To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

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

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

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.

SH7262/SH7264 Group

Interfacing Serial Flash Memory

Using the Renesas Serial Peripheral Interface

Summary

This application note describes how to connect serial flash memory using the SH7262/SH7264 Microcomputers (MCUs) Renesas Serial Peripheral Interface (RSPI).

Target Device

SH7262/SH7264 MCU (In this document, SH7262/SH7264 are described as "SH7264").

Contents

1. Introduction

1.1 Specifications

- Use the serial flash memory of 2 MB (64 KB x 32 sectors, 256 bytes per page) to connect with the SH7264 MCU.
- Use channel 0 of the RSPI to access serial flash memory.

1.2 Modules Used

- Renesas Serial Peripheral Interface (RSPI)
- General-purpose I/O ports

1.3 Applicable Conditions

MCU	SH7262/SH7264
Operating Frequency	Internal clock: 144 MHz Bus clock: 72 MHz Peripheral clock: 36 MHz
Integrated Development Environment	Renesas Technology Corp. High-performance Embedded Workshop Ver.4.04.01
C compiler	Renesas Technology SuperH RISC engine Family C/C++ compiler package Ver.9.02 Release 00
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)

1.4 Related Application Note

Refer to the related application notes as follows:

- SH7262/SH7264 Group Example of Initialization
- SH7262/SH7264 Group Boot from the Serial Flash Memory

2. Applications

Connect the SH7264 MCU (Master) with the SPI-compatible serial flash memory (Slave) for read/write access using the Renesas Serial Peripheral Interface (RSPI). This chapter describes the pin connection example and flow charts of the sample program.

2.1 RSPI Operation

SH7264 RSPI allows full-duplex, synchronous, serial communications with peripheral devices in SPI operation using the MOSI (Master Out Slave In), MISO (Master In Slave Out), SSL (Slave Select), and RSPCK (SPI Clock) pins.

The RSPI has the following features to support SPI-compliant devices:

- Master/slave modes
- Serial transfer clock with programmable polarity and phase (change SPI modes)
- Transfer bit length selectable (8-bit, 16-bit, and 32-bit)

As the RSPI has two channels, channel 0 and channel 1, this application uses channel 0.

2.2 Serial Flash Memory Pin Connection

The following table lists the specifications of the SPI-compliant serial flash memory (AT26DF161A, ATMEL) used in this application.

Table 1 Serial Flash Memory Specifications

Item	Description
SPI modes	Supports SPI modes 0 and 3
Clock frequency	70 MHz (at maximum)
Capacity	2 MB
Sector size	64 KB
Page size	256 bytes
Erase architecture	Chip Erase, 64 KB, 32 KB, 4 KB
Programming options	Byte/Page Program (1 to 256 bytes), Sequential Program
Protect feature	In sectors

The figure below shows an example of serial flash memory circuit. Set the SH7264 pin functions as shown in Table 2.

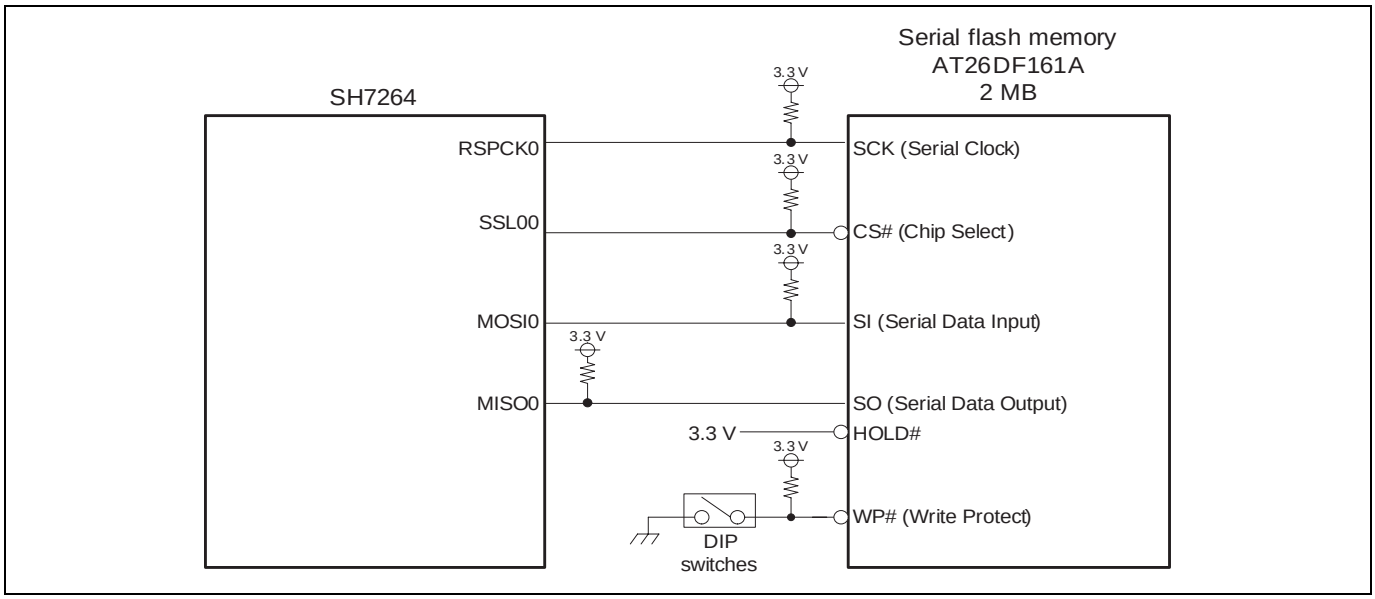


Figure 1 Serial Flash Memory Circuit

Note: Pull-up or pull-down the control signal pins by the external resistor
To pull up or pull down the control signal pins, determine the signal line level not to cause the external device malfunction when the MCU pin status is in high-impedance. SSL00 pin is pulled up by the external resistor to High-level. Pull up or down the RSPCK0 and MOSI0 pins. As the MISO0 pin is an input pin, pull up or down it to avoid floating to the midpoint voltage.

Table 2 Multiplexed Output

Peripheral Functions	Pin Name	SH7264 Port Control Register		SH7264 Multiplexed Pin Name
		Register Name	MD bit Setting	
RSPI	MISO0	PFCR3	PF12MD[2:0] = B'011	PF12/BS#/MISO0/TIOC3D/SPDIF_OUT
	MOSI0	PFCR2	PF11MD[2:0] = B'011	PF11/A25/SSIDATA3/MOSI0/TIOC3C/SPDIF_IN
	SSL00	PFCR2	PF10MD[2:0] = B'011	PF10/A24/SSIWS3/SSL00/TIOC3B/FCE#
	RSPCK0	PFCR2	PF9MD[2:0] = B'011	PF9/A23/SSISCK3/RSPCK0/TIOC3A/FRB

Note: SH7264 Multiplexed Pins
MISO0, MOSI0, SSL00, and RSPCK0 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 RSPI pins.

2.3 Interface Timing Example

This section describes an example of the interface timing between the SH7264 and serial flash memory. Initialize the RSPI and the clock frequency according to serial flash memory, which is used as a slave device.

Figure 2 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 the CPOL and CPHA bits in the command register (SPCMD). By this setting, RSPCK is specified to 1 when it is idling, and the timing to vary the data in the RSPI can be set to the odd edge (falling edge). Initialize the RSPI to satisfy the timing conditions shown in Table 3 and Table 4.

The bit rate is set to 18 Mbps, the access width of the data register (SPDR) is set to 8-bit in this application. Optimizing these settings allows for accessing serial flash memory in high-speed. The setup time may not be enough with the transfer setting shown in Figure 2. Extend the cycle between data output and data latch to one cycle when the setup time is not enough.

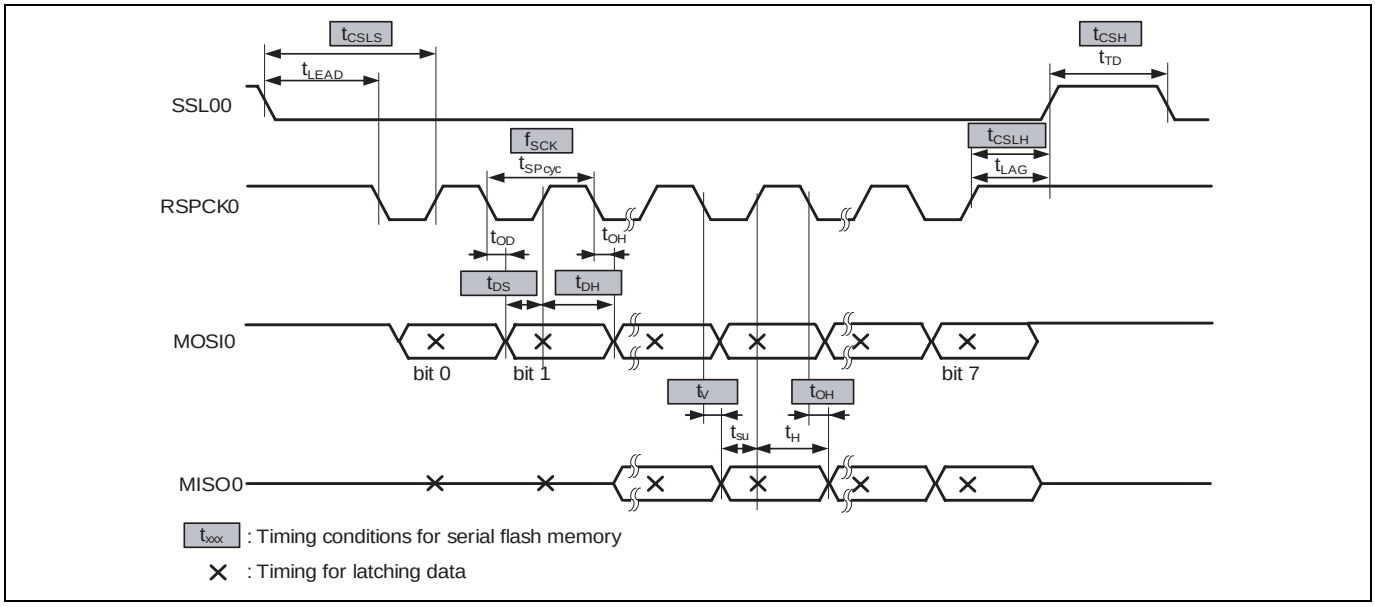


Figure 2 Data Transfer Timing Example (CPOL = 1, CPHA =1)

Table 3 Timing Conditions for Serial Flash Memory when Transferring Data

Symbol	Item	Description	Related registers
t_{CSLS}	Chip Select Low Setup Time	Time required for the slave device to latch data from asserting SSL to the RSPCK rising. The following formula must be fulfilled: t_{LEAD} (=RSPCK delay) + 1/2 x t_{SPcyc} > t_{CSLS} (min)	SPCKD register SPCMD register
t_{CSH}	Chip Select High Time	Time required for SSL negation. The following formula must be fulfilled: t_{TD} (=2 x Bφ + next access delay) > t_{CSH} (min)	SPND register SPCMD register
f_{SCK}	Serial Clock Frequency	The maximum operating frequency supported by the slave device. The following formula must be fulfilled: f_{SCK}(max) > 1/ t_{SPcyc}	SPBR register SPCMD register
t_{CSLH}	Chip select Low Hold Time	Hold time required from the last RSPCK rising to the SSL negation. The following formula must be fulfilled: t_{LAG} (=SSL negation delay) > t_{CSLH} (min)	SSLND register SPCMD register
t_{DS}	Data Input Setup Time	Time required for the master device from outputting data to latching data. The following formula must be fulfilled: 1/2 x t_{SPcyc} - t_{OD}(max) > t_{DS} (min)	
t_{DH}	Data Input Hold Time	Time required for the master device from latching data to stop the data output. The following formula must be fulfilled: t_{OH}(min) + 1/2 x t_{SPcyc} > t_{DH} (min)	

Table 4 Timing Conditions for the SH7264 MCU when Transferring Data

Symbol	Item	Description	Related registers
t_{SU}	Data Input Setup Time	Time required for the slave device from outputting data to latching data. The following formula must be fulfilled: 1/2 x t_{SPcyc} – t_{V} (max) > t_{SU} (min)	
t_{H}	Data Input Hold Time	Time required for the slave device from latching data to stop the data output. The following formula must be fulfilled: t_{OH}(min) + 1/2 x t_{SPcyc} > t_{H}(min)	

2.4 Sample Program Operation

2.4.1 RSPI Initialization Example

Figure 3 and Figure 4 show flow charts of initializing the RSPI in the sample program. This setting enables the SPI operation in master mode.

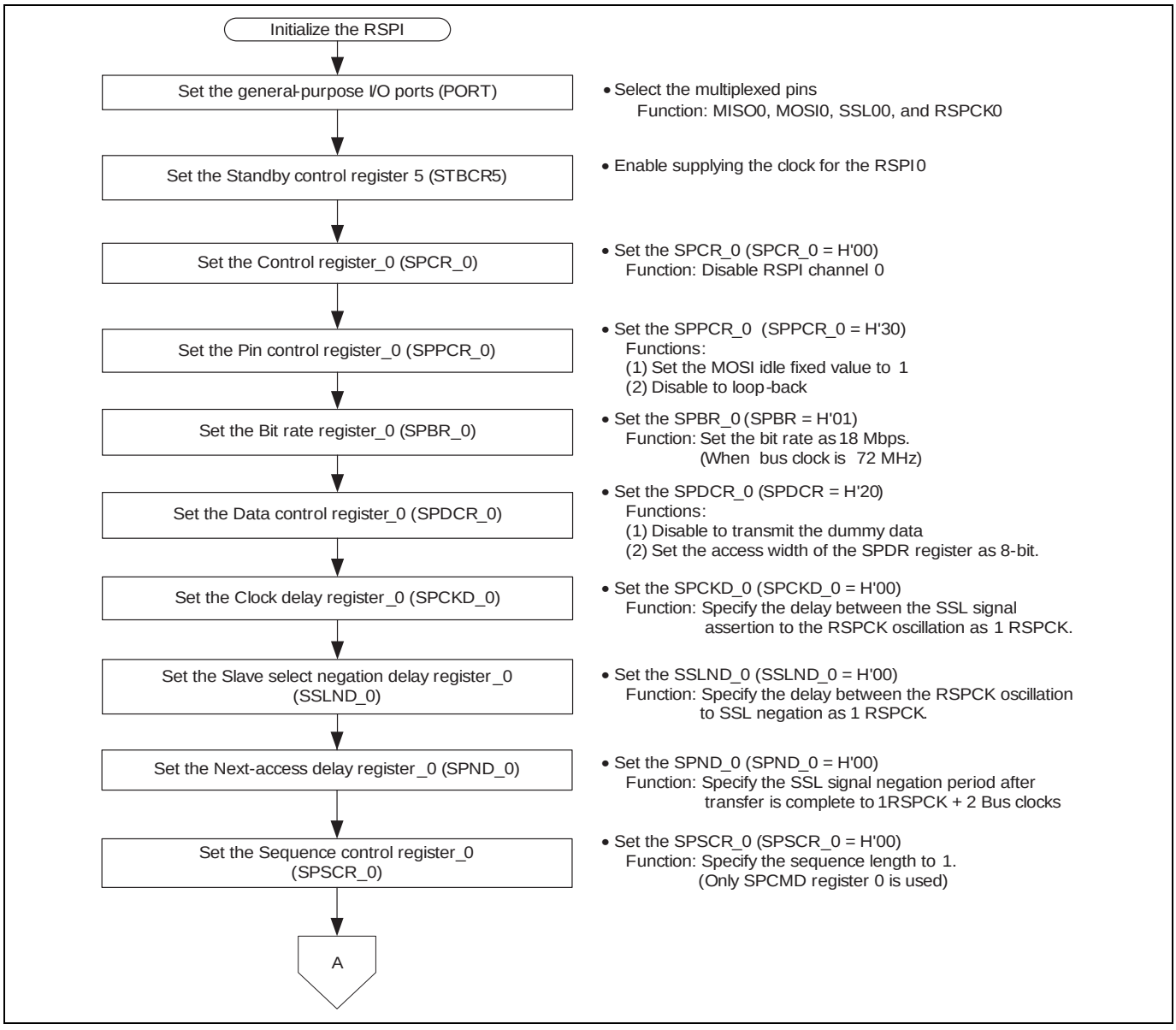


Figure 3 RSPI Initialization Flow Chart (1/2)

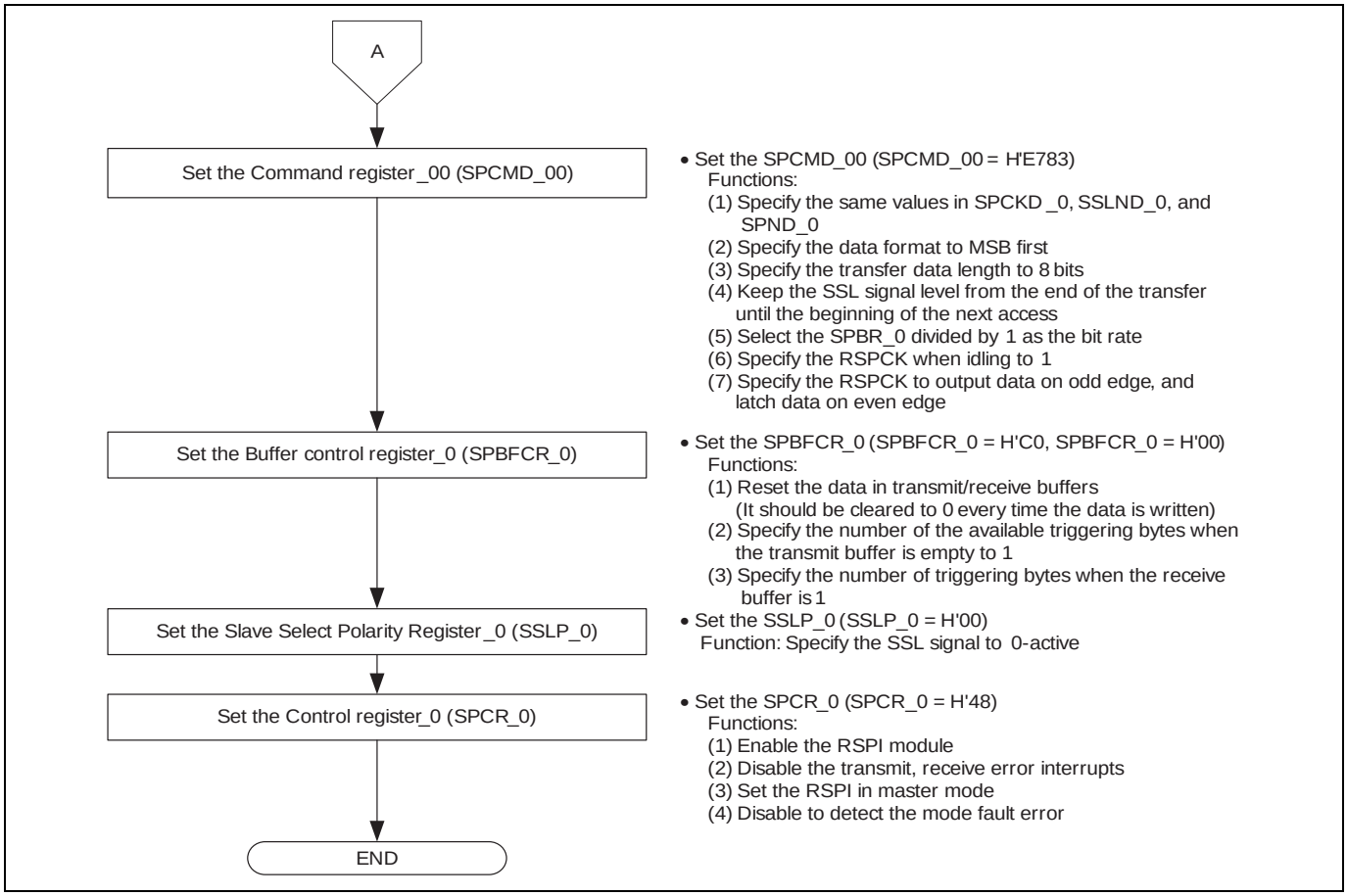


Figure 4 RSPI Initialization Flow Chart (2/2)

2.4.2 Command Transfer Example

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

This application refers to the commands of the ATMEL AT26DF161A. For details on commands, refer to the datasheet provided by the serial flash memory manufacturer.

A. Major Commands

The following table lists the major commands for the AT26DF161A.

Table 5 AT26DF161A Major Commands

Command Name	Opcode	Address Bytes	Dummy Bytes	Data Bytes	Function
Read Array	H'0B	3	1	1+ (1)	Reads the data
Read Array (Low Frequency)	H'03	3	0	1+ (1)	Reads the data (reading in high-speed for low frequency)
Write Enable	H'06	0	0	0	Enables the program/erase command
Write Disable	H'04	0	0	0	Disables the program/erase command
Block Erase (64 Kbytes)	H'D8	3	0	0	Erases the data in blocks (64 KB)
Chip Erase	H'C7	0	0	0	Erases the entire memory array
Byte/Page Program	H'02	3	0	1+ (2)	Programs the data
Read Status Register	H'05	0	0	1+	Reads the status register
Write Status Register	H'01	0	0	1	Writes the data in the status 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) Writes 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 5 shows the sequence example of the Read Array (Low Frequency) command.

When issuing the Read Array (Low Frequency) command, the master device transfers the opcode (H'03) and three address bytes after the SSL signal is asserted. Then, the slave device transfers the read data in every falling edge of the RSPCK.

Although commands can be sequentially issued by repeating to transfer the data in the specified access width, pay special attention to the SSL signal level. Do not negate the SSL signal between the assertion of the SSL signal at the beginning of the command and the transfer end of the last byte of the command. The sample program sets the SSLKP bit in the SPCMD register to 1 to keep the SSL signal. SSL signal is negated by clearing the SPE bit in the SPCR register to 0 after all data transfer is complete.

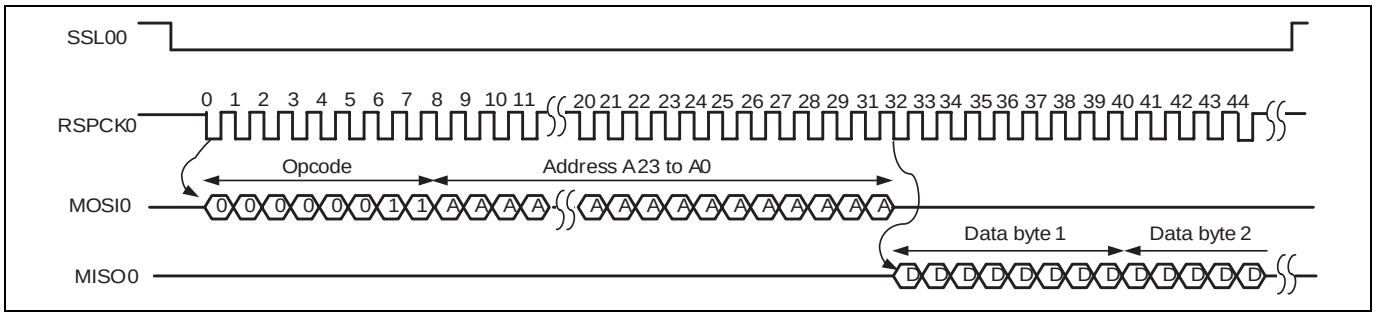


Figure 5 Read Command Sequence (Opcode: H'03)

C. Command Transfer Example in the Sample Program

The Read command that uses both master output and slave output, and the Write command that uses the master output are supported by the sample program. Figure 6 shows the flow chart of the read command transfer. The Read Array (Low Frequency) command follows this flow chart. Figure 7 shows the flow chart of the write command transfer.

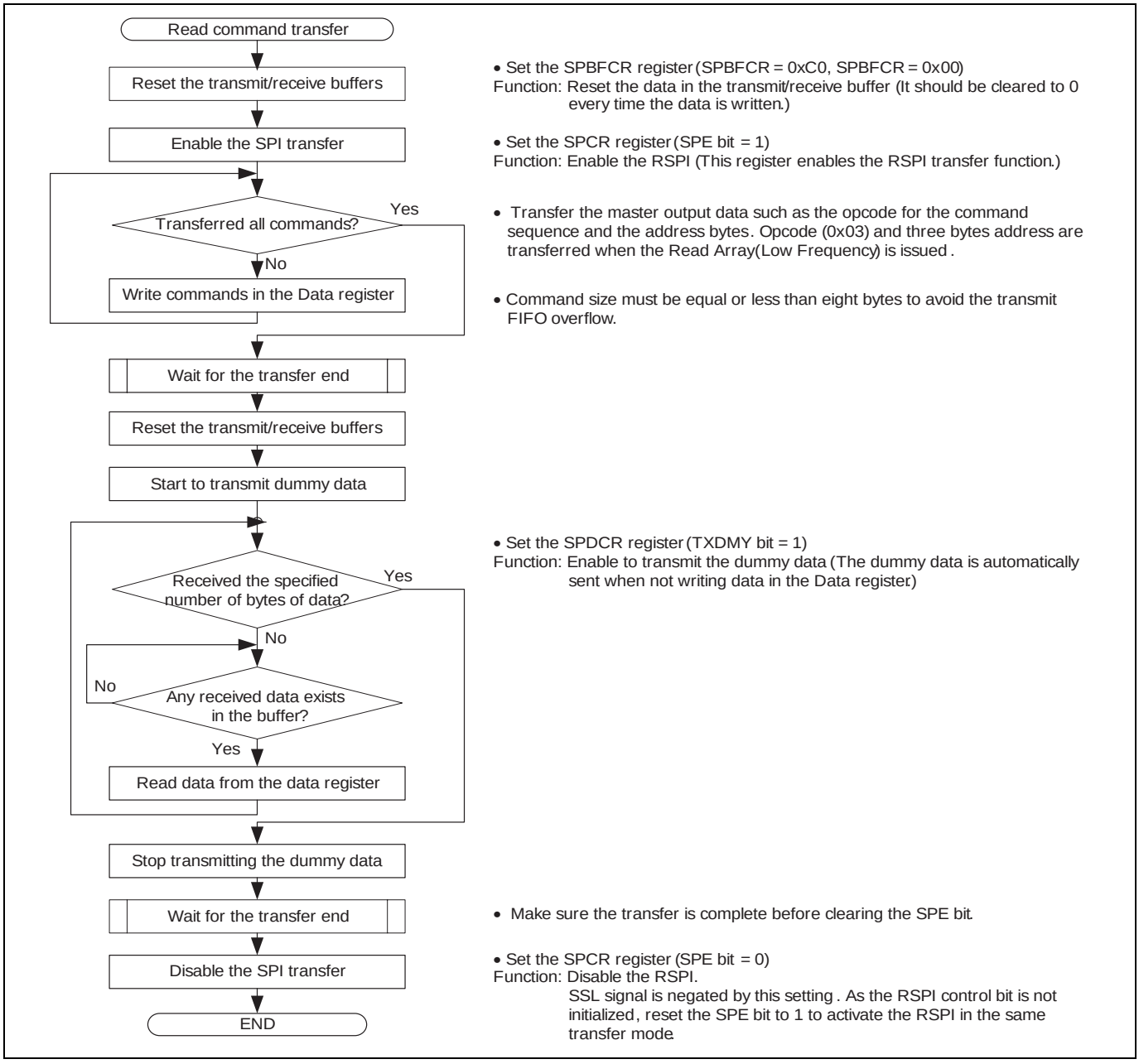
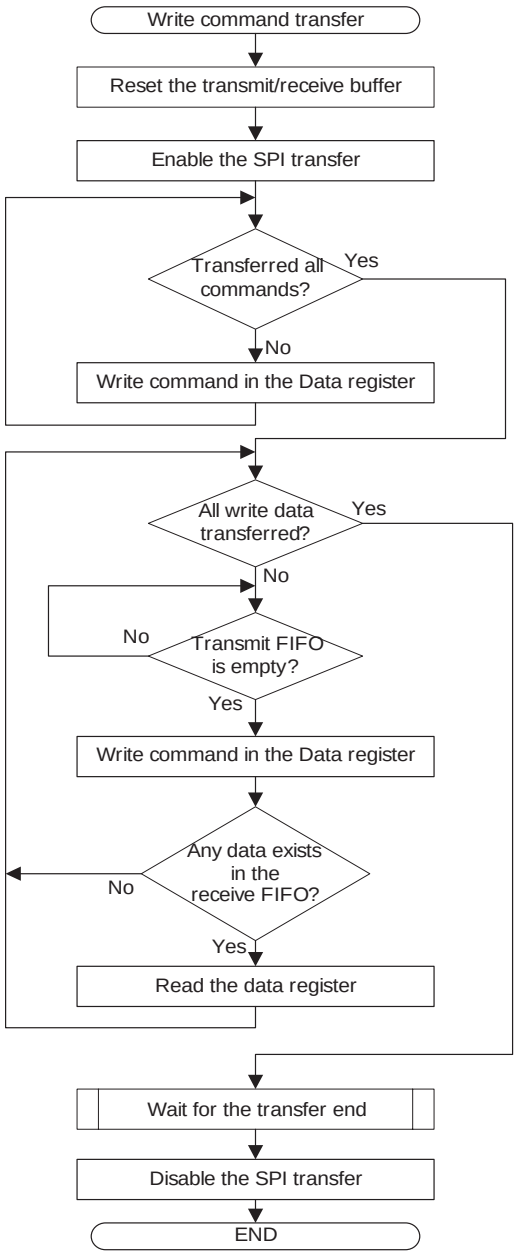


Figure 6 Flow Chart of the Read Command Transfer



• Set the SPBFCR register (SPBFCR = 0xC0, SPBFCR = 0x00)
Function: Reset the transmit/receive buffer (It should be cleared to 0 every time the data is written.)

• Set the SPCR register (SPE bit = 1)
Function: Enable the RSPI
(This register enables the RSPI transfer function. As there is a register not allowing to write data when the SPE bit is 1, pay close attention when setting this register.) The Command register (SPCMD) does not restrict the SPI value when the TEND bit is 1.

• Transfer the master output data such as the opcode for the command sequence and the address bytes

• Command size must be equal or less than eight bytes to avoid the transmit FIFO overflow.

Read the dummy data as when the receive buffer overflows upon the RSPI is operating as the master device, the transfer stops.

• Set the SPCR register (SPE bit = 0)
Function: Disable the RSPI
This register disables the RSPI transfer function. The SSL signal is negated by this setting.

Figure 7 Flow Chart of the Write Command Transfer

2.4.3 Main Function

The figure below shows the flow chart of the main function in the sample program. The sample program writes data in the entire memory array, and compares the written value to the read value.

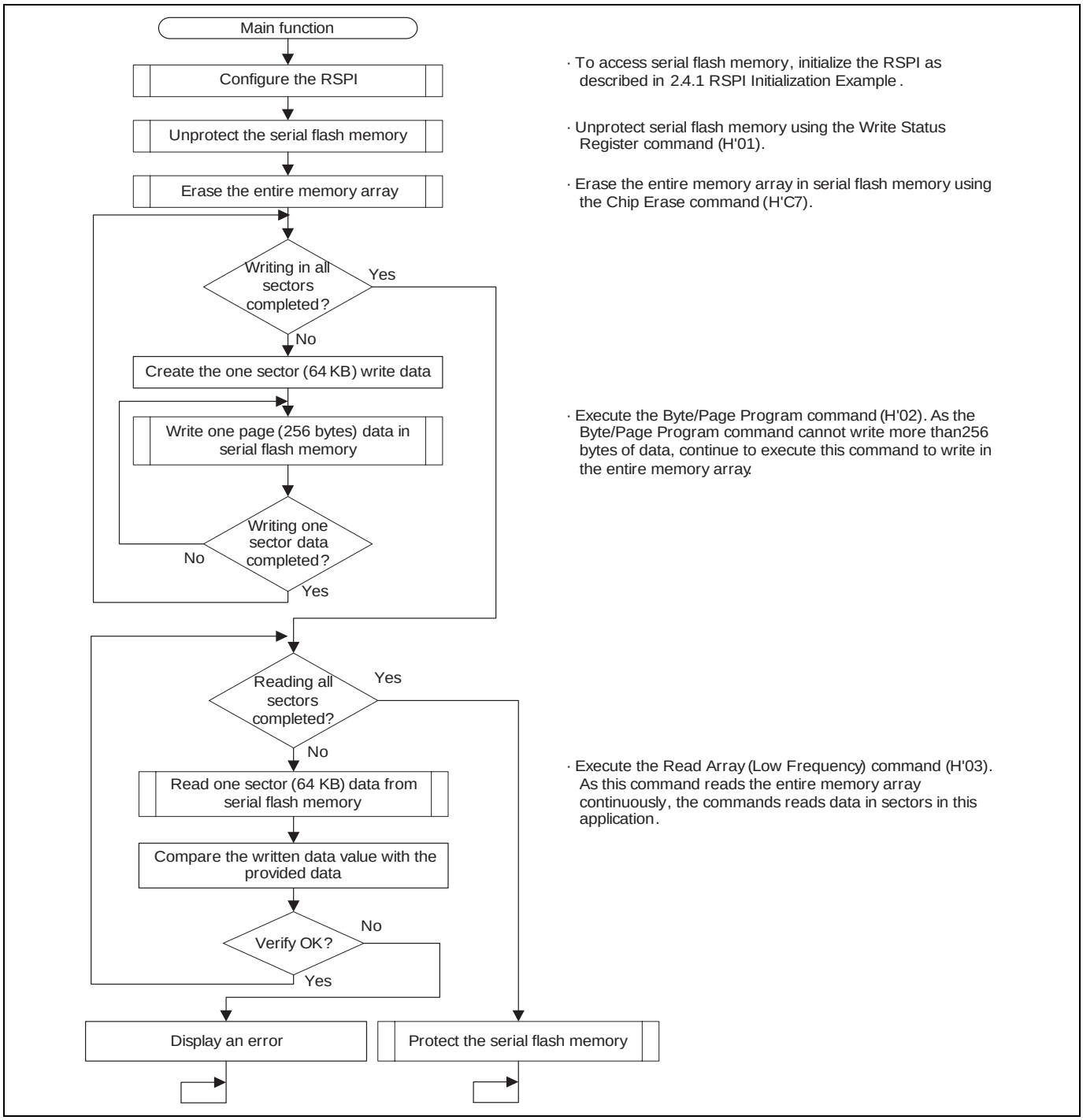


Figure 8 Main Function Flow Chart in the Sample Program

3. Sample Program Listing

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

```
1       /*""FILE COMMENT""*********** Technical reference data ************************
2       *
3       *       System Name : SH7264 Sample Program
4       *       File Name   : main.c
5       *       Abstract    : Interfacing Serial Flash Memory Using the Renesas Serial
6       *                   : Peripheral Interface
7       *       Version     : 1.00.00
8       *       Device      : SH7262/SH7264
9       *       Tool-Chain  : High-performance Embedded Workshop (Version 4.04.01)
10      *                   : C/C++ compiler package for the SuperH RISC engine family
11      *                   :                              (Ver.9.02 Release00).
12      *       OS          : None
13      *       H/W Platform: M3A-HS64G50 (CPU board)
14      *       Disclaimer  :
15      *
16      *       The information described here may contain technical inaccuracies or
17      *       typographical errors. Renesas Technology Corporation and Renesas Solutions
18      *       assume no responsibility for any damage, liability, or other loss rising
19      *       from these inaccuracies or errors.
20      *
21      *       Copyright (C) 2009 Renesas Technology Corp. All Rights Reserved
22      *       AND Renesas Solutions Corp. All Rights Reserved
23      *
24      *       History     : Feb.23,2009 Ver.1.00.00
25      *""FILE COMMENT END""*************************************************************/
26      #include <stdio.h>
27      #include "serial_flash.h"
28
29      /* ==== Macro definition ==== */
30      #define TOP_ADDRESS    0             /* Start address of serial flash memory */
31
32      /* ==== Function prototype declaration ==== */
33      void main(void);
34
35      /* ==== Variable definition ==== */
36      #pragma section DEBUG_128K_BYTES
37      static unsigned char data[SF_SECTOR_SIZE];
38      static unsigned char rbuf[SF_SECTOR_SIZE];
39      #pragma section
40
```

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

```
41    /*""FUNC COMMENT""*****************************************************************
42     * ID          :
43     * Outline     : Accessing serial flash memory main
44     *-------------------------------------------------------------------------------
45     * Include     :
46     *-------------------------------------------------------------------------------
47     * Declaration : void main(void);
48     *-------------------------------------------------------------------------------
49     * Function    : Erases, programs, and reads serial flash memory.
50     *             : After initializing the RSPI channel 0, erases the entire memory
51                   : array, and writes data from the start address. Reads the
52                   : written data to compare to the provided data.
53     *-------------------------------------------------------------------------------
54     * Argument    : void
55     *-------------------------------------------------------------------------------
56     * Return Value: void
57     *""FUNC COMMENT END""*************************************************************/
58    void main(void)
59    {
60      int i, j;
61      static unsigned long addr;
62
63      /* ==== Initializes the RSPI ==== */
64      sf_init_serial_flash();
65
66      /* ==== Unprotects serial flash memory ==== */
67      sf_protect_ctrl( SF_REQ_UNPROTECT );
68
69      /* ==== Chip erase (2 MB, it takes about 10 seconds to complete) ==== */
70      sf_chip_erase();
71
72      /* ==== Writes data (2 MB, it takes about 10 seconds to complete) ==== */
73      addr = TOP_ADDRESS;
74      for(i = 0; i < SF_NUM_OF_SECTOR; i++){
75        /* ---- Initializes the data (64 KB) ---- */
76        for(j = 0; j < SF_SECTOR_SIZE; j++){
77            data[j] = (i + j) % 100;
78        }
79        /* ---- Writes one sector (64KB) data ---- */
80        for(j = 0; j < ( SF_SECTOR_SIZE / SF_PAGE_SIZE ); j++){
81            /* ---- Writes one page (256 bytes) data ---- */
82            sf_byte_program( addr, data+(j*SF_PAGE_SIZE), SF_PAGE_SIZE );
83            addr += SF_PAGE_SIZE;             /* Updates the destination address to write */
84        }
85      }
```

3.3 Sample Program Listing “main.c” (3/3)

```
86      /* ==== Reads data (2 MB) ==== */
87      addr = TOP_ADDRESS;
88      for(i = 0; i < SF_NUM_OF_SECTOR; i++){
89        /* ---- Reads one sector (64 KB) data ---- */
90        sf_byte_read( addr, rbuf, SF_SECTOR_SIZE );
91        addr += SF_SECTOR_SIZE;                   /* Updates the source address to read */
92
93        /* ---- Verifies data ---- */
94        for(j = 0; j < SF_SECTOR_SIZE; j++){
95            data[j] = (i + j) % 100;          /* Outputs the written data */
96            if( data[j] != rbuf[j] ){
97                puts("Error: verify error\n");
98                fflush(stdout);
99                while(1);
100           }
101       }
102     }
103     /* ==== Protects serial flash memory ==== */
104     sf_protect_ctrl( SF_REQ_PROTECT );
105
106     while(1){
107           /* loop */
108       }
109   }
110
111   /* End of File */
```

3.4 Sample Program Listing "serial_flash.c" (1/13)

```
1    /*""FILE COMMENT""************ Technical reference data ************************
2    *
3    *       System Name : SH7264 Sample Program
4    *       File Name   : serial_flash.c
5    *       Abstract    : Interfacing Serial Flash Memory Using the Renesas Serial
6    *                     Peripheral Interface
7    *       Version     : 1.00.00
8    *       Device      : SH7262/SH7264
9    *       Tool-Chain  : High-performance Embedded Workshop (Version 4.04.01)
10   *                   : C/C++ compiler package for the SuperH RISC engine family
11   *                   :                              (Ver.9.02 Release00).
12   *       OS          : None
13   *       H/W Platform: M3A-HS64G50 (CPU board)
14   *       Disclaimer  :
15   *
16   *       The information described here may contain technical inaccuracies or
17   *       typographical errors. Renesas Technology Corporation and Renesas Solutions
18   *       assume no responsibility for any damage, liability, or other loss rising
19   *       from these inaccuracies or errors.
20   *
21   *       Copyright (C) 2009 Renesas Technology Corp. All Rights Reserved
22   *       AND Renesas Solutions Corp. All Rights Reserved
23   *
24   *       History     : Feb.23,2009 Ver.1.00.00
25   *""FILE COMMENT END""*************************************************************/
26   #include <stdio.h>
27   #include <machine.h>
28   #include "iodefine.h"
29   #include "serial_flash.h"
30
31   /* ==== Macro definition ==== */
32   #define SFLASHCMD_CHIP_ERASE  0xc7
33   #define SFLASHCMD_SECTOR_ERASE0xd8
34   #define SFLASHCMD_BYTE_PROGRAM0x02
35   #define SFLASHCMD_BYTE_READ       0x0B
36   #define SFLASHCMD_BYTE_READ_LOW   0x03
37   #define SFLASHCMD_WRITE_ENABLE0x06
38   #define SFLASHCMD_WRITE_DISABLE   0x04
39   #define SFLASHCMD_READ_STATUS 0x05
40   #define SFLASHCMD_WRITE_STATUS0x01
41   #define UNPROTECT_WR_STATUS       0x00
42   #define PROTECT_WR_STATUS     0x3C
43
```

3.5 Sample Program Listing "serial_flash.c" (2/13)

```
44    /* ==== Function prototype declaration ==== */
45    /*** Local function ***/
46    static void write_enable(void);
47    static void write_disable(void);
48    static void busy_wait(void);
49    static unsigned char read_status(void);
50    static void write_status(unsigned char status);
51    static void io_init_rspi(void);
52    static void io_cmd_exe(unsigned char *ope, int ope_sz, unsigned char *data, int data_sz);
53    static void io_cmd_exe_rdmode(unsigned char *ope, int ope_sz, unsigned char *rd, int rd_sz);
54    static void io_wait_tx_end(void);
55
56    /* ==== Variable definition ==== */
57
58    /*""FUNC COMMENT""*****************************************************************
59     * ID          :
60     * Outline     : Serial flash memory initialization
61     *-----------------------------------------------------------------------------
62     * Include     :
63     *-----------------------------------------------------------------------------
64     * Declaration : void sf_init_serial_flash(void);
65     *-----------------------------------------------------------------------------
66     * Function    : Initializes serial flash memory for being accessed.
67     *             : Initializes the channel 0 of the Renesas Serial Peripheral
68     *             : Interface (RSPI).
69     *-----------------------------------------------------------------------------
70     * Argument    : void
71     *-----------------------------------------------------------------------------
72     * Return Value: void
73     *""FUNC COMMENT END""*************************************************************/
74    void sf_init_serial_flash(void)
75    {
76      /* ==== Initializes the RSPI0 ==== */
77      io_init_rspi();
78    }
```

3.6 Sample Program Listing "serial_flash.c" (3/13)

```
79    /*""FUNC COMMENT""*****************************************************************
80     * ID          :
81     * Outline     : Protect/unprotect operation
82     *-------------------------------------------------------------------------------
83     * Include     :
84     *-------------------------------------------------------------------------------
85     * Declaration : void sf_protect_ctrl(enum sf_req req);
86     *-------------------------------------------------------------------------------
87     * Function    : Protects or unprotects serial flash memory.
88     *             : Use the argument req to specify. Default setting and unprotecting
89     *             : method depends on the specifications of the serial flash memory.
90     *-------------------------------------------------------------------------------
91     * Argument    : enum sf_req req ; I : SF_REQ_UNPROTECT -> Write-enable all sectors
92     *             :                       SF_REQ_PROTECT   -> Write-protect all sectors
93     *-------------------------------------------------------------------------------
94     * Return Value: void
95     *""FUNC COMMENT END""*************************************************************/
96    void sf_protect_ctrl(enum sf_req req)
97    {
98      if( req == SF_REQ_UNPROTECT ){
99        write_status( UNPROTECT_WR_STATUS);       /* Protect total area */
100     }
101     else{
102       write_status( PROTECT_WR_STATUS );    /* Unprotect total area */
103     }
104   }
105   /*""FUNC COMMENT""*****************************************************************
106    * ID          :
107    * Outline     : Chip erase
108    *-------------------------------------------------------------------------------
109    * Include     :
110    *-------------------------------------------------------------------------------
111    * Declaration : void sf_chip_erase(void);
112    *-------------------------------------------------------------------------------
113    * Function    : Erases all bits in serial flash memory.
114    *             : Before erasing or programming, issue the Write Enable command.
115    *             : After erasing or programming, make sure to check the status of
116    *             : serial flash memory if the busy status is reset.
117    *-------------------------------------------------------------------------------
118    * Argument    : void
119    *-------------------------------------------------------------------------------
120    * Return Value: void
121    *""FUNC COMMENT END""*************************************************************/
```

3.7 Sample Program Listing "serial_flash.c" (4/13)

```
122   void sf_chip_erase(void)
123   {
124     unsigned char cmd[1];
125     cmd[0] = SFLASHCMD_CHIP_ERASE;
126
127     write_enable();
128     io_cmd_exe(cmd, 1, NULL, 0);
129     busy_wait();
130   }
131
132   /*""FUNC COMMENT""*****************************************************************
133    * ID          :
134    * Outline     : Sector erase
135    *-------------------------------------------------------------------------------
136    * Include     :
137    *-------------------------------------------------------------------------------
138    * Declaration : void sf_sector_erase(int sector_no);
139    *-------------------------------------------------------------------------------
140    * Function    : Erases the specified sector in serial flash memory.
141    *             : Before erasing or programming, issue the Write Enable command.
142    *             : After erasing or programming, make sure to check the status of
143    *             : serial flash memory if the busy status is reset.
144    *-------------------------------------------------------------------------------
145    * Argument    : int sector_no ; I : Sector number
146    *-------------------------------------------------------------------------------
147    * Return Value: void
148    *""FUNC COMMENT END""*************************************************************/
149   void sf_sector_erase(int sector_no)
150   {
151     unsigned char cmd[4];
152     unsigned long addr = sector_no * SF_SECTOR_SIZE;
153
154     cmd[0] = SFLASHCMD_SECTOR_ERASE;
155     cmd[1] = (addr >> 16) & 0xff;
156     cmd[2] = (addr >>  8) & 0xff;
157     cmd[3] =  addr        & 0xff;
158
159     write_enable();
160     io_cmd_exe(cmd, 4, NULL, 0);
161     busy_wait();
162   }
163
```

3.8 Sample Program Listing "serial_flash.c" (5/13)

```
164   /*""FUNC COMMENT""*****************************************************************
165    * ID          :
166    * Outline     : Program data
167    *-------------------------------------------------------------------------------
168    * Include     :
169    *-------------------------------------------------------------------------------
170    * Declaration : void sf_byte_program(unsigned long addr, unsigned char *buf, int size);
171    *-------------------------------------------------------------------------------
172    * Function    : Programs the specified data in serial flash memory.
173    *             : Before erasing or programming, issue the Write Enable command.
174    *             : After erasing or programming, make sure to check the status of
175    *             : serial flash memory if the busy status is reset.
176    *             : The maximum write data size depends on the type of the device.
177    *-------------------------------------------------------------------------------
178    * Argument    : unsigned long addr ; I : Address in serial flash memory to write
179    *             : unsigned char *buf ; I : Buffer address to store the write data
180    *             : int size           ; I : Number of bytes to write
181    *-------------------------------------------------------------------------------
182    * Return Value: void
183    *""FUNC COMMENT END""*************************************************************/
184   void sf_byte_program(unsigned long addr, unsigned char *buf, int size)
185   {
186     unsigned char cmd[4];
187
188     cmd[0] = SFLASHCMD_BYTE_PROGRAM;
189     cmd[1] = (unsigned char)((addr >> 16) & 0xff);
190     cmd[2] = (unsigned char)((addr >>  8) & 0xff);
191     cmd[3] = (unsigned char)( addr        & 0xff);
192     write_enable();
193     io_cmd_exe(cmd, 4, buf, size);
194     busy_wait();
195   }
196
```

3.9 Sample Program Listing "serial_flash.c" (6/13)

```
197   /*""FUNC COMMENT""*****************************************************************
198    * ID          :
199    * Outline     : Read data
200    *-------------------------------------------------------------------------------
201    * Include     :
202    *-------------------------------------------------------------------------------
203    * Declaration : void sf_byte_read(unsigned long addr, unsigned char *buf, int size);
204    *-------------------------------------------------------------------------------
205    * Function    : Reads the specified number of bytes from serial flash memory.
206    *-------------------------------------------------------------------------------
207    * Argument    : unsigned long addr ; I : Address in serial flash memory to read
208    *             : unsigned char *buf ; I : Buffer address to store the read data
209    *             : int size           ; I : Number of bytes to read
210    *-------------------------------------------------------------------------------
211    * Return Value: void
212    *""FUNC COMMENT END""*************************************************************/
213   void sf_byte_read(unsigned long addr, unsigned char *buf, int size)
214   {
215     unsigned char cmd[4];
216
217     cmd[0] = SFLASHCMD_BYTE_READ_LOW;
218     cmd[1] = (unsigned char)((addr >> 16) & 0xff);
219     cmd[2] = (unsigned char)((addr >>  8) & 0xff);
220     cmd[3] = (unsigned char)( addr        & 0xff);
221     io_cmd_exe_rdmode(cmd, 4, buf, size);
222   }
223
224   /*""FUNC COMMENT""*****************************************************************
225    * ID          :
226    * Outline     : Write enable
227    *-------------------------------------------------------------------------------
228    * Include     :
229    *-------------------------------------------------------------------------------
230    * Declaration : static void write_enable(void);
231    *-------------------------------------------------------------------------------
232    * Function    : Issues the Write Enable command to enable erasing or programming
233    *             : serial flash memory.
234    *-------------------------------------------------------------------------------
235    * Argument    : void
236    *-------------------------------------------------------------------------------
237    * Return Value: void
238    *""FUNC COMMENT END""*************************************************************/
239   static void write_enable(void)
240   {
241     unsigned char cmd[1];
242     cmd[0] = SFLASHCMD_WRITE_ENABLE;
243     io_cmd_exe(cmd, 1, NULL, 0);
244   }
```

3.10 Sample Program Listing "serial_flash.c" (7/13)

```
245
246   /*""FUNC COMMENT""*****************************************************************
247    * ID          :
248    * Outline     : Write disable
249    *-------------------------------------------------------------------------------
250    * Include     :
251    *-------------------------------------------------------------------------------
252    * Declaration : static void write_disable(void);
253    *-------------------------------------------------------------------------------
254    * Function    : Issues the Write Disable command to disable erasing or programming
255    *             : serial flash memory.
256    *-------------------------------------------------------------------------------
257    * Argument    : void
258    *-------------------------------------------------------------------------------
259    * Return Value: void
260    *""FUNC COMMENT END""*************************************************************/
261   static void write_disable(void)
262   {
263     unsigned char cmd[1];
264     cmd[0] = SFLASHCMD_WRITE_DISABLE;
265     io_cmd_exe(cmd, 1, NULL, 0);
266   }
267
268   /*""FUNC COMMENT""*****************************************************************
269    * ID          :
270    * Outline     : Busy waiting
271    *-------------------------------------------------------------------------------
272    * Include     :
273    *-------------------------------------------------------------------------------
274    * Declaration : static void busy_wait(void);
275    *-------------------------------------------------------------------------------
276    * Function    : Loops internally when the serial flash memory is busy.
277    *-------------------------------------------------------------------------------
278    * Argument    : void
279    *-------------------------------------------------------------------------------
280    * Return Value: void
281    *""FUNC COMMENT END""*************************************************************/
282   static void busy_wait(void)
283   {
284     while ((read_status() & 0x01) != 0) {   /* RDY/BSY */
285       /* serial flash is busy */
286     }
287   }
288
```

3.11 Sample Program Listing "serial_flash.c" (8/13)

```
289   /*""FUNC COMMENT""*****************************************************************
290    * ID          :
291    * Outline     : Read status
292    *-------------------------------------------------------------------------------
293    * Include     :
294    *-------------------------------------------------------------------------------
295    * Declaration : static unsigned char read_status(void);
296    *-------------------------------------------------------------------------------
297    * Function    : Reads the status of serial flash memory.
298    *-------------------------------------------------------------------------------
299    * Argument    : void
300    *-------------------------------------------------------------------------------
301    * Return Value: Status register value
302    *""FUNC COMMENT END""*************************************************************/
303   static unsigned char read_status(void)
304   {
305     unsigned char buf;
306     unsigned char cmd[1];
307
308     cmd[0] = SFLASHCMD_READ_STATUS;
309     io_cmd_exe_rdmode(cmd, 1, &buf, 1);
310     return buf;
311   }
312
313   /*""FUNC COMMENT""*****************************************************************
314    * ID          :
315    * Outline     : Write status
316    *-------------------------------------------------------------------------------
317    * Include     :
318    *-------------------------------------------------------------------------------
319    * Declaration : static void write_status(unsigned char status);
320    *-------------------------------------------------------------------------------
321    * Function    : Writes the status of serial flash memory.
322    *-------------------------------------------------------------------------------
323    * Argument    : unsigned char status ; I : status register value
324    *-------------------------------------------------------------------------------
325    * Return Value: void
326    *""FUNC COMMENT END""*************************************************************/
327   static void write_status(unsigned char status)
328   {
329     unsigned char cmd[2];
330
331     cmd[0] = SFLASHCMD_WRITE_STATUS;
332     cmd[1] = status;
333
334     write_enable();
335     io_cmd_exe(cmd, 2, NULL, 0);
336     busy_wait();
```

3.12 Sample Program Listing "serial_flash.c" (9/13)

```
337   }
338   /*""FUNC COMMENT""*****************************************************************
339    * ID          :
340    * Outline     : RSPI initialization
341    *-------------------------------------------------------------------------------
342    * Include     :
343    *-------------------------------------------------------------------------------
344    * Declaration : static void io_init_rspi(void);
345    *-------------------------------------------------------------------------------
346    * Function    : Initializes channel 0 of the RSPI.
347    *             : Sets the RSPI in master mode to set parameters required to transfer
348    *             : according to the specifications of serial flash memory.
349    *-------------------------------------------------------------------------------
350    * Argument    : void
351    *-------------------------------------------------------------------------------
352    * Return Value: void
353    *""FUNC COMMENT END""*************************************************************/
354   static void io_init_rspi(void)
355   {
356     /* ==== PORT ==== */
357     PORT.PFCR3.BIT.PF12MD = 3;  /* PF12:MISO0 */
358     PORT.PFCR2.BIT.PF11MD = 3;  /* PF11:MOSI0 */
359     PORT.PFCR2.BIT.PF10MD = 3;  /* PF10:SSL00 */
360     PORT.PFCR2.BIT.PF9MD  = 3;  /* PF9:RSPCK0 */
361
362     /* ==== CPG ==== */
363     CPG.STBCR5.BIT.MSTP51 = 0;  /* RSPI0 active */
364
365     /* ==== RSPI ==== */
366     RSPI0.SPCR.BYTE   =  0x00;   /* Disables channel 0 of the RSPI */
367     RSPI0.SPPCR.BYTE  =  0x30;   /* MOSI idle fixed value = 1 */
368     RSPI0.SPBR.BYTE   = 0x01;   /* Specifies the base bit rate as 18 MHz
369                                                        (Bus clock = 72 MHz) */
370     RSPI0.SPDCR.BYTE  =  0x20;   /* Disables to send the dummy data */
371                             /* Access width of the SPDR register: 8-bit */
372     RSPI0.SPCKD.BYTE  =  0x00;   /* RSPCK delay: 1 RSPCK */
373     RSPI0.SSLND.BYTE  =  0x00;   /* SSL negate delay: 1 RSPCK */
374     RSPI0.SPND.BYTE   =  0x00;   /* Next access delay: 1 RSPCK + 2 Bus clocks */
375     RSPI0.SPSCR.BYTE  =  0x00;   /* Sequence length: 1 (SPCMD0 is only used) */
376     RSPI0.SPCMD0.WORD = 0xE783; /* MSB first */
377                             /* Data length: 8-bit */
378                             /* Keeps the SSL signal level after transfer
379                                 is completed */
380                             /* Bit rate: Base bit rate is divided by 1 */
381                             /* RSPCK when it is idling is 1 */
382                             /* Latches data on odd edge, outputs data on
383                                 even edge */
```

3.13 Sample Program Listing "serial_flash.c" (10/13)

```
384     RSPI0.SPBFCR.BYTE =  0xC0;   /* Enable to reset data in the
385                                     transmit/receive buffer */
386     RSPI0.SPBFCR.BYTE =  0x00;   /* Disable to reset data in the
387                                     transmit/receive buffer */
388                             /* Number of triggers in transmit buffer:
389                                 more than one byte available */
390                             /* Number of triggers in receive buffer:
391                                 more than one byte received */
392     RSPI0.SSLP.BYTE   =  0x00;   /* SSLP = b'0 SSL signal 0-active */
393     RSPI0.SPCR.BYTE   =  0x48;   /* Master mode */
394                             /* Disables interrupts */
395                             /* Enables channel 0 of the RSPI */
396   }
397
398   /*""FUNC COMMENT""*****************************************************************
399    * ID          :
400    * Outline     : Execute command (No read data).
401    *-------------------------------------------------------------------------------
402    * Include     :
403    *-------------------------------------------------------------------------------
404    * Declaration : static void io_cmd_exe(unsigned char *ope, int ope_sz,
405    *             :                        unsigned char *data,int data_sz)
406    *-------------------------------------------------------------------------------
407    * Function    : Executes the specified command.
408    *             : Transmits the argument ope, and then transmits the argument data.
409    *             : Discards the receive data.
410    *             : Set one of the values between 0 and 8 in the ope_sz.
411    *             : Set one of the values between 0 and 256 in the data_sz.
412    *-------------------------------------------------------------------------------
413    * Argument    : unsigned char *ope ; I : Start address of the opcode block and
414    *                                        address block to transmit
415    *             : int ope_sz         ; I : Number of bytes in the opcode block and
416    *                                        address block
417    *             : unsigned char *data; I : Start address of the data block to transmit
418    *             : int data_sz        ; I : Number of bytes in the data block
419    *-------------------------------------------------------------------------------
420    * Return Value: void
421    *""FUNC COMMENT END""*************************************************************/
422   static void io_cmd_exe(unsigned char *ope, int ope_sz, unsigned char *data, int data_sz)
423   {
424     unsigned char tmp;
425
426     /* ==== Resets buffer ==== */
427     RSPI0.SPBFCR.BYTE  = 0xC0u;
428     RSPI0.SPBFCR.BYTE  = 0x00u;
429
```

3.14 Sample Program Listing "serial_flash.c" (11/13)

```
430     /* ---- Enables the SPI transfer ---- */
431     RSPI0.SPCR.BIT.SPE = 1;
432
433     /* ==== MOSI(command, address, write data) ==== */
434     while(ope_sz--){
435       RSPI0.SPDR.BYTE = *ope++;  /* Command size must be equal or less than 8 bytes */
436     }
437     while(data_sz--){
438       while( RSPI0.SPSR.BIT.SPTEF == 0 ){
439           /* wait */
440       }
441       RSPI0.SPDR.BYTE = *data++;
442       if( RSPI0.SPSR.BIT.SPRF == 1 ){
443           tmp = RSPI0.SPDR.BYTE; /* Dummy read to avoid an overflow of data */
444       }
445     }
446     io_wait_tx_end();               /* Waits for transfer end */
447
448     /* ---- SPI transfer end (SSL negation) ---- */
449     RSPI0.SPCR.BIT.SPE = 0;
450   }
451   /*""FUNC COMMENT""*****************************************************************
452    * ID          :
453    * Outline     : Execute command (With read data).
454    *------------------------------------------------------------------------------
455    * Include     :
456    *------------------------------------------------------------------------------
457    * Declaration : static void io_cmd_exe_rdmode(unsigned char *ope, int ope_sz,
458    *             :                               unsigned char *rd,  int rd_sz)
459    *------------------------------------------------------------------------------
460    * Function    : Executes the specified command.
461    *             : Transmits the argument ope, and then receives data in the argument rd.
462    *             : Set one of the values between 0 and 8 in the ope_sz.
463    *             : More than 0 can be set in the rd_sz.
464    *------------------------------------------------------------------------------
465    * Argument    : unsigned char *ope ; I : Start address of the opcode block and
466    *                                        address block to transmit
467    *             : int ope_sz         ; I : Number of bytes in the opcode block and
468    *                                        address block
469    *             : unsigned char *rd  ; I : Buffer address to store the received data
470    *             : int rd_sz          ; I : Number of bytes in the data block
471    *------------------------------------------------------------------------------
472    * Return Value: void
473    *""FUNC COMMENT END""*************************************************************/
474   static void io_cmd_exe_rdmode(unsigned char *ope, int ope_sz, unsigned char *rd, int rd_sz)
475   {
```

3.15 Sample Program Listing "serial_flash.c" (12/13)

```
476     /* ==== Resets buffer ==== */
477     RSPI0.SPBFCR.BYTE  = 0xC0u;
478     RSPI0.SPBFCR.BYTE  = 0x00u;
479
480     /* ---- Enables the SPI transfer ---- */
481     RSPI0.SPCR.BIT.SPE = 1;
482
483     /* ---- MOSI (command, address, dummy) ---- */
484     while(ope_sz--){
485       RSPI0.SPDR.BYTE = *ope++;  /* Command size must be equal or less than 8 bytes */
486     }
487     io_wait_tx_end();               /* Waits for transfer end */
488
489     /* ---- MISO(read data) ---- */
490     RSPI0.SPBFCR.BYTE  = 0xC0u;     /* Resets buffer */
491     RSPI0.SPBFCR.BYTE  = 0x00u;
492
493     RSPI0.SPDCR.BIT.TXDMY = 1;      /* Enables to transmit the dummy data */
494     while(rd_sz--){
495       while( RSPI0.SPSR.BIT.SPRF == 0){
496           /* wait */
497       }
498       *rd++ = RSPI0.SPDR.BYTE;
499     }
500     RSPI0.SPDCR.BIT.TXDMY = 0;      /* Disable to transmit the dummy data */
501     io_wait_tx_end();               /* Waits for transfer end */
502
503     /* ---- SPI transfer end (SSL negation) ---- */
504     RSPI0.SPCR.BIT.SPE = 0;
505   }
506
507   /*""FUNC COMMENT""*****************************************************************
508    * ID          :
509    * Outline     : Transfer end waiting
510    *-------------------------------------------------------------------------------
511    * Include     :
512    *-------------------------------------------------------------------------------
513    * Declaration : static void io_wait_tx_end(void);
514    *-------------------------------------------------------------------------------
515    * Function    : Loops internally until the transmission is completed.
516    *-------------------------------------------------------------------------------
517    * Argument    : void
518    *-------------------------------------------------------------------------------
519    * Return Value: void
520    *""FUNC COMMENT END""*************************************************************/
```

3.16 Sample Program Listing "serial_flash.c" (13/13)

```
521   static void io_wait_tx_end(void)
522   {
523     while(RSPI0.SPSR.BIT.TEND == 0){
524       /* wait */
525     }
526   }
527
528   /* End of File */
```

3.17 Sample Program Listing "serial_flash.h"

```
1     /*""FILE COMMENT""************ Technical reference data ************************
2     *
3     *       System Name : SH7264 Sample Program
4     *       File Name   : serial_flash.h
5     *       Abstract    : Interfacing Serial Flash Memory Using the Renesas Serial
6     *                   : Peripheral Interface
7     *       Version     : 1.00.00
8     *       Device      : SH7262/SH7264
9     *       Tool-Chain  : High-performance Embedded Workshop (Version 4.04.01)
10    *                   : C/C++ compiler package for the SuperH RISC engine family
11    *                   :                               (Ver.9.02 Release00).
12    *       OS          : None
13    *       H/W Platform: M3A-HS64G50 (CPU board)
14    *       Disclaimer  :
15    *
16    *       The information described here may contain technical inaccuracies or
17    *       typographical errors. Renesas Technology Corporation and Renesas Solutions
18    *       assume no responsibility for any damage, liability, or other loss rising
19    *       from these inaccuracies or errors.
20    *
21    *       Copyright (C) 2009 Renesas Technology Corp. All Rights Reserved
22    *       AND Renesas Solutions Corp. All Rights Reserved
23    *
24    *       History     : Feb.23,2009 Ver.1.00.00
25    *""FILE COMMENT END""*************************************************************/
26    #ifndef _SERIAL_FLASH_H_
27    #define _SERIAL_FLASH_H_
28
29    /* ==== Macro definition ==== */
30    #define SF_PAGE_SIZE      256        /* Page size of serial flash memory */
31    #define SF_SECTOR_SIZE    0x10000    /* Sector size = 64KB   */
32    #define SF_NUM_OF_SECTOR  32         /* Number of sectors 32   */
33    enum sf_req{
34      SF_REQ_PROTECT = 0,            /* Requests to protect */
35      SF_REQ_UNPROTECT               /* Requests to unprotect */
36    };
37    /* ==== Function prototype declaration ==== */
38    void sf_init_serial_flash(void);
39    void sf_protect_ctrl(enum sf_req req);
40    void sf_chip_erase(void);
41    void sf_sector_erase(int sector_no);
42    void sf_byte_program(unsigned long addr, unsigned char *buf, int size);
43    void sf_byte_read(unsigned long addr, unsigned char *buf, int size);
44
45    #endif /* _SERIAL_FLASH_H_ */
46    /* End of File */
```

4. References

- Software Manual
  SH-2A/SH-2A-FPU Software Manual Rev. 3.00
  (Download the latest version from the Renesas website.)

- Hardware Manual
  SH7262 Group, SH7264 Group Hardware Manual Rev. 1.00
  (Download the latest version from the Renesas website.)

Website and Support

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

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

Revision History

		Description	
Rev.	**Date**	**Page**	**Summary**
1.00	Apr 14, 2009	—	First edition issued

Notes regarding these materials

1. This document is provided for reference purposes only so that Renesas customers may select the appropriate Renesas products for their use. Renesas neither makes warranties or representations with respect to the accuracy or completeness of the information contained in this document nor grants any license to any intellectual property rights or any other rights of Renesas or any third party with respect to the information in this document.
2. Renesas shall have no liability for damages or infringement of any intellectual property or other rights arising out of the use of any information in this document, including, but not limited to, product data, diagrams, charts, programs, algorithms, and application circuit examples.
3. You should not use the products or the technology described in this document for the purpose of military applications such as the development of weapons of mass destruction or for the purpose of any other military use. When exporting the products or technology described herein, you should follow the applicable export control laws and regulations, and procedures required by such laws and regulations.
4. All information included in this document such as product data, diagrams, charts, programs, algorithms, and application circuit examples, 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 products listed in this document, please confirm the latest product information with a Renesas sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas such as that disclosed through our website. (http://www.renesas.com)
5. Renesas has used reasonable care in compiling the information included in this document, but Renesas assumes no liability whatsoever for any damages incurred as a result of errors or omissions in the information included in this document.
6. When using or otherwise relying on the information in this document, you should evaluate the information in light of the total system before deciding about the applicability of such information to the intended application. Renesas makes no representations, warranties or guaranties regarding the suitability of its products for any particular application and specifically disclaims any liability arising out of the application and use of the information in this document or Renesas products.
7. With the exception of products specified by Renesas as suitable for automobile applications, Renesas products are not designed, manufactured or tested for applications or otherwise in systems the failure or malfunction of which may cause a direct threat to human life or create a risk of human injury or which require especially high quality and reliability such as safety systems, or equipment or systems for transportation and traffic, healthcare, combustion control, aerospace and aeronautics, nuclear power, or undersea communication transmission. If you are considering the use of our products for such purposes, please contact a Renesas sales office beforehand. Renesas shall have no liability for damages arising out of the uses set forth above.
8. Notwithstanding the preceding paragraph, you should not use Renesas products for the purposes listed below:
   (1) artificial life support devices or systems
   (2) surgical implantations
   (3) healthcare intervention (e.g., excision, administration of medication, etc.)
   (4) any other purposes that pose a direct threat to human life
   Renesas shall have no liability for damages arising out of the uses set forth in the above and purchasers who elect to use Renesas products in any of the foregoing applications shall indemnify and hold harmless Renesas Technology Corp., its affiliated companies and their officers, directors, and employees against any and all damages arising out of such applications.
9. You should use the products described herein within the range specified by Renesas, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas shall have no liability for malfunctions or damages arising out of the use of Renesas products beyond such specified ranges.
10. Although Renesas endeavors to improve the quality and reliability of its products, IC products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Please be sure to implement safety measures to guard against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas 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 applicable measures. Among others, since the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
11. In case Renesas products listed in this document are detached from the products to which the Renesas products are attached or affixed, the risk of accident such as swallowing by infants and small children is very high. You should implement safety measures so that Renesas products may not be easily detached from your products. Renesas shall have no liability for damages arising out of such detachment.
12. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written approval from Renesas.
13. Please contact a Renesas sales office if you have any questions regarding the information contained in this document, Renesas semiconductor products, or if you have any other inquiries.