SH7266/SH7267 Group

Interfacing Serial Flash Memory Using the Renesas Quad Serial Peripheral Interface

R01AN0167EJ0100
Rev. 1.00
Dec. 27, 2010

Summary

This application note describes how to connect serial flash memory using the SH7266/SH7267 Microcomputers (MCUs) Renesas Quad Serial Peripheral Interface (RQSPI).

Target Device

SH7266/SH7267 MCU (In this document, SH7266/SH7267 are described as "SH7267".)

Contents

1. Introduction

1.1 Specifications

- Connects a serial flash memory which is compliant with the Serial Peripheral Interface (SPI) multi I/O bus with the SH7267
- Uses the Renesas Serial Peripheral Interface (RSPI) to boot the SH7267
- After booting the SH7267, uses RQSPI quad-SPI operation to read or program the serial flash memory

1.2 Modules Used

- Renesas Quad Serial Peripheral Interface (RQSPI)
- Renesas Serial Peripheral Interface (RSPI)
- Boot mode (serial flash boot)
- General-purpose I/O ports

1.3 Applicable Conditions

MCU	SH7266/SH7267
Operating Frequency	Internal clock: 144 MHz Bus clock: 72 MHz Peripheral clock: 36 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)

1.4 Related Application Note

For more information, refer to the following application note:

- SH7266/SH7267 Group Boot from the Serial Flash Memory

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. Applications

Connect the SH7267 (master) with the SPI-compatible serial flash memory (slave) for read/write access using the Renesas Quad Serial Peripheral Interface (RQSPI). This chapter describes the pin connection example and sample program flow charts.

2.1 RQSPI Operation

SH7267 RQSPI is allowed to communicate with the serial flash memory by single-, dual-, and quad-SPI operation. Set the transfer data length between 8 bits to 128 Gbits, set the bit rate to the bus clock divided by between 1 and 4080. Note that the bit rate cannot be set to the bus clock divided by 1 when transmitting data. Use four Command registers (SPCMDn, n = 0 to 3) to execute several different transfer modes sequentially in a loop.

Figure 1 to Figure 3 shows transfer formats in single-, dual-, and quad-SPI modes. Single-SPI mode allows full-duplex communication to set QMO pin to output and QMI pin to input. Dual-SPI mode allows 2-bit wide half-duplex communication to set pins QIO0 and QIO1 to I/O pins. Quad-SPI mode allows 4-bit wide to set pins QIO0, QIO1, QIO2, and QIO3 to I/O pins. QMO pin is multiplexed with QIO0 pin and QMI pin is multiplexed with QIO1 pin.

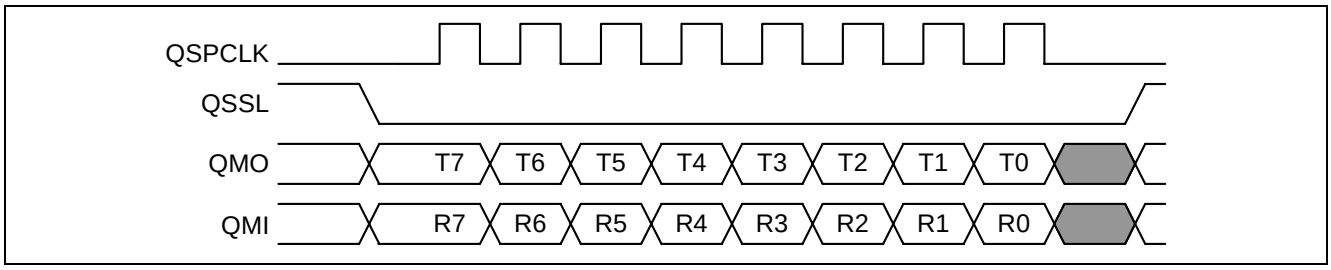

Figure 1 Single-SPI Mode Transfer Format

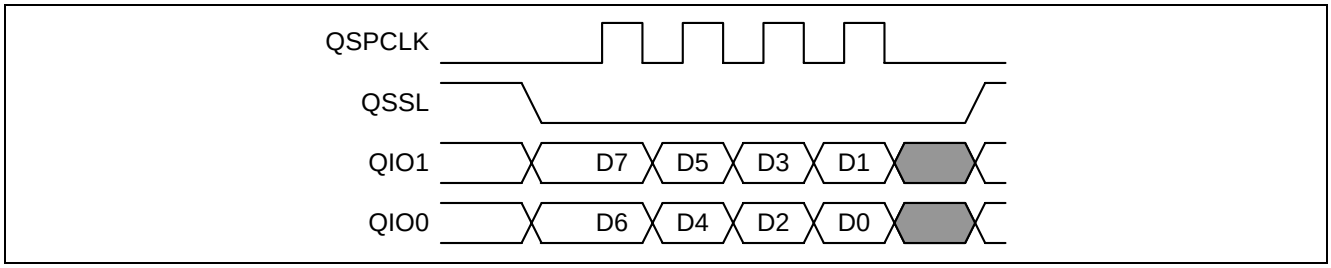

Figure 2 Dual-SPI Mode Transfer Format

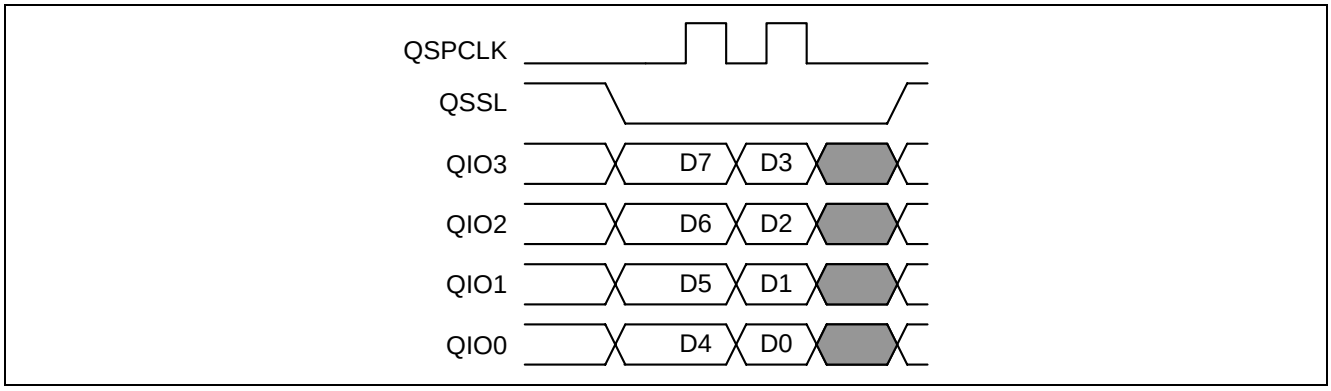

Figure 3 Quad-SPI Mode Transfer Format

2.2 Serial Flash Memory Pin Connection

Table 1 lists the specifications of the multi I/O bus-compatible serial flash memory (S25F032P, Spansion) used in this application.

Table 1 Serial Flash Memory Specifications

Item	Description
Bus I/O	Serial I/O (full-duplex), dual I/O (half-duplex), quad I/O (half-duplex)
SPI modes	SPI modes 0 and 3 supported
Clock frequency	Serial I/O: 104 MHz (max.), dual/quad I/O: 80 MHz (max.)
Capacity	4 MB
Sector size	64 KB
Page size	256 KB
Erase size	Bulk erase, 64 KB, 8 KB, 4 KB
Program size	Page Program (1 to 256 bytes)
Data protection modes	Write Enable command (in commands) Software Protected Mode, Hardware Protected Mode (in blocks)

Figure 4 shows an example of serial flash memory connection circuit. Set the SH7267 pin functions as shown in Table 2.

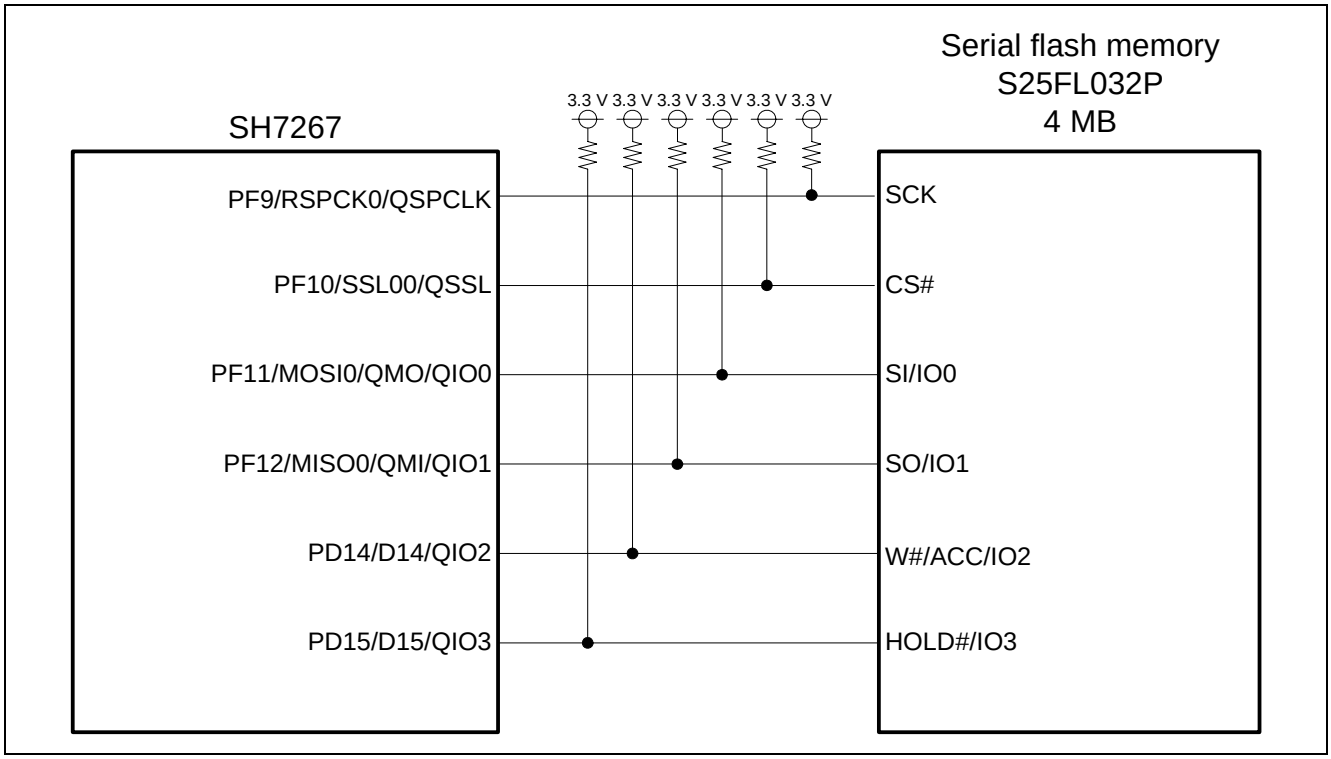

Figure 4 Serial Flash Memory Circuit

Note: Pull up or pull down the control signal pins using external resistors.
Pull up or pull down the control signal pins, so the external device does not malfunction when the MCU pins are in the high impedance state. Use external resistors to pull up the control signal pins.

Table 2 Multiplexed Pins

Peripheral Functions	Pin Name	SH7264 Port Control Register		SH7267 Multiplexed Pin Name
		Register Name	MD Bit Setting	
RQSPI	QSPCLK	PFCR2	PF9MD[2:0] = B'110	PF9/A23/SSISCK3/RSPCL0/TIOC3A/FRB/QSPCLK
	QSSL	PFCR2	PF10MD[2:0] = B'110	PF10/A24/SSIWS3/SSL00/TIOC3C/SPDIF_IN/QMO/QIO0
	QMO/QIO0	PFCR2	PF11MD[2:0] = B'110	PF12/BS/MISO0/TIOC3D/SPDIF_OUT/QMI/QIO1
	QMI/QIO1	PFCR3	PF12MD[2:0] = B'110	PF12/BS/MISO0/TIOC3D/SPDIF_OUT/QMI/QIO1
	QIO2	PFCR3	PF14MD[1:0] = B'11	PD14/D14/NAF6/PWM2G/QIO2
	QIO3	PFCR3	PF15MD[1:0] = B'11	PD15/D15/NAF7/PWM2H/QIO3

Note: SH7267 Multiplexed Pins
QSPCLK, QSSL, QMO/QIO0, QMI/QIO1, QIO2, and QIO3 pins are multiplexed, and set to general-purpose I/O ports as default. Before accessing serial flash memory, use the general-purpose I/O port control register to set the multiplexed pins to RQSPI pins. QIO2 and QIO3 pins cannot be set to RQSPI function in boot mode 0 (boot from the memory which is connected to CS0 space). Use boot modes 1 or 3 (serial flash boot) to set these pins to RQSPI function.

2.3 Interface Timing Example

This section describes an example of the interface timing between the SH7267 and serial flash memory. Configure the RQSPI and set the clock frequency according to the timing conditions of the serial flash memory, which is used as a slave.

Figure 5 shows an example of the data transfer timing. As the serial flash memory used in this application latches data at the rising edge of the clock, and outputs data at the falling edge of the clock, specify 1 to bits CPOL and CPHA in the Command register (SPCMD). By this setting, QSPCLK is specified to 1 when it is idling, and the timing to output data in the RSPI can be set to the odd edge (falling edge). Configure the RQSPI to satisfy the timing conditions shown in Table 3 and Table 4.

This application sets the bit rate to 36 Mbps (when the bus clock is 72 MHz). As the SH7267 does not allow the user to set the bit rate to the bus clock divide-by-1 when transmitting data, do not set the bit rate to 72 Mbps when the SH7267 is in Single-SPI operation. As dual-SPI and quad-SPI operations are half-duplex communication, the bit rate can be set to 72 Mbps only for reading data.

When the setup time is not enough for the transfer mode shown in Figure 5, extend the setup time from outputting data to latching data to one cycle. For more information, refer to the application note "SH7262/SH7264 Group High-speed Read/Write Serial Flash Memory Using the Renesas Serial Peripheral Interface".

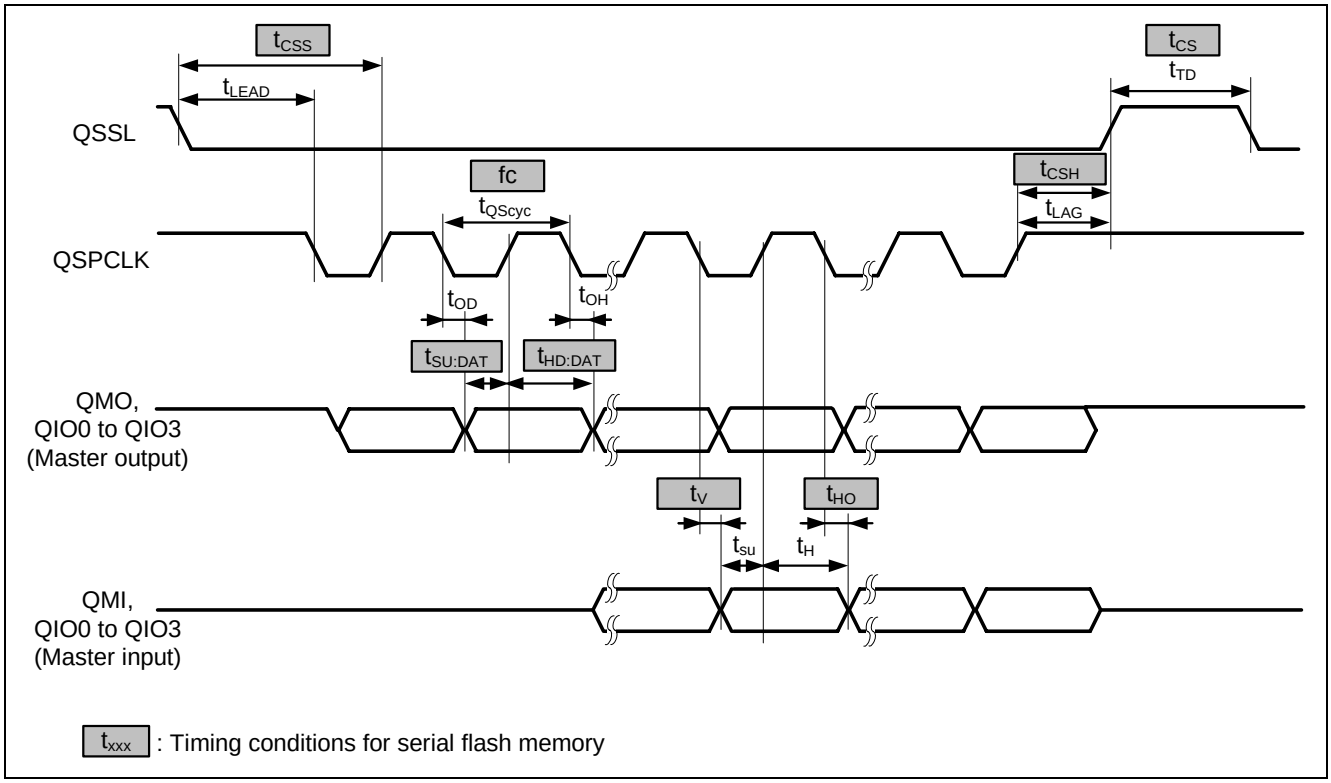

Figure 5 Data Transfer Timing Example (CPOL = 1, CPHA = 1)

Table 3 Timing Conditions for Serial Flash Memory When Transferring Data

Symbol	Item	Description	Related Registers
t_{CSS}	CS# Active Setup Time	Time required by the slave device to latch data from asserting SSL to the QSPCLK rising The following formula must be fulfilled: $[t_{LEAD}$ (=QSPCLK delay) + 1/2] × t_{QScyc} × $t_{cyc} \geq t_{CSS}$ (min.)	SPCKD register SPCMD register SPBR register
t_{CS}	CS# High Time	Time required for QSSL negation The following formula must be fulfilled: t_{TD} (= next access delay) × t_{QScyc} × $t_{cyc} \geq t_{CS}$ (min.)	SPND register SPCMD register
f_C	Serial clock frequency	The maximum operating frequency supported by the slave device The following formula must be fulfilled: f_C (max.) ≥ 1/(t_{QScyc} × t_{cyc})	SPBR register SPCMD register
t_{CSH}	CS# Active Hold Time	Hold time required from the last QSPCLK rising to the QSSL negation The following formula must be fulfilled: t_{LAG} (QSSL negate delay) × t_{QScyc} × $t_{cyc} \geq t_{CSH}$ (min.)	SSLND register SPCMD register
$t_{SU:DAT}$	Data in Setup Time	Time required by the master device from outputting data to latching data The following formula must be fulfilled: 1/2 × t_{QScyc} × t_{cyc} − t_{OD} (max.) ≥ $t_{SU:DAT}$	SPBR register SPCMD register
$t_{HD:DAT}$	Data in Hold Time	Time required by the master device to hold the data output The following formula must be fulfilled: t_{OH} (min.) + 1/2 × t_{QScyc} × $t_{cyc} \geq t_{HD:DAT}$	SPBR register SPCMD register

Note: t indicates one cycle time of the bus clock (Bφ).

Table 4 Timing Conditions for the SH7267 When Transferring Data

Symbol	Item	Description	Related Registers
t_{SU}	Data in Setup Time	Time required by the slave device to latch data from asserting SSL to the QSPCLK rising The following formula must be fulfilled: 1/2 × t_{QScyc} × t_{cyc} − t_V (max.) ≥ t_{SU} (min.)	SPBR register SPCMD register
t_H	Data in Hold Time	Time required by the slave device to hold the data output The following formula must be fulfilled: t_{HO} (min.) + 1/2 × t_{QScyc} × $t_{cyc} \geq t_H$ (min.)	SPBR register SPCMD register

Note: t indicates one cycle time of the bus clock (Bφ).

2.4 Sample Program Operation

2.4.1 RQSPI Configuration

Figure 6 shows the flow chart for configuring the RQSPI in the sample program. Set the Command register according to the transfer mode before starting the transfer.

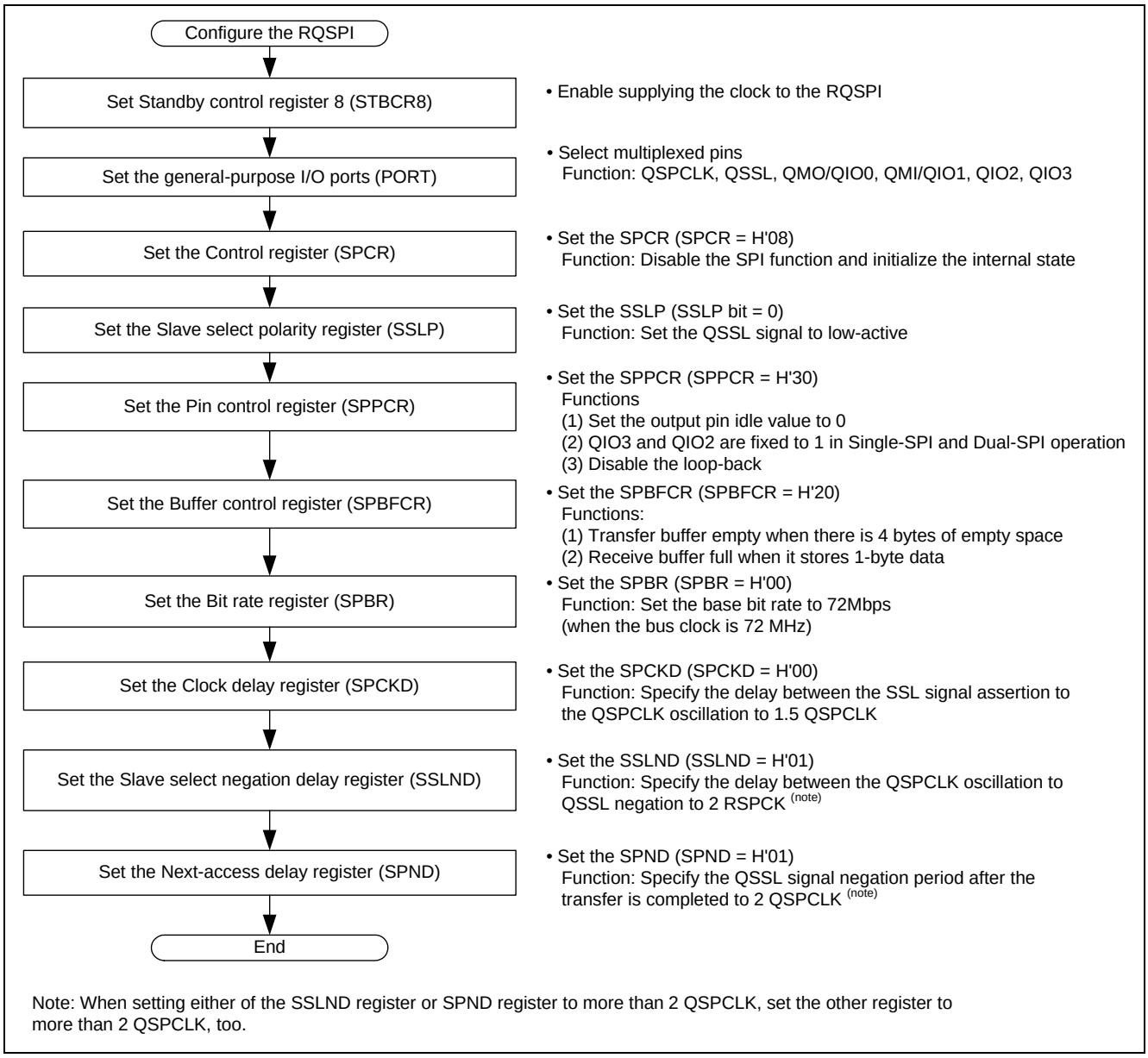

Figure 6 Flow Chart for Configuring the RQSPI

2.4.2 Sequence Control

Use commands to access serial flash memory. This section describes the major commands and command sequence example, and shows the sequence control in the sample program.

This application refers to the commands of the Spansion S25FL032P. For more information, refer to the data sheet provided by the serial flash memory manufacturer.

A. Major commands

Table 5 lists the major commands for the S25FL032P.

Table 5 S25FL032P Major Commands

Command Name	Command Code	Address Bytes	Dummy Bytes	Data Bytes	Description
Read Data bytes at Fast Speed	H'0B	3	1	More than 1 (1)	Reads the data
Quad Output Read	H'6B	3	1	More than 1 (1)	Reads the data (quad-SPI)
Write Enable	H'06	0	0	0	Enables the program/erase command
Write Disable	H'04	0	0	0	Disables the program/erase command
64KB Sector Erase	H'D8	3	0	0	Erases the data in blocks (64 KB)
Bulk Erase	H'C7	0	0	0	Erases the entire memory array
Page Programming	H'02	3	0	More than 1 (2)	Programs the data
Quad Page Programming	H'32	3	0	More than 1 (2)	Programs the data (quad-SPI)
Read Status Register	H'05	0	0	More than 1	Reads the Status register
Read Configuration Register	H'35	0	0	More than 1	Reads the Configuration register
Write (Status & Configuration) Register	H'01	0	0	1 or 2	Programs the data in the Status and Configuration register

Notes: 1. Reads the address incremented from the specified address (When the last byte of the memory array has been read, the device will continue reading back at the beginning of the array).
2. Programs the data in the incremented address in the same page (When the device goes beyond the end of the page, it will wrap around back to the beginning of the same page).

B. Command sequence example

Figure 7 shows the sequence example of the Quad Output Read (H'6B).

Execute the S25FL032P Quad Output Read command as follows:

Assert the QSSL signal, and transfer the command code and address (3 bytes) in Single-SPI mode. Next, insert the dummy clocks, and execute the dummy clock in quad-SPI mode (read) to set QMO/QIO0 pin to input. Then, read the data in quad-SPI mode (read).

Such complicated transfer is enabled by the sequence control with the combination of the Command register n (SPCMDn) and Transfer data length multiplier setting register n (SPBMULn).

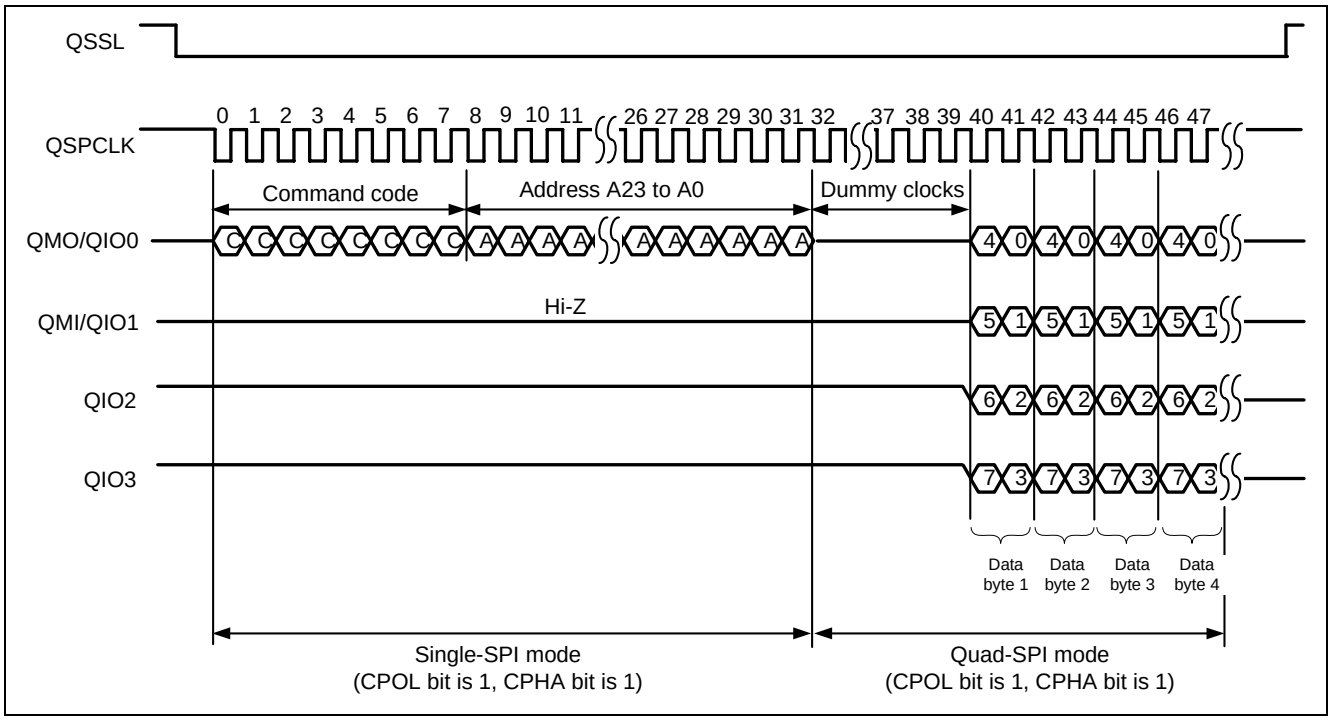

Figure 7 Read Command Sequence

C. Sequence control example in the sample program

Figure 8 shows the sequence control example used in the sample program.

Use SPCMD0 register to transfer the command code and address by the master device in single-SPI mode. Use SPCMD1 and SPCMD2 registers to transfer Quad Output Read command (H'6B) and Quad Page Programming (H'32) to use the SH7267 in quad-SPI mode. Set the number of transfers for each SPCMDn register to the SPBMULn register.

The SH7267 immediately oscillates the clock and starts receiving data in master mode when the receive buffer is empty during reading data in dual-, and quad-SPI mode. Thus, note that the SH7267 may discard the receive data depending on the timing when the receive buffer in the SPBFCR register is reset.

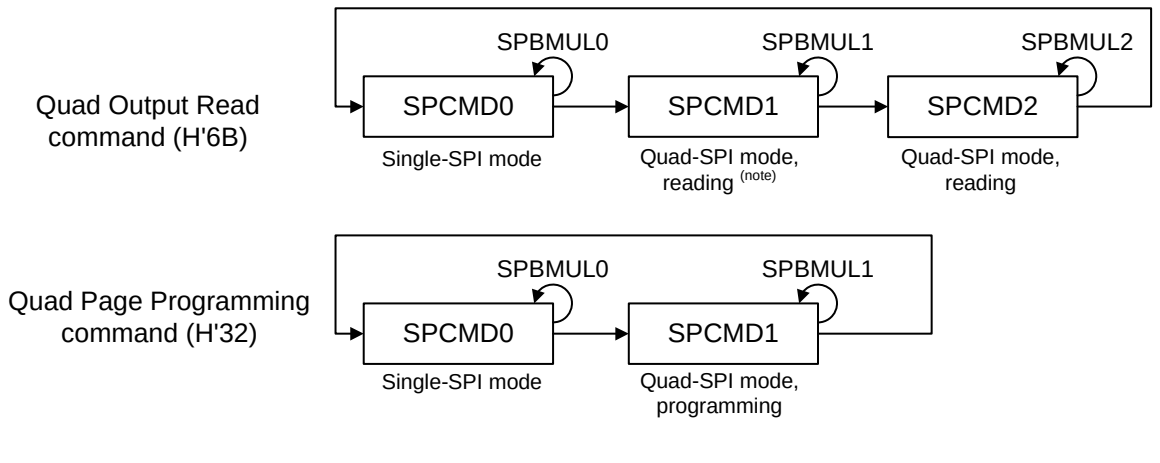

Figure 8 Sequence Control in the Sample Program

Figure 9 shows the structure to define the sequence control information in the sample program. The structure controls the transfer setting for each SPCMDn register.

Use the rqspi_cmd_set[4] structure when executing the RQSPI transfer by the io_rqspi_transfer function. Set the value in the rqspi_cmd_set[4] structure by the io_rqspi_set_cmd function before executing the io_rqspi_transfer function.

```
/* ---- Structure to define the RQSPI sequence control information ---- */
typedef struct{
        unsigned short spcmd;     /* SPCMDn register setting */
        unsigned long  spbmul;    /* SPBMULn register setting */
        void    *wr_ptr;          /* Address storing the transmit data */
        void    *rd_ptr;          /* Address storing the receive data */
}RQSPI_CMD_ST;

RQSPI_CMD_ST rqspi_cmd_set[4];
```

Figure 9 Structure to Define the Sample Program Sequence Control Information

Figure 10 shows the macro definition used to set the spcmd member.

Specify either of the SPI_SINGLE/SPI_QUAD_WR/SPI_QUAD_RD macro as the argument of the io_rqspi_set_cmd function. The io_rqspi_set_cmd function sets the value to specify the SPI mode to the rqspi_cmd_set[4] structure.

Macro definition for the basis to set the SPCMD register
Macro name: SPCMD_DEFAULT_SET
Setting: 0xE087

```
/* bit 15: Clock delay: SPCKD (1.5 QSPCLK)     */
/* bit 14: QSSL negation delay: SSLND (2 QSPCLK)                    */
/* bit 13: Next access delay: SPND (2 QSPCLK)                       */
/* bit 12: Format: MSB first                                        */
/* bits 11 to 8: Transfer data length: 8 bits                       */
/* bit 7: QSSL signal: Retained after the transfer is completed     */
/* bits 6, 5: SPI mode: Single-SPI                                  */
/* bit 4: Read/program: Program (invalid)                           */
/* bits 3, 2: Bit rate: SPBR divided by 2 (36 Mbps)                 */
/* bit 1: CPOL (QSPCLK  polarity): 1 when it is idling              */
/* bit 0: CPHA (QSPCLK phase): Shifts data on the odd edge          */
/*                                : Latches data on the even edge   */
```

Macro definition to use single-SPI mode
Macro name: (SPCMD_DEFAULT_SET | SPI_SINGLE)
Setting: 0xE087 | 0x0000

Macro definition to program data in quad-SPI mode
Macro name: (SPCMD_DEFAULT_SET | SPI_QUAD_WR)
Setting: 0xE087 | 0x0040

Macro definition to read data in quad-SPI mode
Macro name: (SPCMD_DEFAULT_SET | SPI_QUAD_RD)
Setting: 0xE087 | 0x0050

Note: To negate the QSSL signal after the transfer is completed, bit 7 (SSLKP bit) in the SPCMD_DEFAULT_SET is set to 0.

Figure 10 Macro Definition to Set the SPCMDn Register in the Sample Program

2.4.3 Main Function Flow

Figure 11 shows the main function flow chart. The sample program programs the data in the entire memory array, and compares the programmed value to the read value.

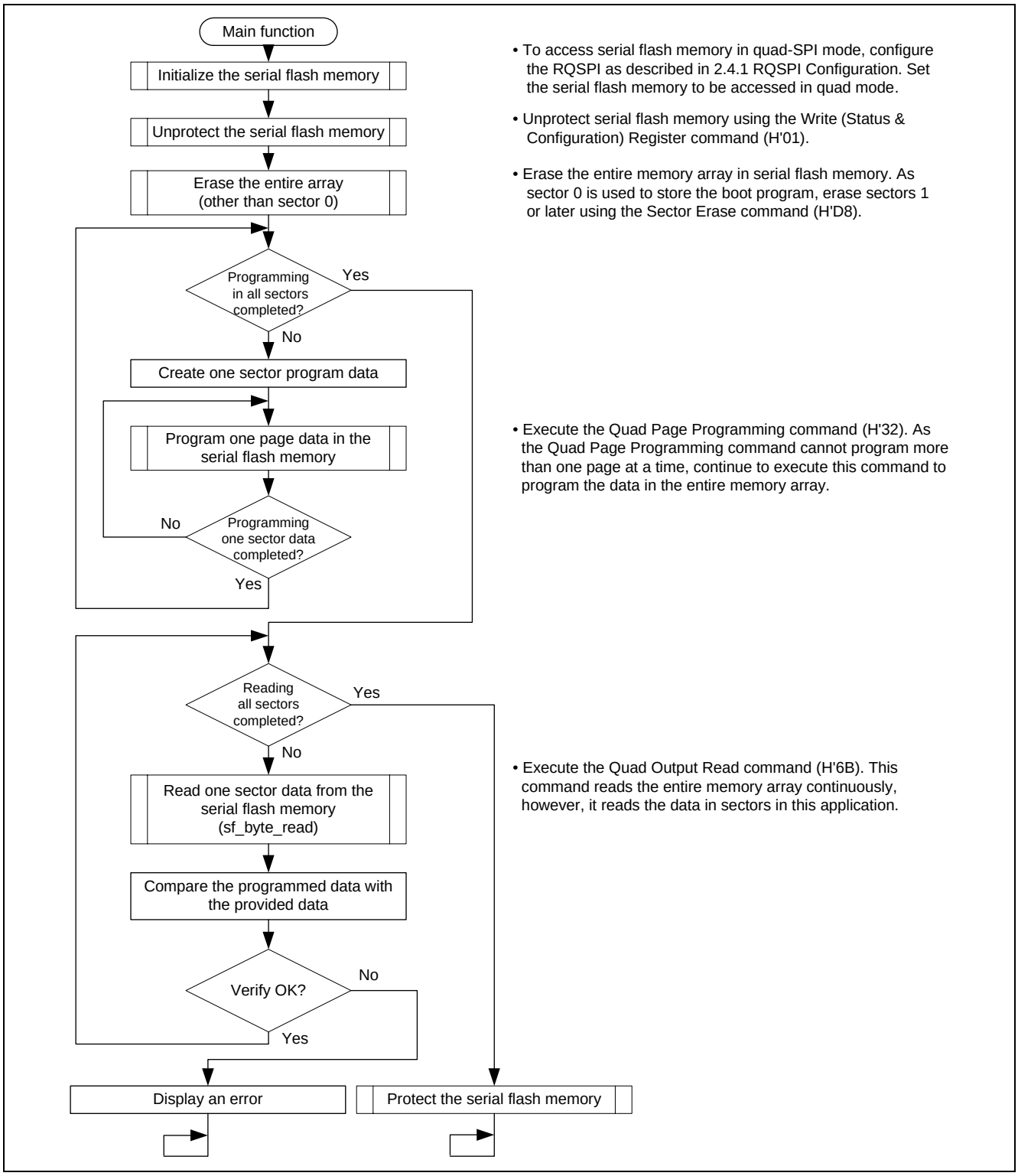

Figure 11 Sample Program Main Function Flow Chart

2.4.4 Serial Flash Memory Command Flow Chart

Figure 12 shows the flow chart for executing the command in the sample program. Serial flash memory has several commands, and this section shows the flow chart for executing the Quad_Output_Read command (H'6B).

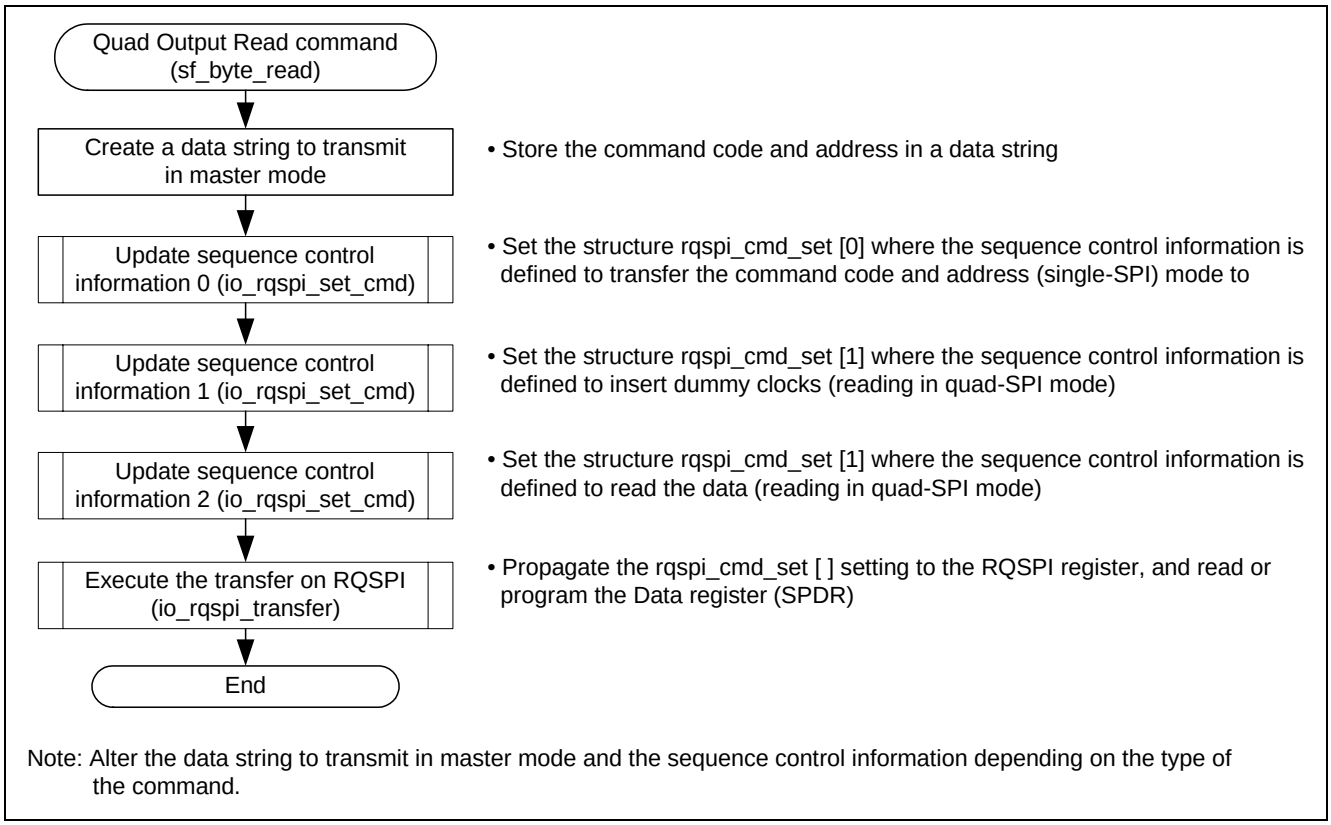

Figure 12 Flow Chart for Executing the Quad Output Read Command in the Sample Program

2.4.5 RQSPI Transfer Flow Chart

Figure 13 shows the flow chart for updating the sequence control information in the sample program. Execute this processing for the required number of Command registers before the RQSPI transfers data.

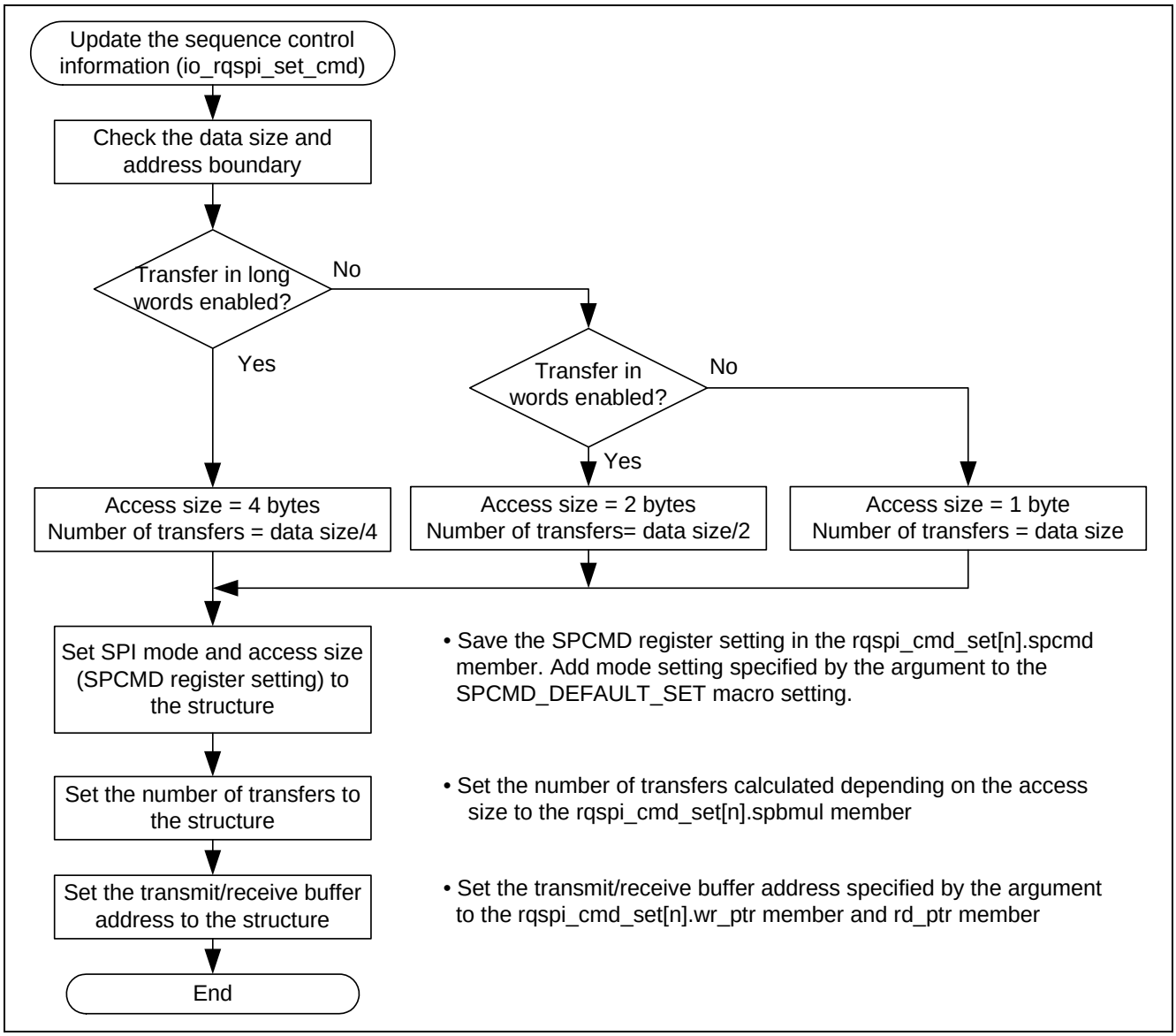

Figure 13 Flow Chart for Updating the Sequence Control Information in the Sample Program

Figure 14 and Figure 15 show RQSPI transfer flow charts in the sample program.

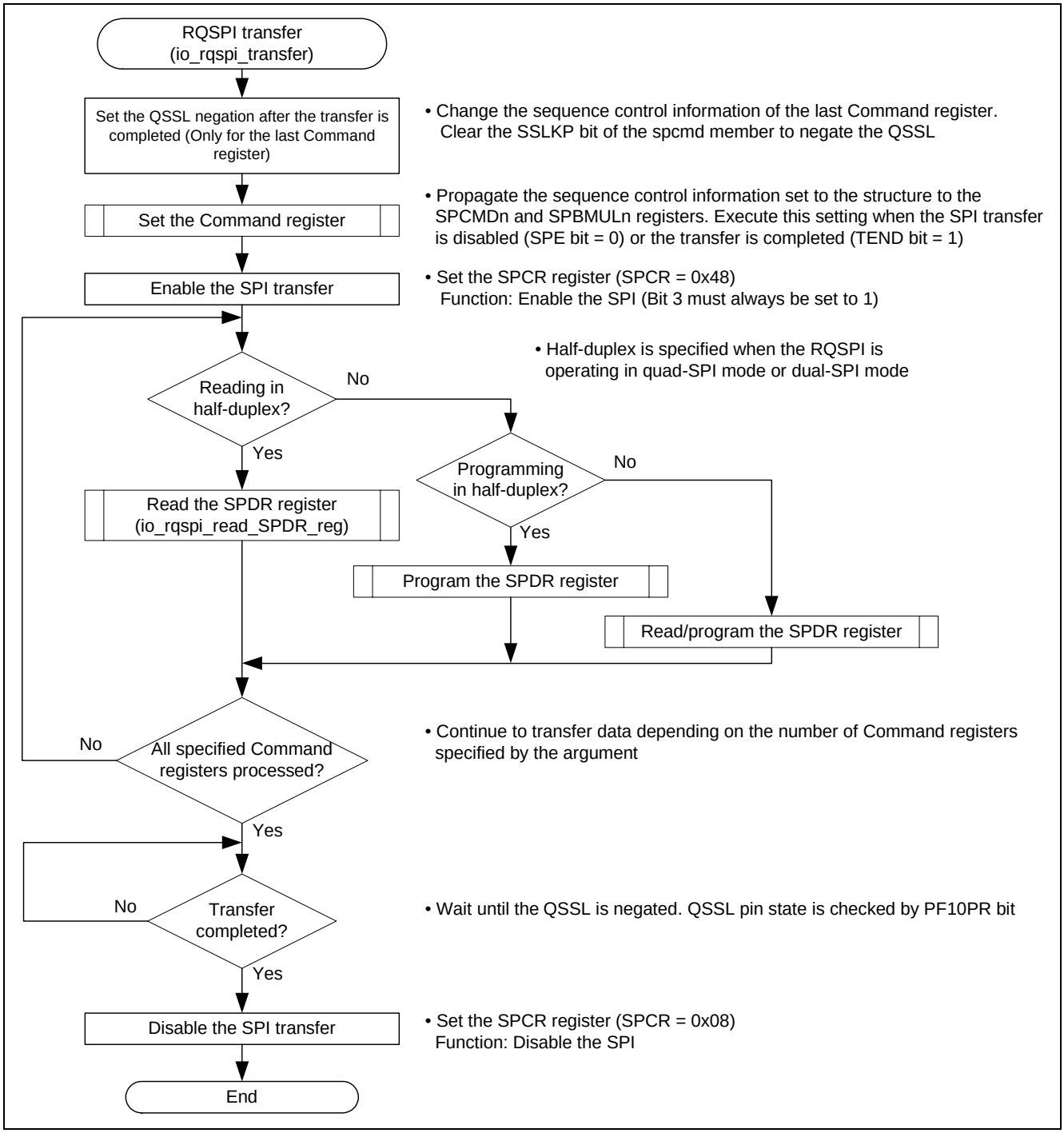

Figure 14 RQSPI Transfer Flow Chart (1/2)

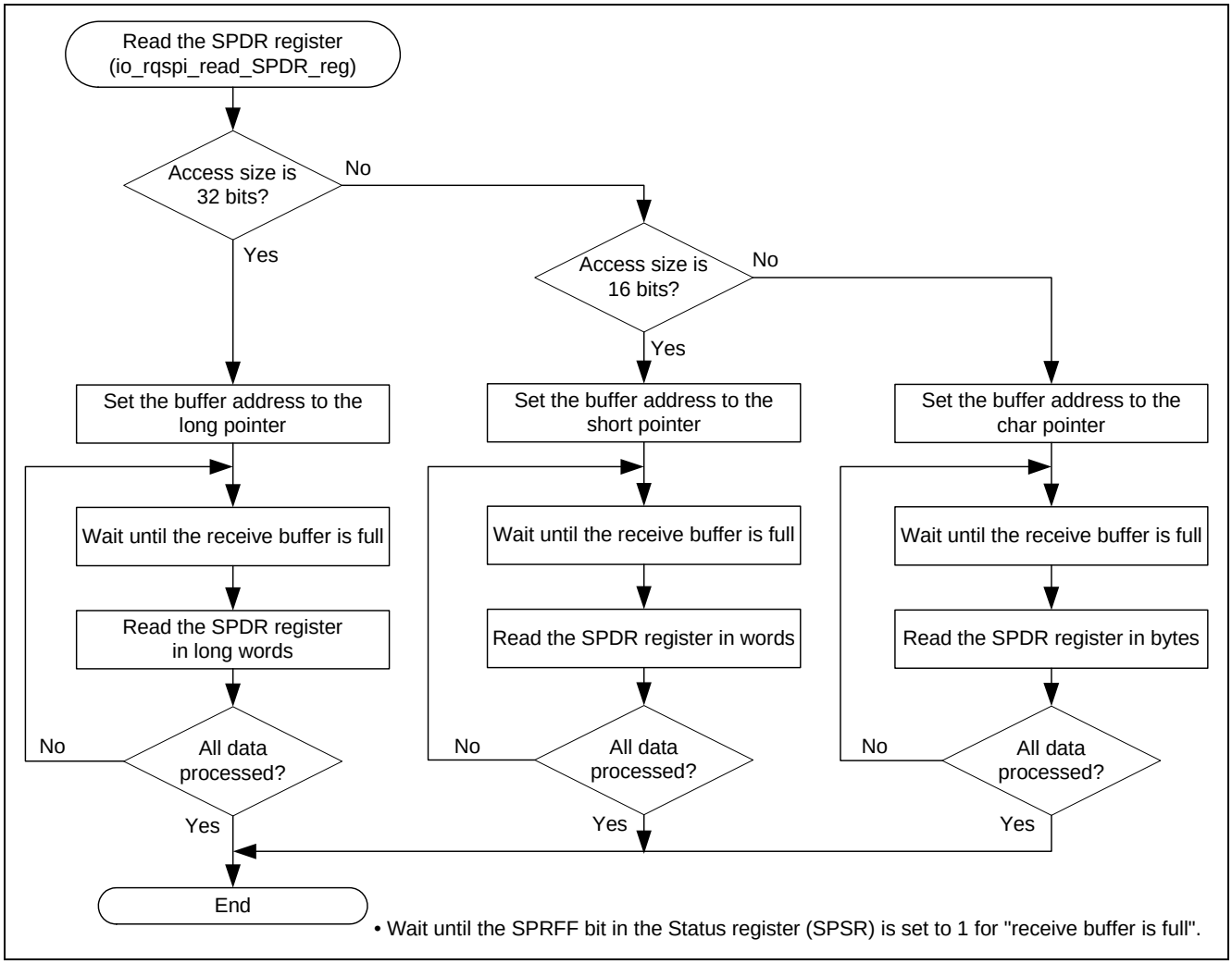

Figure 15 RQSPI Transfer Flow Chart (2/2)

3. Sample Program Listing

3.1 Supplement to the Sample Program

When using the SH7267 in boot mode 0, pins QIO2 and QIO3 cannot be set to RQSPI. Thus, the sample program boots the SH7267 in boot modes 1 or 3 (serial flash boot).

For the procedure to boot the MCU from the serial flash memory and program the serial flash memory, refer to the application note "SH7267 Group Boot from the Serial Flash Memory".

3.2 Sample Program Listing "main.c" (1/3)

```
/***********************************************************************
*   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) 2010 Renesas Electronics Corporation. All Rights Reserved.
*""FILE COMMENT""*********** Technical reference data *************************
*   System Name : SH7267 Sample Program
*   File Name   : main.c
*   Abstract    : Interfacing Serial Flash Memory Using the Renesas Quad Serial
*               : Peripheral Interface
*   Version     : 1.00.00
*   Device      : SH7266/SH7267
*   Tool-Chain  : High-performance Embedded Workshop (Ver.4.07.00).
*               : C/C++ compiler package for the SuperH RISC engine family
*               :                             (Ver.9.03 Release02).
*   OS          : None
*   H/W Platform: R0K57267(CPU board), Spansion[S25FL032P](serial flash)
*   Description :
***********************************************************************
*   History     : Aug.20,2010 Ver.1.00.00
*""FILE COMMENT END""*****************************************************/
#include <stdio.h>
#include "qserial_flash.h"

```

3.3 Sample Program Listing "main.c" (2/3)

```
48    /* ==== Macro definition ==== */
49    #define TOP_ADDRESS    0              /* Serial flash memory start address */
50
51    /* ==== Function prototype declaration ==== */
52    void main(void);
53
54    /* ==== Variable definition ==== */
55    #pragma section DEBUG_BUFFER
56    static unsigned char data[SF_SECTOR_SIZE];
57    static unsigned char rbuf[SF_SECTOR_SIZE];
58    #pragma section
59
60    /*""FUNC COMMENT""*****************************************************************
61     * ID           :
62     * Outline      : Accessing serial flash memory main
63     *-----------------------------------------------------------------------------
64     * Include      :
65     *-----------------------------------------------------------------------------
66     * Declaration  : void main(void);
67     *-----------------------------------------------------------------------------
68     * Description  : Erases, programs, and reads serial flash memory.
69     *              : After initializing RSPI channel 0, it erases the entire memory
70     *              : array other than sector 0, and programs data from the start
71     *              : address. Reads the program data to compare to the provided data.
72     *-----------------------------------------------------------------------------
73     * Argument     : void
74     *-----------------------------------------------------------------------------
75     * Return Value : void
76     *-----------------------------------------------------------------------------
77     * Note         : None
78     *""FUNC COMMENT END""*************************************************************/
79    void main(void)
80    {
81      int i, j;
82      static unsigned long addr;
83
84      /* ==== Initializes the serial flash memory ==== */
85      sf_init_serial_flash();
86
87      /* ==== Unprotects the serial flash memory ==== */
88      sf_protect_ctrl( SF_REQ_UNPROTECT );
89
90      /* ==== Erases sectors (entire memory array other than sector 0) ==== */
91      for(i = 1; i < SF_NUM_OF_SECTOR; i++){
92        sf_sector_erase( i );
93      }
```

3.4 Sample Program Listing "main.c" (3/3)

```
94      /* ==== Programs the data (entire memory array other than sector 0) ==== */
95      addr = TOP_ADDRESS + SF_SECTOR_SIZE;       /* sector 1 */
96      for(i = 1; i < SF_NUM_OF_SECTOR; i++){     /* sector 1 or later */
97
98        /* ---- Initializes the data (one sector) ---- */
99        for(j = 0; j < SF_SECTOR_SIZE; j++){
100           data[j] = (i + j) % 100;
101       }
102       /* ---- Programs one sector data ---- */
103       for(j = 0; j < ( SF_SECTOR_SIZE / SF_PAGE_SIZE ); j++){
104
105           /* ---- Programs one page size data ---- */
106           sf_byte_program( addr, data+(j*SF_PAGE_SIZE), SF_PAGE_SIZE );
107           addr += SF_PAGE_SIZE;             /* Updates the program destination address */
108       }
109     }
110     /* ==== Reads the data (entire memory array other than sector 0) ==== */
111     addr = TOP_ADDRESS + SF_SECTOR_SIZE;       /* sector 1 */
112     for(i = 1; i < SF_NUM_OF_SECTOR; i++){     /* sector 1 or later */
113
114       /* ---- Reads one sector data ---- */
115       sf_byte_read( addr, rbuf, SF_SECTOR_SIZE );
116       addr += SF_SECTOR_SIZE;                  /* Updates the read destination address */
117
118       /* ---- Verifies the data ---- */
119       for(j = 0; j < SF_SECTOR_SIZE; j++){
120           data[j] = (i + j) % 100;          /* Outputs the program data */
121           if( data[j] != rbuf[j] ){
122               puts("Error: verify error\n");
123               fflush(stdout);
124               while(1){
125                   /* error */
126               }
127           }
128       }
129     }
130     /* ==== Protects the serial flash memory ==== */
131     sf_protect_ctrl( SF_REQ_PROTECT );
132
133     while(1){
134          /* loop */
135       }
136   }
137
138   /* End of File */
```

3.5 Sample Program Listing "qserial_flash.c" (1/2)

```
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) 2010 Renesas Electronics Corporation. All Rights Reserved.
29  *""FILE COMMENT""*********** Technical reference data *************************
30  *   System Name : SH7267 Sample Program
31  *   File Name   : qserial_flash.c
32  *   Abstract    : Interfacing Serial Flash Memory Using the Renesas Quad Serial
33  *               : Periferal Interface
34  *   Version     : 1.01.00
35  *   Device      : SH7266/SH7267
36  *   Tool-Chain  : High-performance Embedded Workshop (Ver.4.07.00).
37  *               : C/C++ compiler package for the SuperH RISC engine family
38  *               :                             (Ver.9.03 Release02).
39  *   OS          : None
40  *   H/W Platform: R0K57267(CPU board), Spansion[S25FL032P](serial flash)
41  *   Description :
42  ***********************************************************************
43  *   History     : Aug.20,2010 Ver.1.00.00
44  *               : Oct.13,2010 Ver.1.01.00 Sequence control when reading in
45  *               : quad-SPI mode modified
46  *""FILE COMMENT END""*************************************************************/
47  #include <stdio.h>
48  #include <machine.h>
49  #include "iodefine.h"
50  #include "qserial_flash.h"
51  #include "rqspi.h"
52
```

3.6 Sample Program Listing "qserial_flash.c" (2/2)

```
53    /* ==== Macro definition ==== */
54    /* ---- Serial flash memory commands [S25FL032P(Spansion)] ---- */
55    #define SFLASHCMD_CHIP_ERASE  0xc7
56    #define SFLASHCMD_SECTOR_ERASE0xd8
57    #define SFLASHCMD_BYTE_PROGRAM0x02
58    #define SFLASHCMD_BYTE_READ       0x0B
59    #define SFLASHCMD_QUAD_PROGRAM0x32
60    #define SFLASHCMD_QUAD_READ       0x6B
61    #define SFLASHCMD_WRITE_ENABLE0x06
62    #define SFLASHCMD_WRITE_DISABLE   0x04

      (omitted)

284   /*""FUNC COMMENT""*****************************************************************
285    * ID           :
286    * Outline      : Read data
287    *-------------------------------------------------------------------------------
288    * Include      :
289    *-------------------------------------------------------------------------------
290    * Declaration  : void sf_byte_read(unsigned long addr, unsigned char *buf, int size);
291    *-------------------------------------------------------------------------------
292    * Description  : Reads the specified number of bytes from the serial flash memory.
293    *-------------------------------------------------------------------------------
294    * Argument     : unsigned long addr ; I : Address in the serial flash memory to read
295    *              : unsigned char *buf ; I : Buffer address to store the read data
296    *              : int size           ; I : Number of bytes to read
297    *-------------------------------------------------------------------------------
298    * Return Value : void
299    *-------------------------------------------------------------------------------
300    * Note         : None
301    *""FUNC COMMENT END""*************************************************************/
302   void sf_byte_read(unsigned long addr, unsigned char *buf, int size)
303   {
304     unsigned char cmd[4];
305
306     cmd[0] = SFLASHCMD_QUAD_READ;
307     cmd[1] = (unsigned char)((addr >> 16) & 0xff);
308     cmd[2] = (unsigned char)((addr >>  8) & 0xff);
309     cmd[3] = (unsigned char)( addr        & 0xff);
310
311     io_rqspi_set_cmd( 0, SPI_SINGLE,  cmd, NULL, sizeof(cmd));
312     io_rqspi_set_cmd( 1, SPI_QUAD_RD, NULL, NULL, 4);  /* (2clk/byte)×4 = 8clk */
313     io_rqspi_set_cmd( 2, SPI_QUAD_RD, NULL, buf, size);
314     io_rqspi_transfer(2);
315   }

      (omitted)

476   /* End of File */
```

3.7 Sample Program Listing "qserial_flash.h" (1)

```
1   /***********************************************************************
2   *   DISCLAIMER

    (omitted)

27  ***********************************************************************
28  *   Copyright (C) 2010 Renesas Electronics Corporation. All Rights Reserved.
29  *""FILE COMMENT""*********** Technical reference data ********************
30  *   System Name : SH7267 Sample Program
31  *   File Name   : qserial_flash.h
32  *   Abstract    : Interfacing Serial Flash Memory Using the Renesas Quad Serial
33  *               : Peripheral Interface
34  *   Version     : 1.00.00
35  *   Device      : SH7266/SH7267
36  *   Tool-Chain  : High-performance Embedded Workshop (Ver.4.07.00).
37  *               : C/C++ compiler package for the SuperH RISC engine family
38  *               :                              (Ver.9.03 Release02).
39  *   OS          : None
40  *   H/W Platform: R0K57267(CPU board), Spansion[S25FL032P](serial flash)
41  *   Description :
42  ***********************************************************************
43  *   History     : Aug.20,2010 Ver.1.00.00
44  *""FILE COMMENT END""*************************************************************/
45  #ifndef _QSERIAL_FLASH_H_
46  #define _QSERIAL_FLASH_H_
47
48  /* ==== Macro definition ==== */
49  #define SF_PAGE_SIZE      256        /* Page size */
50  #define SF_SECTOR_SIZE    0x10000    /* Sector size = 64 KB   */
51  #define SF_NUM_OF_SECTOR  64         /* Number of sectors: 64   */
52
53  enum sf_req{
54    SF_REQ_PROTECT = 0,             /* Requests to protect */
55    SF_REQ_UNPROTECT,                   /* Requests to unprotect */
56    SF_REQ_SERIALMODE,                  /* Requests serial/dual mode */
57    SF_REQ_QUADMODE,                /* Requests quad mode */
58  };
59
60  /* ==== Function prototype declaration ==== */
61  void sf_init_serial_flash(void);
62  void sf_protect_ctrl(enum sf_req req);
63  void sf_set_mode(enum sf_req req);
64  void sf_chip_erase(void);
65  void sf_sector_erase(int sector_no);
66  void sf_byte_program(unsigned long addr, unsigned char *buf, int size);
67  void sf_byte_read(unsigned long addr, unsigned char *buf, int size);
68
69  #endif /* _QSERIAL_FLASH_H_ */
70  /* End of File */
```

3.8 Sample Program Listing "rqspi.c" (1/12)

```
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) 2010 Renesas Electronics Corporation. All Rights Reserved.
29    *""FILE COMMENT""*********** Technical reference data *************************
30    *   System Name : SH7267 Sample Program
31    *   File Name   : rqspi.c
32    *   Abstract    : Interfacing Serial Flash Memory Using the Renesas Quad Serial
33    *               : Peripheral Interface
34    *   Version     : 1.00.00
35    *   Device      : SH7266/SH7267
36    *   Tool-Chain  : High-performance Embedded Workshop (Ver.4.07.00).
37    *               : C/C++ compiler package for the SuperH RISC engine family
38    *               :                              (Ver.9.03 Release02).
39    *   OS          : None
40    *   H/W Platform: R0K57267(CPU board), Spansion[S25FL032P](serial flash)
41    *   Description :
42    ***********************************************************************
43    *   History     : Aug.20,2010 Ver.1.00.00
44    *""FILE COMMENT END""*****************************************************/
45    #include <stdio.h>
46    #include <machine.h>
47    #include "iodefine.h"
48    #include "rqspi.h"
```

3.9 Sample Program Listing "rqspi.c" (2/12)

```
49
50    /* ==== Macro definition ==== */
51    #define SPCR_SPI_ENABLE  0x48     /* Always write 1 to bit 3 */
52    #define SPCR_SPI_DISABLE 0x08     /* Always write 1 to bit 3 */
53    #define SPCMD_SPIRW_BIT  0x0070       /* SPI mode and read/program target bit */
54    #define SPCMD_SPB_BIT    0x0f00       /* Transfer data length (access size) target bit */
55    #define SPCMD_SPB_8BITS  0x0000
56    #define SPCMD_SPB_16BITS 0x0100
57    #define SPCMD_SPB_32BITS 0x0200
58    #define SPCMD_DEFAULT_SET 0xe087
59                  /* bit 15: Clock delay: SPCKD (1.5 QSPCLK)   */
60                  /* bit 14: QSSL negation delay: SSLND (2 QSPCLK)            */
61                  /* bit 13: Next access delay: SPND (2 QSPCLK)               */
62                  /* bit 12: Format: MSB first      */
63                  /* bits 11 to 8: Transfer data length: 8 bits               */
64                  /* bit 7: QSSL signal: Retained after the transfer is completed */
65                  /* bits 6, 5: SPI mode: Single-SPI                          */
66                  /* bit 4: Read/program: Program (invalid)                   */
67                  /* bits 3, 2: Bit rate: SPBR divided by 2 (36 Mbps)             */
68                  /* bit 1: CPOL (QSPCLK polarity): 1 when it is idling           */
69                  /* bit 0: CPHA (QSPCLK phase): Shifts data on the odd edge      */
70                  /*                           : Latches data on the even edge */
71
72    /* ---- Structure to define the RQSPI sequence control information ---- */
73    typedef struct{
74      unsigned short spcmd;           /* SPCMDn register setting */
75      unsigned long  spbmul;          /* SPBMULn register setting */
76      void    *wr_ptr;            /* Address storing the transmit data */
77      void    *rd_ptr;            /* Address storing the receive data */
78    }RQSPI_CMD_ST;
79
80    /* ==== Function prototype declaration ==== */
81    static int io_rqspi_update_SPCMD_reg( int seq );
82    static void io_rqspi_write_SPDR_reg( void *wrp, int cnt, unsigned short bitsz );
83    static void io_rqspi_read_SPDR_reg( void *rdp, int cnt , unsigned short bitsz );
84    static void io_rqspi_rdwr_SPDR_reg( void *wrp, void *rdp, int cnt, unsigned short bitsz );
85
86    /* ==== Variable definition ==== */
87    RQSPI_CMD_ST rqspi_cmd_set[4];
88
```

3.10 Sample Program Listing "rqspi.c" (3/12)

```
/*""FUNC COMMENT""*****************************************************************
 * ID          :
 * Outline     : Configure the RQSPI
 *-----------------------------------------------------------------------------
 * Include     : iodefine.h
 *-----------------------------------------------------------------------------
 * Declaration : static void io_init_rqspi(void);
 *-----------------------------------------------------------------------------
 * Description : Configures the Renesas Quad Serial Peripheral Interface.
 *             : Configures the RQSPI in master mode, and executes the transfer
 *             : setting according to the specifications of the serial flash memory.
 *-----------------------------------------------------------------------------
 * Argument    : void
 *-----------------------------------------------------------------------------
 * Return Value: void
 *-----------------------------------------------------------------------------
 * Note         : None
 *""FUNC COMMENT END""*************************************************************/
void io_rqspi_initialize(void)
{
  /* ==== Supplies the clock ==== */
  CPG.STBCR8.BIT.MSTP82 = 0;

  /* ==== Port ==== */
  PORT.PFCR3.BIT.PF12MD = 0x06u;  /* QMI/QIO1 */
  PORT.PFCR2.BIT.PF11MD = 0x06u;  /* QMO/QIO0 */
  PORT.PFCR2.BIT.PF10MD = 0x06u;  /* QSSL */
  PORT.PFCR2.BIT.PF9MD  = 0x06u;  /* QSPCLK */
  PORT.PDCR3.BIT.PD15MD = 0x03u;  /* QIO3 */
  PORT.PDCR3.BIT.PD14MD = 0x03u;  /* QIO2 */

  /* ==== Disables the SPI function and initializes the internal state ==== */
  RQSPI.SPCR.BYTE = SPCR_SPI_DISABLE;

  /* ==== Slave select polarity register (SSLP) ==== */
  RQSPI.SSLP.BIT.SSLP = 0;    /* Sets the QSSL signal to low-active */

  /* ==== Pin control register (SPPCR) ==== */
  RQSPI.SPPCR.BYTE = 0x26;    /* Sets the output pin idle value to 0 */
                              /* QIO3 is fixed to 1 in single-SPI and dual-SPI operation */
                              /* QIO2 is fixed to 1 in single-SPI and dual-SPI operation */
                              /* Normal operating mode (disables the loop-back) */
  /* ==== Buffer control register (SPBFCR) ==== */
  RQSPI.SPBFCR.BYTE = 0x20;   /* Transfer buffer empty when there are 4 bytes of empty
space */
                           /* Receive buffer full when it stores 1-byte data */
  /* ==== Bit rate register (SPBR) ==== */
  RQSPI.SPBR.BYTE = 0;        /* Base bit rate 72 Mbps (B-clock is 72 MHz) */

  /* ==== Clock delay register (SPCKD) ==== */
  RQSPI.SPCKD.BYTE = 0x00;    /* SSL setup time = 1.5 QSPCLK */
```

3.11 Sample Program Listing "rqspi.c" (4/12)

```
140     /* ==== Slave select negation delay register (SSLND) ==== */
141     RQSPI.SSLND.BYTE = 0x01;    /* SSL hold time = 2 QSPCLK */
142
143     /* ==== Next-access delay register (SPND) ==== */
144     RQSPI.SPND.BYTE = 0x01;         /* Continuous transfer delay time = 2 QSPCLK */
145
146     /* ==== Command register n (SPCMDn) ==== */
147     RQSPI.SPCMD0.WORD = SPCMD_DEFAULT_SET;     /* (Reset before the transfer) */
148     RQSPI.SPCMD1.WORD = SPCMD_DEFAULT_SET;     /* (Reset before the transfer) */
149     RQSPI.SPCMD2.WORD = SPCMD_DEFAULT_SET;     /* (Reset before the transfer) */
150     RQSPI.SPCMD3.WORD = SPCMD_DEFAULT_SET;     /* (Reset before the transfer) */
151   }
152   /*""FUNC COMMENT""*****************************************************************
153    * ID           :
154    * Outline      : Update the serquence control information
155    *-------------------------------------------------------------------------------
156    * Include      :
157    *-------------------------------------------------------------------------------
158    * Declaration  : void io_rqspi_set_cmd( int idx, unsigned short mode, void *wrp,
159    *              :                                    void *rdp, unsigned long sz)
160    *-------------------------------------------------------------------------------
161    * Description  : Updates the structure rqspi_cmd_set to define the serquence
162    *              : control information.
163    *-------------------------------------------------------------------------------
164    * Argument     : int            idx  ; I : Target command register number (0 to 3)
165    *              : unsigned short mode ; I : SPI mode and program/write setting to set the SPCMD
166    *              : void          *wrp  ; I : Address storing the transmit data
167    *              : void          *rdp  ; O : Address storing the receive data
168    *              : unsigned long  sz   ; I : Number of transmit/receive data (bytes)
169    *-------------------------------------------------------------------------------
170    * Return Value : void
171    *-------------------------------------------------------------------------------
172    * Note         :
173    *""FUNC COMMENT END""*************************************************************/
174   void io_rqspi_set_cmd( int idx, unsigned short mode, void *wrp, void *rdp, unsigned long sz)
175   {
176     RQSPI_CMD_ST *cmd = rqspi_cmd_set;
177     unsigned short bitsz = SPCMD_SPB_8BITS;
178     unsigned long  trncnt = sz;
179
```

3.12 Sample Program Listing "rqspi.c" (5/12)

```
180     /* ---- Sets the most appropriate data size ---- */
181     if( ((sz&0x3)==0) && (((int)wrp&0x3)==0) && (((int)rdp&0x3)==0) ){
182       bitsz = SPCMD_SPB_32BITS;
183       trncnt = sz >> 2;
184     }
185     else if( ((sz&0x1)==0) && (((int)wrp&0x1)==0) && (((int)rdp&0x1)==0) ){
186       bitsz = SPCMD_SPB_16BITS;
187       trncnt = sz >> 1;
188     }
189
190     cmd[idx].spcmd  = (mode | SPCMD_DEFAULT_SET | bitsz);
191     cmd[idx].spbmul = trncnt;
192     cmd[idx].wr_ptr = wrp;
193     cmd[idx].rd_ptr = rdp;
194   }
195   /*""FUNC COMMENT""*****************************************************************
196    * ID           :
197    * Outline      : RQSPI transfer
198    *-------------------------------------------------------------------------------
199    * Include      : iodefine.h
200    *-------------------------------------------------------------------------------
201    * Declaration  : int io_rqspi_transfer( int seq );
202    *-------------------------------------------------------------------------------
203    * Description  : Executes the transfer on the RQSPI.
204    *              : Use the structure rqspi_cmd_set to define the sequence control
205    *              : information, and execute the transfer on the RQSPI.
206    *              : Reads and programs the Data register as required, and transmits
207    *              : or receives the data.
208    *-------------------------------------------------------------------------------
209    * Argument     : int seq ; I : Number of command registers to use -1 (Only SPCMD0: 0)
210    *-------------------------------------------------------------------------------
211    * Return Value : 0 : Normal end
212    *-------------------------------------------------------------------------------
213    * Note         : None
214    *""FUNC COMMENT END""*************************************************************/
215   int io_rqspi_transfer( int seq )
216   {
217     int i, cnt;
218     unsigned short spirw, bitsz;
219
220     /* ==== Sets the Command register ==== */
221     rqspi_cmd_set[seq].spcmd &= ~(0x0080);     /* Adds the QSSL negation setting */
222     io_rqspi_update_SPCMD_reg(seq);
223
```

3.13 Sample Program Listing "rqspi.c" (6/12)

```
224     /* ==== Enables the SPI transfer ==== */
225     RQSPI.SPCR.BYTE = SPCR_SPI_ENABLE;
226
227     /* ==== Reads and programs the Data register ==== */
228     for( i=0; i<=seq; i++){
229       bitsz = rqspi_cmd_set[i].spcmd & SPCMD_SPB_BIT;
230       spirw = rqspi_cmd_set[i].spcmd & SPCMD_SPIRW_BIT;
231       cnt   = rqspi_cmd_set[i].spbmul;
232
233       /* ---- Programming in half-duplex  --- */
234       if( spirw==SPI_QUAD_WR || spirw==SPI_DUAL_WR ){
235           io_rqspi_write_SPDR_reg( rqspi_cmd_set[i].wr_ptr, cnt, bitsz);
236       }
237       /* ---- Reading in half-duplex --- */
238       else if( spirw==SPI_QUAD_RD || spirw==SPI_DUAL_RD ){
239           io_rqspi_read_SPDR_reg( rqspi_cmd_set[i].rd_ptr, cnt, bitsz);
240       }
241       /* ---- Full-duplex --- */
242       else{
243           io_rqspi_rdwr_SPDR_reg( rqspi_cmd_set[i].wr_ptr, rqspi_cmd_set[i].rd_ptr, cnt, bitsz);
244       }
245     }
246     /* ==== Waits until the transfer is completed (Until the QSSL is negated) ==== */
247     while( PORT.PFPR0.BIT.PF10PR == 0 ){
248       /* wait */
249     }
250     /* ==== SPI transfer is completed ==== */
251     RQSPI.SPCR.BYTE = SPCR_SPI_DISABLE;
252   }
```

3.14 Sample Program Listing "rqspi.c" (7/12)

```
253  /*""FUNC COMMENT""*************************************************************
254   * ID           :
255   * Outline      : Update the Command register
256   *-----------------------------------------------------------------------------
257   * Include      : iodefine.h
258   *-----------------------------------------------------------------------------
259   * Declaration  : static int io_rqspi_update_SPCMD_reg( int seq );
260   *-----------------------------------------------------------------------------
261   * Description  : Propagates the sequence control information to the Sequence
262   *              : control register and Command register.
263   *-----------------------------------------------------------------------------
264   * Argument     : int seq ; I : Number of Command registers to use -1 (Only SPCMD0: 0)
265   *-----------------------------------------------------------------------------
266   * Return Value : 0 : Normal end
267   *-----------------------------------------------------------------------------
268   * Note         : None
269   *""FUNC COMMENT END""*********************************************************/
270  static int io_rqspi_update_SPCMD_reg( int seq )
271  {
272    RQSPI_CMD_ST *cmd = rqspi_cmd_set;
273
274    /* ==== Sets the Sequence control register ==== */
275    RQSPI.SPSCR.BYTE = seq;
276
277    /* ==== Sets the Command register ==== */
278    /* ---- CMD3 ---- */
279    RQSPI.SPCMD3.WORD  = cmd[3].spcmd;
280    RQSPI.SPBMUL3.LONG = cmd[3].spbmul;
281    /* ---- CMD2 ---- */
282    RQSPI.SPCMD2.WORD  = cmd[2].spcmd;
283    RQSPI.SPBMUL2.LONG = cmd[2].spbmul;
284    /* ---- CMD1 ---- */
285    RQSPI.SPCMD1.WORD  = cmd[1].spcmd;
286    RQSPI.SPBMUL1.LONG = cmd[1].spbmul;
287    /* ---- CMD0 ---- */
288    RQSPI.SPCMD0.WORD  = cmd[0].spcmd;
289    RQSPI.SPBMUL0.LONG = cmd[0].spbmul;
290
291    return 0;
292  }
```

3.15 Sample Program Listing "rqspi.c" (8/12)

```
/*""FUNC COMMENT""*************************************************************
 * ID           :
 * Outline      : Program the Data register (half-duplex).
 *-----------------------------------------------------------------------------
 * Include      : iodefine.h
 *-----------------------------------------------------------------------------
 * Declaration  : static void io_rqspi_write_SPDR_reg( void *wrp, int cnt,
 *              :                                      unsigned short bitsz );
 *-----------------------------------------------------------------------------
 * Description  : Program the data stored in the argument wrp to the Data register
 *              : in size specified by the argument bitsz. Continues the processing
 *              : for the number of times specified by the argument cnt.
 *-----------------------------------------------------------------------------
 * Argument     : void          *wrp   ; I : Pointer to the program data
 *              : int            cnt   ; I : Number of times to program
 *              : unsigned short bitsz ; I : Access size to the Data register
 *-----------------------------------------------------------------------------
 * Return Value : void
 *-----------------------------------------------------------------------------
 * Note         : None
 *""FUNC COMMENT END""*********************************************************/
static void io_rqspi_write_SPDR_reg( void *wrp, int cnt, unsigned short bitsz )
{
  if( bitsz == SPCMD_SPB_32BITS ){
    unsigned long *wrp_l = (unsigned long *)wrp;
    while( cnt-- ){
        while(RQSPI.SPSR.BIT.SPTEF == 0){
            /* Waits until the transmit buffer is empty */
        }
        RQSPI.SPDR.LONG = *wrp_l++;
    }
  }
  else if( bitsz == SPCMD_SPB_16BITS ){
    unsigned short *wrp_w = (unsigned short *)wrp;
    while( cnt-- ){
        while(RQSPI.SPSR.BIT.SPTEF == 0){
            /* Waits untl the transmit buffer is empty */
        }
        RQSPI.SPDR.WORD = *wrp_w++;
    }
  }
  else{
    unsigned char *wrp_c = (unsigned char *)wrp;
    while( cnt-- ){
        while(RQSPI.SPSR.BIT.SPTEF == 0){
            /* Waits until the transmit buffer is empty */
        }
        RQSPI.SPDR.BYTE = *wrp_c++;
    }
  }
}
```

3.16 Sample Program Listing "rqspi.c" (9/12)

```
/*""FUNC COMMENT""*****************************************************************
 * ID           :
 * Outline      : Read the Data register (half-duplex).
 *-----------------------------------------------------------------------------
 * Include      : iodefine.h
 *-----------------------------------------------------------------------------
 * Declaration  : static void io_rqspi_read_SPDR_reg( void *rdp, int cnt,
 *              :                                     unsigned short bitsz );
 *-----------------------------------------------------------------------------
 * Description  : Reads the Data register in size specified by the argument bitsz,
 *              : and stores the data in the buffer area specified by the argument
 *              : rdp. Continues the processing for the number of times specified
 *              : by the argument cnt.
 *-----------------------------------------------------------------------------
 * Argument     : void          *rdp   ; I : Buffer address to store the read data
 *              : int            cnt   ; I : Number of times to read
 *              : unsigned short bitsz ; I : Access size to the Data register
 *-----------------------------------------------------------------------------
 * Return Value : void
 *-----------------------------------------------------------------------------
 * Note         : None
 *""FUNC COMMENT END""*************************************************************/
static void io_rqspi_read_SPDR_reg( void *rdp, int cnt , unsigned short bitsz )
{
  if( bitsz == SPCMD_SPB_32BITS ){
    unsigned long *rdp_l = (unsigned long *)rdp;
    while( cnt-- ){
        while( RQSPI.SPSR.BIT.SPRFF == 0 ){
            /* Waits until the receive buffer is full */
        }
        *rdp_l++ = RQSPI.SPDR.LONG;
    }
  }
  else if( bitsz == SPCMD_SPB_16BITS ){
    unsigned short *rdp_w = (unsigned short *)rdp;
    while( cnt-- ){
        while( RQSPI.SPSR.BIT.SPRFF == 0 ){
            /* Waits until the receive buffer is full */
        }
        *rdp_w++ = RQSPI.SPDR.WORD;
    }
  }
  else{
    unsigned char *rdp_c = (unsigned char *)rdp;
    while( cnt-- ){
        while( RQSPI.SPSR.BIT.SPRFF == 0 ){
            /* Waits until the receive buffer is full */
        }
        *rdp_c++ = RQSPI.SPDR.BYTE;
    }
  }
}
```

3.17 Sample Program Listing "rqspi.c" (10/12)

```
/*""FUNC COMMENT""*****************************************************************
 * ID           :
 * Outline      : Read or program the Data register (full-duplex).
 *-----------------------------------------------------------------------------
 * Include      : iodefine.h
 *-----------------------------------------------------------------------------
 * Declaration  : static void io_rqspi_rdwr_SPDR_reg( void *wrp, void *rdp,
 *              :                                     int cnt, unsigned short bitsz );
 *-----------------------------------------------------------------------------
 * Description  : Reads the Data register in size specified by the argument bitsz,
 *              : and stores the data in the buffer area specified by the argument
 *              : rdp. Continues the processing for the number of times specified
 *              : by the argument cnt.
 *-----------------------------------------------------------------------------
 * Argument     : void          *wrp   ; I : Pointer to the program data
 *              : void          *rdp   ; I : Buffer address to store the read data
 *              : int            cnt   ; I : Number of times to program and read
 *              : unsigned short bitsz ; I : Access size to the Data register
 *-----------------------------------------------------------------------------
 * Return Value : void
 *-----------------------------------------------------------------------------
 * Note         : None
 *""FUNC COMMENT END""*************************************************************/
static void io_rqspi_rdwr_SPDR_reg( void *wrp, void *rdp, int cnt, unsigned short bitsz )
{
  unsigned long tmp = 0;

  /* ==== When the access size is 32-bit ==== */
  if( bitsz == SPCMD_SPB_32BITS ){
    unsigned long *wrp_l = (unsigned long *)wrp;
    unsigned long *rdp_l = (unsigned long *)rdp;
    while( cnt-- ){
        /* ---- Programs the data ---- */
        while(RQSPI.SPSR.BIT.SPTEF == 0){
            /* Waits until the transmit buffer is empty */
        }
        if( wrp != NULL){
            RQSPI.SPDR.LONG = *wrp_l++;
        }
        else{
            RQSPI.SPDR.LONG = tmp; /* Transmits the dummy data */
        }
        /* ---- Reads the data ---- */
        while( RQSPI.SPSR.BIT.SPRFF == 0 ){
            /* Waits until the receive buffer is full */
        }
        if(rdp != NULL){
            *rdp_l++ = RQSPI.SPDR.LONG;
        }
```

3.18 Sample Program Listing "rqspi.c" (11/12)

```
445           else{
446               tmp = RQSPI.SPDR.LONG; /* Receives the dummy data */
447           }
448       }
449     }
450     /* ==== When the access size is 16-bit ==== */
451     else if( bitsz == SPCMD_SPB_16BITS ){
452       unsigned short *wrp_w = (unsigned short *)wrp;
453       unsigned short *rdp_w = (unsigned short *)rdp;
454       while( cnt-- ){
455           /* ---- Programs the data ---- */
456           while(RQSPI.SPSR.BIT.SPTEF == 0){
457               /* Waits until the transmit buffer is empty */
458           }
459           if( wrp != NULL){
460               RQSPI.SPDR.WORD = *wrp_w++;
461           }
462           else{
463               RQSPI.SPDR.WORD = (unsigned short)tmp;    /* Transmits the dummy data */
464           }
465           /* ---- Reads the data ---- */
466           while( RQSPI.SPSR.BIT.SPRFF == 0 ){
467               /* Waits until the receive buffer is full */
468           }
469           if(rdp != NULL){
470               *rdp_w++ = RQSPI.SPDR.WORD;
471           }
472           else{
473               tmp = RQSPI.SPDR.WORD; /* Receives the dummy data */
474           }
475       }
476     }
477     /* ==== When the access size is 8-bit ==== */
478     else{
479       unsigned char *wrp_c = (unsigned char *)wrp;
480       unsigned char *rdp_c = (unsigned char *)rdp;
481       while( cnt-- ){
482           /* ---- Programs the data ---- */
483           while(RQSPI.SPSR.BIT.SPTEF == 0){
484               /* Waits until the transmit buffer is empty */
485           }
486           if( wrp != NULL){
487               RQSPI.SPDR.BYTE = *wrp_c++;
488           }
489           else{
490               RQSPI.SPDR.BYTE = (unsigned char)tmp; /* Transmits the dummy data */
491           }
```

3.19 Sample Program Listing "rqspi.c" (12/12)

```
492           /* ---- Reads the data ---- */
493           while( RQSPI.SPSR.BIT.SPRFF == 0 ){
494               /* Waits until the receive buffer is full */
495           }
496           if(rdp != NULL){
497               *rdp_c++ = RQSPI.SPDR.BYTE;
498           }
499           else{
500               tmp = RQSPI.SPDR.BYTE; /* Receives the dummy data */
501           }
502       }
503     }
504   }
505   /* End of File */
```

3.20 Sample Program Listing "rqspi.h" (1/1)

```
/***********************************************************************
*   DISCLAIMER
(omitted)


***********************************************************************
*   Copyright (C) 2010 Renesas Electronics Corporation. All Rights Reserved.
*""FILE COMMENT""*********** Technical reference data *************************
*   System Name : SH7267 Sample Program
*   File Name   : rqspi.h
*   Abstract    : Interfacing Serial Flash Memory Using the Renesas Quad Serial
*               : Peripheral Interface
*   Version     : 1.00.00
*   Device      : SH7266/SH7267
*   Tool-Chain  : High-performance Embedded Workshop (Ver.4.07.00).
*               : C/C++ compiler package for the SuperH RISC engine family
*               :                             (Ver.9.03 Release02).
*   OS          : None
*   H/W Platform: R0K57267(CPU board), Spansion[S25FL032P](serial flash)
*   Description :
***********************************************************************
*   History     : Aug.20,2010 Ver.1.00.00
*""FILE COMMENT END""*****************************************************/
#ifndef _RQSPI_H_
#define _RQSPI_H_

/* ==== Macro definition ==== */
#define SPI_SINGLE 0x0000  /* Single-SPI mode */
#define SPI_DUAL_WR    0x0020  /* Dual-SPI mode, programming */
#define SPI_DUAL_RD    0x0030  /* Dual-SPI mode, reading */
#define SPI_QUAD_WR    0x0040  /* Quad-SPI mode, programming */
#define SPI_QUAD_RD    0x0050  /* Quad-SPI mode, reading */

/* ==== Funcion prototype declaration ==== */
void io_rqspi_initialize(void);
void io_rqspi_set_cmd( int idx, unsigned short mode, void *wrp, void *rdp, unsigned long sz);
int io_rqspi_transfer( int seq );

#endif /* _RQSPI_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 Hardware Manual
  SH7266 Group, SH7267 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	Dec.27.10	—	First edition issued

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: +49-211-6503-0, 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.
7F, No. 363 Fu Shing North Road Taipei, Taiwan, R.O.C.
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 1.0