

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.....	2
2. Explanation of Application Program	3
3. Sample Program List.....	22
4. References	51

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 (I ϕ) : 266.67 MHz Internal bus clock (B ϕ) : 133.33 MHz Peripheral clock 1 (P1 ϕ) : 66.67 MHz Peripheral clock 0 (P0 ϕ) : 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

Serial Flash Memory Connection Sample Program

Figure 1 shows 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.

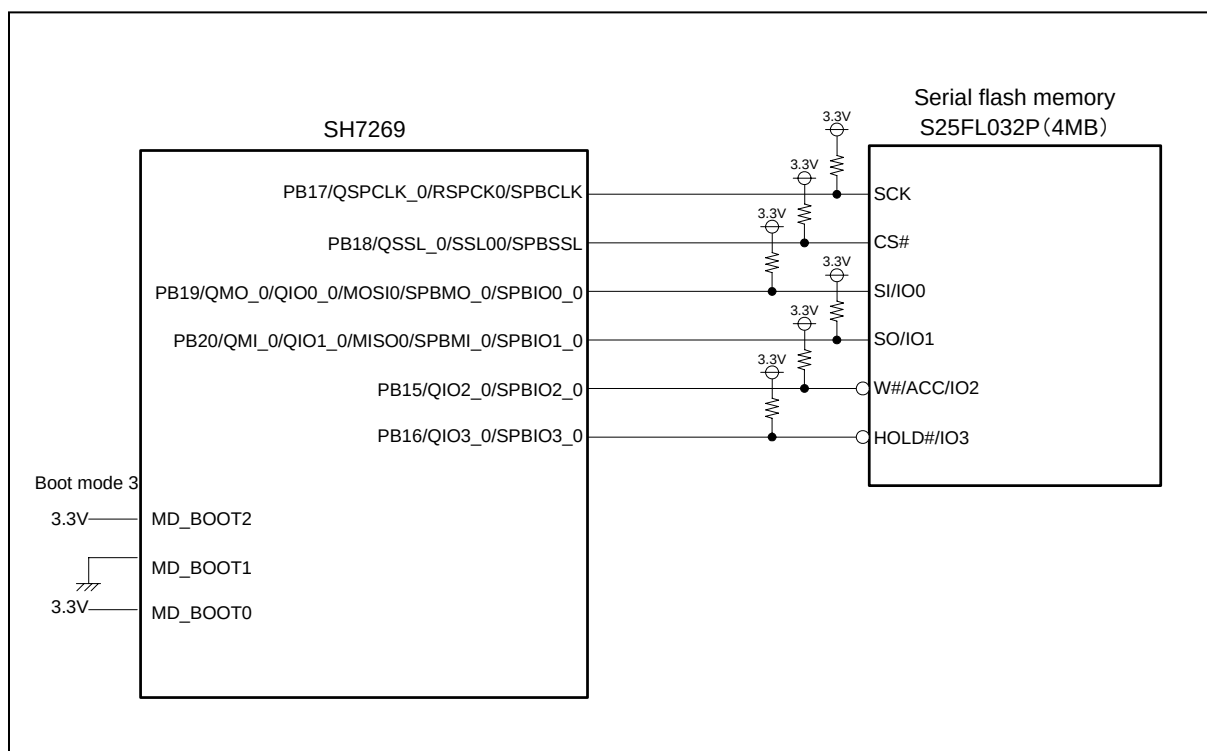


Figure 1 Example of Serial Flash memory Circuit

Note: Treating pull-up/down by the resistor 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 resistor attached externally.

Table 2 Multiplexed Output Pins

Periphera I 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 CS0 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.

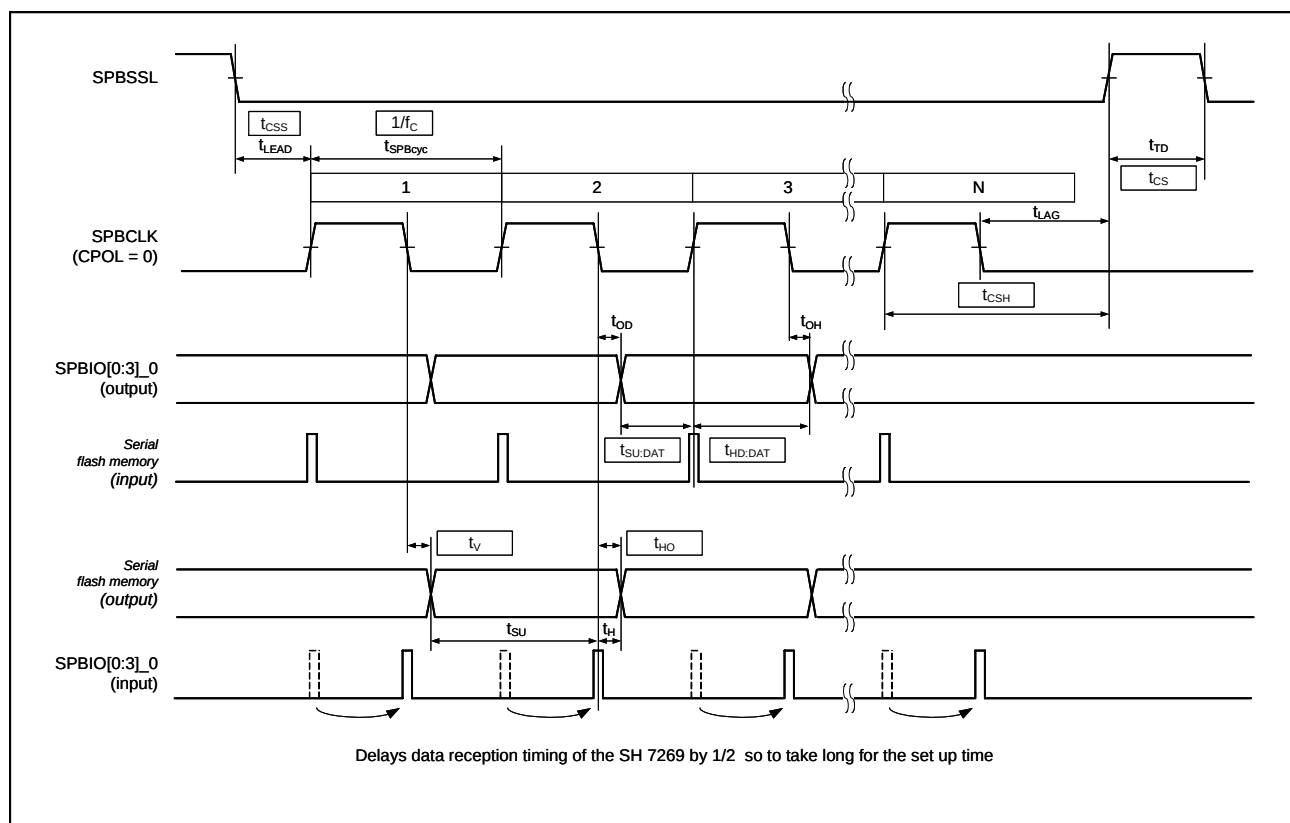


Figure 2 Interface Timing in this Application Program

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

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

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. $t_{SPBcyc} - t_v(\text{maximum}) \geq t_{SU}(\text{minimum})$
t_H	Data input hold time	Necessary hold time for data input. The following requirement should be met. $t_{HO}(\text{min}) \geq t_H(\text{min})$

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

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 $B \phi$ (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 (SSLDL)	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.

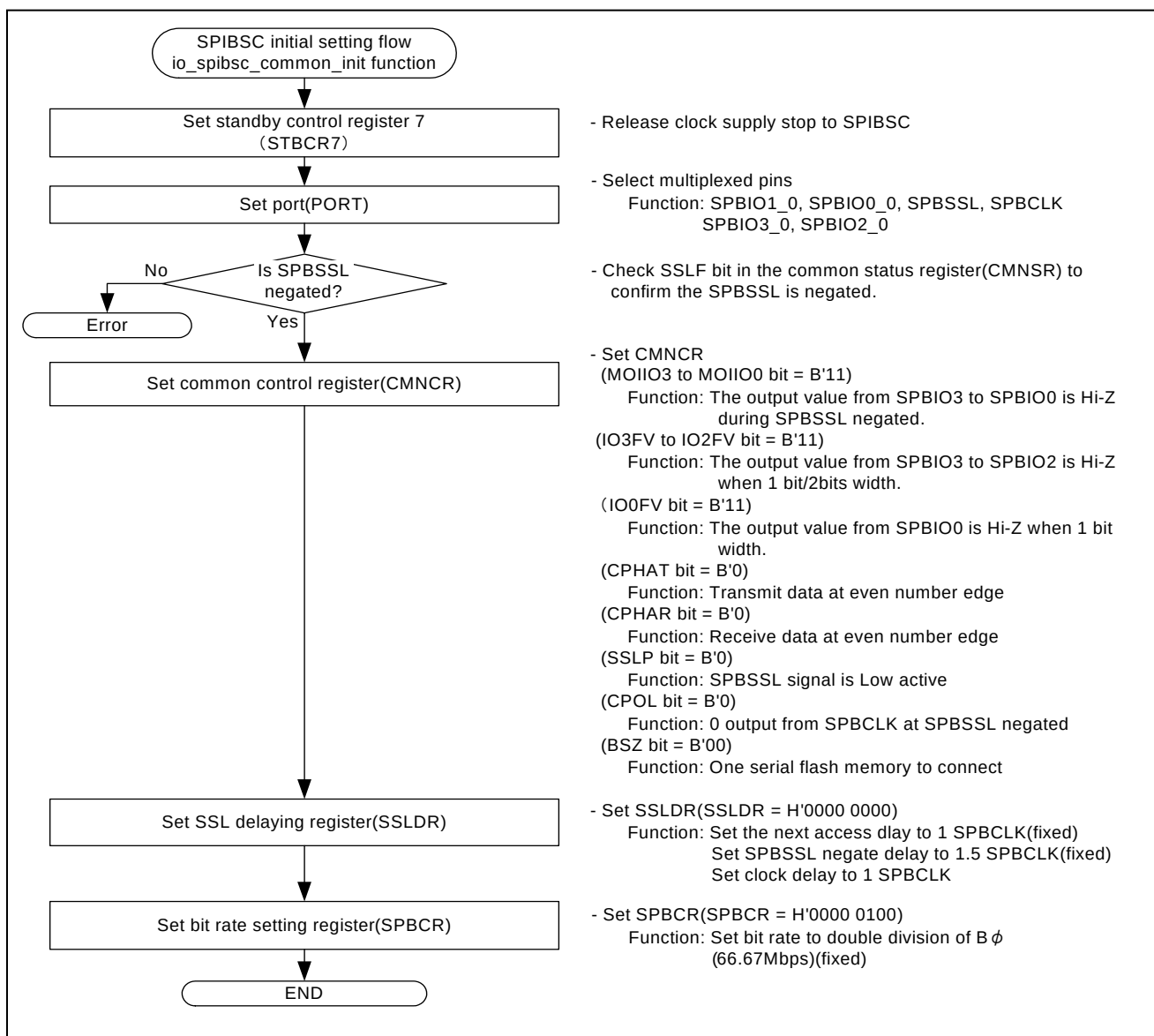


Figure 3 Flow Chart of SPIBSC Initial Setting in this Application Program

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.

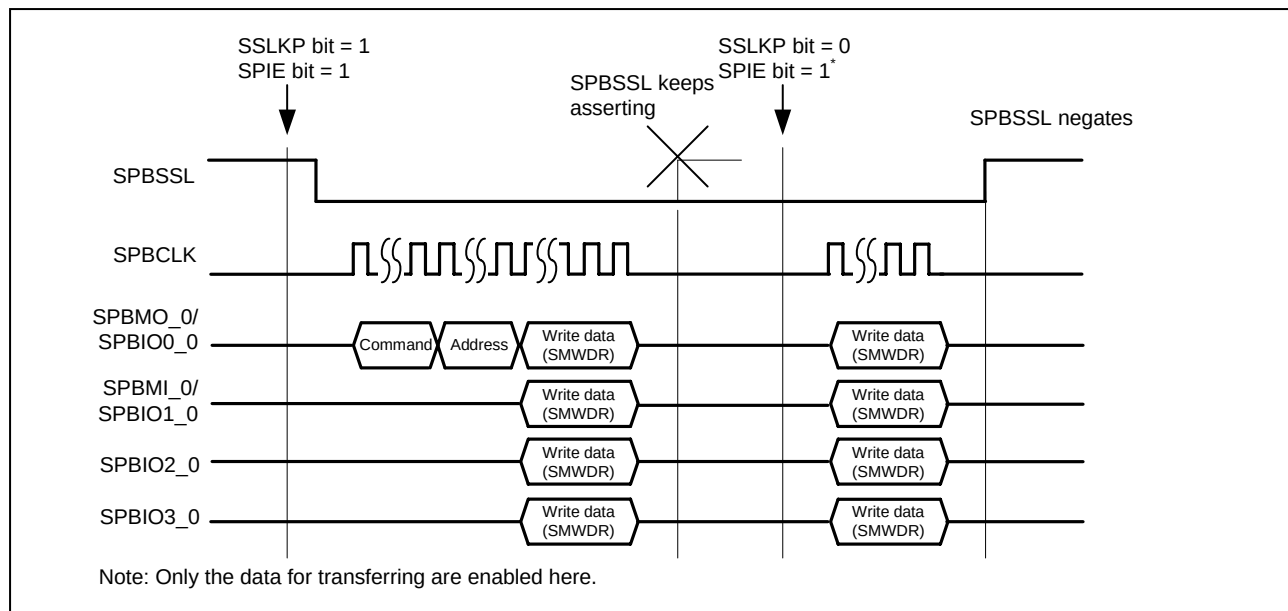


Figure 4 SPBSSL Assert Retention in the SPI Mode

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.

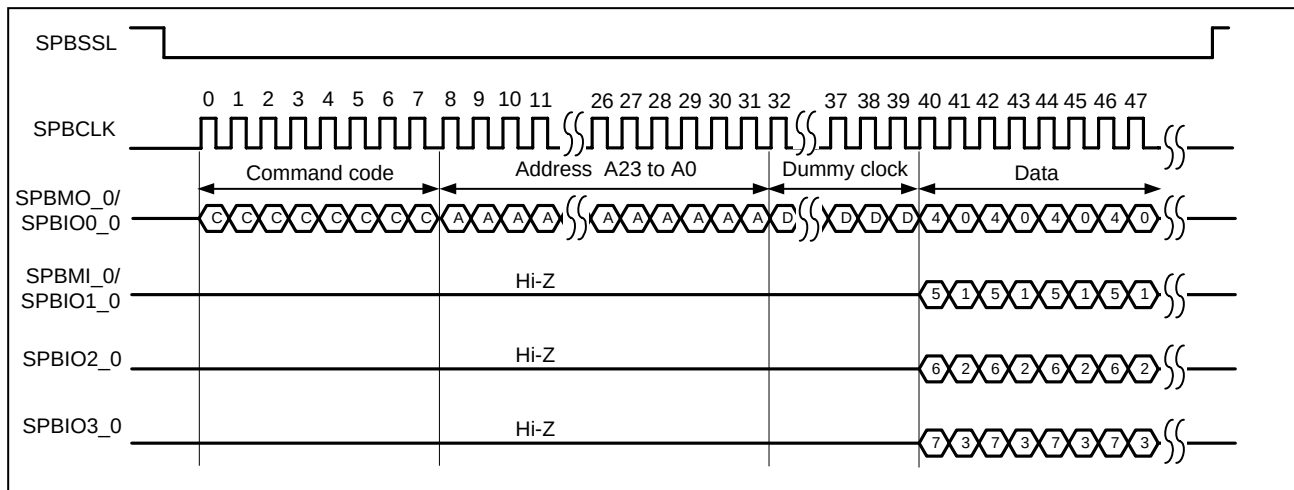


Figure 5 Quad Output Read Command Sequence

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.

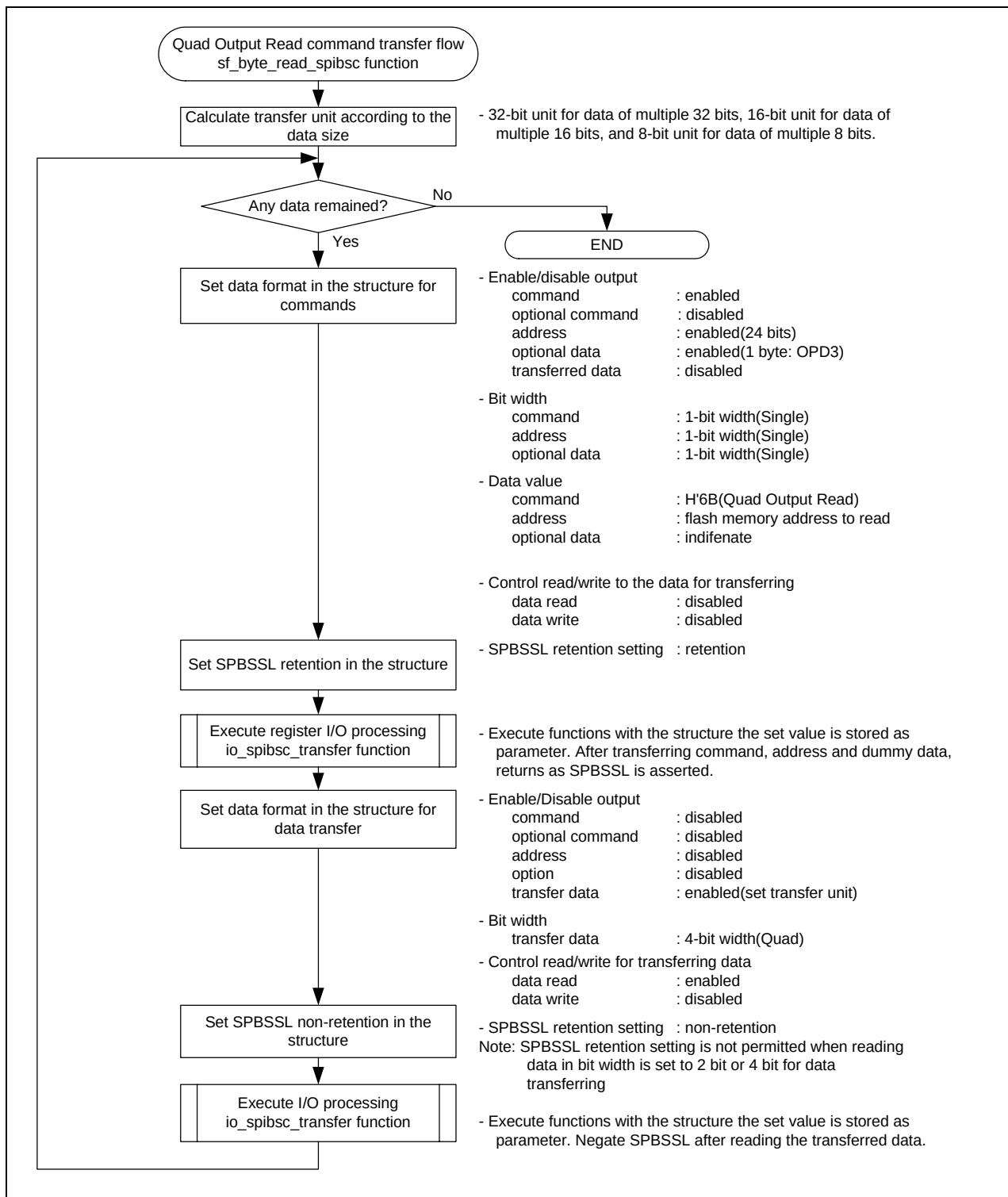


Figure 6 Read Command Low Chart in the SPI Operation Mode

Serial Flash Memory Connection Sample Program

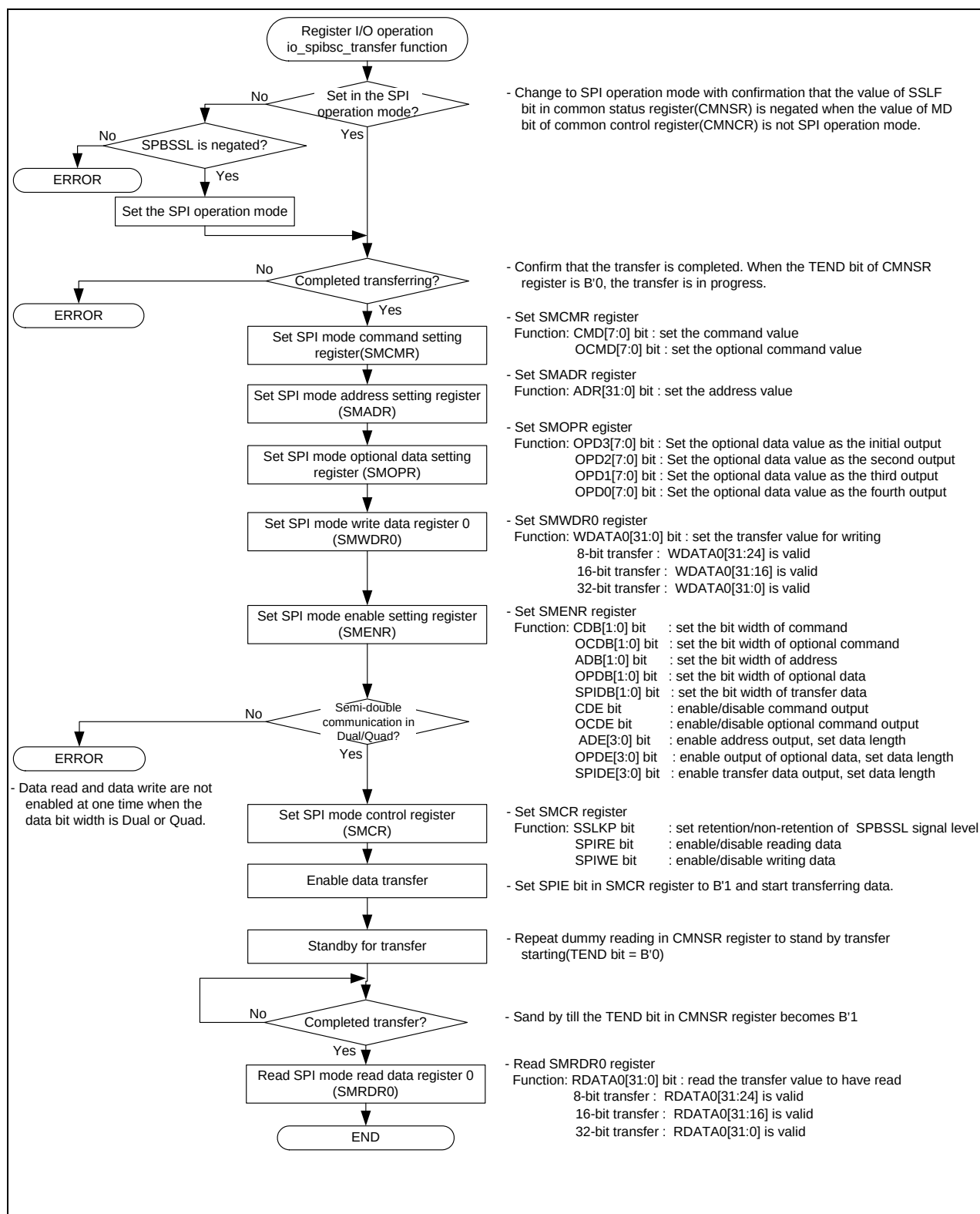


Figure 7 Register I/O Flow Chart in SPI Operation Mode

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.

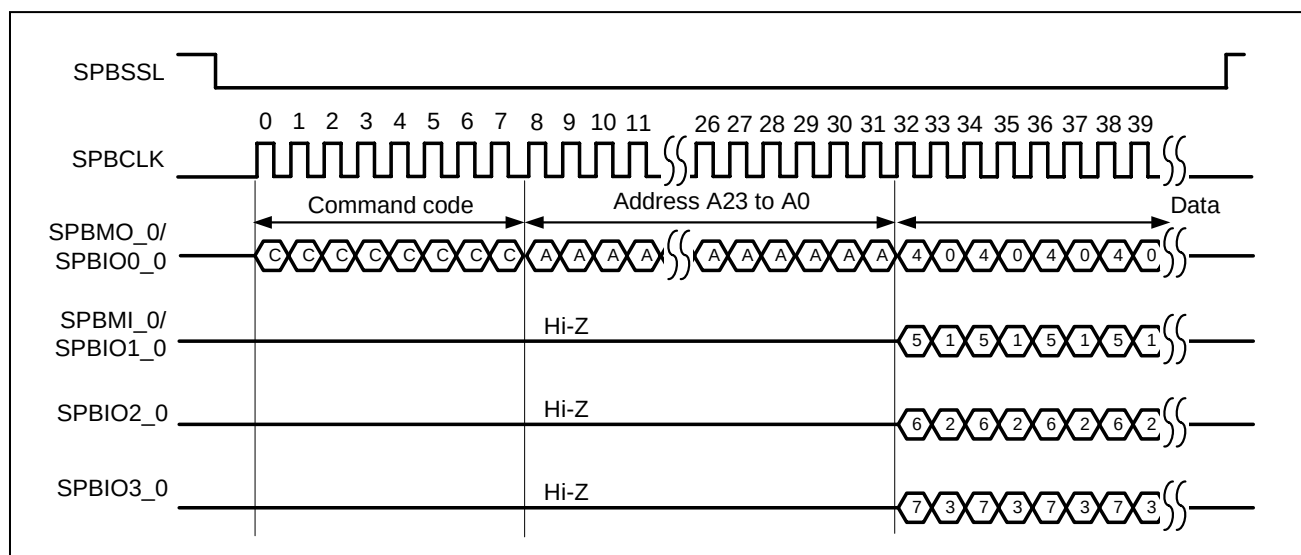


Figure 8 Quad Page Programming Command Sequence

(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.

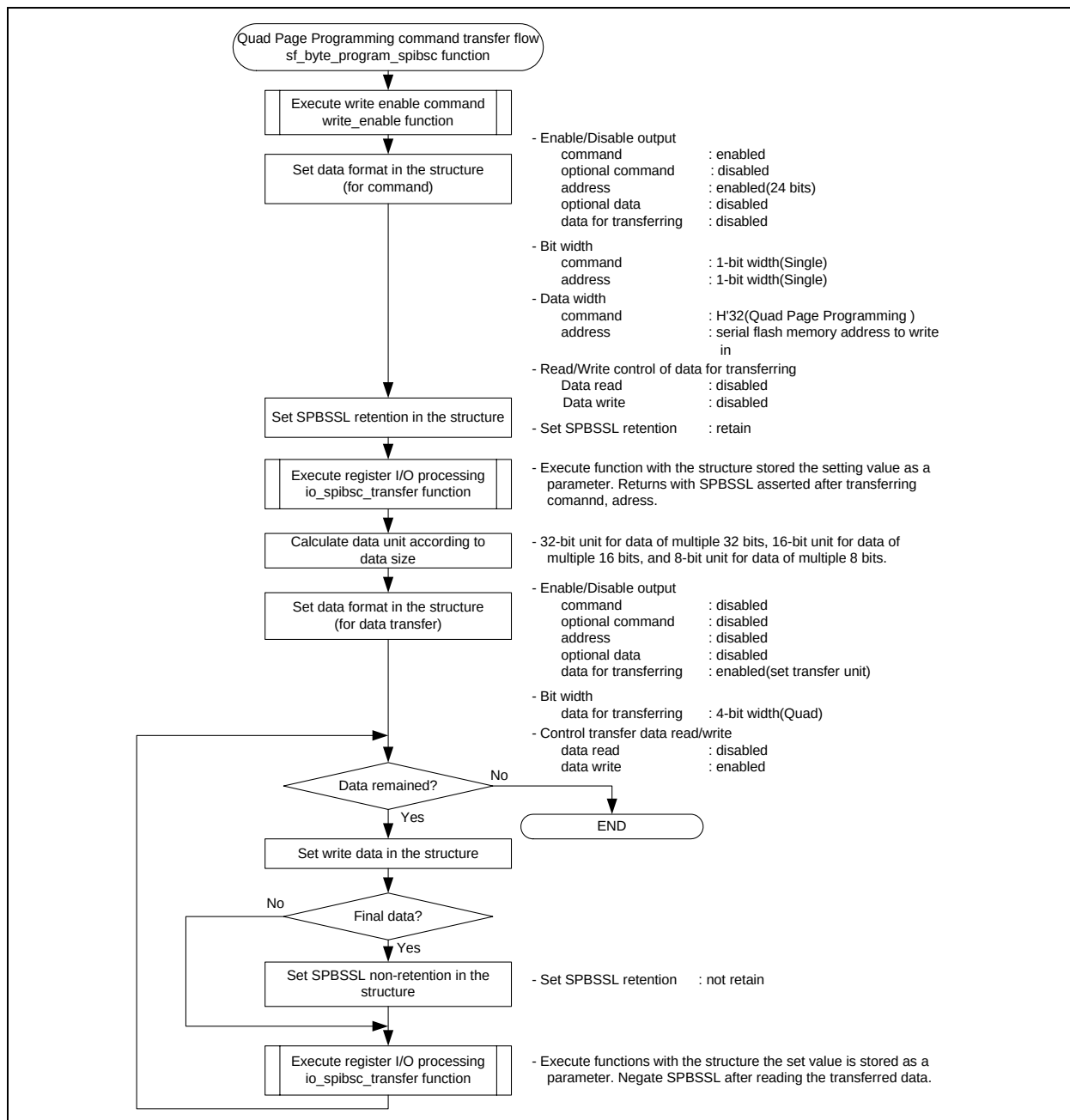


Figure 9 Flow Chart of Write Command Transfer in SPI Operation Mode

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.

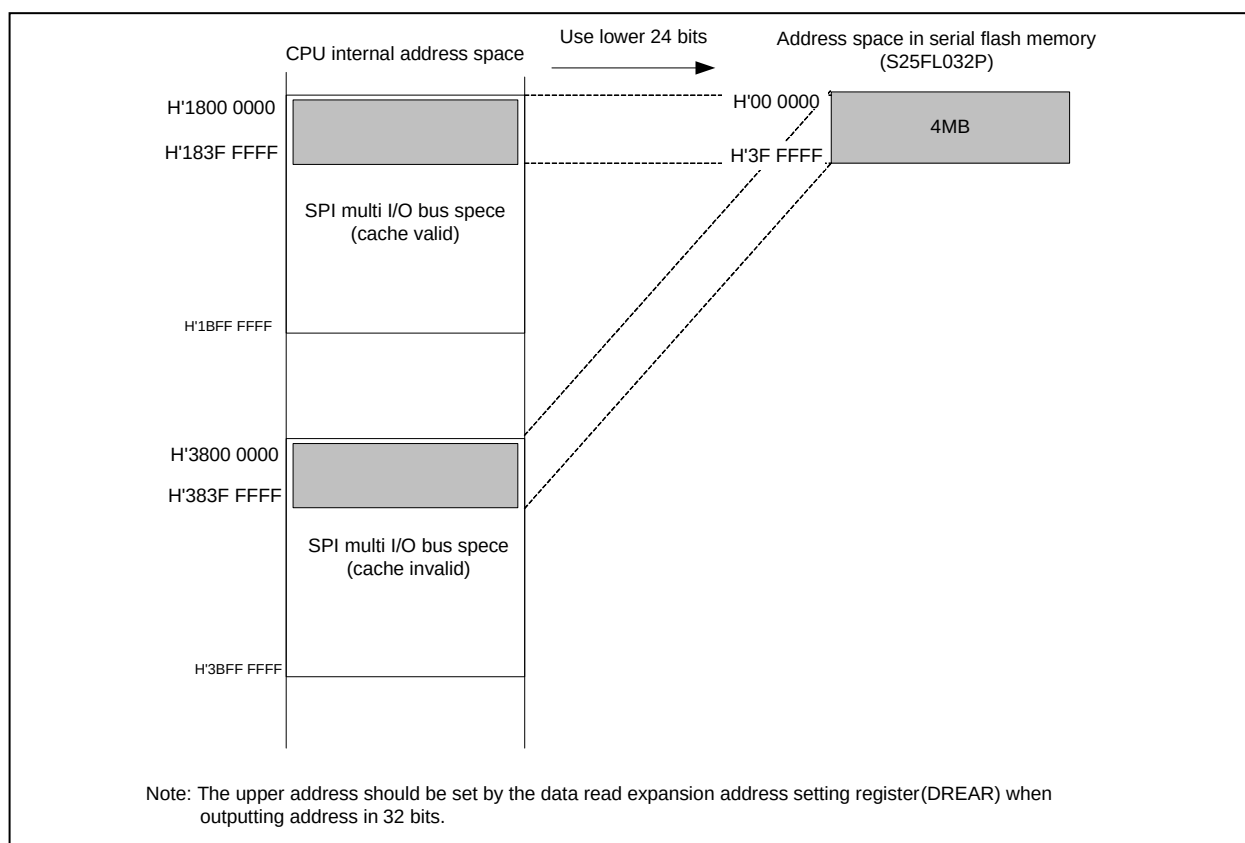


Figure 10 Address Conversion Image in External Address Space Read Mode

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)	DREN.R.CDB[1:0] bit	DREN.R.OCDB [1:0] bit	DREN.R.ADB[1:0] bit	DREN.R.OPDB[1:0] bit	DREN.R.DRDB[1:0] bit
Enable data input	Output always				Input always
Enable transfer	DREN.R.CDE bit	DREN.R.OCDE bit	DREN.R.ADE[3:0] bit (set bit length as well)	DREN.R.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.

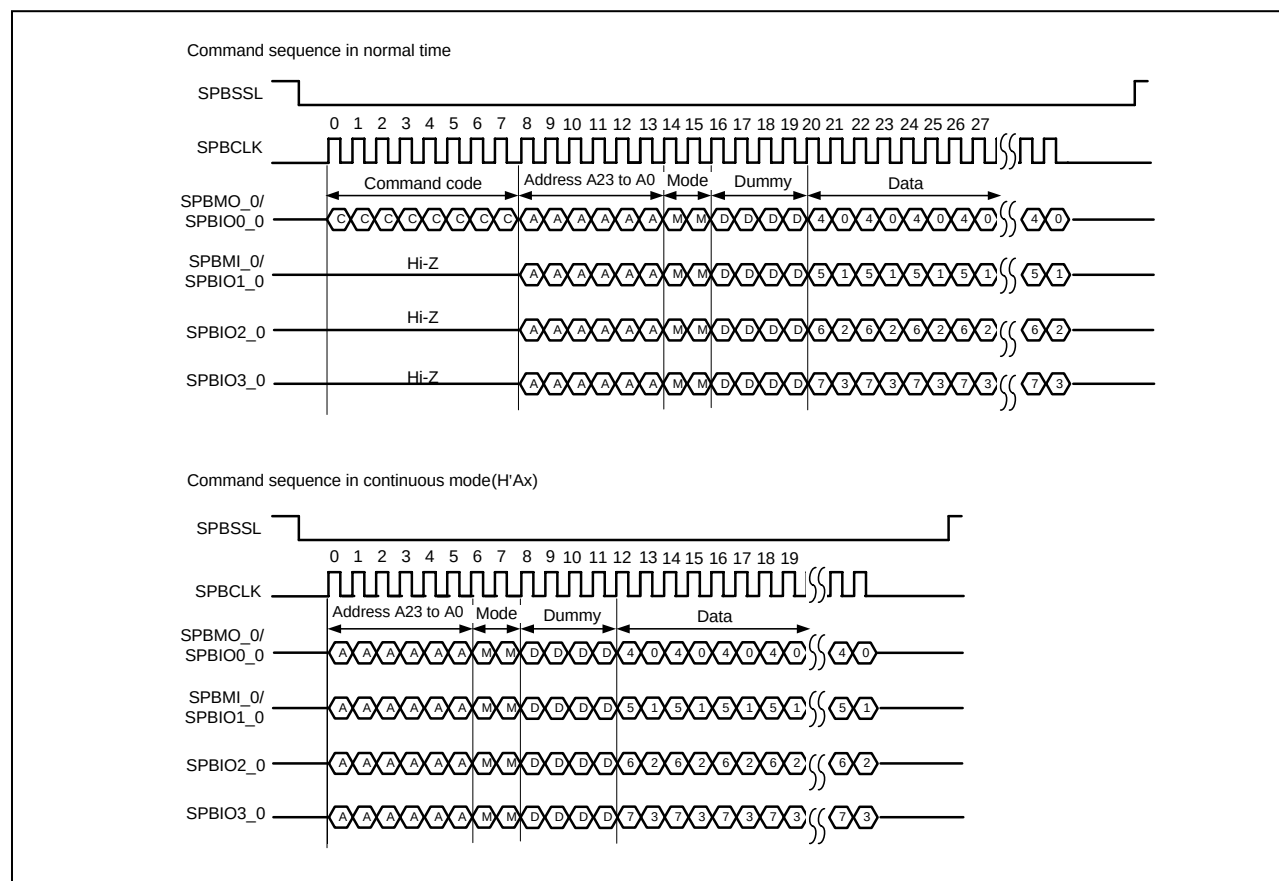


Figure 11 Quad I/O high Performance Read Command Sequence

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 read 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.

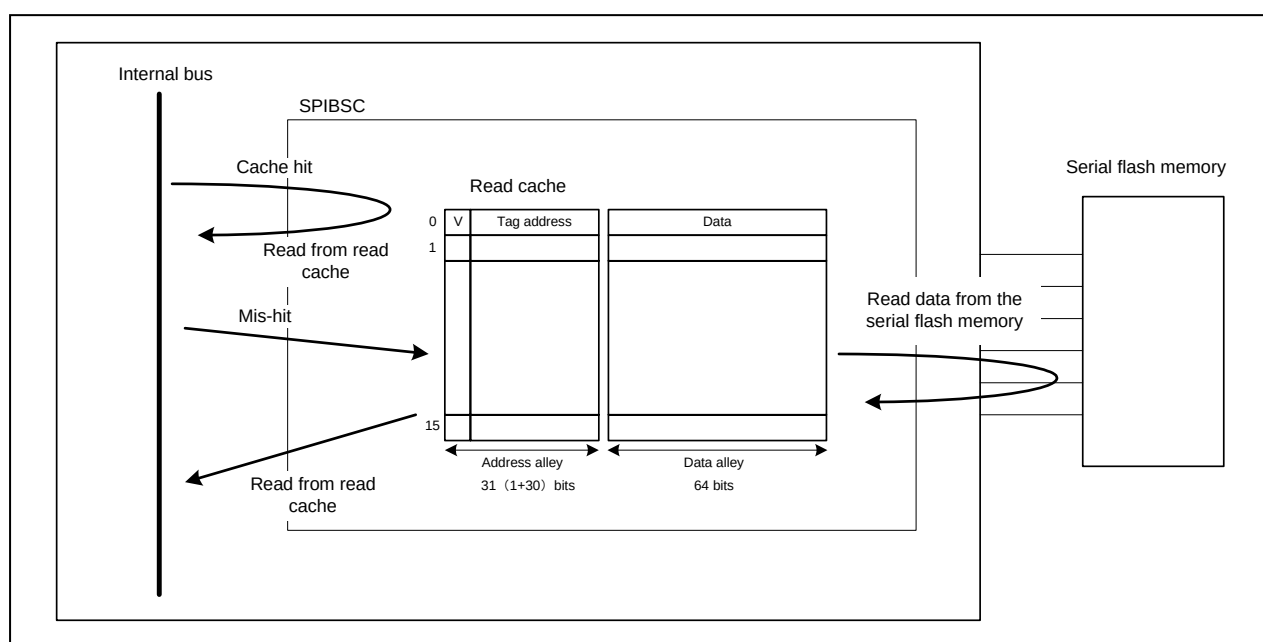


Figure 12 Operation of Burst Read and Read Cache

(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.

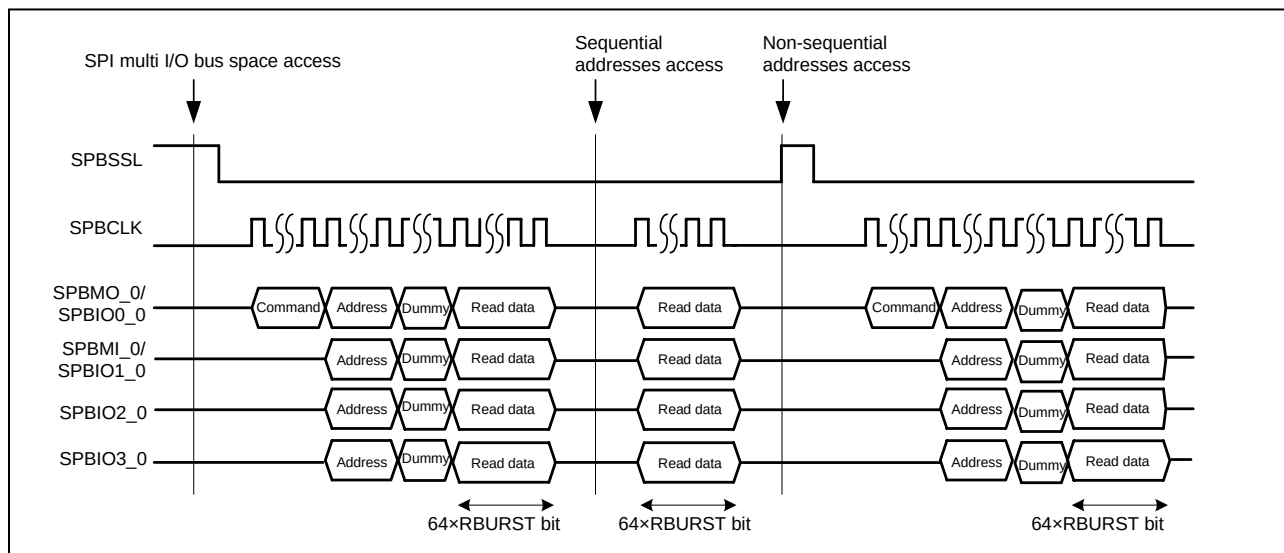


Figure 13 SPBSSL Automatic Negation in Burst Read Operation

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.

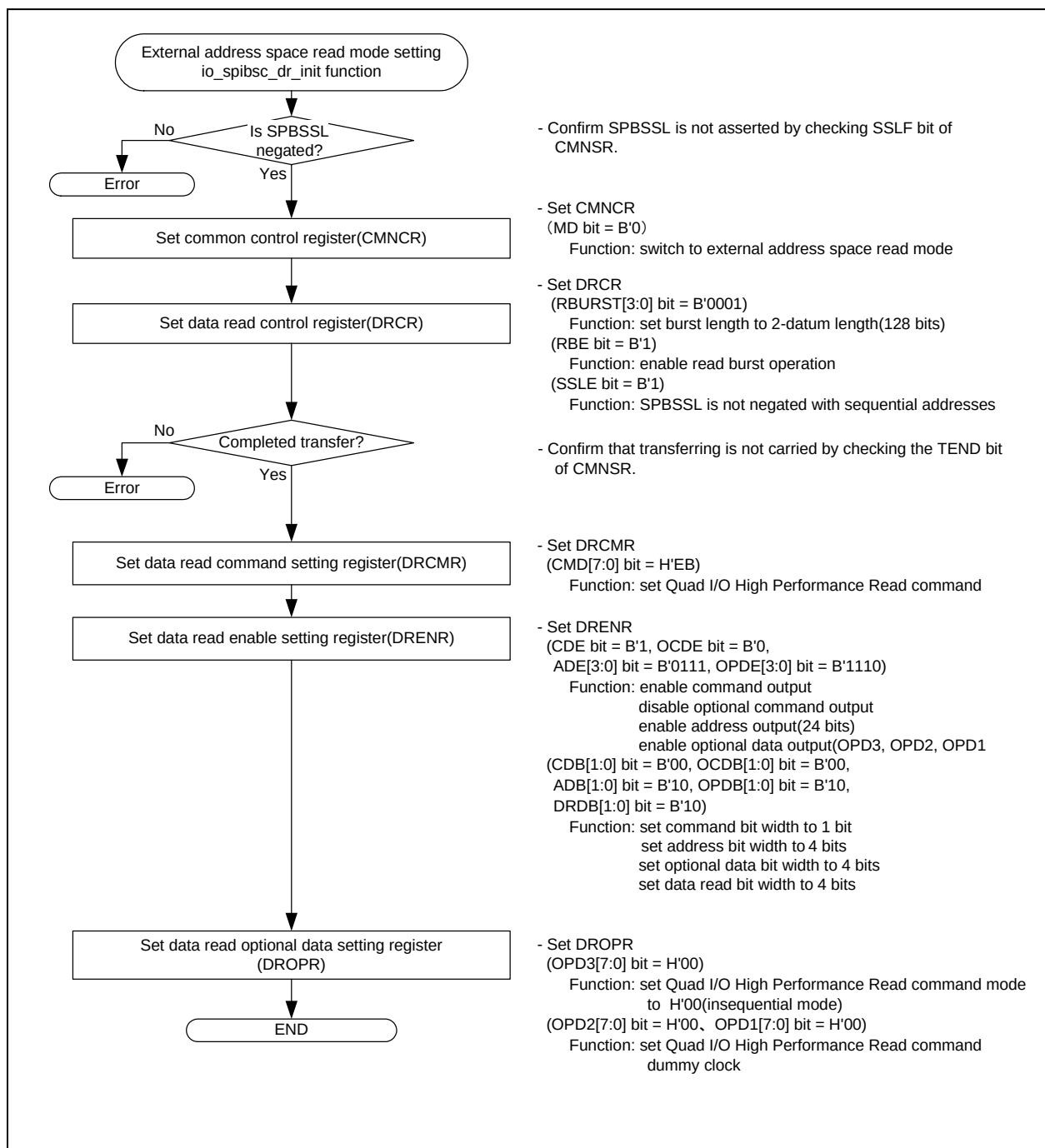


Figure 14 External Address Space Read Mode Setting Flow Chart

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.

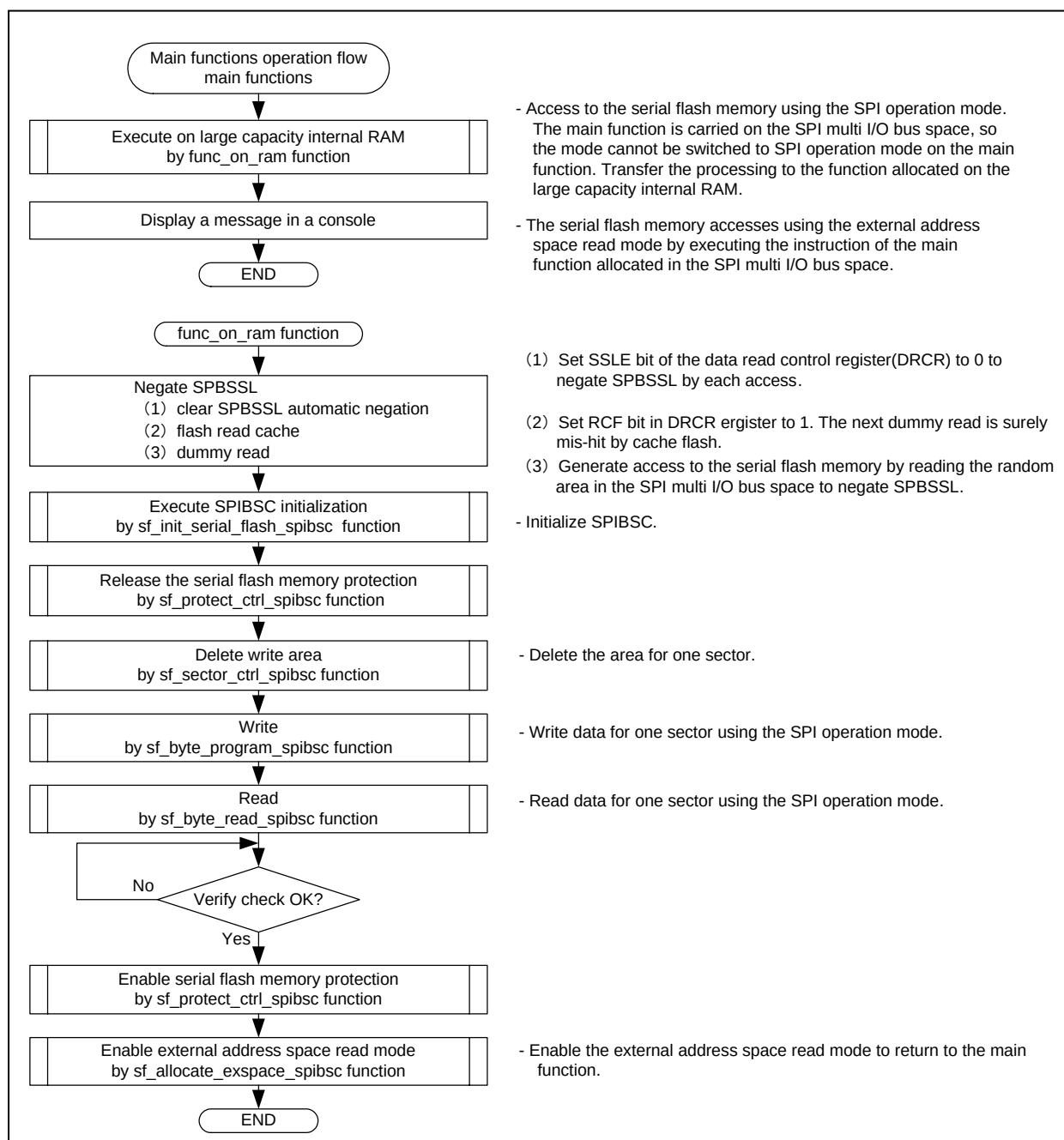


Figure 15 Main Function Operation Flow in Sample Program

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)

```

1  /*****
2  *   DISCLAIMER
3  *
4  *   This software is supplied by Renesas Electronics Corporation and is only
5  *   intended for use with Renesas products. No other uses are authorized.
6  *
7  *   This software is owned by Renesas Electronics Corporation and is protected under
8  *   all applicable laws, including copyright laws.
9  *
10 *   THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
11 *   REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
12 *   INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
13 *   PARTICULAR PURPOSE AND NON-INFRINGEMENT.  ALL SUCH WARRANTIES ARE EXPRESSLY
14 *   DISCLAIMED.
15 *
16 *   TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
17 *   ELECTRONICS CORPORATION NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
18 *   FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
19 *   FOR ANY REASON RELATED TO THIS SOFTWARE, EVEN IF RENESAS OR ITS
20 *   AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
21 *
22 *   Renesas reserves the right, without notice, to make changes to this
23 *   software and to discontinue the availability of this software.
24 *   By using this software, you agree to the additional terms and
25 *   conditions found by accessing the following link:
26 *   http://www.renesas.com/disclaimer
27 *****/
28 *   Copyright (C) 2011 Renesas Electronics Corporation. All rights reserved.
29 *****/
30 *   System Name : SH7268/SH7269 Sample Program
31 *   File Name   : main.c
32 *   Abstract    : Sample Program Main
33 *   Version     : 1.00.00
34 *   Device      : SH7268/SH7269
35 *   Tool-Chain  : High-performance Embedded Workshop (Ver.4.07.00).
36 *               : C/C++ compiler package for the SuperH RISC engine family
37 *               :                               (Ver.9.03Release02).
38 *   OS          : None
39 *   H/W Platform: R0K57269(CPU board)
40 *   Description :
41 *****/
42 *   History     : Jul.06,2011 Ver.1.00.00
43 *****/
44 #include <stdio.h>
45 #include <string.h>
46 #include <machine.h>
47 #include "serial_flash.h"
48 #include "iodefine.h"
49

```

3.3 Sample Program List “main.c” (2)

```

50  /* ==== prototype declaration ==== */
51  void main(void);
52  void func_on_ram(void);
53
54  /*****
55   * ID          :
56   * Outline     : main
57   * Include     :
58   * Declaration : void main(void);
59   * Description :
60   * Argument    : void
61   * Return Value: void
62   * Note       : None
63   *****/
64  void main(void)
65  {
66      func_on_ram();
67
68      puts("\nSH7269 SPIBSC Sample Program. Ver.1.00.00");
69      puts("Copyright (C) 2011 Renesas Electronics Corporation. All rights reserved.");
70      puts("\n");
71
72      while(1){
73          /* loop */
74      }
75  }
76
77
78  #pragma section SPIBSC
79  /*****
80   * ID          :
81   * Outline     : SPI operating mode
82   * Include     :
83   * Declaration : void exe_spibsc_spi(void) ;
84   * Description :
85   * Argument    : void
86   * Return Value: void
87   * Note       : None
88   *****/
89  void func_on_ram(void)
90  {
91      volatile short dummy;
92      int w_size  = SF_PAGE_SIZE;
93      int w_sctno = (SF_NUM_OF_SECTOR - 1);
94      int w_addr, bsz, i;
95      static char r_data[ SF_PAGE_SIZE ];
96      static char w_data[ SF_PAGE_SIZE ];
97

```


3.4 Sample Program List “main.c” (3)

```
98      /* ==== Use SPI operating mode ==== */
99
100     /* Initialize data */
101     for(i=0; i<w_size; i++){
102         r_data[i] = 'R';
103         w_data[i] = 'W';
104     }
105     bsz = 1;
106     w_addr = (w_sctno * SF_SECTOR_SIZE * bsz);
107
108     /* Negate SPBSSL */
109     SPIBSC.DRCR.BIT.SSLE = 0;          /* No keep SSL */
110     SPIBSC.DRCR.BIT.RCF = 1;          /* Chach flush */
111     dummy = *(short *)0x18000000;     /* Dummy read */
112
113     /* Initializes the SPIBSC */
114     sf_init_serial_flash_spibsc();
115
116     /* Disables the software protection in serial flash memory */
117     sf_protect_ctrl_spibsc(SF_REQ_UNPROTECT);
118
119     /* Erase */
120     sf_sector_erase_spibsc(w_sctno);
121
122     /* Write */
123     sf_byte_program_spibsc(w_addr, w_data, w_size );
124
125     /* Read */
126     sf_byte_read_spibsc(w_addr, r_data, w_size);
127
128     /* Verifies data */
129     for(i=0; i<w_size; i++){
130         if( r_data[i] != w_data[i] ){
131             while(1){
132                 /* error */
133             }
134         }
135     }
136     /* Enables the software protection in serial flash memory */
137     sf_protect_ctrl_spibsc(SF_REQ_PROTECT);
138
139
140     /* ==== Enable external address space read mode ==== */
141     sf_allocate_exspace_spibsc();
142
143 }
144
145 /* End of File */
```

3.5 Sample Program List "qserial_flash_spibsc.c" (1)

```

1  /*****
2  *   DISCLAIMER
3  *
4  *   This software is supplied by Renesas Electronics Corporation and is only
5  *   intended for use with Renesas products. No other uses are authorized.
6  *
7  *   This software is owned by Renesas Electronics Corporation and is protected under
8  *   all applicable laws, including copyright laws.
9  *
10 *   THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
11 *   REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
12 *   INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
13 *   PARTICULAR PURPOSE AND NON-INFRINGEMENT. ALL SUCH WARRANTIES ARE EXPRESSLY
14 *   DISCLAIMED.
15 *
16 *   TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
17 *   ELECTRONICS CORPORATION NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
18 *   FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
19 *   FOR ANY REASON RELATED TO THIS SOFTWARE, EVEN IF RENESAS OR ITS
20 *   AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
21 *
22 *   Renesas reserves the right, without notice, to make changes to this
23 *   software and to discontinue the availability of this software.
24 *   By using this software, you agree to the additional terms and
25 *   conditions found by accessing the following link:
26 *   http://www.renesas.com/disclaimer
27 *****/
28 *   Copyright (C) 2011 Renesas Electronics Corporation. All rights reserved.
29 *****/
30 *   System Name : SH7268/SH7269 Firm Update Sample Program
31 *   File Name  : qserial_flash_spibsc.c
32 *   Abstract   :
33 *   Version    : 1.00.00
34 *   Device     : SH7268/SH7269
35 *   Tool-Chain : High-performance Embedded Workshop (Ver.4.07.00).
36 *              : C/C++ compiler package for the SuperH RISC engine family
37 *              :                      (Ver.9.03Release02).
38 *   OS         : None
39 *   H/W Platform: R0K57269(CPU board)
40 *   Description :
41 *****/
42 *   History    : Jul.06,2011 Ver.1.00.00
43 *****/
44 #include "io_spibsc.h"
45 #include "serial_flash.h"
46 #include "qserial_flash_spibsc.h"
47

```

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

```

48  #pragma section SPIBSC
49
50  /* ---- serial flash command[S25FL032P(Spansion)] ---- */
51  #define SFLASHCMD_CHIP_ERASE  0xc7
52  #define SFLASHCMD_SECTOR_ERASE 0xd8
53  #define SFLASHCMD_BYTE_PROGRAM 0x02
54  #define SFLASHCMD_BYTE_READ    0x0B    /* fast read    */
55  #define SFLASHCMD_DUAL_READ    0x3B
56  #define SFLASHCMD_QUAD_READ    0x6B
57  #define SFLASHCMD_DUAL_IO_READ 0xBB
58  #define SFLASHCMD_QUAD_IO_READ 0xEB
59  #define SFLASHCMD_WRITE_ENABLE 0x06
60  #define SFLASHCMD_READ_STATUS  0x05
61  #define SFLASHCMD_READ_CONFIG  0x35
62  #define SFLASHCMD_WRITE_STATUS 0x01
63  #define SFLASHCMD_QUAD_PROGRAM 0x32
64  /* ---- serial flash register definitions ---- */
65  #define CFREG_QUAD_BIT          0x02    /* Quad mode bit(Configuration Register) */
66  #define CFREG_FREEZE_BIT        0x01    /* freeze bit(Configuration Register) */
67  #define STREG_BPROTECT_BIT      0x1c    /* protect bit(Status Register) */
68
69  /* ==== Prototype Declaration ==== */
70  static void read_status(unsigned char* status1,unsigned char* status2);
71  static void read_config(unsigned char* config1,unsigned char* config2);
72  static void busy_wait(void);
73  static void write_status(unsigned char status, unsigned char config);
74  static void sf_set_mode(enum sf_req req);
75  static void write_enable(void);
76  #if (SPI_QUAD != 0)
77  static void sf_byte_read_spibsc_quad(unsigned long addr, unsigned char *buf, int unit);
78  #endif
79
80  /* ==== Global variable ==== */
81  ST_SPIBSC_SM SpibscSm;
82
83
84
85  (The rest is omitted)
86

```

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

```

127  /*****
128  * ID          :
129  * Outline      : External address space read mode
130  * Include      :
131  * Declaration  : void sf_allocate_exspace_spibsc (void);
132  * Description  : Set to the external address space read mode
133  * Argument     : void
134  * Return Value : void
135  * Note        : None
136  *****/
137  void sf_allocate_exspace_spibsc (void)
138  {
139  #if (SFLASH_DUAL == 0)
140      sf_bsz_set_spibsc(1);          /* s-flash x 1 */
141  #else
142      sf_bsz_set_spibsc(2);          /* s-flash x 2 */
143  #endif
144
145  #if (SPI_QUAD == 0)
146      io_spibsc_dr_init(SFLASHCMD_BYTE_READ); /* Single-SPI */
147  #else
148      io_spibsc_dr_init(SFLASHCMD_QUAD_IO_READ); /* Quad-SPI */
149  #endif
150  }
151
152  /*****
153  * ID          :
154  * Outline      : Initialize the serial flash memory
155  * Include      :
156  * Declaration  : void sf_init_serial_flash_spibsc(void);
157  * Description  : Initialize to access to the serial flash memory
158  *              : Initialize the SPI multi bus I/O bus controller(SPIBSC)
159  *              : to set the serial flash memory to Quad mode
160  * Argument     : void
161  * Return Value : void
162  * Note        : None
163  *****/
164  void sf_init_serial_flash_spibsc(void)
165  {
166      /* ==== Initialize SPIBSC ==== */
167  #if (SFLASH_DUAL == 0)
168      io_spibsc_common_init(SPIBSC_CMNCR_BSZ_SINGLE); /* s-flash x 1 */
169  #else
170      io_spibsc_common_init(SPIBSC_CMNCR_BSZ_DUAL); /* s-flash x 2 */
171  #endif
172
173  #if (SPI_QUAD == 0)
174      sf_set_mode( SF_REQ_SERIALMODE );
175  #else
176      /* ==== setting serial-flash quad mode ==== */
177      sf_set_mode( SF_REQ_QUADMODE );
178  #endif
179  }

```

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

```

180
181  /*****
182   * ID          :
183   * Outline     : Protection
184   * Include     :
185   * Declaration : void sf_protect_ctrl_spibsc(enum sf_req req);
186   * Description : Serial flash memory protect setting/clearing the setting
187   *              : Specify the setting by the argument, reg. The initial value of
188   *              : protection/clearance differ to the specification of the serial
189   *              : flash memory.
190   * Argument    : enum sf_req req ; I : SF_REQ_UNPROTECT -> clear all sector protection
191   *              :                      SF_REQ_PROTECT  -> protect all sectors
192   * Return Value : void
193   * Note        : None
194   *****/
195 void sf_protect_ctrl_spibsc(enum sf_req req)
196 {
197     unsigned char st_reg1, st_reg2;
198     unsigned char cf_reg1, cf_reg2;
199
200     read_status(&st_reg1,&st_reg2);
201     read_config(&cf_reg1,&cf_reg2);
202
203     /* ==== Set value of Serial Flash(0) ==== */
204
205     /* ---- clear freeze bit in configuration register ---- */
206     write_status( st_reg1 , (unsigned char)(cf_reg1 & (~CFREG_FREEZE_BIT)) );
207
208     if( req == SF_REQ_UNPROTECT ){
209         st_reg1 &= ~STREG_BPROTECT_BIT;    /* un-protect in all area */
210     }
211     else{
212         st_reg1 |= STREG_BPROTECT_BIT;    /* protect in all area */
213     }
214
215     /* ---- clear or set protect bit in status register ---- */
216     /* ---- with freeze bit in configuration register ---- */
217     write_status( st_reg1 , (unsigned char)(cf_reg1 | CFREG_FREEZE_BIT) );
218 }

    (The rest is omitted)

```

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

```

304  /*****
305  * ID      :
306  * Outline : Sector erase
307  * Include :
308  * Declaration : void sf_sector_erase_spibsc(int sector_no);
309  * Description : Erase the specified sector in the serial flash memory
310  *             : A write enable command should be issued before erasing or programming.
311  *             : After erasing or programming, check the serial flash memory status
312  *             : with the busy state is cleared.
313  * Argument  : int sector_no ; I :sector number
314  * Return Value : void
315  * Note      : None
316  *****/
317  void sf_sector_erase_spibsc(int sector_no)
318  {
319      unsigned long addr = sector_no * SF_SECTOR_SIZE;
320
321      #if (SFLASH_DUAL == 1)
322          int bsz;
323
324          /* set BE in both of serial-flash */
325          bsz = sf_bsz_get_spibsc();
326          sf_bsz_set_spibsc(2);          /* s-flash x 2 */
327      #endif
328
329      /* sector erase in Single-SPI */
330
331      write_enable();          /* WREN Command */
332
333      SpibscSm.cdb = SPIBSC_1BIT;          /* Command bit-width = Single */
334      SpibscSm.adb = SPIBSC_1BIT;          /* Address bit-width = Single */
335
336      SpibscSm.cde = SPIBSC_OUTPUT_ENABLE; /* Command Enable */
337      SpibscSm.ocde = SPIBSC_OUTPUT_DISABLE; /* Optional-Command Disable */
338      SpibscSm.ade = SPIBSC_OUTPUT_ADDR_24; /* Enable(Adr[23:0]) */
339      SpibscSm.opde = SPIBSC_OUTPUT_DISABLE; /* Option-Data Disable */
340      SpibscSm.spide = SPIBSC_OUTPUT_DISABLE; /* Disable */
341
342      SpibscSm.sslkp = SPIBSC_SPISSL_NEGATE; /* Negate after transfer */
343      SpibscSm.spire = SPIBSC_SPIDATA_DISABLE; /* Data Access (Read Disable) */
344      SpibscSm.spiwe = SPIBSC_SPIDATA_DISABLE; /* Data Access (Write Disable) */
345
346      SpibscSm.cmd = SFLASHCMD_SECTOR_ERASE; /* SE:Sector Erase */
347
348      SpibscSm.addr = addr;          /* dont care in dual mode */
349                                     /* because address is calcurated with sector_no */
350

```

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

```

351     io_spibsc_transfer(&SpibscSm);
352
353     busy_wait();
354
355     #if (SFLASH_DUAL == 1)
356         sf_bsz_set_spibsc(bsz);
357     #endif
358 }
359
360 /*****
361  * ID          :
362  * Outline     : Data program
363  * Include     :
364  * Declaration: void sf_byte_program_spibsc(unsigned long addr, unsigned char *buf, int size);
365  * Description : Program the assigned program in the serial flash memory
366  *              : Erase the specified sector in the serial flash memory
367  *              : A write enable command should be issued before erasing or programming.
368  *              : After erasing or programming, check the serial flash memory status
369  *              : with the busy state is cleared.
370  *              : The maximum write data size is limited by the device.
371  * Argument    : unsigned long addr ; I : address in the serial flash memory to write to
372  *              : unsigned char *buf ; I : address of the buffer to store write data
373  *              : int size           ; I : number of byte to write
374  * Return Value: void
375  * Note        : None
376  *****/
377 void sf_byte_program_spibsc(unsigned long addr, unsigned char *buf, int size)
378 {
379     int unit;
380
381     write_enable();                /* WREN Command */
382
383     /* ---- Command, Address ---- */
384     SpibscSm.cdb = SPIBSC_1BIT;    /* Command bit-width = Single */
385     SpibscSm.adb = SPIBSC_1BIT;    /* Address bit-width = Single */
386
387     SpibscSm.cde = SPIBSC_OUTPUT_ENABLE; /* Command Enable */
388     SpibscSm.ocde = SPIBSC_OUTPUT_DISABLE; /* Optional-Command Disable */
389     SpibscSm.ade = SPIBSC_OUTPUT_ADDR_24; /* Enable Adr[23:0] */
390     SpibscSm.opde = SPIBSC_OUTPUT_DISABLE; /* Option-Data Disable */
391     SpibscSm.spide = SPIBSC_OUTPUT_DISABLE; /* Disable */
392
393     SpibscSm.sslkp = SPIBSC_SPISSL_KEEP; /* Keep after transfer */
394     SpibscSm.spire = SPIBSC_SPIDATA_DISABLE; /* Data Access (Read Disable) */
395     SpibscSm.spiwe = SPIBSC_SPIDATA_DISABLE; /* Data Access (Write Disable) */
396
397     #if (SPI_QUAD == 0)
398         SpibscSm.cmd = SFLASHCMD_BYTE_PROGRAM; /* PP: Page Program */
399     #else
400         SpibscSm.cmd = SFLASHCMD_QUAD_PROGRAM; /* QPP: Quad Page Program */
401     #endif
402

```

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

```

403     if(io_spibsc_bsz_get() == SPIBSC_CMNCR_BSZ_DUAL){
404         SpibscSm.addr = (unsigned long)(addr >> 1);
405     }
406     else{
407         SpibscSm.addr = addr;
408     }
409
410     io_spibsc_transfer(&SpibscSm);           /* Command,Address */
411
412     /* ---- Data ---- */
413     #if (SPI_QUAD == 0)
414         SpibscSm.spidb = SPIBSC_1BIT;        /* Single */
415     #else
416         SpibscSm.spidb = SPIBSC_4BIT;        /* Quad */
417     #endif
418
419     SpibscSm.cde = SPIBSC_OUTPUT_DISABLE;    /* Command Disable */
420     SpibscSm.ocde = SPIBSC_OUTPUT_DISABLE;   /* Optional-Command Disable */
421     SpibscSm.ade = SPIBSC_OUTPUT_DISABLE;    /* Disable Adr */
422     SpibscSm.opde = SPIBSC_OUTPUT_DISABLE;   /* Option-Data Disable */
423
424     SpibscSm.spire = SPIBSC_SPIDATA_DISABLE; /* Data Access (Read Disable) */
425     SpibscSm.spiwe = SPIBSC_SPIDATA_ENABLE;  /* Data Access (Write Enable) */
426
427     if(io_spibsc_bsz_get() == SPIBSC_CMNCR_BSZ_DUAL){
428         if((size % 8) == 0){
429             SpibscSm.spide = SPIBSC_OUTPUT_SPID_32; /* Enable(64bit) */
430             unit = 8;
431         }
432         else if((size % 4) == 0){
433             SpibscSm.spide = SPIBSC_OUTPUT_SPID_16; /* Enable(32bit) */
434             unit = 4;
435         }
436         else if((size % 2) == 0){
437             SpibscSm.spide = SPIBSC_OUTPUT_SPID_8;  /* Enable(16bit) */
438             unit = 2;
439         }
440         else{
441             return;
442         }
443     }

```


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

```

444     else{
445         if((size % 4) == 0){
446             SpibscSm.spide = SPIBSC_OUTPUT_SPID_32; /* Enable(32bit) */
447             unit = 4;
448         }
449         else if((size % 2) == 0){
450             SpibscSm.spide = SPIBSC_OUTPUT_SPID_16; /* Enable(16bit) */
451             unit = 2;
452         }
453         else{
454             SpibscSm.spide = SPIBSC_OUTPUT_SPID_8; /* Enable(8bit) */
455             unit = 1;
456         }
457     }
458
459     while(size > 0){
460         SpibscSm.smwdr[0] = (unsigned long)((((unsigned long)*buf++ << 24) & 0xff000000ul);
461                                     /* Data[63:56] or Data[31:24] */
462         if(unit >= 2){
463             SpibscSm.smwdr[0]|=(unsigned long)((((unsigned long)*buf++ << 16)& 0x00ff0000ul);
464                                     /* Data[55:48] or Data[23:16] */
465         }
466         if(unit >= 4){
467             SpibscSm.smwdr[0] |= (unsigned long)(
468                                     (((unsigned long)*buf++ << 8 ) & 0x0000ff00ul) |
469                                     (((unsigned long)*buf++      ) & 0x000000fful));
470                                     /* Data[47:40] or Data[15:0] */
471         }
472         if(unit >= 8){
473             SpibscSm.smwdr[1] = (unsigned long)(
474                                     (((unsigned long)*buf++ << 24) & 0xff000000ul) |
475                                     (((unsigned long)*buf++ << 16) & 0x00ff0000ul) |
476                                     (((unsigned long)*buf++ << 8 ) & 0x0000ff00ul) |
477                                     (((unsigned long)*buf++      ) & 0x000000fful));
478                                     /*Data[31: 0] or nothing */
479         }
480
481         size -= unit;
482         if(size <= 0){
483             SpibscSm.sslkp = SPIBSC_SPISSL_NEGATE;
484         }
485         io_spibsc_transfer(&SpibscSm); /* Data */
486     }
487
488     busy_wait();
489
490 }
491

```

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

```

492  /*****
493  * ID      :
494  * Outline : Data read
495  * Include :
496  *Declaration: void sf_byte_read_spibsc(unsigned long addr, unsigned char *buf, int size);
497  * Description : Read the serial memory by the specified number of byte
498  * Argument    : unsigned long addr ; I : address of the serial flash memory to read
499  *              : unsigned char *buf ; I : address of the buffer to store the read data
500  *              : int size ; I : number of byte to read
501  * Return Value : void
502  * Note        : None
503  *****/
504  #if (SPI_QUAD == 0)

        (Omitted)

606  #else
607
608  void sf_byte_read_spibsc(unsigned long addr, unsigned char *buf, int size)
609  {
610      int unit;
611
612      if(io_spibsc_bsz_get() == SPIBSC_CMNCR_BSZ_DUAL){
613          if((size % 8) == 0){
614              unit = 8;
615          }
616          else if((size % 4) == 0){
617              unit = 4;
618          }
619          else if((size % 2) == 0){
620              unit = 2;
621          }
622          else{
623              return;
624          }
625      }
626      else{
627          if((size % 4) == 0){
628              unit = 4;
629          }
630          else if((size % 2) == 0){
631              unit = 2;
632          }
633          else{
634              unit = 1;
635          }
636      }
637

```

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

```

638     while(size > 0){
639         sf_byte_read_spibsc_quad(addr, buf, unit);
640
641         /* increment address and buf */
642         addr += unit;
643         buf += unit;
644
645         size -= unit;
646     }
647 }
648
649 static void sf_byte_read_spibsc_quad(unsigned long addr, unsigned char *buf, int unit)
650 {
651     /* ---- Command,Address,Dummy ---- */
652     SpibscSm.cdb = SPIBSC_1BIT;          /* Command bit-width = Single */
653     SpibscSm.adb = SPIBSC_1BIT;          /* Address bit-width = Single */
654
655     SpibscSm.cde = SPIBSC_OUTPUT_ENABLE; /* Command Enable */
656     SpibscSm.ocde = SPIBSC_OUTPUT_DISABLE; /* Optional-Command Disable */
657     SpibscSm.ade = SPIBSC_OUTPUT_ADDR_24; /* Enable Adr[23:0] */
658     SpibscSm.opde = SPIBSC_OUTPUT_OPD_3; /* Option-Data OPD3 */
659     SpibscSm.spide = SPIBSC_OUTPUT_DISABLE; /* Disable */
660
661     SpibscSm.sslkp = SPIBSC_SPISSL_KEEP; /* Keep after transfer */
662     SpibscSm.spire = SPIBSC_SPIDATA_DISABLE; /* Data Access (Read Disable) */
663     SpibscSm.spiwe = SPIBSC_SPIDATA_DISABLE; /* Data Access (Write Disable) */
664
665     SpibscSm.cmd = SFLASHCMD_QUAD_READ; /* QOR: Quad Output Read */
666
667     if(io_spibsc_bsz_get() == SPIBSC_CMNCR_BSZ_DUAL){
668         SpibscSm.addr = (unsigned long)(addr >> 1);
669     }
670     else{
671         SpibscSm.addr = addr;
672     }
673
674     io_spibsc_transfer(&SpibscSm); /* Command,Address */
675
676
677     /* ---- Data ---- */
678     SpibscSm.spidb = SPIBSC_4BIT; /* Quad */
679
680     SpibscSm.cde = SPIBSC_OUTPUT_DISABLE; /* Command Disable */
681     SpibscSm.ocde = SPIBSC_OUTPUT_DISABLE; /* Optional-Command Disable */
682     SpibscSm.ade = SPIBSC_OUTPUT_DISABLE; /* Disable Adr */
683     SpibscSm.opde = SPIBSC_OUTPUT_DISABLE; /* Option-Data Disable */
684
685     SpibscSm.spire = SPIBSC_SPIDATA_ENABLE; /* Data Access (Read Enable) */
686     SpibscSm.spiwe = SPIBSC_SPIDATA_DISABLE; /* Data Access (Write Disable) */
687

```

3.15 Sample Program List "qserial_flash_spibsc.c" (11)

```

688     if(io_spibsc_bsz_get() == SPIBSC_CMNCR_BSZ_DUAL){
689         if( unit == 8 ){
690             SpibscSm.spide = SPIBSC_OUTPUT_SPID_32; /* Enable(64bit) */
691         }
692         else if( unit == 4 ){
693             SpibscSm.spide = SPIBSC_OUTPUT_SPID_16; /* Enable(32bit) */
694         }
695         else if( unit == 2 ){
696             SpibscSm.spide = SPIBSC_OUTPUT_SPID_8; /* Enable(16bit) */
697         }
698         else{
699             return;
700         }
701     }
702     else{
703         if( unit == 4 ){
704             SpibscSm.spide = SPIBSC_OUTPUT_SPID_32; /* Enable(32bit) */
705         }
706         else if( unit == 2 ){
707             SpibscSm.spide = SPIBSC_OUTPUT_SPID_16; /* Enable(16bit) */
708         }
709         else{
710             SpibscSm.spide = SPIBSC_OUTPUT_SPID_8; /* Enable(8bit) */
711         }
712     }
713
714     SpibscSm.sslkp = SPIBSC_SPISSL_NEGATE;
715     io_spibsc_transfer(&SpibscSm); /* Data input */
716
717     *buf++ = (unsigned char)((SpibscSm.smrdr[0] >> 24) & 0x000000fful);
718                                     /* Data[63:56],Data[31:24] */
719     if(unit >= 2){
720         *buf++ = (unsigned char)((SpibscSm.smrdr[0] >> 16) & 0x000000fful);
721                                     /* Data[55:48],Data[23:16] */
722     }
723     if(unit >= 4){
724         *buf++ = (unsigned char)((SpibscSm.smrdr[0] >> 8 ) & 0x000000fful);
725         *buf++ = (unsigned char)((SpibscSm.smrdr[0]      ) & 0x000000fful);
726                                     /* Data[47:40],Data[15:0] */
727     }
728     if(unit >= 8){
729         *buf++ = (unsigned char)((SpibscSm.smrdr[1] >> 24) & 0x000000fful);
730         *buf++ = (unsigned char)((SpibscSm.smrdr[1] >> 16) & 0x000000fful);
731         *buf++ = (unsigned char)((SpibscSm.smrdr[1] >> 8 ) & 0x000000fful);
732         *buf++ = (unsigned char)((SpibscSm.smrdr[1]      ) & 0x000000fful);
733                                     /*Data[31:0] */
734     }
735 }
736 #endif
737

```

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

```

      (Omitted)

860  /*****
861   * ID          :
862   * Outline     : Enable writing
863   * Include     :
864   * Declaration : static void write_enable(void);
865   * Description : Issuing the write enable command to permit to erase/program
866   *             : in the serial flash memory
867   * Argument    : void
868   * Return Value: void
869   * Note        : None
870   *****/
871  static void write_enable(void)
872  {
873      SpibscSm.cdb = SPIBSC_1BIT;          /* Single */
874
875      SpibscSm.cde = SPIBSC_OUTPUT_ENABLE; /* Command Enable */
876      SpibscSm.ocde = SPIBSC_OUTPUT_DISABLE; /* Optional-Command Disable */
877      SpibscSm.ade = SPIBSC_OUTPUT_DISABLE; /* Address Disable */
878      SpibscSm.opde = SPIBSC_OUTPUT_DISABLE; /* Option-Data Disable */
879      SpibscSm.spide = SPIBSC_OUTPUT_DISABLE; /* Disable */
880
881      SpibscSm.sslkp = SPIBSC_SPISSL_NEGATE; /* Negate after transfer */
882      SpibscSm.spire = SPIBSC_SPIDATA_DISABLE; /* Data Access (Read Disable) */
883      SpibscSm.spiwe = SPIBSC_SPIDATA_DISABLE; /* Data Access (Write Disable) */
884
885      SpibscSm.cmd = SFLASHCMD_WRITE_ENABLE; /* WREN:Write Enable */
886
887      io_spibsc_transfer(&SpibscSm);
888  }

      (The rest is omitted)

```

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

```

1  /*****
2  *   DISCLAIMER

   (Omitted)

27 ****
28 *   Copyright (C) 2011 Renesas Electronics Corporation. All rights reserved.
29 **** Technical reference data ****
30 *   System Name : SH7268/SH7269 Firm Update Sample Program
31 *   File Name   : qserial_flash_spibsc.h
32 *   Abstract    :
33 *   Version     : 1.00.00
34 *   Device      : SH7268/SH7269
35 *   Tool-Chain  : High-performance Embedded Workshop (Ver.4.07.00).
36 *               : C/C++ compiler package for the SuperH RISC engine family
37 *               :                               (Ver.9.03Release02).
38 *   OS          : None
39 *   H/W Platform: R0K57269(CPU board)
40 *   Description :
41 ****
42 *   History     : Jul.06,2011 Ver.1.00.00
43 ****/
44 #ifndef _QSERIAL_FLASH_SPIBSC_H_
45 #define _QSERIAL_FLASH_SPIBSC_H_
46
47 /* ==== Function prototype declaration ==== */
48 int sf_bsz_get_spibsc(void);
49 void sf_bsz_set_spibsc(int bsz);
50 void sf_allocate_exspace_spibsc (void);
51 void sf_init_serial_flash_spibsc(void);
52 void sf_protect_ctrl_spibsc(enum sf_req req);
53 void sf_chip_erase_spibsc(void);
54 void sf_sector_erase_spibsc(int sector_no);
55 void sf_byte_program_spibsc(unsigned long addr, unsigned char *buf, int size);
56 void sf_byte_read_spibsc(unsigned long addr, unsigned char *buf, int size);
57
58 #endif /* _QSERIAL_FLASH_SPIBSC_H_ */
59 /* End of File */
60
61
62

```

3.18 Sample Program List "io_spibsc.c" (1)

```

1  /*****
2  *   DISCLAIMER
3  *
4  *   This software is supplied by Renesas Electronics Corporation and is only
5  *   intended for use with Renesas products. No other uses are authorized.
6  *
7  *   This software is owned by Renesas Electronics Corporation and is protected under
8  *   all applicable laws, including copyright laws.
9  *
10 *   THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
11 *   REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
12 *   INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
13 *   PARTICULAR PURPOSE AND NON-INFRINGEMENT.  ALL SUCH WARRANTIES ARE EXPRESSLY
14 *   DISCLAIMED.
15 *
16 *   TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
17 *   ELECTRONICS CORPORATION NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
18 *   FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
19 *   FOR ANY REASON RELATED TO THIS SOFTWARE, EVEN IF RENESAS OR ITS
20 *   AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
21 *
22 *   Renesas reserves the right, without notice, to make changes to this
23 *   software and to discontinue the availability of this software.
24 *   By using this software, you agree to the additional terms and
25 *   conditions found by accessing the following link:
26 *   http://www.renesas.com/disclaimer
27 *****/
28 *   Copyright (C) 2011 Renesas Electronics Corporation. All rights reserved.
29 *****/
30 *   System Name : SH7268/SH7269 Firm Update Sample Program
31 *   File Name   : io_spibsc.c
32 *   Abstract    : loader program for spibsc
33 *   Version     : 1.00.00
34 *   Device      : SH7268/SH7269
35 *   Tool-Chain  : High-performance Embedded Workshop (Ver.4.07.00).
36 *               : C/C++ compiler package for the SuperH RISC engine family
37 *               :                               (Ver.9.03Release02).
38 *   OS          : None
39 *   H/W Platform: R0K57269(CPU board)
40 *   Description :
41 *****/
42 *   History     : Jul.06,2011 Ver.1.00.00
43 *****/
44 #include "iodefine.h"
45 #include "io_spibsc.h"
46 #include "machine.h"
47

```

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

```

48  #pragma section SPIBSC
49
50  /* ==== define values ==== */
51
52  /* ==== Prototype Declaration ==== */
53
54  /* ==== Global variable ==== */
55
56  /*****
57   * ID          :
58   * Outline     :
59   * Include      : io_spibsc.h
60   * Declaration : int io_spibsc_bsz_set(unsigned long bsz);
61   * Description :
62   * Argument     : unsigned long bsz : BSZ bit
63   * Return Value :
64   * Note         : None
65   *****/
66  int io_spibsc_bsz_set(unsigned long bsz)
67  {
68      if(SPIBSC.CMNSR.BIT.SSLF != SPIBSC_SSL_NEGATE){
69          return -1;
70      }
71      if(SPIBSC.CMNCR.BIT.BSZ != bsz){
72          if(bsz == SPIBSC_CMNCR_BSZ_DUAL){
73              /* s-flash x 2 (4bit x 2) */
74              PORT.PBCR3.BIT.PB14MD = 6; /* PB14:SPBI03_1 */
75              PORT.PBCR3.BIT.PB13MD = 6; /* PB13:SPBI02_1 */
76              PORT.PFCR0.BIT.PF3MD = 6; /* PF3:SPBMI_1/SPBI01_1 */
77              PORT.PFCR0.BIT.PF2MD = 6; /* PF2:SPBM0_1/SPBI00_1 */
78          }
79          SPIBSC.CMNCR.BIT.BSZ = bsz;
80          SPIBSC.DRCR.BIT.RCF = SPIBSC_DRCR_RCF_EXE; /* flush read-cache */
81      }
82      return 0;
83  }
84

```


3.20 Sample Program List "io_spibsc.c" (3)

```

85  /*****
86  * ID          :
87  * Outline     :
88  * Include     : io_spibsc.h
89  * Declaration : unsigned long io_spibsc_bsz_get(void);
90  * Description  :
91  * Argument    : void
92  * Return Value : BSZ bit
93  * Note       : None
94  *****/
95  unsigned long io_spibsc_bsz_get(void)
96  {
97      return (unsigned long)SPIBSC.CMNCR.BIT.BSZ;
98  }
99
100 /*****
101 * ID          :
102 * Outline     :
103 * Include     : io_spibsc.h
104 * Declaration : int io_spibsc_common_init(unsigned long bsz);
105 * Description  :
106 * Argument    : unsigned long bsz : BSZ bit
107 * Return Value :
108 * Note       : None
109 *****/
110 int io_spibsc_common_init(unsigned long bsz)
111 {
112     CPG.STBCR7.BIT.MSTP75 = 0;
113
114     PORT.PBCR5.BIT.PB20MD = 6; /* PB20:SPBMI_0/SPBIO1_0 */
115     PORT.PBCR4.BIT.PB19MD = 6; /* PB19:SPBMO_0/SPBIO0_0 */
116     PORT.PBCR4.BIT.PB18MD = 6; /* PB18:SPBSSL */
117     PORT.PBCR4.BIT.PB17MD = 6; /* PB17:SPBCLK */
118     PORT.PBCR4.BIT.PB16MD = 6; /* PB16:SPBIO3_0 */
119     PORT.PBCR3.BIT.PB15MD = 6; /* PB15:SPBIO2_0 */
120
121     if(bsz == SPIBSC_CMNCR_BSZ_DUAL){
122         /* s-flash x 2 (4bit x 2) */
123         PORT.PBCR3.BIT.PB14MD = 6; /* PB14:SPBIO3_1 */
124         PORT.PBCR3.BIT.PB13MD = 6; /* PB13:SPBIO2_1 */
125         PORT.PFCR0.BIT.PF3MD = 6; /* PF3:SPBMI_1/SPBIO1_1 */
126         PORT.PFCR0.BIT.PF2MD = 6; /* PF2:SPBMO_1/SPBIO0_1 */
127     }
128
129     if(SPIBSC.CMNSR.BIT.SSLF != SPIBSC_SSL_NEGATE){
130         return -1;
131     }
132

```

3.21 Sample Program List "io_spibsc.c" (4)

```

133     SPIBSC.CMNCR.BIT.M0II03 = SPIBSC_OUTPUT_HiZ;
134     SPIBSC.CMNCR.BIT.M0II02 = SPIBSC_OUTPUT_HiZ;
135     SPIBSC.CMNCR.BIT.M0II01 = SPIBSC_OUTPUT_HiZ;
136     SPIBSC.CMNCR.BIT.M0II00 = SPIBSC_OUTPUT_HiZ;
137
138     SPIBSC.CMNCR.BIT.I03FV = SPIBSC_OUTPUT_HiZ;
139     SPIBSC.CMNCR.BIT.I02FV = SPIBSC_OUTPUT_HiZ;
140     SPIBSC.CMNCR.BIT.I00FV = SPIBSC_OUTPUT_HiZ;
141
142     /* S-flash mode 0 */
143     SPIBSC.CMNCR.BIT.CPHAT = SPIBSC_CMNCR_CPHAT_EVEN;
144                                     /* even edge : write */
145     SPIBSC.CMNCR.BIT.CPHAR = SPIBSC_CMNCR_CPHAR_EVEN;
146                                     /* even edge : read */
147     SPIBSC.CMNCR.BIT.SSLP = SPIBSC_CMNCR_SSLP_LOW;
148                                     /* SPBSSL : low active */
149     SPIBSC.CMNCR.BIT.CPOL = SPIBSC_CMNCR_CPOL_LOW;
150                                     /* SPBCLK : low at negate */
151
152     io_spibsc_bsz_set(bsz);
153
154     SPIBSC.SSLDR.BIT.SPNDL = SPIBSC_DELAY_1SPBCLK;
155                                     /* next access delay */
156     SPIBSC.SSLDR.BIT.SLNDL = SPIBSC_DELAY_1SPBCLK;
157                                     /* SPBSSL negate delay */
158     SPIBSC.SSLDR.BIT.SCKDL = SPIBSC_DELAY_1SPBCLK;
159                                     /* clock delay */
160
161     /* ---- Bit rate 66.67Mbps ---- */
162     SPIBSC.SPBCR.BIT.SPBR = 1;          /* divide 2 base bit rate B clock(133.33MHz) */
163     SPIBSC.SPBCR.BIT.BRDV = 0;
164
165     return 0;
166 }
167

```

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

```

168  /*****
169  * ID          :
170  * Outline     :
171  * Include     : io_spibsc.h
172  * Declaration : int io_spibsc_dr_init(unsigned long cmd);
173  * Description :
174  * Argument    : void
175  * Return Value :
176  * Note        : None
177  *****/
178  int io_spibsc_dr_init(unsigned long cmd)
179  {
180      if(SPIBSC.CMNSR.BIT.SSLF != SPIBSC_SSL_NEGATE){
181          return -1;
182      }
183
184      SPIBSC.CMNCR.BIT.MD = SPIBSC_CMNCR_MD_EXTRD;    /* SPI I/O mode*/
185
186      SPIBSC.DRCR.BIT.RBURST = SPIBSC_BURST_2;
187      SPIBSC.DRCR.BIT.RBE = SPIBSC_BURST_ENABLE;
188      SPIBSC.DRCR.BIT.SSLE = SPIBSC_SPISSL_KEEP;      /* Keep SSL after read */
189                                          /* if not continuous address it negeted */
190
191      if(SPIBSC.CMNSR.BIT.TEND != SPIBSC_TRANS_END){
192          return -1;
193      }
194
195      /* ---- Command ---- */
196      SPIBSC.DRCMR.BIT.CMD = cmd;                    /* Command */
197      SPIBSC.DREN.R.BIT.CDB = SPIBSC_1BIT;           /* Single */
198      SPIBSC.DREN.R.BIT.CDE = SPIBSC_OUTPUT_ENABLE;
199                                          /* Enable */
200
201      /* ---- Option Command ---- */
202      SPIBSC.DRCMR.BIT.OCMD = 0x00;
203      SPIBSC.DREN.R.BIT.OCDB = SPIBSC_1BIT;          /* Single */
204      SPIBSC.DREN.R.BIT.OCDE = SPIBSC_OUTPUT_DISABLE;
205                                          /* Disable */
206
207      /* ---- Address ---- */
208      if(cmd == 0xBB){
209                                          /* Dual I/O High Performance */
210          SPIBSC.DREN.R.BIT.ADB = SPIBSC_2BIT;       /* Dual */
211          SPIBSC.DREN.R.BIT.ADE = SPIBSC_OUTPUT_ADDR_24;
212                                          /* S-flash x 1 Enable(ADDR[23:0]) */
213                                          /* S-flash x 2 Enable(ADDR[24:1]) */
213      }

```

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

```

214     else if(cmd == 0xEB){
215                                     /* Quad I/O High Performance */
216         SPIBSC.DREN.R.BIT.ADB = SPIBSC_4BIT; /* Quad */
217         SPIBSC.DREN.R.BIT.ADE = SPIBSC_OUTPUT_ADDR_24;
218                                     /* S-flash x 1 Enable(ADDR[23:0]) */
219                                     /* S-flash x 2 Enable(ADDR[24:1]) */
220     }
221     else{
222         SPIBSC.DREN.R.BIT.ADB = SPIBSC_1BIT; /* Single */
223         SPIBSC.DREN.R.BIT.ADE = SPIBSC_OUTPUT_ADDR_24;
224                                     /* S-flash x 1 Enable(ADDR[23:0]) */
225                                     /* S-flash x 2 Enable(ADDR[24:1]) */
226     }
227
228     /* ---- Option Data ---- */
229     if(cmd == 0xBB){
230                                     /* Dual I/O High Performance */
231         SPIBSC.DROPR.BIT.OPD3 = 0x00; /* Option Data(Mode bit) */
232         SPIBSC.DROPR.BIT.OPD2 = 0x00; /* Option Data */
233         SPIBSC.DROPR.BIT.OPD1 = 0x00; /* Option Data */
234         SPIBSC.DROPR.BIT.OPD0 = 0x00; /* Option Data */
235         SPIBSC.DREN.R.BIT.OPDB = SPIBSC_2BIT; /* Dual */
236         SPIBSC.DREN.R.BIT.OPDE = SPIBSC_OUTPUT_OPD_3;
237                                     /* Enable(OPD3) */
238     }
239     else if(cmd == 0xEB){
240                                     /* Quad I/O High Performance */
241         SPIBSC.DROPR.BIT.OPD3 = 0x00; /* Option Data(Mode bit) */
242         SPIBSC.DROPR.BIT.OPD2 = 0x00; /* Option Data(Dummy) */
243         SPIBSC.DROPR.BIT.OPD1 = 0x00; /* Option Data(Dummy) */
244         SPIBSC.DROPR.BIT.OPD0 = 0x00; /* Option Data */
245         SPIBSC.DREN.R.BIT.OPDB = SPIBSC_4BIT; /* Quad */
246         SPIBSC.DREN.R.BIT.OPDE = SPIBSC_OUTPUT_OPD_321;
247                                     /* Enable(OPD3, OPD2, OPD1) */
248     }
249     else{
250         SPIBSC.DROPR.BIT.OPD3 = 0x00; /* Option Data(Dummy) */
251         SPIBSC.DROPR.BIT.OPD2 = 0x00; /* Option Data */
252         SPIBSC.DROPR.BIT.OPD1 = 0x00; /* Option Data */
253         SPIBSC.DROPR.BIT.OPD0 = 0x00; /* Option Data */
254         SPIBSC.DREN.R.BIT.OPDB = SPIBSC_1BIT; /* Single */
255         SPIBSC.DREN.R.BIT.OPDE = SPIBSC_OUTPUT_OPD_3;
256                                     /* Enable(OPD3) */
257     }

```

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

```

258  /* ---- Data ---- */
259  if(cmd == 0x6B){
260      SPIBSC.DREN.R.BIT.DRDB = SPIBSC_4BIT;      /* Quad */
261  }
262  else if(cmd == 0x3B){
263      SPIBSC.DREN.R.BIT.DRDB = SPIBSC_2BIT;      /* Dual */
264  }
265  else if(cmd == 0x0B){
266      SPIBSC.DREN.R.BIT.DRDB = SPIBSC_1BIT;      /* Single */
267  }
268  else if(cmd == 0xBB){
269      SPIBSC.DREN.R.BIT.DRDB = SPIBSC_2BIT;      /* Dual I/O High Performance */
270  }
271  else if(cmd == 0xEB){
272      SPIBSC.DREN.R.BIT.DRDB = SPIBSC_4BIT;      /* Quad I/O High Performance */
273  }
274  else{
275      return -1;
276  }
277  return 0;
278  }
279
280  /*****
281   * ID          :
282   * Outline     :
283   * Include     : io_spibsc.h
284   * Declaration : int io_spibsc_transfer(ST_SPIBSC_SM *SpibscSm);
285   * Description :
286   * Argument    : void
287   * Return Value :
288   * Note       : None
289   *****/
290  int io_spibsc_transfer(ST_SPIBSC_SM *SpibscSm)
291  {
292      int i;
293      volatile unsigned long dummy;
294
295      if(SPIBSC.CMNCR.BIT.MD != SPIBSC_CMNCR_MD_SPI){
296          if(SPIBSC.CMNSR.BIT.SSLF != SPIBSC_SSL_NEGATE){
297              return -1;
298          }
299          SPIBSC.CMNCR.BIT.MD = SPIBSC_CMNCR_MD_SPI;
300                                  /* SPI Mode */
301      }
302
303      if(SPIBSC.CMNSR.BIT.TEND != SPIBSC_TRANS_END){
304          return -1;
305      }
306

```

3.25 Sample Program List "io_spibsc.c" (8)

```

307      /* ---- Command ---- */
308      SPIBSC.SMENR.BIT.CDE = SpibscSm->cde;          /* Enable/Disable */
309      if(SpibscSm->cde != SPIBSC_OUTPUT_DISABLE){
310          SPIBSC.SCMR.BIT.CMD = SpibscSm->cmd;        /* Command */
311          SPIBSC.SMENR.BIT.CDB = SpibscSm->cdb;        /* Single/Dual/Quad */
312      }
313
314      /* ---- Option Command ---- */
315      SPIBSC.SMENR.BIT.OCDE = SpibscSm->ocde;         /* Enable/Disable */
316      if(SpibscSm->ocde != SPIBSC_OUTPUT_DISABLE){
317          SPIBSC.SCMR.BIT.OCMD = SpibscSm->ocmd;       /* Option Command */
318          SPIBSC.SMENR.BIT.OCDB = SpibscSm->ocdb;      /* Single/Dual/Quad */
319      }
320
321      /* ---- Address ---- */
322      SPIBSC.SMENR.BIT.ADE = SpibscSm->ade;           /* Enable/Disable */
323      if(SpibscSm->ade != SPIBSC_OUTPUT_DISABLE){
324          SPIBSC.SMADR.BIT.ADR = SpibscSm->addr;        /* Address */
325          SPIBSC.SMENR.BIT.ADB = SpibscSm->adb;        /* Single/Dual/Quad */
326      }
327
328      /* ---- Option Data ---- */
329      SPIBSC.SMENR.BIT.OPDE = SpibscSm->opde;         /* Enable/Disable */
330      if(SpibscSm->opde != SPIBSC_OUTPUT_DISABLE){
331          SPIBSC.SMOPR.BIT.OPD3 = SpibscSm->opd[0];    /* Option Data */
332          SPIBSC.SMOPR.BIT.OPD2 = SpibscSm->opd[1];    /* Option Data */
333          SPIBSC.SMOPR.BIT.OPD1 = SpibscSm->opd[2];    /* Option Data */
334          SPIBSC.SMOPR.BIT.OPD0 = SpibscSm->opd[3];    /* Option Data */
335          SPIBSC.SMENR.BIT.OPDB = SpibscSm->opdb;      /* Single/Dual/Quad */
336      }
337
338      /* ---- Data ---- */
339      SPIBSC.SMENR.BIT.SPIDE = SpibscSm->spide;       /* Enable/Disable */
340      if(SpibscSm->spide != SPIBSC_OUTPUT_DISABLE){
341          SPIBSC.SMWDR0.LONG = SpibscSm->smwdr[0];
342          SPIBSC.SMWDR1.LONG = SpibscSm->smwdr[1];     /* Valid in two serial-flash */
343          SPIBSC.SMENR.BIT.SPIDB = SpibscSm->spidb;    /* Single/Dual/Quad */
344      }
345
346      SPIBSC.SMCR.BIT.SSLKP = SpibscSm->sslkp;
347
348      if((SpibscSm->spidb != SPIBSC_1BIT) && (SpibscSm->spide != SPIBSC_OUTPUT_DISABLE)){
349          if((SpibscSm->spire == SPIBSC_SPIDATA_ENABLE) &&
350             (SpibscSm->spiwe == SPIBSC_SPIDATA_ENABLE)){
351              /* not set in same time */
352              return -1;
353          }
354      }

```

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

```
354     SPIBSC.SMCR.BIT.SPIRE = SpibscSm->spire;
355     SPIBSC.SMCR.BIT.SPIWE = SpibscSm->spiwe;
356
357     SPIBSC.SMCR.BIT.SPIE = SPIBSC_SPI_ENABLE;          /* execute after setting SPNDL bit */
358
359     /* wait for transfer-start */
360     dummy = SPIBSC.CMNSR.LONG;
361     dummy = SPIBSC.CMNSR.LONG;
362     dummy = SPIBSC.CMNSR.LONG;
363     dummy = SPIBSC.CMNSR.LONG;
364
365     while(SPIBSC.CMNSR.BIT.TEND != SPIBSC_TRANS_END){
366         /* wait for transfer-end */
367     }
368     SpibscSm->smrdr[0] = SPIBSC.SMRDR0.LONG;
369     SpibscSm->smrdr[1] = SPIBSC.SMRDR1.LONG;          /* valid in two serial-flash */
370
371     return 0;
372 }
373
374 /* End of File */
375
```

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

```

1  /*****
2  *   DISCLAIMER

   (Omitted)

27 ****
28 *   Copyright (C) 2011 Renesas Electronics Corporation. All rights reserved.
29 **** Technical reference data ****
30 *   System Name : SH7268/SH7269 Firm Update Sample Program
31 *   File Name   : io_spibsc.h
32 *   Abstract    : spibsc structure
33 *   Version     : 1.00.00
34 *   Device      : SH7268/SH7269
35 *   Tool-Chain  : High-performance Embedded Workshop (Ver.4.07.00).
36 *               : C/C++ compiler package for the SuperH RISC engine family
37 *               :                               (Ver.9.03Release02).
38 *   OS          : None
39 *   H/W Platform: R0K57269(CPU board)
40 *   Description :
41 ****
42 *   History     : Jul.06,2011 Ver.1.00.00
43 *****/
44 #ifndef _IO_SPIBSC_H_
45 #define _IO_SPIBSC_H_
46
47 /* ==== define values ==== */
48 #define SPIBSC_CMNCR_MD_EXTRD      0
49 #define SPIBSC_CMNCR_MD_SPI        1
50
51 #define SPIBSC_OUTPUT_LOW          0
52 #define SPIBSC_OUTPUT_HIGH         1
53 #define SPIBSC_OUTPUT_LAST         2
54 #define SPIBSC_OUTPUT_HiZ          3
55
56 #define SPIBSC_CMNCR_CPHAT_EVEN    0
57 #define SPIBSC_CMNCR_CPHAT_ODD
58
59 #define SPIBSC_CMNCR_CPHAR_ODD     0
60 #define SPIBSC_CMNCR_CPHAR_EVEN    1
61
62 #define SPIBSC_CMNCR_SSLP_LOW      0
63 #define SPIBSC_CMNCR_SSLP_HIGH     1
64
65 #define SPIBSC_CMNCR_CPOL_LOW      0
66 #define SPIBSC_CMNCR_CPOL_HIGH     1
67
68 #define SPIBSC_CMNCR_BSZ_SINGLE     0
69 #define SPIBSC_CMNCR_BSZ_DUAL       1
70

```


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

```

71  #define SPIBSC_DELAY_1SPBCLK      0
72  #define SPIBSC_DELAY_2SPBCLK      1
73  #define SPIBSC_DELAY_3SPBCLK      2
74  #define SPIBSC_DELAY_4SPBCLK      3
75  #define SPIBSC_DELAY_5SPBCLK      4
76  #define SPIBSC_DELAY_6SPBCLK      5
77  #define SPIBSC_DELAY_7SPBCLK      6
78  #define SPIBSC_DELAY_8SPBCLK      7
79
80
81  #define SPIBSC_BURST_1              0x00
82  #define SPIBSC_BURST_2              0x01
83  #define SPIBSC_BURST_3              0x02
84  #define SPIBSC_BURST_4              0x03
85  #define SPIBSC_BURST_5              0x04
86  #define SPIBSC_BURST_6              0x05
87  #define SPIBSC_BURST_7              0x06
88  #define SPIBSC_BURST_8              0x07
89  #define SPIBSC_BURST_9              0x08
90  #define SPIBSC_BURST_10             0x09
91  #define SPIBSC_BURST_11             0x0a
92  #define SPIBSC_BURST_12             0x0b
93  #define SPIBSC_BURST_13             0x0c
94  #define SPIBSC_BURST_14             0x0d
95  #define SPIBSC_BURST_15             0x0e
96  #define SPIBSC_BURST_16             0x0f
97
98  #define SPIBSC_BURST_DISABLE         0
99  #define SPIBSC_BURST_ENABLE         1
100
101  #define SPIBSC_DRCR_RCF_EXE          1
102
103  #define SPIBSC_SSL_NEGATE            0
104  #define SPIBSC_TRANS_END             1
105
106  #define SPIBSC_1BIT                  0
107  #define SPIBSC_2BIT                  1
108  #define SPIBSC_4BIT                  2
109
110  #define SPIBSC_OUTPUT_DISABLE        0
111  #define SPIBSC_OUTPUT_ENABLE         1
112  #define SPIBSC_OUTPUT_ADDR_24        0x07
113  #define SPIBSC_OUTPUT_ADDR_32        0x0f
114  #define SPIBSC_OUTPUT_OPD_3          0x08
115  #define SPIBSC_OUTPUT_OPD_32         0x0c
116  #define SPIBSC_OUTPUT_OPD_321        0x0e
117  #define SPIBSC_OUTPUT_OPD_3210       0x0f
118

```

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

```

119  #define SPIBSC_OUTPUT_SPID_8      0x08
120  #define SPIBSC_OUTPUT_SPID_16    0x0c
121  #define SPIBSC_OUTPUT_SPID_32    0x0f
122
123  #define SPIBSC_SPISSL_NEGATE      0
124  #define SPIBSC_SPISSL_KEEP       1
125
126  #define SPIBSC_SPIDATA_DISABLE    0
127  #define SPIBSC_SPIDATA_ENABLE    1
128
129  #define SPIBSC_SPI_DISABLE        0
130  #define SPIBSC_SPI_ENABLE        1
131
132  typedef struct{
133      unsigned long cdb:2;
134      unsigned long ocdb:2;
135      unsigned long adb:2;
136      unsigned long opdb:2;
137      unsigned long spidb:2;
138      unsigned long cde:1;
139      unsigned long ocde:1;
140      unsigned long ade:4;
141      unsigned long opde:4;
142      unsigned long spide:4;
143      unsigned long sslkp:1;
144      unsigned long spire:1;
145      unsigned long spiwe:1;
146      unsigned long :5;
147
148      unsigned char cmd;
149      unsigned char ocmd;
150      unsigned long addr;
151      unsigned char opd[4];
152      unsigned long smrdr[2];
153      unsigned long smwdr[2];
154  }ST_SPIBSC_SM;
155
156  int io_spibsc_bsz_set(unsigned long bsz);
157  unsigned long io_spibsc_bsz_get(void);
158  int io_spibsc_common_init(unsigned long bsz);
159  int io_spibsc_dr_init(unsigned long cmd);
160  int io_spibsc_transfer(ST_SPIBSC_SM *SpibscSm);
161
162  #endif /* IO_SPIBSC_H */
163  /* 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>

All trademarks and registered trademarks are the property of their respective owners.

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.



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: +1-408-688-6000, Fax: +1-408-688-6130

Renesas Electronics Canada Limited

1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-6441, 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: +86-10-8236-1155, Fax: +86-10-8236-7679

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 Laviel'or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141