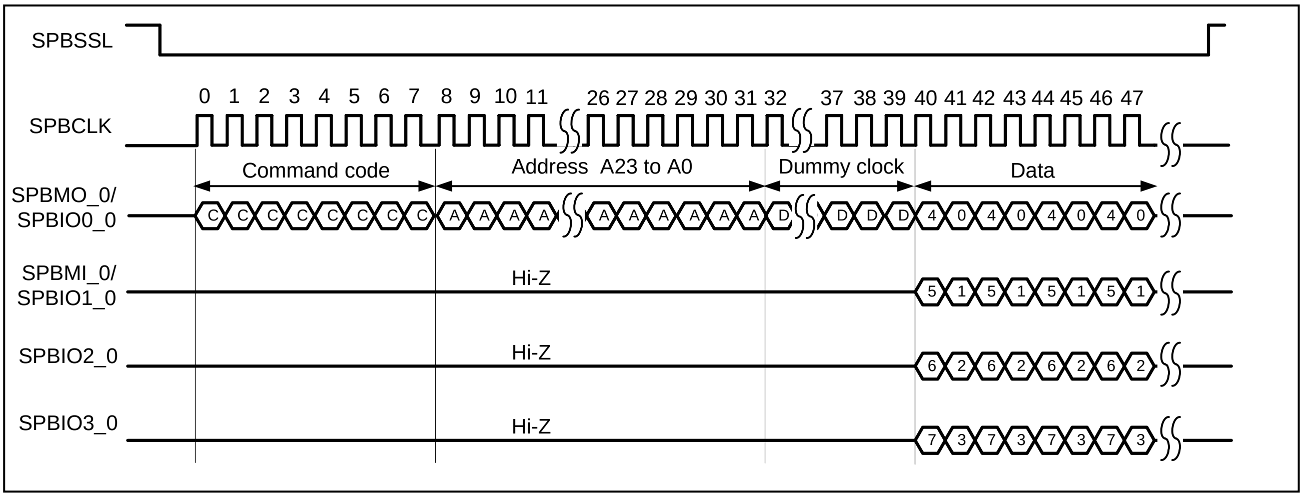
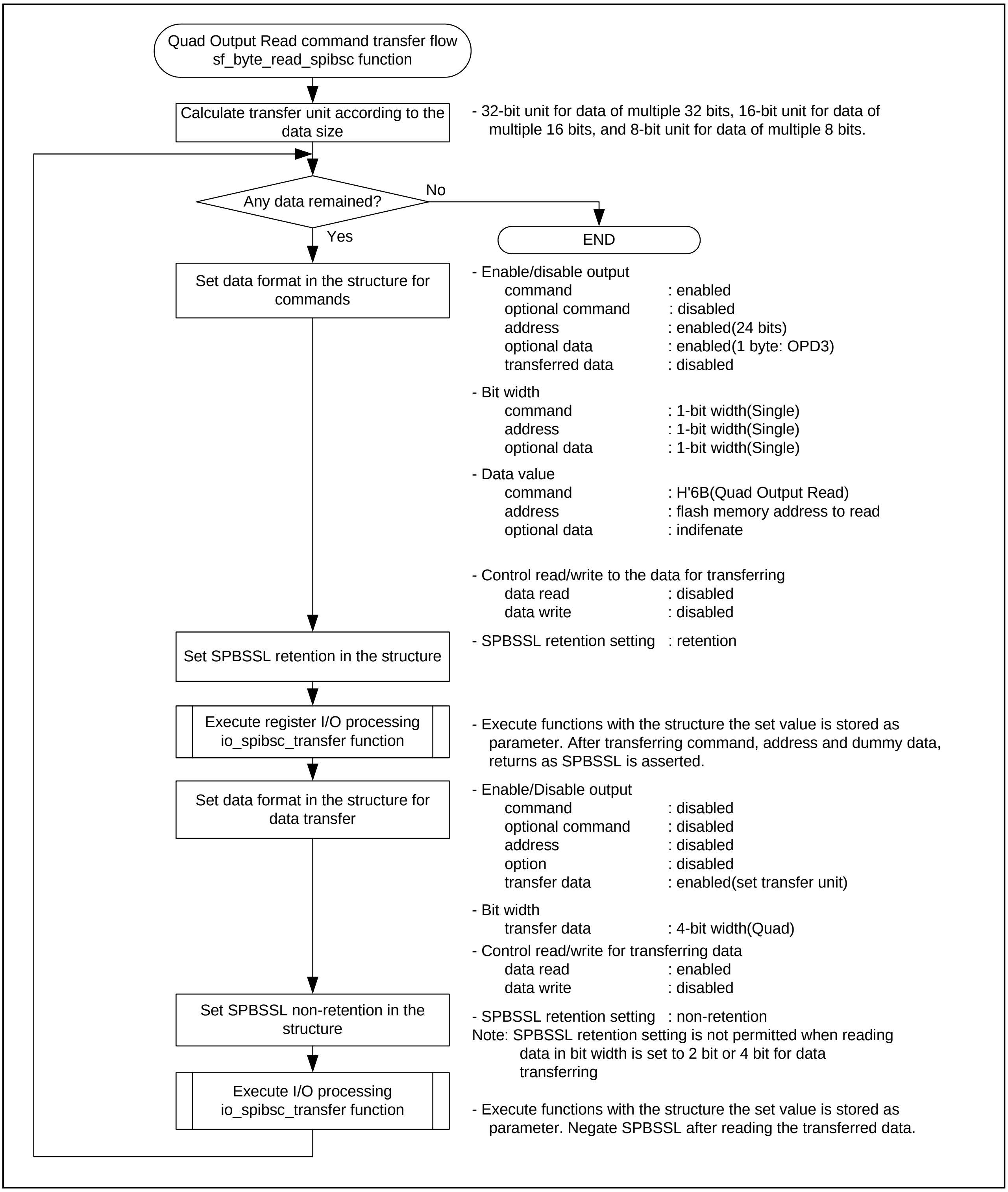


Figure 5 Quad Output Read Command Sequence

Note: Quad Output Read command reads 4 bytes for one issue as the transfer data bit width is 4 bits.

(2) SPI Operation Mode Setting Flow(Read)

Figure 6 and Figure 7 show the flow chart of read command transfer in the SPI mode in this application program.


Quad Output Read command transfer flow
sf_byte_read_spibsc function
Calculate transfer unit according to the data size
Any data remained?
No
Yes
END
Set data format in the structure for commands
Set SPBSSL retention in the structure
Execute register I/O processing
io_spibsc_transfer function
Set data format in the structure for data transfer
Set SPBSSL non-retention in the structure
Execute I/O processing
io_spibsc_transfer function

- 32-bit unit for data of multiple 32 bits, 16-bit unit for data of multiple 16 bits, and 8-bit unit for data of multiple 8 bits.

```
- Enable/disable output
      command                  : enabled
      optional command         : disabled
      address                  : enabled(24 bits)
      optional data            : enabled(1 byte: OPD3)
      transferred data         : disabled
- Bit width
      command                  : 1-bit width(Single)
      address                  : 1-bit width(Single)
      optional data            : 1-bit width(Single)
- Data value
      command                  : H'6B(Quad Output Read)
      address                  : flash memory address to read
      optional data            : indifenate

- Control read/write to the data for transferring
      data read                : disabled
      data write               : disabled
- SPBSSL retention setting     : retention
```

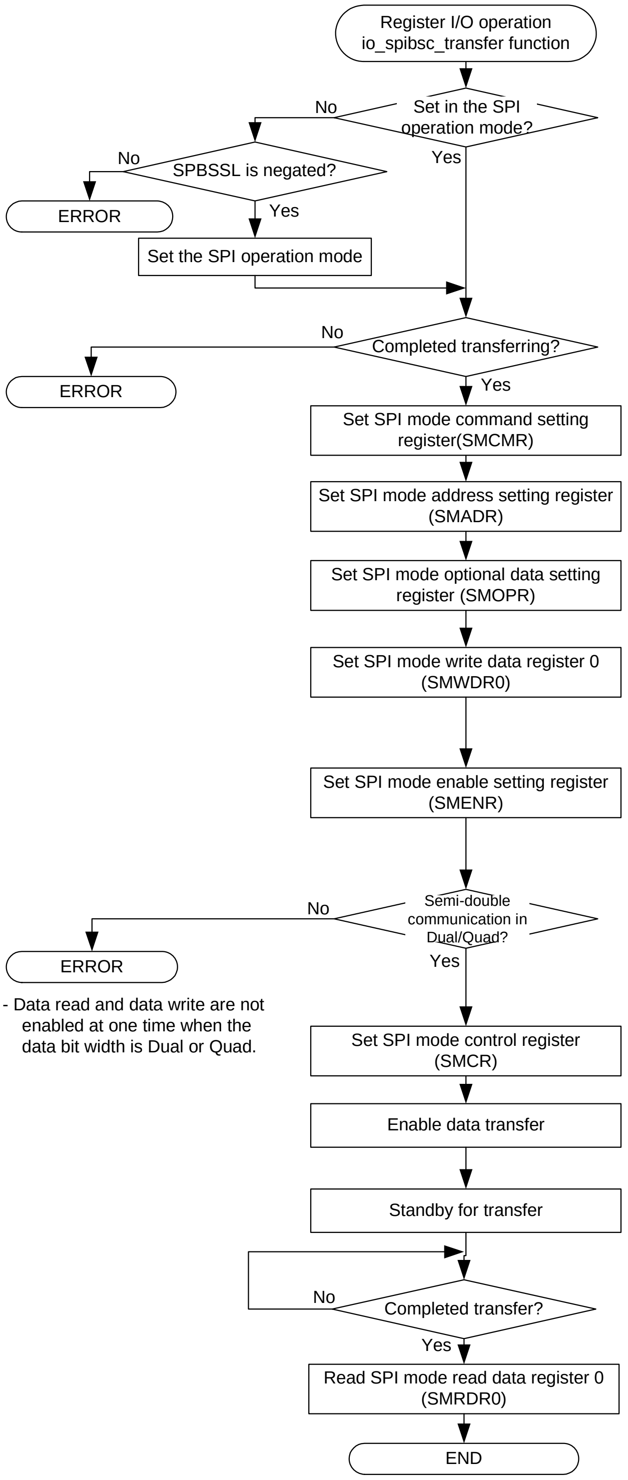
- Execute functions with the structure the set value is stored as parameter. After transferring command, address and dummy data, returns as SPBSSL is asserted.

```
- Enable/Disable output
      command                  : disabled
      optional command         : disabled
      address                  : disabled
      option                   : disabled
      transfer data            : enabled(set transfer unit)
- Bit width
      transfer data            : 4-bit width(Quad)
- Control read/write for transferring data
      data read                : enabled
      data write               : disabled
- SPBSSL retention setting     : non-retention
```

Note: SPBSSL retention setting is not permitted when reading data in bit width is set to 2 bit or 4 bit for data transferring

- Execute functions with the structure the set value is stored as parameter. Negate SPBSSL after reading the transferred data.

Figure 6 Read Command Low Chart in the SPI Operation Mode


Register I/O operation
io_spibsc_transfer function
Set in the SPI operation mode?
No
Yes
SPBSSL is negated?
No
Yes
ERROR
Set the SPI operation mode
Completed transferring?
No
Yes
ERROR
Set SPI mode command setting register(SMCMR)
Set SPI mode address setting register (SMADR)
Set SPI mode optional data setting register (SMOPR)
Set SPI mode write data register 0 (SMWDR0)
Set SPI mode enable setting register (SMENR)
Semi-double communication in Dual/Quad?
No
Yes
ERROR
Set SPI mode control register (SMCR)
Enable data transfer
Standby for transfer
Completed transfer?
No
Yes
Read SPI mode read data register 0 (SMRDR0)
END

- Data read and data write are not enabled at one time when the data bit width is Dual or Quad.

- Change to SPI operation mode with confirmation that the value of SSLF bit in common status register(CMNSR) is negated when the value of MD bit of common control register(CMNCR) is not SPI operation mode.

- Confirm that the transfer is completed. When the TEND bit of CMNSR register is B'0, the transfer is in progress.

- Set SMCMR register
  Function: CMD[7:0] bit : set the command value
  OCMD[7:0] bit : set the optional command value

- Set SMADR register
  Function: ADR[31:0] bit : set the address value

- Set SMOPR egister
  Function: OPD3[7:0] bit : Set the optional data value as the initial output
  OPD2[7:0] bit : Set the optional data value as the second output
  OPD1[7:0] bit : Set the optional data value as the third output
  OPD0[7:0] bit : Set the optional data value as the fourth output

- Set SMWDR0 register
  Function: WDATA0[31:0] bit : set the transfer value for writing
  8-bit transfer : WDATA0[31:24] is valid
  16-bit transfer : WDATA0[31:16] is valid
  32-bit transfer : WDATA0[31:0] is valid

- Set SMENR register
  Function: CDB[1:0] bit : set the bit width of command
  OCDB[1:0] bit : set the bit width of optional command
  ADB[1:0] bit : set the bit width of address
  OPDB[1:0] bit : set the bit width of optional data
  SPIDB[1:0] bit : set the bit width of transfer data
  CDE bit : enable/disable command output
  OCDE bit : enable/disable optional command output
  ADE[3:0] bit : enable address output, set data length
  OPDE[3:0] bit : enable output of optional data, set data length
  SPIDE[3:0] bit : enable transfer data output, set data length

- Set SMCR register
  Function: SSLKP bit : set retention/non-retention of SPBSSL signal level
  SPIRE bit : enable/disable reading data
  SPIWE bit : enable/disable writing data

- Set SPIE bit in SMCR register to B'1 and start transferring data.

- Repeat dummy reading in CMNSR register to stand by transfer starting(TEND bit = B'0)

- Sand by till the TEND bit in CMNSR register becomes B'1

- Read SMRDR0 register
  Function: RDATA0[31:0] bit : read the transfer value to have read
  8-bit transfer : RDATA0[31:24] is valid
  16-bit transfer : RDATA0[31:16] is valid
  32-bit transfer : RDATA0[31:0] is valid

Figure 7 Register I/O Flow Chart in SPI Operation Mode

2.5.5 Data Write Procedure

(1) Write Command

Table 8 describes about S25FL032P write command. Figure 8 shows its command sequence. Only the commands used in the sample program are applied here.

Table 8 S25FL032P Write Command Used in the SPI Operation Mode

Command name	Command code	Number of address byte	Number of dummy byte	Number of data byte	Function
Quad Page Programming	H'32	3	0	1 or bigger*	Write data(Quad-SPI)

Note: Write in the area incremented on the same page as the specified address. Exceeding the final address returns to address 0.

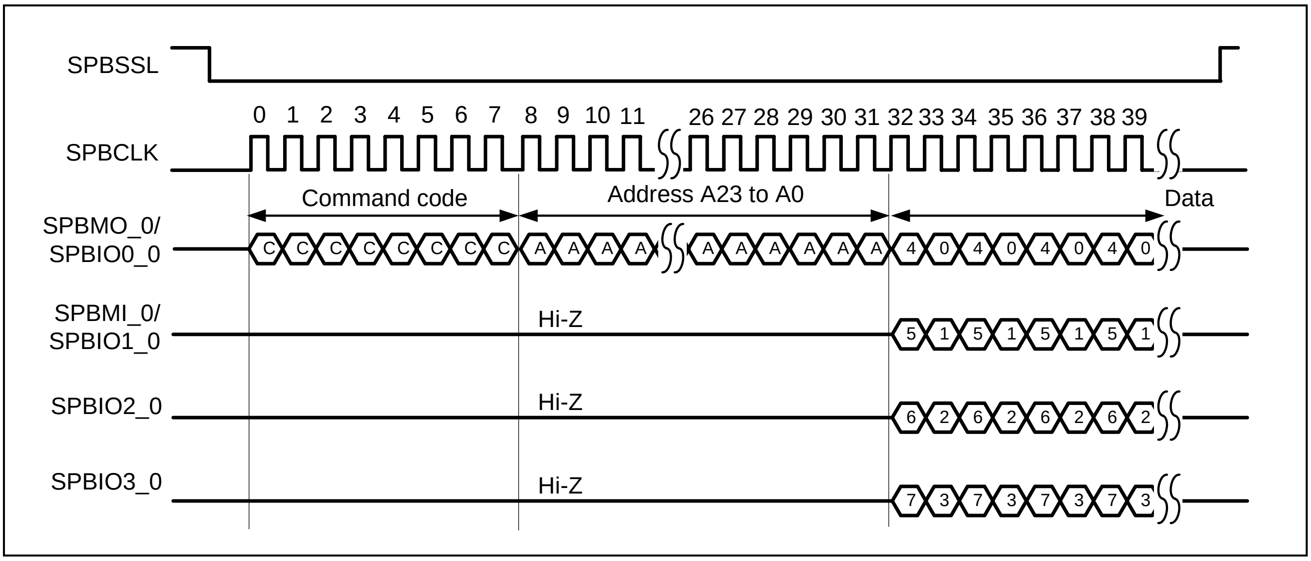


Figure 8 Quad Page Programming Command Sequence

(2) SPI Operation Mode Setting Flow(Write)

Figure 9 shows the flow chart of write command transfer in the SPI mode in this application program. For register I/O operation flow, refer to Figure 7.

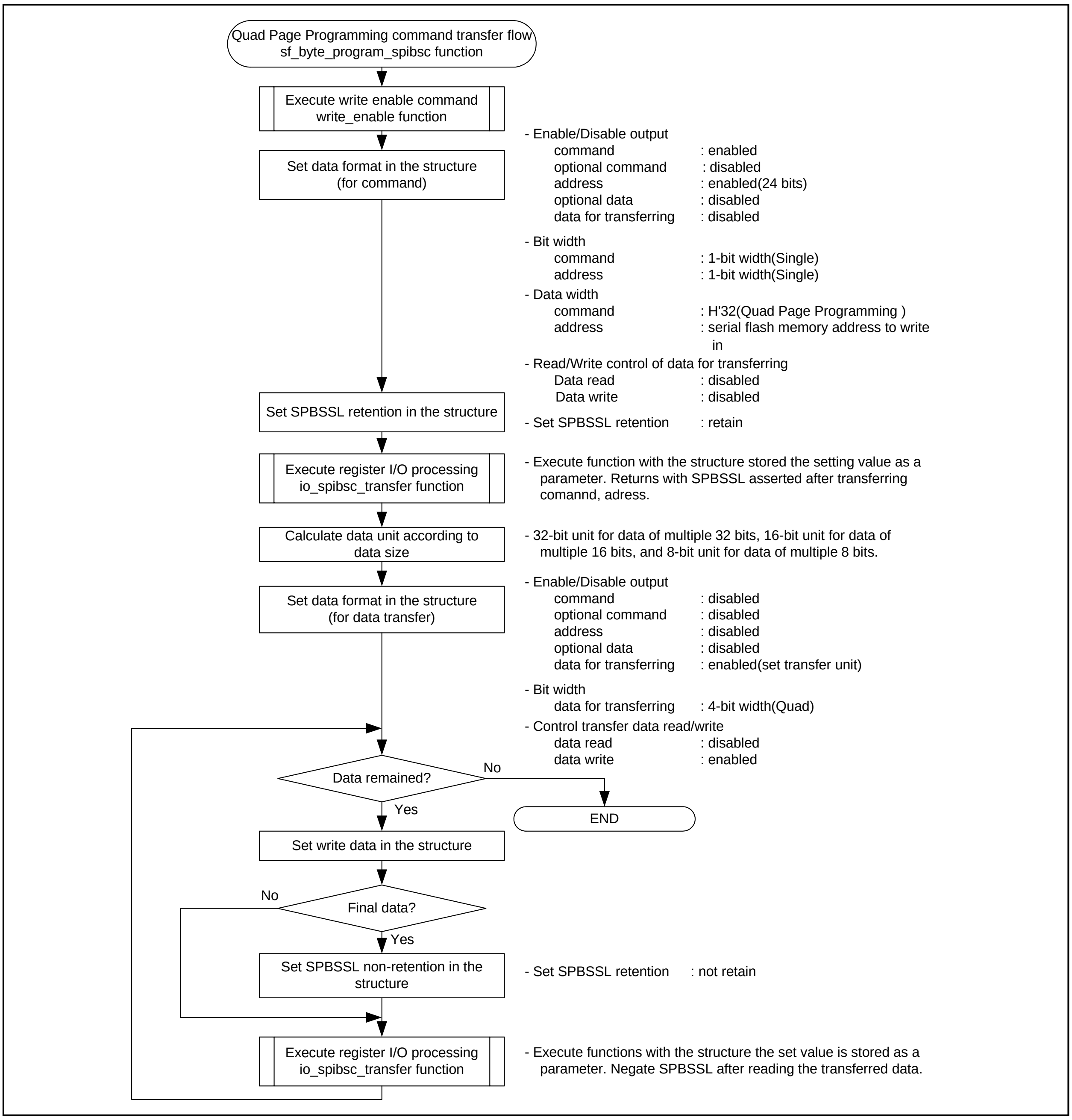

Quad Page Programming command transfer flow
sf_byte_program_spibsc function
Execute write enable command
write_enable function
Set data format in the structure
(for command)
- Enable/Disable output
command : enabled
optional command : disabled
address : enabled(24 bits)
optional data : disabled
data for transferring : disabled
- Bit width
command : 1-bit width(Single)
address : 1-bit width(Single)
- Data width
command : H'32(Quad Page Programming)
address : serial flash memory address to write in
- Read/Write control of data for transferring
Data read : disabled
Data write : disabled
Set SPBSSL retention in the structure
- Set SPBSSL retention : retain
Execute register I/O processing
io_spibsc_transfer function
- Execute function with the structure stored the setting value as a parameter. Returns with SPBSSL asserted after transferring comannd, adress.
Calculate data unit according to data size
- 32-bit unit for data of multiple 32 bits, 16-bit unit for data of multiple 16 bits, and 8-bit unit for data of multiple 8 bits.
Set data format in the structure
(for data transfer)
- Enable/Disable output
command : disabled
optional command : disabled
address : disabled
optional data : disabled
data for transferring : enabled(set transfer unit)
- Bit width
data for transferring : 4-bit width(Quad)
- Control transfer data read/write
data read : disabled
data write : enabled
Data remained?
No
Yes
END
Set write data in the structure
Final data?
No
Yes
Set SPBSSL non-retention in the structure
- Set SPBSSL retention : not retain
Execute register I/O processing
io_spibsc_transfer function
- Execute functions with the structure the set value is stored as a parameter. Negate SPBSSL after reading the transferred data.

Figure 9 Flow Chart of Write Command Transfer in SPI Operation Mode

2.6 External Address Space Read Mode

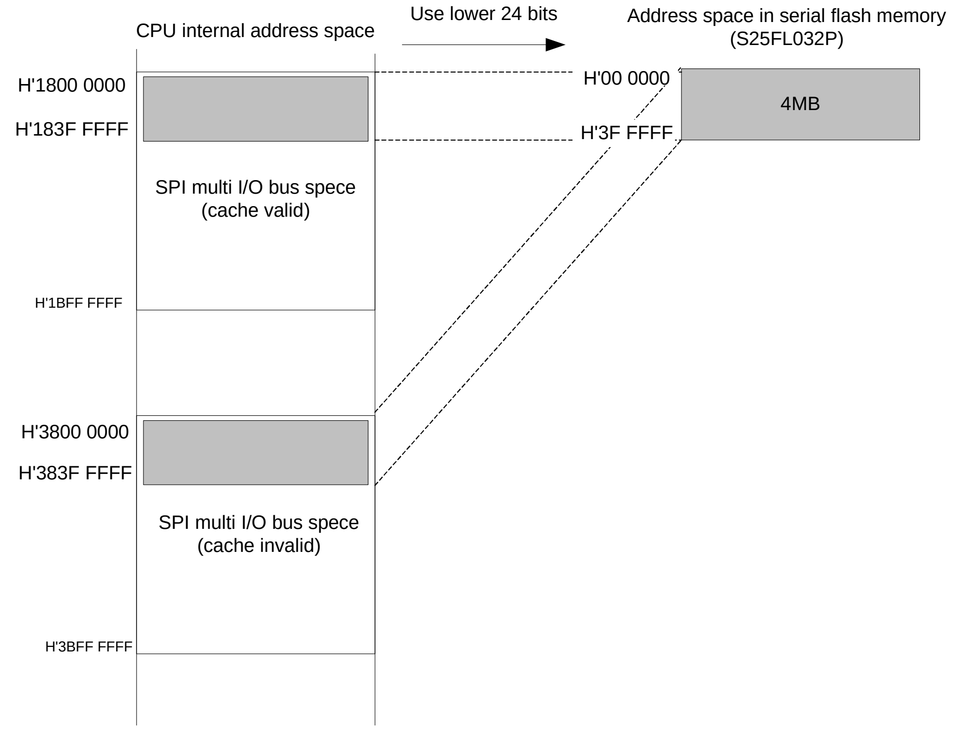
2.6.1 Operation Overview

The external address space read mode converts the read access to the SPI multi I/O bus space to SPI communication automatically. Using this function enables fetching the programs on the serial flash memory directly which is the same as the NOR flash memory. Therefore deploying the programs on the RAM is not necessary, which results in saving RAM capacity.

To use the external address space read mode, setting shown in Figure 14 is necessary besides Figure 3 “Flow Chart of SPIBSC Initial Setting in this Application”.

2.6.2 Automatic Address Conversion

Figure 10 shows the address conversion image in the external address space read mode with the serial flash memory as 24-bit address. SPIBSC uses the lower 24 bits for accessing to the serial flash memory when detecting read access to H’1800 0000 to address H’1BFF FFFF which are in the SPI multi I/O bus space, and so does in the cache invalid space as address H’3800 0000 to address H’3BFF FFFF.



Note: The upper address should be set by the data read expansion address setting register(DREAR) when outputting address in 32 bits.

Figure 10 Address Conversion Image in External Address Space Read Mode

2.6.3 Data Format and Related Registers

For accessing to the serial flash memory, access commands are necessary. The command data formats are set in SPIBSC register. Table 9 describes about the data format and related registers in external address space read mode. Pay attention that the registers used are different from those used in the SPI operation mode.

Table 9 Data Format in External Address Space Read Mode and Related Registers

Item	Command	Optional command	Address	Optional data	Data for transferring
Data	DRCMR.CMD[7:0] bit	DRCMR. OCMD[7:0] bit	With 24 bits The lower address which was read [23:0] bit	DROPR.OPDn[7:0]bit （n = 0 to 3）	When normal reading Transfer number of bits according to access size （8/16/32/64 bits） When burst reading DRCR.RBURST[3:0] bit （RBURST×64 bit）
Set bit width （Single/Dual/Quad）	DRENR.CDB[1:0] bit	DRENR.OCDB [1:0] bit	DRENR.ADB[1:0] bit	DRENR.OPDB[1:0] bit	DRENR.DRDB[1:0] bit
Enable data input	Output always				Input always
Enable transfer	DRENR.CDE bit	DRENR.OCDE bit	DRENR.ADE[3:0] bit （set bit length as well）	DRENR.OPDE[3:0] bit	Enable always

2.6.4 Read Command

Table 10 describes about S25FL032P read command used in the external address space read mode. Figure 11 shows its command sequence. Only the commands used in the sample program are applied here.

Table 10 S25FL032O Read Command used in the External Address Space Read Mode

Command name	Command code	Number of address byte	Number of dummy byte	Number of data byte	Function	Command name
Quad I/O High Performance Read	H'EB	3	1[1]	2	More than one[2]	High speed data read (Quad-SP)

Note: (1) Setting H'A0 to H'AF leads to set continuous mode, and for asserting SPBSSL next time, the command code is not necessary. This is not used in this application program.

(2) Read the area incremented from the specified address. Exceeding final address leads returning to the address 0.

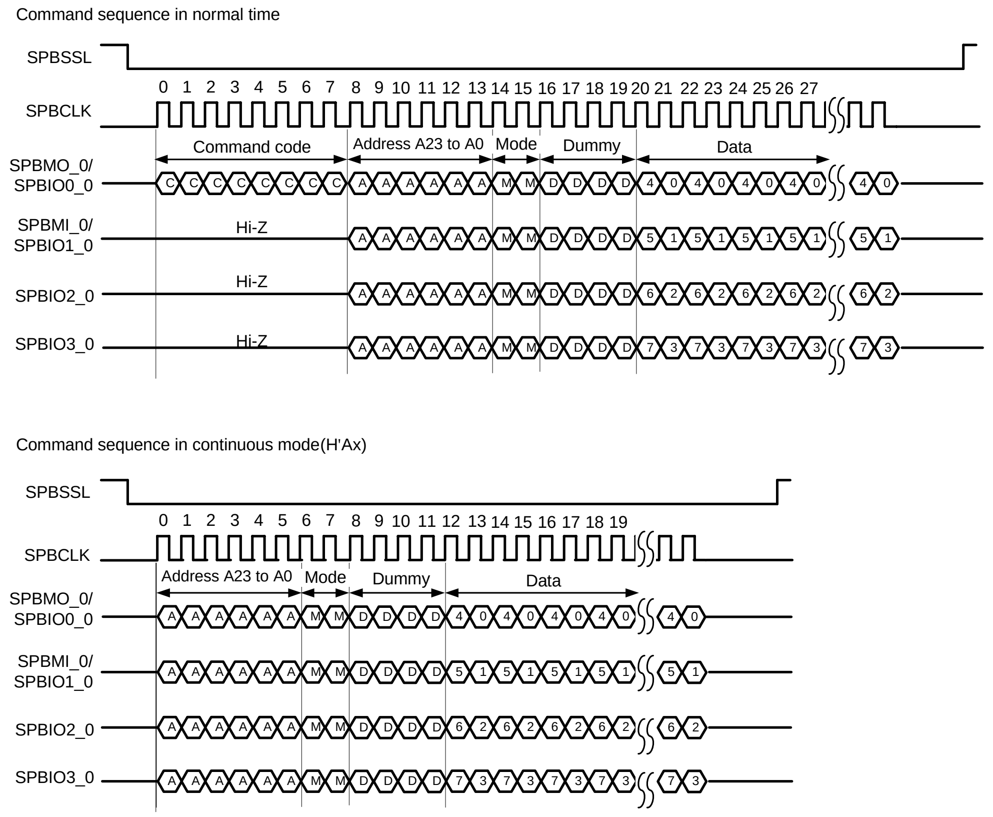


Figure 11 Quad I/O high Performance Read Command Sequence

2.6.5 Burst Read Operation

Burst read operation is enacted when RBE bit in the data read control register(DRCR) is set to 1. At the same time, read cache is effective. The operation overview of burst read and read cache is described as follows.

(1) Burst Read and Read Cache

Figure 12 shows the operation of burst read and read cache.

Detecting read access to the SPI multi I/O bus space, SPIBSC first refers to data in read cache. When the read cache contains data, SPIBSC does not access to the serial flash memory but read data from the read cache. When the reach cache is without data, SPIBSC reads burst reads the serial flash memory to store the data in the read cache. The data transfer length at this time is 64 bits x BRUST[3:0]. The read starts from 64-bit boundary.

To flash the read cache, RCF bit in DRCR register should be set.

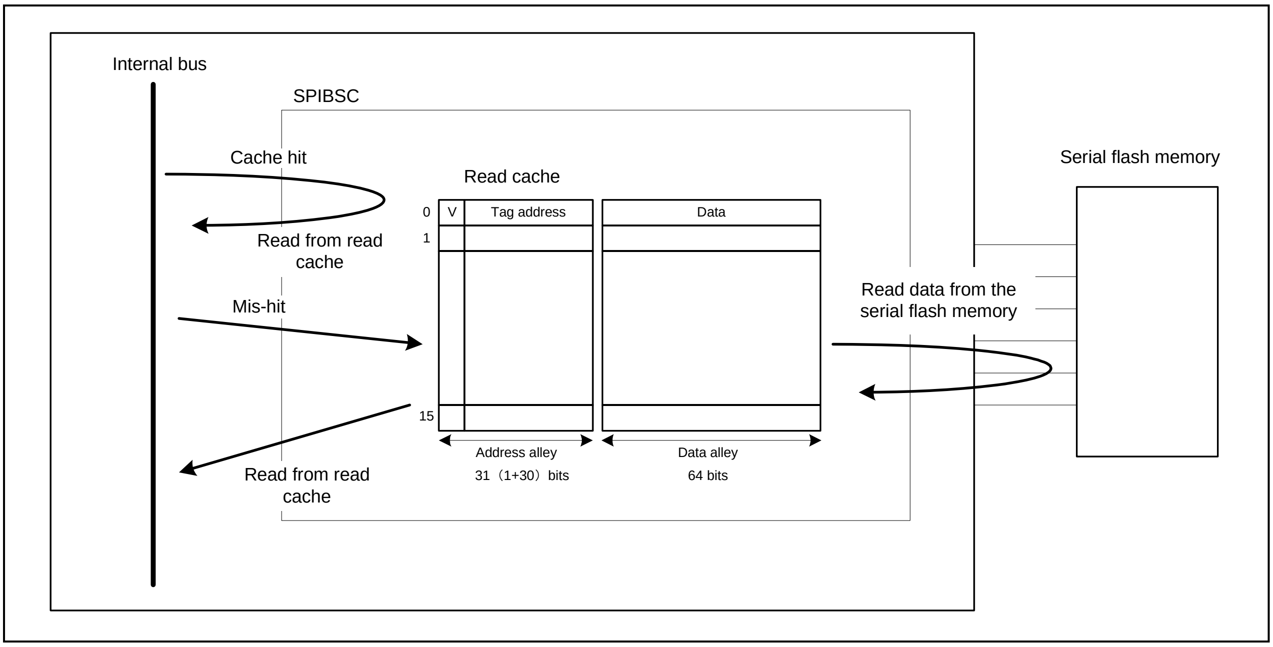


Figure 12 Operation of Burst Read and Read Cache

(2) SPBSSL Automatic Negation

Figure 13 shows the SPBSSL automatic negation in burst read operation. Setting SSLE bit in DRCR register to 1 does not lead negation SPBSSL pin after burst read transfer. At the next accessing, when the address is sequential to the previous read address, burst read is carried without issuing command/optional command/address/optional data. When the addresses are not sequential, SPBSSL pin should be negated once and later burst read is carried after issuing command/optional command/address/optional data.

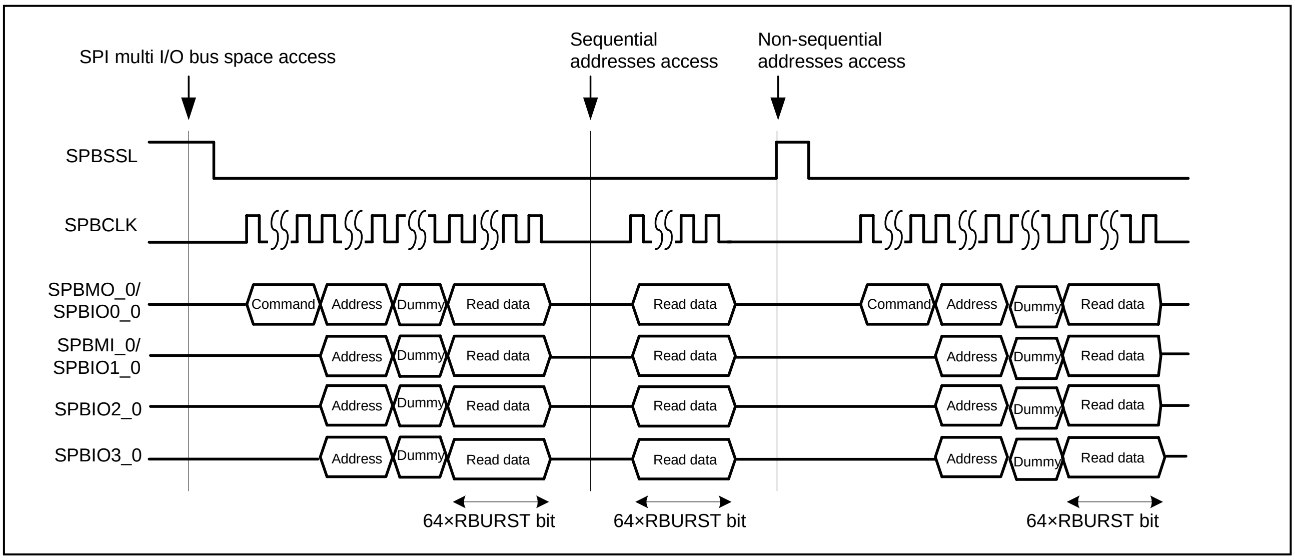
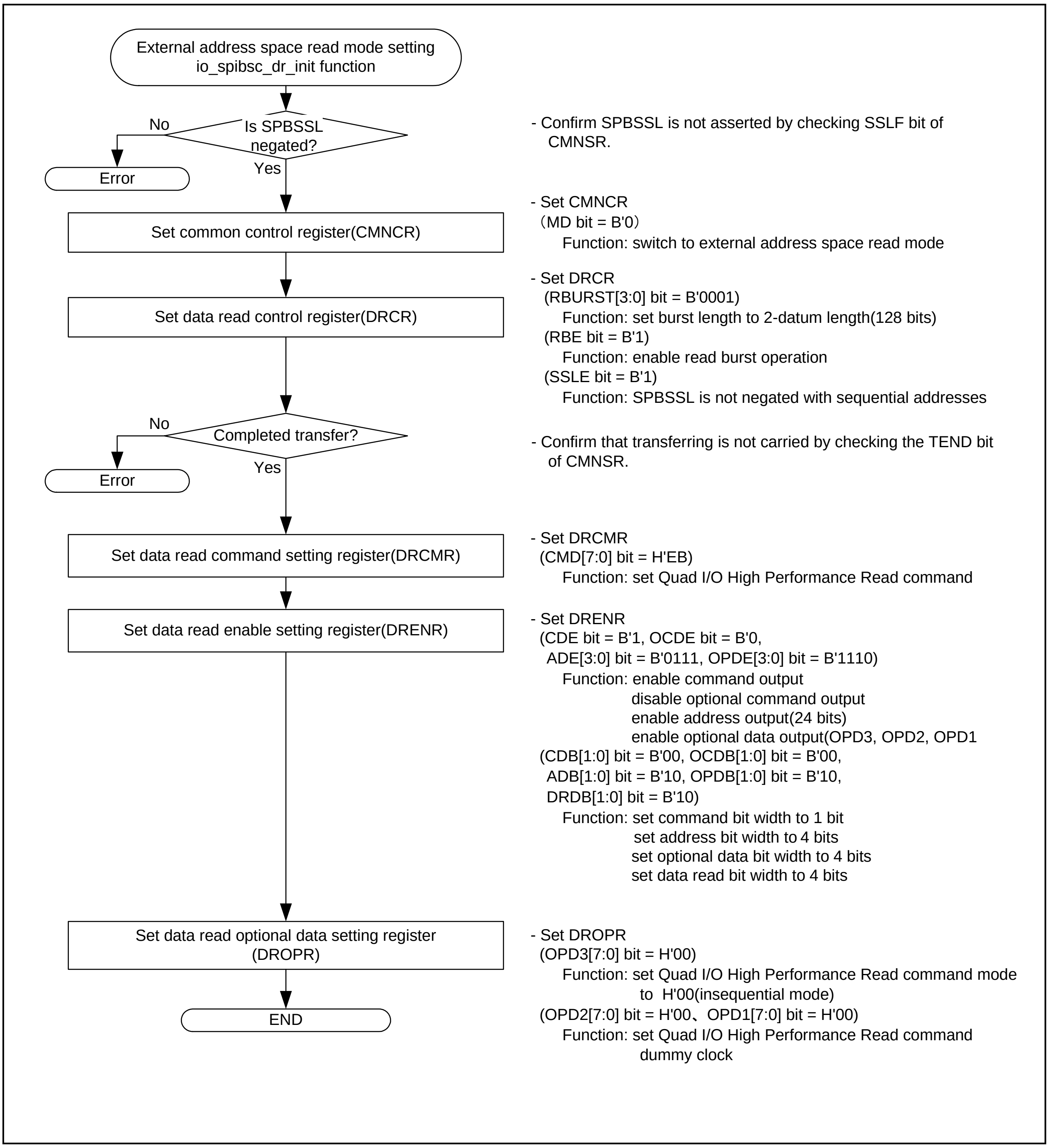


Figure 13 SPBSSL Automatic Negation in Burst Read Operation

2.6.6 External Address Space Read Mode Setting Flow

Figure 14 shows the external address space read mode setting flow chart in this application program.


External address space read mode setting
io_spibsc_dr_init function
Is SPBSSL negated?
No
Yes
Error
Set common control register(CMNCR)
Set data read control register(DRCR)
Completed transfer?
No
Yes
Error
Set data read command setting register(DRCMR)
Set data read enable setting register(DRENR)
Set data read optional data setting register (DROPR)
END

- Confirm SPBSSL is not asserted by checking SSLF bit of CMNSR.

- Set CMNCR
（MD bit = B'0）
Function: switch to external address space read mode

- Set DRCR
(RBURST[3:0] bit = B'0001)
Function: set burst length to 2-datum length(128 bits)
(RBE bit = B'1)
Function: enable read burst operation
(SSLE bit = B'1)
Function: SPBSSL is not negated with sequential addresses

- Confirm that transferring is not carried by checking the TEND bit of CMNSR.

- Set DRCMR
(CMD[7:0] bit = H'EB)
Function: set Quad I/O High Performance Read command

- Set DRENR
(CDE bit = B'1, OCDE bit = B'0,
ADE[3:0] bit = B'0111, OPDE[3:0] bit = B'1110)
Function: enable command output
disable optional command output
enable address output(24 bits)
enable optional data output(OPD3, OPD2, OPD1
(CDB[1:0] bit = B'00, OCDB[1:0] bit = B'00,
ADB[1:0] bit = B'10, OPDB[1:0] bit = B'10,
DRDB[1:0] bit = B'10)
Function: set command bit width to 1 bit
set address bit width to 4 bits
set optional data bit width to 4 bits
set data read bit width to 4 bits

- Set DROPR
(OPD3[7:0] bit = H'00)
Function: set Quad I/O High Performance Read command mode to H'00(insequential mode)
(OPD2[7:0] bit = H'00、OPD1[7:0] bit = H'00)
Function: set Quad I/O High Performance Read command dummy clock

Figure 14 External Address Space Read Mode Setting Flow Chart

2.7 Sample Program Operation Overview

In this section, sample program operation overview is described. First, the sample program is started in external address space read mode effective to execute the main function on the SPI multi I/O bus space. Then, read/write the serial flash memory using the SPI operation mode. This is executed in the function on the large capacity internal RAM as the SPI operation mode is not switched on the SPI multi I/O space. Finally, the external address space read mode is enacted to return to the main functions.

2.7.1 Main Function Flow

Figure 15 shows the main function flow chart of the sample program.

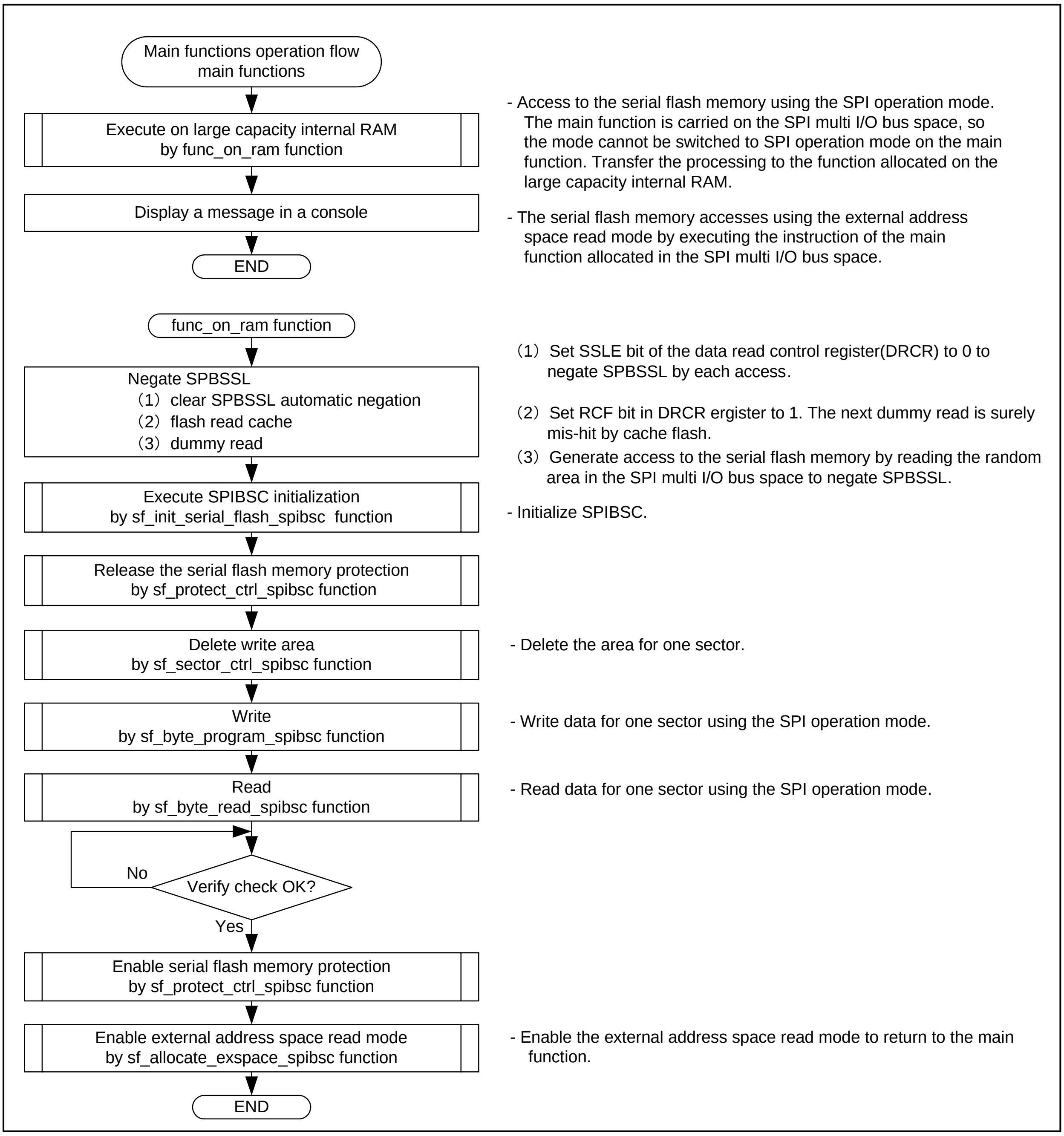

Main functions operation flow
main functions
Execute on large capacity internal RAM
by func_on_ram function
Display a message in a console
END
func_on_ram function
Negate SPBSSL
（1）clear SPBSSL automatic negation
（2）flash read cache
（3）dummy read
Execute SPIBSC initialization
by sf_init_serial_flash_spibsc function
Release the serial flash memory protection
by sf_protect_ctrl_spibsc function
Delete write area
by sf_sector_ctrl_spibsc function
Write
by sf_byte_program_spibsc function
Read
by sf_byte_read_spibsc function
No
Verify check OK?
Yes
Enable serial flash memory protection
by sf_protect_ctrl_spibsc function
Enable external address space read mode
by sf_allocate_exspace_spibsc function
END
- Access to the serial flash memory using the SPI operation mode. The main function is carried on the SPI multi I/O bus space, so the mode cannot be switched to SPI operation mode on the main function. Transfer the processing to the function allocated on the large capacity internal RAM.
- The serial flash memory accesses using the external address space read mode by executing the instruction of the main function allocated in the SPI multi I/O bus space.
（1）Set SSLE bit of the data read control register(DRCR) to 0 to negate SPBSSL by each access.
（2）Set RCF bit in DRCR ergister to 1. The next dummy read is surely mis-hit by cache flash.
（3）Generate access to the serial flash memory by reading the random area in the SPI multi I/O bus space to negate SPBSSL.
- Initialize SPIBSC.
- Delete the area for one sector.
- Write data for one sector using the SPI operation mode.
- Read data for one sector using the SPI operation mode.
- Enable the external address space read mode to return to the main function.

Figure 15 Main Function Operation Flow in Sample Program

3. Sample Program List

3.1 Additional Description about Sample Program

For using boot mode 0 and boot mode 1(booting from the memory connected to CS0 space), setting the pin to SPIBSC function is not available. Therefore, the sample program is booted from the boot mode 3 as a serial flash boot.

For the procedure of using the serial flash boot and for the method to write the program in the serial flash memory, refer to the section “SH7268/SH7269 Group Example for Booting from the Serial Flash Memory Using the SPI Multi I/O Bus Controller”.

3.2 Sample Program List “main.c” (1)

```
/*******************************************************************************
*   DISCLAIMER
*
*   This software is supplied by Renesas Electronics Corporation and is only
*   intended for use with Renesas products. No other uses are authorized.
*
*   This software is owned by Renesas Electronics Corporation and is protected under
*   all applicable laws, including copyright laws.
*
*   THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
*   REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
*   INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
*   PARTICULAR PURPOSE AND NON-INFRINGEMENT.  ALL SUCH WARRANTIES ARE EXPRESSLY
*   DISCLAIMED.
*
*   TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
*   ELECTRONICS CORPORATION NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
*   FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
*   FOR ANY REASON RELATED TO THIS SOFTWARE, EVEN IF RENESAS OR ITS
*   AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
*
*   Renesas reserves the right, without notice, to make changes to this
*   software and to discontinue the availability of this software.
*   By using this software, you agree to the additional terms and
*   conditions found by accessing the following link:
*   http://www.renesas.com/disclaimer
********************************************************************************
*   Copyright (C) 2011 Renesas Electronics Corporation. All rights reserved.
**************************** Technical reference data **************************
*   System Name : SH7268/SH7269 Sample Program
*   File Name   : main.c
*   Abstract    : Sample Program Main
*   Version     : 1.00.00
*   Device      : SH7268/SH7269
*   Tool-Chain  : High-performance Embedded Workshop (Ver.4.07.00).
*               : C/C++ compiler package for the SuperH RISC engine family
*               :                              (Ver.9.03Release02).
*   OS          : None
*   H/W Platform: R0K57269(CPU board)
*   Description :
********************************************************************************
*   History     : Jul.06,2011 Ver.1.00.00
*******************************************************************************/
#include <stdio.h>
#include <string.h>
#include <machine.h>
#include "serial_flash.h"
#include "iodefine.h"

```

3.3 Sample Program List “main.c” (2)

```
/* ==== prototype declaration ==== */
void main(void);
void func_on_ram(void);

/*******************************************************************************
 * ID           :
 * Outline      : main
 * Include      :
 * Declaration : void main(void);
 * Description :
 * Argument     : void
 * Return Value: void
 * Note         : None
 ******************************************************************************/
void main(void)
{
    func_on_ram();

    puts("\nSH7269 SPIBSC Sample Program. Ver.1.00.00");
    puts("Copyright (C) 2011 Renesas Electronics Corporation. All rights reserved.");
    puts("\n");

    while(1){
        /* loop */
    }
}


#pragma section SPIBSC
/*******************************************************************************
 * ID           :
 * Outline      : SPI operating mode
 * Include      :
 * Declaration : void exe_spibsc_spi(void) ;
 * Description :
 * Argument     : void
 * Return Value: void
 * Note         : None
 ******************************************************************************/
void func_on_ram(void)
{
    volatile short dummy;
    int w_size  = SF_PAGE_SIZE;
    int w_sctno = (SF_NUM_OF_SECTOR - 1);
    int w_addr, bsz, i;
    static char r_data[ SF_PAGE_SIZE ];
    static char w_data[ SF_PAGE_SIZE ];

```

3.4 Sample Program List “main.c” (3)

```
    /* ==== Use SPI operating mode ==== */

    /* Initialize data */
    for(i=0; i<w_size; i++){
        r_data[i] = 'R';
        w_data[i] = 'W';
    }
    bsz = 1;
    w_addr = (w_sctno * SF_SECTOR_SIZE * bsz);

    /* Negate SPBSSL */
    SPIBSC.DRCR.BIT.SSLE = 0;        /* No keep SSL */
    SPIBSC.DRCR.BIT.RCF = 1;         /* Chach flush */
    dummy = *(short *)0x18000000;    /* Dummy read */

    /* Initializes the SPIBSC */
    sf_init_serial_flash_spibsc();

    /* Disables the software protection in serial flash memory */
    sf_protect_ctrl_spibsc(SF_REQ_UNPROTECT);

    /* Erase */
    sf_sector_erase_spibsc(w_sctno);

    /* Write */
    sf_byte_program_spibsc(w_addr, w_data, w_size );

    /* Read */
    sf_byte_read_spibsc(w_addr,r_data, w_size);

    /* Verifies data */
    for(i=0; i<w_size; i++){
        if( r_data[i] != w_data[i] ){
            while(1){
                /* error */
            }
        }
    }
    /* Enables the software protection in serial flash memory */
    sf_protect_ctrl_spibsc(SF_REQ_PROTECT);


    /* ==== Enable external address space read mode ==== */
    sf_allocate_exspace_spibsc();

}

/* End of File */
```

3.5 Sample Program List “qserial_flash_spibsc.c” (1)

```
/*******************************************************************************
*   DISCLAIMER
*
*   This software is supplied by Renesas Electronics Corporation and is only
*   intended for use with Renesas products. No other uses are authorized.
*
*   This software is owned by Renesas Electronics Corporation and is protected under
*   all applicable laws, including copyright laws.
*
*   THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
*   REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
*   INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
*   PARTICULAR PURPOSE AND NON-INFRINGEMENT.  ALL SUCH WARRANTIES ARE EXPRESSLY
*   DISCLAIMED.
*
*   TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
*   ELECTRONICS CORPORATION NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
*   FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
*   FOR ANY REASON RELATED TO THIS SOFTWARE, EVEN IF RENESAS OR ITS
*   AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
*
*   Renesas reserves the right, without notice, to make changes to this
*   software and to discontinue the availability of this software.
*   By using this software, you agree to the additional terms and
*   conditions found by accessing the following link:
*   http://www.renesas.com/disclaimer
********************************************************************************
*   Copyright (C) 2011 Renesas Electronics Corporation. All rights reserved.
**************************** Technical reference data **************************
*   System Name : SH7268/SH7269 Firm Update Sample Program
*   File Name   : qserial_flash_spibsc.c
*   Abstract    :
*   Version     : 1.00.00
*   Device      : SH7268/SH7269
*   Tool-Chain  : High-performance Embedded Workshop (Ver.4.07.00).
*               : C/C++ compiler package for the SuperH RISC engine family
*               :                              (Ver.9.03Release02).
*   OS          : None
*   H/W Platform: R0K57269(CPU board)
*   Description :
********************************************************************************
*   History     : Jul.06,2011 Ver.1.00.00
*******************************************************************************/
#include "io_spibsc.h"
#include "serial_flash.h"
#include "qserial_flash_spibsc.h"

```

3.6 Sample Program List “qserial_flash_spibsc.c” (2)

```
#pragma section SPIBSC

/* ---- serial flash command[S25FL032P(Spansion)] ---- */
#define SFLASHCMD_CHIP_ERASE  0xc7
#define SFLASHCMD_SECTOR_ERASE0xd8
#define SFLASHCMD_BYTE_PROGRAM0x02
#define SFLASHCMD_BYTE_READ       0x0B       /* fast read   */
#define SFLASHCMD_DUAL_READ       0x3B
#define SFLASHCMD_QUAD_READ       0x6B
#define SFLASHCMD_DUAL_IO_READ0xBB
#define SFLASHCMD_QUAD_IO_READ0xEB
#define SFLASHCMD_WRITE_ENABLE0x06
#define SFLASHCMD_READ_STATUS 0x05
#define SFLASHCMD_READ_CONFIG 0x35
#define SFLASHCMD_WRITE_STATUS0x01
#define SFLASHCMD_QUAD_PROGRAM0x32
/* ---- serial flash register definitions ---- */
#define CFREG_QUAD_BIT        0x02       /* Quad mode bit(Configuration Register) */
#define CFREG_FREEZE_BIT      0x01       /* freeze bit(Configuration Register) */
#define STREG_BPROTECT_BIT    0x1c       /* protect bit(Status Register) */

/* ==== Prototype Declaration ==== */
static void read_status(unsigned char* status1,unsigned char* status2);
static void read_config(unsigned char* config1,unsigned char* config2);
static void busy_wait(void);
static void write_status(unsigned char status, unsigned char config);
static void sf_set_mode(enum sf_req req);
static void write_enable(void);
#if (SPI_QUAD != 0)
static void sf_byte_read_spibsc_quad(unsigned long addr, unsigned char *buf, int unit);
#endif

/* ==== Global variable ==== */
ST_SPIBSC_SM SpibscSm;



     (The rest is omitted)

```

3.7 Sample Program List “qserial_flash_spibsc.c” (3)

```
/********************************************************************************
 * ID           :
 * Outline      : External address space read mode
 * Include      :
 * Declaration  : void sf_allocate_exspace_spibsc (void);
 * Description  : Set to the external address space read mode
 * Argument     : void
 * Return Value : void
 * Note         : None
 ******************************************************************************/
void sf_allocate_exspace_spibsc (void)
{
#if (SFLASH_DUAL == 0)
    sf_bsz_set_spibsc(1);                       /* s-flash x 1  */
#else
    sf_bsz_set_spibsc(2);                       /* s-flash x 2  */
#endif

#if (SPI_QUAD == 0)
    io_spibsc_dr_init(SFLASHCMD_BYTE_READ);     /* Single-SPI   */
#else
    io_spibsc_dr_init(SFLASHCMD_QUAD_IO_READ);  /* Quad-SPI     */
#endif
}

/********************************************************************************
 * ID            :
 * Outline      : Initialize the serial flash memory
 * Include      :
 * Declaration  : void sf_init_serial_flash_spibsc(void);
 * Description  : Initialize to access to the serial flash memory
 *                : Initialize the SPI multi bus I/O bus controller(SPIBSC)
 *                : to set the serial flash memory to Quad mode
 * Argument     : void
 * Return Value : void
 * Note          : None
 ******************************************************************************/
void sf_init_serial_flash_spibsc(void)
{
    /* ==== Initialize SPIBSC ==== */
#if (SFLASH_DUAL == 0)
    io_spibsc_common_init(SPIBSC_CMNCR_BSZ_SINGLE); /* s-flash x 1  */
#else
    io_spibsc_common_init(SPIBSC_CMNCR_BSZ_DUAL);   /* s-flash x 2  */
#endif

#if (SPI_QUAD == 0)
    sf_set_mode( SF_REQ_SERIALMODE );
#else
    /* ==== setting serial-flash quad mode ==== */
    sf_set_mode( SF_REQ_QUADMODE );
#endif
}
```

3.8 Sample Program List “qserial_flash_spibsc.c” (4)

```

/********************************************************************************
 * ID             :
 * Outline        : Protection
 * Include        :
 * Declaration    : void sf_protect_ctrl_spibsc(enum sf_req req);
 * Description    : Serial flash memory protect setting/clearing the setting
 *                  : Specify the setting by the argument, reg. The initial value of
 *                  : protection/clearance differ to the specification of the serial
 *                  : flash memory.
 * Argument       : enum sf_req req ; I : SF_REQ_UNPROTECT -> clear all sector protection
 *                  :                       SF_REQ_PROTECT   -> protect all sectors
 * Return Value   : void
 * Note           : None
 *******************************************************************************/
void sf_protect_ctrl_spibsc(enum sf_req req)
{
    unsigned char st_reg1, st_reg2;
    unsigned char cf_reg1, cf_reg2;

    read_status(&st_reg1,&st_reg2);
    read_config(&cf_reg1,&cf_reg2);

    /* ==== Set value of Serial Flash(0) ==== */

    /* ---- clear freeze bit in configuration register  ---- */
    write_status( st_reg1 , (unsigned char)(cf_reg1 & (~CFREG_FREEZE_BIT)) );

    if( req == SF_REQ_UNPROTECT ){
        st_reg1 &= ~STREG_BPROTECT_BIT;     /* un-protect in all area   */
    }
    else{
        st_reg1 |= STREG_BPROTECT_BIT;      /* protect in all area      */
    }

    /* ---- clear or set protect bit in status register ---- */
    /* ---- with freeze bit in configuration register   ---- */
    write_status( st_reg1 , (unsigned char)(cf_reg1 | CFREG_FREEZE_BIT) );
}


     (The rest is omitted)
```

3.9 Sample Program List “qserial_flash_spibsc.c” (5)

```
/*******************************************************************************
 * ID            :
 * Outline       : Sector erase
 * Include       :
 * Declaration   : void sf_sector_erase_spibsc(int sector_no);
 * Description   : Erase the specified sector in the serial flash memory
 *               : A write enable command should be issued before erasing or programming.
 *               : After erasing or programming, check the serial flash memory status
 *               : with the busy state is cleared.
 * Argument      : int sector_no ; I :sector number
 * Return Value  : void
 * Note          : None
 ******************************************************************************/
void sf_sector_erase_spibsc(int sector_no)
{
    unsigned long addr = sector_no * SF_SECTOR_SIZE;

#if (SFLASH_DUAL == 1)
    int bsz;

    /*  set BE in both of serial-flash */
    bsz = sf_bsz_get_spibsc();
    sf_bsz_set_spibsc(2);                       /* s-flash x 2  */
#endif

    /* sector erase in Single-SPI   */

    write_enable();                             /* WREN Command */

    SpibscSm.cdb = SPIBSC_1BIT;                 /* Commmand bit-width = Single */
    SpibscSm.adb = SPIBSC_1BIT;                 /* Address bit-width = Single */

    SpibscSm.cde = SPIBSC_OUTPUT_ENABLE;        /* Command Enable */
    SpibscSm.ocde = SPIBSC_OUTPUT_DISABLE;      /* Optional-Command Disable */
    SpibscSm.ade = SPIBSC_OUTPUT_ADDR_24;       /* Enable(Adr[23:0]) */
    SpibscSm.opde = SPIBSC_OUTPUT_DISABLE;      /* Option-Data Disable */
    SpibscSm.spide = SPIBSC_OUTPUT_DISABLE;     /* Disable */

    SpibscSm.sslkp = SPIBSC_SPISSL_NEGATE;      /* Negate after transfer */
    SpibscSm.spire = SPIBSC_SPIDATA_DISABLE;    /* Data Access (Read Disable) */
    SpibscSm.spiwe = SPIBSC_SPIDATA_DISABLE;    /* Data Access (Write Disable) */

    SpibscSm.cmd = SFLASHCMD_SECTOR_ERASE;      /* SE:Sector Erase */

    SpibscSm.addr = addr;       /* dont care in dual mode   */
                                /* because address is calcurated with sector_no */

```

3.10 Sample Program List “qserial_flash_spibsc.c” (6)

```
    io_spibsc_transfer(&SpibscSm);

    busy_wait();

#if (SFLASH_DUAL == 1)
    sf_bsz_set_spibsc(bsz);
#endif
}

/********************************************************************************
 * ID             :
 * Outline        : Data program
 * Include        :
 *Declaration:void sf_byte_program_spibsc(unsigned long addr,unsigned char *buf,int size);
 * Description  : Program the assigned program in the serial flash memory
 *                : Erase the specified sector in the serial flash memory
 *                : A write enable command should be issued before erasing or programming.
 *                : After erasing or programming, check the serial flash memory status
 *                : with the busy state is cleared.
 *                : The maximum write data size is limited by the device.
 * Argument     : unsigned long addr ; I : address in the serial flash memory to write to
 *                : unsigned char *buf ; I : address of the buffer to store write data
 *                : int size           ; I : number of byte to write
 * Return Value : void
 * Note           : None
 *******************************************************************************/
void sf_byte_program_spibsc(unsigned long addr, unsigned char *buf, int size)
{
    int unit;

    write_enable();                             /* WREN Command */

    /* ---- Command,Address ---- */
    SpibscSm.cdb = SPIBSC_1BIT;                 /* Commmand bit-width = Single */
    SpibscSm.adb = SPIBSC_1BIT;                 /* Address bit-width = Single */

    SpibscSm.cde = SPIBSC_OUTPUT_ENABLE;        /* Command Enable */
    SpibscSm.ocde = SPIBSC_OUTPUT_DISABLE;      /* Optional-Command Disable */
    SpibscSm.ade = SPIBSC_OUTPUT_ADDR_24;       /* Enable Adr[23:0] */
    SpibscSm.opde = SPIBSC_OUTPUT_DISABLE;      /* Option-Data Disable */
    SpibscSm.spide = SPIBSC_OUTPUT_DISABLE;     /* Disable */

    SpibscSm.sslkp = SPIBSC_SPISSL_KEEP;        /* Keep after transfer */
    SpibscSm.spire = SPIBSC_SPIDATA_DISABLE;    /* Data Access (Read Disable) */
    SpibscSm.spiwe = SPIBSC_SPIDATA_DISABLE;    /* Data Access (Write Disable) */

#if (SPI_QUAD == 0)
    SpibscSm.cmd = SFLASHCMD_BYTE_PROGRAM;      /* PP: Page Program */
#else
    SpibscSm.cmd = SFLASHCMD_QUAD_PROGRAM;      /* QPP: Quad Page Program */
#endif

```

3.11 Sample Program List “qserial_flash_spibsc.c” (7)

```
    if(io_spibsc_bsz_get() == SPIBSC_CMNCR_BSZ_DUAL){
        SpibscSm.addr = (unsigned long)(addr >> 1);
    }
    else{
        SpibscSm.addr = addr;
    }

    io_spibsc_transfer(&SpibscSm);              /* Command,Address */

    /* ---- Data ---- */
#if (SPI_QUAD == 0)
    SpibscSm.spidb = SPIBSC_1BIT;               /* Single */
#else
    SpibscSm.spidb = SPIBSC_4BIT;               /* Quad */
#endif

    SpibscSm.cde = SPIBSC_OUTPUT_DISABLE;       /* Command Disable */
    SpibscSm.ocde = SPIBSC_OUTPUT_DISABLE;      /* Optional-Command Disable */
    SpibscSm.ade = SPIBSC_OUTPUT_DISABLE;       /* Disable Adr */
    SpibscSm.opde = SPIBSC_OUTPUT_DISABLE;      /* Option-Data Disable */

    SpibscSm.spire = SPIBSC_SPIDATA_DISABLE;    /* Data Access (Read Disable) */
    SpibscSm.spiwe = SPIBSC_SPIDATA_ENABLE;     /* Data Access (Write Ensable) */

    if(io_spibsc_bsz_get() == SPIBSC_CMNCR_BSZ_DUAL){
        if((size % 8) == 0){
            SpibscSm.spide = SPIBSC_OUTPUT_SPID_32; /* Enable(64bit) */
            unit = 8;
        }
        else if((size % 4) == 0){
            SpibscSm.spide = SPIBSC_OUTPUT_SPID_16; /* Enable(32bit) */
            unit = 4;
        }
        else if((size % 2) == 0){
            SpibscSm.spide = SPIBSC_OUTPUT_SPID_8;  /* Enable(16bit) */
            unit = 2;
        }
        else{
            return;
        }
    }
```

3.12 Sample Program List “qserial_flash_spibsc.c” (8)

```
    else{
        if((size % 4) == 0){
            SpibscSm.spide = SPIBSC_OUTPUT_SPID_32; /* Enable(32bit) */
            unit = 4;
        }
        else if((size % 2) == 0){
            SpibscSm.spide = SPIBSC_OUTPUT_SPID_16; /* Enable(16bit) */
            unit = 2;
        }
        else{
            SpibscSm.spide = SPIBSC_OUTPUT_SPID_8;  /* Enable(8bit) */
            unit = 1;
        }
    }

    while(size > 0){
        SpibscSm.smwdr[0] = (unsigned long)(((unsigned long)*buf++ << 24) & 0xff000000ul);
                                                /* Data[63:56] or Data[31:24] */
        if(unit >= 2){
          SpibscSm.smwdr[0]|=(unsigned long)(((unsigned long)*buf++ << 16)& 0x00ff0000ul);
                                                /* Data[55:48] or Data[23:16] */
        }
        if(unit >= 4){
            SpibscSm.smwdr[0] |= (unsigned long)(
                                (((unsigned long)*buf++ << 8 ) & 0x0000ff00ul) |
                                (((unsigned long)*buf++      ) & 0x000000fful));
                                                /* Data[47:40] or Data[15:0] */
        }
        if(unit >= 8){
            SpibscSm.smwdr[1] = (unsigned long)(
                                (((unsigned long)*buf++ << 24) & 0xff000000ul) |
                                (((unsigned long)*buf++ << 16) & 0x00ff0000ul) |
                                (((unsigned long)*buf++ << 8 ) & 0x0000ff00ul) |
                                (((unsigned long)*buf++      ) & 0x000000fful));
                                                /*Data[31: 0] or nothing */
        }

        size -= unit;
        if(size <= 0){
            SpibscSm.sslkp = SPIBSC_SPISSL_NEGATE;
        }
        io_spibsc_transfer(&SpibscSm);          /* Data */
    }

    busy_wait();

}

```

3.13 Sample Program List “qserial_flash_spibsc.c” (9)

```
/*******************************************************************************
 * ID             :
 * Outline      : Data read
 * Include      :
 *Declaration: void sf_byte_read_spibsc(unsigned long addr, unsigned char *buf, int size);
 * Description  : Read the serial memory by the specified number of byte
 * Argument     : unsigned long addr ; I : address of the serial flash memory to read
 *                 : unsigned char *buf ; I : address of the buffer to store the read data
 *                 : int size           ; I : number of byte to read
 * Return Value : void
 * Note          : None
 ******************************************************************************/
#if (SPI_QUAD == 0)


     (Omitted)


#else

void sf_byte_read_spibsc(unsigned long addr, unsigned char *buf, int size)
{
    int unit;

    if(io_spibsc_bsz_get() == SPIBSC_CMNCR_BSZ_DUAL){
        if((size % 8) == 0){
            unit = 8;
        }
        else if((size % 4) == 0){
            unit = 4;
        }
        else if((size % 2) == 0){
            unit = 2;
        }
        else{
            return;
        }
    }
    else{
        if((size % 4) == 0){
            unit = 4;
        }
        else if((size % 2) == 0){
            unit = 2;
        }
        else{
            unit = 1;
        }
    }

```

3.14 Sample Program List “qserial_flash_spibsc.c” (10)

```
    while(size > 0){
        sf_byte_read_spibsc_quad(addr, buf, unit);

        /* increment address and buf */
        addr += unit;
        buf  += unit;

        size -= unit;
    }
}

static void sf_byte_read_spibsc_quad(unsigned long addr, unsigned char *buf, int unit)
{
    /* ---- Command,Address,Dummy ---- */
    SpibscSm.cdb = SPIBSC_1BIT;                 /* Commmand bit-width = Single */
    SpibscSm.adb = SPIBSC_1BIT;                 /* Address bit-width = Single */

    SpibscSm.cde = SPIBSC_OUTPUT_ENABLE;        /* Command Enable */
    SpibscSm.ocde = SPIBSC_OUTPUT_DISABLE;      /* Optional-Command Disable */
    SpibscSm.ade = SPIBSC_OUTPUT_ADDR_24;       /* Enable Adr[23:0] */
    SpibscSm.opde = SPIBSC_OUTPUT_OPD_3;        /* Option-Data OPD3 */
    SpibscSm.spide = SPIBSC_OUTPUT_DISABLE;     /* Disable */

    SpibscSm.sslkp = SPIBSC_SPISSL_KEEP;        /* Keep after transfer */
    SpibscSm.spire = SPIBSC_SPIDATA_DISABLE;    /* Data Access (Read Disable) */
    SpibscSm.spiwe = SPIBSC_SPIDATA_DISABLE;    /* Data Access (Write Disable) */

    SpibscSm.cmd = SFLASHCMD_QUAD_READ;         /* QOR: Quad Output Read */

    if(io_spibsc_bsz_get() == SPIBSC_CMNCR_BSZ_DUAL){
        SpibscSm.addr = (unsigned long)(addr >> 1);
    }
    else{
        SpibscSm.addr = addr;
    }

    io_spibsc_transfer(&SpibscSm);              /* Command,Address */


    /* ---- Data ---- */
    SpibscSm.spidb = SPIBSC_4BIT;               /* Quad */

    SpibscSm.cde = SPIBSC_OUTPUT_DISABLE;       /* Command Disable */
    SpibscSm.ocde = SPIBSC_OUTPUT_DISABLE;      /* Optional-Command Disable */
    SpibscSm.ade = SPIBSC_OUTPUT_DISABLE;       /* Disable Adr */
    SpibscSm.opde = SPIBSC_OUTPUT_DISABLE;      /* Option-Data Disable */

    SpibscSm.spire = SPIBSC_SPIDATA_ENABLE;     /* Data Access (Read Enable) */
    SpibscSm.spiwe = SPIBSC_SPIDATA_DISABLE;    /* Data Access (Write Disable) */

```

3.15 Sample Program List “qserial_flash_spibsc.c” (11)

```
    if(io_spibsc_bsz_get() == SPIBSC_CMNCR_BSZ_DUAL){
        if( unit == 8 ){
            SpibscSm.spide = SPIBSC_OUTPUT_SPID_32; /* Enable(64bit) */
        }
        else if( unit == 4 ){
            SpibscSm.spide = SPIBSC_OUTPUT_SPID_16; /* Enable(32bit) */
        }
        else if( unit == 2 ){
            SpibscSm.spide = SPIBSC_OUTPUT_SPID_8;  /* Enable(16bit) */
        }
        else{
            return;
        }
    }
    else{
        if( unit == 4 ){
            SpibscSm.spide = SPIBSC_OUTPUT_SPID_32; /* Enable(32bit) */
        }
        else if( unit == 2 ){
            SpibscSm.spide = SPIBSC_OUTPUT_SPID_16; /* Enable(16bit) */
        }
        else{
            SpibscSm.spide = SPIBSC_OUTPUT_SPID_8;  /* Enable(8bit) */
        }
    }

    SpibscSm.sslkp = SPIBSC_SPISSL_NEGATE;
    io_spibsc_transfer(&SpibscSm);                  /* Data input */

    *buf++ = (unsigned char)((SpibscSm.smrdr[0] >> 24) & 0x000000fful);
                                                    /* Data[63:56],Data[31:24] */
    if(unit >= 2){
        *buf++ = (unsigned char)((SpibscSm.smrdr[0] >> 16) & 0x000000fful);
                                                    /* Data[55:48],Data[23:16] */
    }
    if(unit >= 4){
        *buf++ = (unsigned char)((SpibscSm.smrdr[0] >> 8 ) & 0x000000fful);
        *buf++ = (unsigned char)((SpibscSm.smrdr[0]      ) & 0x000000fful);
                                                    /* Data[47:40],Data[15:0] */
    }
    if(unit >= 8){
        *buf++ = (unsigned char)((SpibscSm.smrdr[1] >> 24) & 0x000000fful);
        *buf++ = (unsigned char)((SpibscSm.smrdr[1] >> 16) & 0x000000fful);
        *buf++ = (unsigned char)((SpibscSm.smrdr[1] >> 8 ) & 0x000000fful);
        *buf++ = (unsigned char)((SpibscSm.smrdr[1]      ) & 0x000000fful);
                                                    /*Data[31:0] */
    }
}
#endif

```

3.16 Sample Program List “qserial_flash_spibsc.c” (12)

(Omitted)

```
/*******************************************************************************
 * ID              :
 * Outline        : Enable writing
 * Include        :
 * Declaration  : static void write_enable(void);
 * Description  : Issuing the write enable command to permit to erase/program
 *                 : in the serial flash memory
 * Argument      : void
 * Return Value : void
 * Note           : None
 ******************************************************************************/
static void write_enable(void)
{
    SpibscSm.cdb = SPIBSC_1BIT;                 /* Single */

    SpibscSm.cde = SPIBSC_OUTPUT_ENABLE;        /* Command Enable */
    SpibscSm.ocde = SPIBSC_OUTPUT_DISABLE;      /* Optional-Command Disable */
    SpibscSm.ade = SPIBSC_OUTPUT_DISABLE;       /* Address Disable */
    SpibscSm.opde = SPIBSC_OUTPUT_DISABLE;      /* Option-Data Disable */
    SpibscSm.spide = SPIBSC_OUTPUT_DISABLE;     /* Disable */

    SpibscSm.sslkp = SPIBSC_SPISSL_NEGATE;      /* Negate after transfer */
    SpibscSm.spire = SPIBSC_SPIDATA_DISABLE;    /* Data Access (Read Disable) */
    SpibscSm.spiwe = SPIBSC_SPIDATA_DISABLE;    /* Data Access (Write Disable) */

    SpibscSm.cmd = SFLASHCMD_WRITE_ENABLE;      /* WREN:Write Enable */

    io_spibsc_transfer(&SpibscSm);
}
```

(The rest is omitted)

3.17 Sample Program List “qserial_flash_spibsc.h” (1)

```
/*******************************************************************************
*   DISCLAIMER

(Omitted)

********************************************************************************
*   Copyright (C) 2011 Renesas Electronics Corporation. All rights reserved.
**************************** Technical reference data **************************
*   System Name : SH7268/SH7269 Firm Update Sample Program
*   File Name   : qserial_flash_spibsc.h
*   Abstract    :
*   Version     : 1.00.00
*   Device      : SH7268/SH7269
*   Tool-Chain  : High-performance Embedded Workshop (Ver.4.07.00).
*               : C/C++ compiler package for the SuperH RISC engine family
*               :                             (Ver.9.03Release02).
*   OS          : None
*   H/W Platform: R0K57269(CPU board)
*   Description :
********************************************************************************
*   History     : Jul.06,2011 Ver.1.00.00
*******************************************************************************/
#ifndef _QSERIAL_FLASH_SPIBSC_H_
#define _QSERIAL_FLASH_SPIBSC_H_

/* ==== Function prototype declaration ==== */
int sf_bsz_get_spibsc(void);
void sf_bsz_set_spibsc(int bsz);
void sf_allocate_exspace_spibsc (void);
void sf_init_serial_flash_spibsc(void);
void sf_protect_ctrl_spibsc(enum sf_req req);
void sf_chip_erase_spibsc(void);
void sf_sector_erase_spibsc(int sector_no);
void sf_byte_program_spibsc(unsigned long addr, unsigned char *buf, int size);
void sf_byte_read_spibsc(unsigned long addr, unsigned char *buf, int size);

#endif /* _QSERIAL_FLASH_SPIBSC_H_ */
/* End of File */



```

3.18 Sample Program List “io_spibsc.c” (1)

```
/******************************************************************************
*   DISCLAIMER
*
*   This software is supplied by Renesas Electronics Corporation and is only
*   intended for use with Renesas products. No other uses are authorized.
*
*   This software is owned by Renesas Electronics Corporation and is protected under
*   all applicable laws, including copyright laws.
*
*   THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
*   REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
*   INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
*   PARTICULAR PURPOSE AND NON-INFRINGEMENT.  ALL SUCH WARRANTIES ARE EXPRESSLY
*   DISCLAIMED.
*
*   TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
*   ELECTRONICS CORPORATION NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
*   FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
*   FOR ANY REASON RELATED TO THIS SOFTWARE, EVEN IF RENESAS OR ITS
*   AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
*
*   Renesas reserves the right, without notice, to make changes to this
*   software and to discontinue the availability of this software.
*   By using this software, you agree to the additional terms and
*   conditions found by accessing the following link:
*   http://www.renesas.com/disclaimer
********************************************************************************
*   Copyright (C) 2011 Renesas Electronics Corporation. All rights reserved.
**************************** Technical reference data **************************
*   System Name : SH7268/SH7269 Firm Update Sample Program
*   File Name   : io_spibsc.c
*   Abstract    : loader program for spibsc
*   Version     : 1.00.00
*   Device      : SH7268/SH7269
*   Tool-Chain  : High-performance Embedded Workshop (Ver.4.07.00).
*               : C/C++ compiler package for the SuperH RISC engine family
*               :                              (Ver.9.03Release02).
*   OS          : None
*   H/W Platform: R0K57269(CPU board)
*   Description :
********************************************************************************
*   History     : Jul.06,2011 Ver.1.00.00
*******************************************************************************/
#include "iodefine.h"
#include "io_spibsc.h"
#include "machine.h"

```

3.19 Sample Program List “io_spibsc.c” (2)

```
#pragma section SPIBSC

/* ==== define values ==== */

/* ==== Prototype Declaration ==== */

/* ==== Global variable ==== */

/*******************************************************************************
 * ID            :
 * Outline       :
 * Include       : io_spibsc.h
 * Declaration   : int io_spibsc_bsz_set(unsigned long bsz);
 * Description   :
 * Argument      : unsigned long bsz : BSZ bit
 * Return Value  :
 * Note           : None
 ******************************************************************************/
int io_spibsc_bsz_set(unsigned long bsz)
{
  if(SPIBSC.CMNSR.BIT.SSLF != SPIBSC_SSL_NEGATE){
    return -1;
  }
  if(SPIBSC.CMNCR.BIT.BSZ != bsz){
    if(bsz == SPIBSC_CMNCR_BSZ_DUAL){
        /* s-flash x 2 (4bit x 2)  */
        PORT.PBCR3.BIT.PB14MD = 6; /* PB14:SPBIO3_1 */
        PORT.PBCR3.BIT.PB13MD = 6; /* PB13:SPBIO2_1 */
        PORT.PFCR0.BIT.PF3MD = 6;  /* PF3:SPBMI_1/SPBIO1_1 */
        PORT.PFCR0.BIT.PF2MD = 6;  /* PF2:SPBMO_1/SPBIO0_1 */
    }
    SPIBSC.CMNCR.BIT.BSZ = bsz;
    SPIBSC.DRCR.BIT.RCF = SPIBSC_DRCR_RCF_EXE;    /* flush read-cache    */
  }
  return 0;
}

```

3.20 Sample Program List “io_spibsc.c” (3)

```
/****************************************************************************
 * ID              :
 * Outline        :
 * Include        : io_spibsc.h
 * Declaration  : unsigned long io_spibsc_bsz_get(void);
 * Description  :
 * Argument      : void
 * Return Value : BSZ bit
 * Note           : None
 ***************************************************************************/
unsigned long io_spibsc_bsz_get(void)
{
  return (unsigned long)SPIBSC.CMNCR.BIT.BSZ;
}

/****************************************************************************
 * ID              :
 * Outline        :
 * Include        : io_spibsc.h
 * Declaration  : int io_spibsc_common_init(unsigned long bsz);
 * Description  :
 * Argument      : unsigned long bsz : BSZ bit
 * Return Value :
 * Note            : None
 ***************************************************************************/
int io_spibsc_common_init(unsigned long bsz)
{
  CPG.STBCR7.BIT.MSTP75 = 0;

  PORT.PBCR5.BIT.PB20MD = 6;  /* PB20:SPBMI_0/SPBIO1_0  */
  PORT.PBCR4.BIT.PB19MD = 6;  /* PB19:SPBMO_0/SPBIO0_0  */
  PORT.PBCR4.BIT.PB18MD = 6;  /* PB18:SPBSSL  */
  PORT.PBCR4.BIT.PB17MD = 6;  /* PB17:SPBCLK */
  PORT.PBCR4.BIT.PB16MD = 6;  /* PB16:SPBIO3_0 */
  PORT.PBCR3.BIT.PB15MD = 6;  /* PB15:SPBIO2_0 */

  if(bsz == SPIBSC_CMNCR_BSZ_DUAL){
    /* s-flash x 2 (4bit x 2)  */
    PORT.PBCR3.BIT.PB14MD = 6; /* PB14:SPBIO3_1 */
    PORT.PBCR3.BIT.PB13MD = 6; /* PB13:SPBIO2_1 */
    PORT.PFCR0.BIT.PF3MD = 6;  /* PF3:SPBMI_1/SPBIO1_1 */
    PORT.PFCR0.BIT.PF2MD = 6;  /* PF2:SPBMO_1/SPBIO0_1 */
  }

  if(SPIBSC.CMNSR.BIT.SSLF != SPIBSC_SSL_NEGATE){
    return -1;
  }

```

3.21 Sample Program List “io_spibsc.c” (4)

```
  SPIBSC.CMNCR.BIT.MOIIO3 = SPIBSC_OUTPUT_HiZ;
  SPIBSC.CMNCR.BIT.MOIIO2 = SPIBSC_OUTPUT_HiZ;
  SPIBSC.CMNCR.BIT.MOIIO1 = SPIBSC_OUTPUT_HiZ;
  SPIBSC.CMNCR.BIT.MOIIO0 = SPIBSC_OUTPUT_HiZ;

  SPIBSC.CMNCR.BIT.IO3FV = SPIBSC_OUTPUT_HiZ;
  SPIBSC.CMNCR.BIT.IO2FV = SPIBSC_OUTPUT_HiZ;
  SPIBSC.CMNCR.BIT.IO0FV = SPIBSC_OUTPUT_HiZ;

  /* S-flash mode 0    */
  SPIBSC.CMNCR.BIT.CPHAT = SPIBSC_CMNCR_CPHAT_EVEN;
                                          /* even edge : write */
  SPIBSC.CMNCR.BIT.CPHAR = SPIBSC_CMNCR_CPHAR_EVEN;
                                          /* even edge : read */
  SPIBSC.CMNCR.BIT.SSLP = SPIBSC_CMNCR_SSLP_LOW;
                                          /* SPBSSL : low active */
  SPIBSC.CMNCR.BIT.CPOL = SPIBSC_CMNCR_CPOL_LOW;
                                          /* SPBCLK : low at negate */

  io_spibsc_bsz_set(bsz);

  SPIBSC.SSLDR.BIT.SPNDL = SPIBSC_DELAY_1SPBCLK;
                                          /* next access delay */
  SPIBSC.SSLDR.BIT.SLNDL = SPIBSC_DELAY_1SPBCLK;
                                          /* SPBSSL negate delay */
  SPIBSC.SSLDR.BIT.SCKDL = SPIBSC_DELAY_1SPBCLK;
                                          /* clock delay */

  /* ---- Bit rate 66.67Mbps ---- */
  SPIBSC.SPBCR.BIT.SPBR = 1;              /* divide 2 base bit rate B clock(133.33MHz) */
  SPIBSC.SPBCR.BIT.BRDV = 0;

  return 0;
}

```

3.22 Sample Program List “io_spibsc.c” (5)

```
/*******************************************************************************
 * ID              :
 * Outline        :
 * Include        : io_spibsc.h
 * Declaration    : int io_spibsc_dr_init(unsigned long cmd);
 * Description    :
 * Argument       : void
 * Return Value   :
 * Note            : None
 ******************************************************************************/
int io_spibsc_dr_init(unsigned long cmd)
{
  if(SPIBSC.CMNSR.BIT.SSLF != SPIBSC_SSL_NEGATE){
    return -1;
  }

  SPIBSC.CMNCR.BIT.MD = SPIBSC_CMNCR_MD_EXTRD;   /* SPI I/O mode*/

  SPIBSC.DRCR.BIT.RBURST = SPIBSC_BURST_2;
  SPIBSC.DRCR.BIT.RBE = SPIBSC_BURST_ENABLE;
  SPIBSC.DRCR.BIT.SSLE = SPIBSC_SPISSL_KEEP;     /* Keep SSL after read */
                              /* if not continuous address it negeted   */

  if(SPIBSC.CMNSR.BIT.TEND != SPIBSC_TRANS_END){
    return -1;
  }

  /* ---- Command ---- */
  SPIBSC.DRCMR.BIT.CMD = cmd;                /* Command */
  SPIBSC.DRENR.BIT.CDB = SPIBSC_1BIT;        /* Single */
  SPIBSC.DRENR.BIT.CDE = SPIBSC_OUTPUT_ENABLE;
                                          /* Enable */
  /* ---- Option Command ---- */
  SPIBSC.DRCMR.BIT.OCMD = 0x00;
  SPIBSC.DRENR.BIT.OCDB = SPIBSC_1BIT;       /* Single */
  SPIBSC.DRENR.BIT.OCDE = SPIBSC_OUTPUT_DISABLE;
                                              /* Disable */

  /* ---- Address ---- */
  if(cmd == 0xBB){
                                              /* Dual I/O High Performance */
    SPIBSC.DRENR.BIT.ADB = SPIBSC_2BIT;       /* Dual */
    SPIBSC.DRENR.BIT.ADE = SPIBSC_OUTPUT_ADDR_24;
                                              /* S-flash x 1 Enable(ADDR[23:0]) */
                                              /* S-flash x 2 Enable(ADDR[24:1]) */
  }
```

3.23 Sample Program List “io_spibsc.c” (6)

```
    else if(cmd == 0xEB){
                                            /* Quad I/O High Performance */
      SPIBSC.DRENR.BIT.ADB = SPIBSC_4BIT;   /* Quad */
      SPIBSC.DRENR.BIT.ADE = SPIBSC_OUTPUT_ADDR_24;
                                            /* S-flash x 1 Enable(ADDR[23:0]) */
                                            /* S-flash x 2 Enable(ADDR[24:1]) */
    }
    else{
      SPIBSC.DRENR.BIT.ADB = SPIBSC_1BIT;   /* Single */
      SPIBSC.DRENR.BIT.ADE = SPIBSC_OUTPUT_ADDR_24;
                                            /* S-flash x 1 Enable(ADDR[23:0]) */
                                            /* S-flash x 2 Enable(ADDR[24:1]) */
    }

    /* ---- Option Data ---- */
    if(cmd == 0xBB){
                                            /* Dual I/O High Performance */
      SPIBSC.DROPR.BIT.OPD3 = 0x00;         /* Option Data(Mode bit) */
      SPIBSC.DROPR.BIT.OPD2 = 0x00;         /* Option Data */
      SPIBSC.DROPR.BIT.OPD1 = 0x00;         /* Option Data */
      SPIBSC.DROPR.BIT.OPD0 = 0x00;         /* Option Data */
      SPIBSC.DRENR.BIT.OPDB = SPIBSC_2BIT;  /* Dual */
      SPIBSC.DRENR.BIT.OPDE = SPIBSC_OUTPUT_OPD_3;
                                            /* Enable(OPD3) */
    }
    else if(cmd == 0xEB){
                                            /* Quad I/O High Performance */
      SPIBSC.DROPR.BIT.OPD3 = 0x00;         /* Option Data(Mode bit) */
      SPIBSC.DROPR.BIT.OPD2 = 0x00;         /* Option Data(Dummy) */
      SPIBSC.DROPR.BIT.OPD1 = 0x00;         /* Option Data(Dummy) */
      SPIBSC.DROPR.BIT.OPD0 = 0x00;         /* Option Data */
      SPIBSC.DRENR.BIT.OPDB = SPIBSC_4BIT;  /* Quad */
      SPIBSC.DRENR.BIT.OPDE = SPIBSC_OUTPUT_OPD_321;
                                            /* Enable(OPD3,OPD2,OPD1) */
    }
    else{
      SPIBSC.DROPR.BIT.OPD3 = 0x00;         /* Option Data(Dummy) */
      SPIBSC.DROPR.BIT.OPD2 = 0x00;         /* Option Data */
      SPIBSC.DROPR.BIT.OPD1 = 0x00;         /* Option Data */
      SPIBSC.DROPR.BIT.OPD0 = 0x00;         /* Option Data */
      SPIBSC.DRENR.BIT.OPDB = SPIBSC_1BIT;  /* Single */
      SPIBSC.DRENR.BIT.OPDE = SPIBSC_OUTPUT_OPD_3;
                                            /* Enable(OPD3) */
    }
```

3.24 Sample Program List “io_spibsc.c” (7)

```
  /* ---- Data ---- */
  if(cmd == 0x6B){
    SPIBSC.DRENR.BIT.DRDB = SPIBSC_4BIT;     /* Quad */
  }
  else if(cmd == 0x3B){
    SPIBSC.DRENR.BIT.DRDB = SPIBSC_2BIT;     /* Dual */
  }
  else if(cmd == 0x0B){
    SPIBSC.DRENR.BIT.DRDB = SPIBSC_1BIT;     /* Single */
  }
  else if(cmd == 0xBB){
    SPIBSC.DRENR.BIT.DRDB = SPIBSC_2BIT;     /* Dual I/O High Performance */
  }
  else if(cmd == 0xEB){
    SPIBSC.DRENR.BIT.DRDB = SPIBSC_4BIT;     /* Quad I/O High Performance */
  }
  else{
    return -1;
  }
  return 0;
}

/********************************************************************************
 * ID             :
 * Outline       :
 * Include       : io_spibsc.h
 * Declaration  : int io_spibsc_transfer(ST_SPIBSC_SM *SpibscSm);
 * Description  :
 * Argument      : void
 * Return Value :
 * Note          : None
 *******************************************************************************/
int io_spibsc_transfer(ST_SPIBSC_SM *SpibscSm)
{
  int i;
  volatile unsigned long dummy;

  if(SPIBSC.CMNCR.BIT.MD != SPIBSC_CMNCR_MD_SPI){
    if(SPIBSC.CMNSR.BIT.SSLF != SPIBSC_SSL_NEGATE){
        return -1;
    }
    SPIBSC.CMNCR.BIT.MD = SPIBSC_CMNCR_MD_SPI;
                                             /* SPI Mode */
  }

  if(SPIBSC.CMNSR.BIT.TEND != SPIBSC_TRANS_END){
    return -1;
  }

```

3.25 Sample Program List “io_spibsc.c” (8)

```
    /* ---- Command ---- */
    SPIBSC.SMENR.BIT.CDE = SpibscSm->cde;       /* Enable/Disable */
    if(SpibscSm->cde != SPIBSC_OUTPUT_DISABLE){
        SPIBSC.SMCMR.BIT.CMD = SpibscSm->cmd;   /* Command */
        SPIBSC.SMENR.BIT.CDB = SpibscSm->cdb;   /* Single/Dual/Quad */
    }

    /* ---- Option Command ---- */
    SPIBSC.SMENR.BIT.OCDE = SpibscSm->ocde;     /* Enable/Disable */
    if(SpibscSm->ocde != SPIBSC_OUTPUT_DISABLE){
        SPIBSC.SMCMR.BIT.OCMD = SpibscSm->ocmd; /* Option Command */
        SPIBSC.SMENR.BIT.OCDB = SpibscSm->ocdb; /* Single/Dual/Quad */
    }

    /* ---- Address ---- */
    SPIBSC.SMENR.BIT.ADE = SpibscSm->ade;           /* Enable/Disable */
    if(SpibscSm->ade != SPIBSC_OUTPUT_DISABLE){
        SPIBSC.SMADR.BIT.ADR = SpibscSm->addr;      /* Address */
        SPIBSC.SMENR.BIT.ADB = SpibscSm->adb;       /* Single/Dual/Quad */
    }

    /* ---- Option Data ---- */
    SPIBSC.SMENR.BIT.OPDE = SpibscSm->opde;         /* Enable/Disable */
    if(SpibscSm->opde != SPIBSC_OUTPUT_DISABLE){
        SPIBSC.SMOPR.BIT.OPD3 = SpibscSm->opd[0];   /* Option Data */
        SPIBSC.SMOPR.BIT.OPD2 = SpibscSm->opd[1];   /* Option Data */
        SPIBSC.SMOPR.BIT.OPD1 = SpibscSm->opd[2];   /* Option Data */
        SPIBSC.SMOPR.BIT.OPD0 = SpibscSm->opd[3];   /* Option Data */
        SPIBSC.SMENR.BIT.OPDB = SpibscSm->opdb;     /* Single/Dual/Quad */
    }

    /* ---- Data ---- */
    SPIBSC.SMENR.BIT.SPIDE = SpibscSm->spide;       /* Enable/Disable */
    if(SpibscSm->spide != SPIBSC_OUTPUT_DISABLE){
        SPIBSC.SMWDR0.LONG = SpibscSm->smwdr[0];
        SPIBSC.SMWDR1.LONG = SpibscSm->smwdr[1];    /* Valid in two serial-flash */
        SPIBSC.SMENR.BIT.SPIDB = SpibscSm->spidb;   /* Single/Dual/Quad */
    }

    SPIBSC.SMCR.BIT.SSLKP = SpibscSm->sslkp;

    if((SpibscSm->spidb != SPIBSC_1BIT) && (SpibscSm->spide != SPIBSC_OUTPUT_DISABLE)){
       if((SpibscSm->spire  == SPIBSC_SPIDATA_ENABLE) &&
          (SpibscSm->spiwe  == SPIBSC_SPIDATA_ENABLE)){
        /* not set in same time */
        return -1;
    }
  }
```

3.26 Sample Program List “io_spibsc.c” (9)

```
    SPIBSC.SMCR.BIT.SPIRE = SpibscSm->spire;
    SPIBSC.SMCR.BIT.SPIWE = SpibscSm->spiwe;

    SPIBSC.SMCR.BIT.SPIE = SPIBSC_SPI_ENABLE;      /* execute after setting SPNDL bit */

    /* wait for transfer-start */
    dummy = SPIBSC.CMNSR.LONG;
    dummy = SPIBSC.CMNSR.LONG;
    dummy = SPIBSC.CMNSR.LONG;
    dummy = SPIBSC.CMNSR.LONG;

    while(SPIBSC.CMNSR.BIT.TEND != SPIBSC_TRANS_END){
      /* wait for transfer-end */
    }
    SpibscSm->smrdr[0] = SPIBSC.SMRDR0.LONG;
    SpibscSm->smrdr[1] = SPIBSC.SMRDR1.LONG;       /* valid in two serial-flash  */

    return 0;
  }

  /* End of File */

```

3.27 Sample Program List “io_spibsc.h” (1)

```
/********************************************************************************
*   DISCLAIMER

(Omitted)

*********************************************************************************
*   Copyright (C) 2011 Renesas Electronics Corporation. All rights reserved.
**************************** Technical reference data ***************************
*   System Name : SH7268/SH7269 Firm Update Sample Program
*   File Name   : io_spibsc.h
*   Abstract    : spibsc structure
*   Version     : 1.00.00
*   Device      : SH7268/SH7269
*   Tool-Chain  : High-performance Embedded Workshop (Ver.4.07.00).
*               : C/C++ compiler package for the SuperH RISC engine family
*               :                              (Ver.9.03Release02).
*   OS          : None
*   H/W Platform: R0K57269(CPU board)
*   Description :
*********************************************************************************
*   History     : Jul.06,2011 Ver.1.00.00
********************************************************************************/
#ifndef _IO_SPIBSC_H_
#define _IO_SPIBSC_H_

/* ==== define values ==== */
#define SPIBSC_CMNCR_MD_EXTRD     0
#define SPIBSC_CMNCR_MD_SPI       1

#define SPIBSC_OUTPUT_LOW         0
#define SPIBSC_OUTPUT_HIGH        1
#define SPIBSC_OUTPUT_LAST        2
#define SPIBSC_OUTPUT_HiZ         3

#define SPIBSC_CMNCR_CPHAT_EVEN   0
#define SPIBSC_CMNCR_CPHAT_ODD

#define SPIBSC_CMNCR_CPHAR_ODD    0
#define SPIBSC_CMNCR_CPHAR_EVEN   1

#define SPIBSC_CMNCR_SSLP_LOW     0
#define SPIBSC_CMNCR_SSLP_HIGH    1

#define SPIBSC_CMNCR_CPOL_LOW     0
#define SPIBSC_CMNCR_CPOL_HIGH    1

#define SPIBSC_CMNCR_BSZ_SINGLE       0
#define SPIBSC_CMNCR_BSZ_DUAL     1

```

3.28 Sample Program List “io_spibsc.h” (2)

```
#define SPIBSC_DELAY_1SPBCLK      0
#define SPIBSC_DELAY_2SPBCLK      1
#define SPIBSC_DELAY_3SPBCLK      2
#define SPIBSC_DELAY_4SPBCLK      3
#define SPIBSC_DELAY_5SPBCLK      4
#define SPIBSC_DELAY_6SPBCLK      5
#define SPIBSC_DELAY_7SPBCLK      6
#define SPIBSC_DELAY_8SPBCLK      7


#define SPIBSC_BURST_1            0x00
#define SPIBSC_BURST_2            0x01
#define SPIBSC_BURST_3            0x02
#define SPIBSC_BURST_4            0x03
#define SPIBSC_BURST_5            0x04
#define SPIBSC_BURST_6            0x05
#define SPIBSC_BURST_7            0x06
#define SPIBSC_BURST_8            0x07
#define SPIBSC_BURST_9            0x08
#define SPIBSC_BURST_10           0x09
#define SPIBSC_BURST_11           0x0a
#define SPIBSC_BURST_12           0x0b
#define SPIBSC_BURST_13           0x0c
#define SPIBSC_BURST_14           0x0d
#define SPIBSC_BURST_15           0x0e
#define SPIBSC_BURST_16           0x0f

#define SPIBSC_BURST_DISABLE      0
#define SPIBSC_BURST_ENABLE       1

#define SPIBSC_DRCR_RCF_EXE       1

#define SPIBSC_SSL_NEGATE         0
#define SPIBSC_TRANS_END          1

#define SPIBSC_1BIT               0
#define SPIBSC_2BIT               1
#define SPIBSC_4BIT               2

#define SPIBSC_OUTPUT_DISABLE     0
#define SPIBSC_OUTPUT_ENABLE      1
#define SPIBSC_OUTPUT_ADDR_24     0x07
#define SPIBSC_OUTPUT_ADDR_32     0x0f
#define SPIBSC_OUTPUT_OPD_3           0x08
#define SPIBSC_OUTPUT_OPD_32      0x0c
#define SPIBSC_OUTPUT_OPD_321     0x0e
#define SPIBSC_OUTPUT_OPD_3210    0x0f
```

3.29 Sample Program List “io_spibsc.h” (3)

```
#define SPIBSC_OUTPUT_SPID_8        0x08
#define SPIBSC_OUTPUT_SPID_16       0x0c
#define SPIBSC_OUTPUT_SPID_32       0x0f

#define SPIBSC_SPISSL_NEGATE        0
#define SPIBSC_SPISSL_KEEP          1

#define SPIBSC_SPIDATA_DISABLE      0
#define SPIBSC_SPIDATA_ENABLE       1

#define SPIBSC_SPI_DISABLE          0
#define SPIBSC_SPI_ENABLE           1

typedef struct{
    unsigned long cdb:2;
    unsigned long ocdb:2;
    unsigned long adb:2;
    unsigned long opdb:2;
    unsigned long spidb:2;
    unsigned long cde:1;
    unsigned long ocde:1;
    unsigned long ade:4;
    unsigned long opde:4;
    unsigned long spide:4;
    unsigned long sslkp:1;
    unsigned long spire:1;
    unsigned long spiwe:1;
    unsigned long :5;

    unsigned char cmd;
    unsigned char ocmd;
    unsigned long addr;
    unsigned char opd[4];
    unsigned long smrdr[2];
    unsigned long smwdr[2];
}ST_SPIBSC_SM;

int io_spibsc_bsz_set(unsigned long bsz);
unsigned long io_spibsc_bsz_get(void);
int io_spibsc_common_init(unsigned long bsz);
int io_spibsc_dr_init(unsigned long cmd);
int io_spibsc_transfer(ST_SPIBSC_SM *SpibscSm);

#endif /* IO_SPIBSC_H */
/* End of File */
```

4. References

- Software Manual
  SH-2A/SH2A-FPU Software Manual Rev. 3.00
  The latest version can be downloaded from the Renesas Electronics website.

- User's Manual for Hardware
  SH7268 Group, SH7269 Group User's Manual: Hardware Rev. 1.00
  The latest version can be downloaded from the Renesas Electronics website.

Website and Support

Renesas Electronics Website
http://www.renesas.com/

Inquiries
http://www.renesas.com/inquiry

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Jul 11.11	—	First edition issued
1.01	Feb.16.12	—	Added sample code of SH726B

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this manual, refer to the relevant sections of the manual. If the descriptions under General Precautions in the Handling of MPU/MCU Products and in the body of the manual differ from each other, the description in the body of the manual takes precedence.

1. Handling of Unused Pins
   Handle unused pins in accord 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 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. Unused pins should be handled as described under Handling of Unused Pins in the manual.
2. Processing at Power-on
   The state of the product is undefined at the moment 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 moment 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 moment 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 moment when power is supplied until the power reaches the level at which resetting has been specified.
3. Prohibition of Access to Reserved Addresses
   Access to reserved addresses is prohibited.
   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.
4. Clock Signals
   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has 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. Moreover, 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.
5. Differences between Products
   Before changing from one product to another, i.e. to one with a different type number, confirm that the change will not lead to problems.
   — The characteristics of MPU/MCU in the same group but having different type numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different type numbers, implement a system-evaluation test for each of the products.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of 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. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. 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 of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may 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.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, 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, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be 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, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

RENESAS

SALES OFFICES

Renesas Electronics Corporation

http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
1 harbourFront Avenue, #06-10, keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 1.1