# SH7239 Group

## Interfacing Serial Flash Memory Using the Renesas Serial Peripheral Interface

R01AN0057EJ0100
Rev. 1.00
Nov. 24, 2010

### Summary

This application note describes how to connect serial flash memory using the SH7239 Microcomputers (MCUs) Renesas Serial Peripheral Interface (RSPI).

### Target Device

SH7239 MCU

### Contents

# 1. Introduction

## 1.1 Specifications

- Use the serial flash memory of 1 Mbit (256 Kbit × 4 sectors) to connect with the SH7239 MCU

## 1.2 Modules Used

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

## 1.3 Applicable Conditions

| | |
|---|---|
| MCU | SH7239 (R5F72395ADFP) |
| Power Supply Voltage | 3.3 V |
| Operating Frequency | Internal clock: 160 MHz<br>Bus clock: 40 MHz<br>Peripheral clock: 40 MHz |
| Integrated Development Environment | Renesas Electronics<br>High-performance Embedded Workshop Ver.4.07.00 |
| C Compiler | Renesas Electronics SuperH RISC engine Family<br>C/C++ compiler package Ver.9.03 Release 02 |
| Compiler Options | Default setting in the High-performance Embedded Workshop<br>(-cpu=sh2afpu -fpu=single -debug -gbr=auto -global_volatile=0<br>-opt_range=all -infinite_loop=0 -del_vacant_loop=0 -struct_alloc=1) |

# 2. Applications

Connect the SH7239 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

SH7239 RSPI allows full-duplex, synchronous, serial communications with peripheral devices in SPI operation using 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- to 16-bit, 20-, 24-, and 32-bit)

## 2.2 Serial Flash Memory Pin Connection

The following table lists the specifications of the SPI-compliant serial flash memory (M25P10, STMicroelectronics) used in this application.

**Table 1 Serial Flash Memory Specifications**

| Item | Description |
|---|---|
| SPI modes | Supports SPI modes 0 and 3 |
| Clock frequency | 20 MHz (at maximum) |
| Capacity | 1 Mbit |
| Sector size | 256 Kbit |
| Page size | 128 bytes |
| Erase architecture | Sector Erase (256 Kbit), Bulk Erase (1 Mbit) |
| Programming options | Page Program (1 to 128 bytes) |
| Protect area size | None, Upper quarter (sector 3), Upper half (sectors 3 and 2), Whole memory (all sectors) |

The figure below shows an example of serial flash memory circuit. Set the SH7239 pin functions as shown in Table 2.

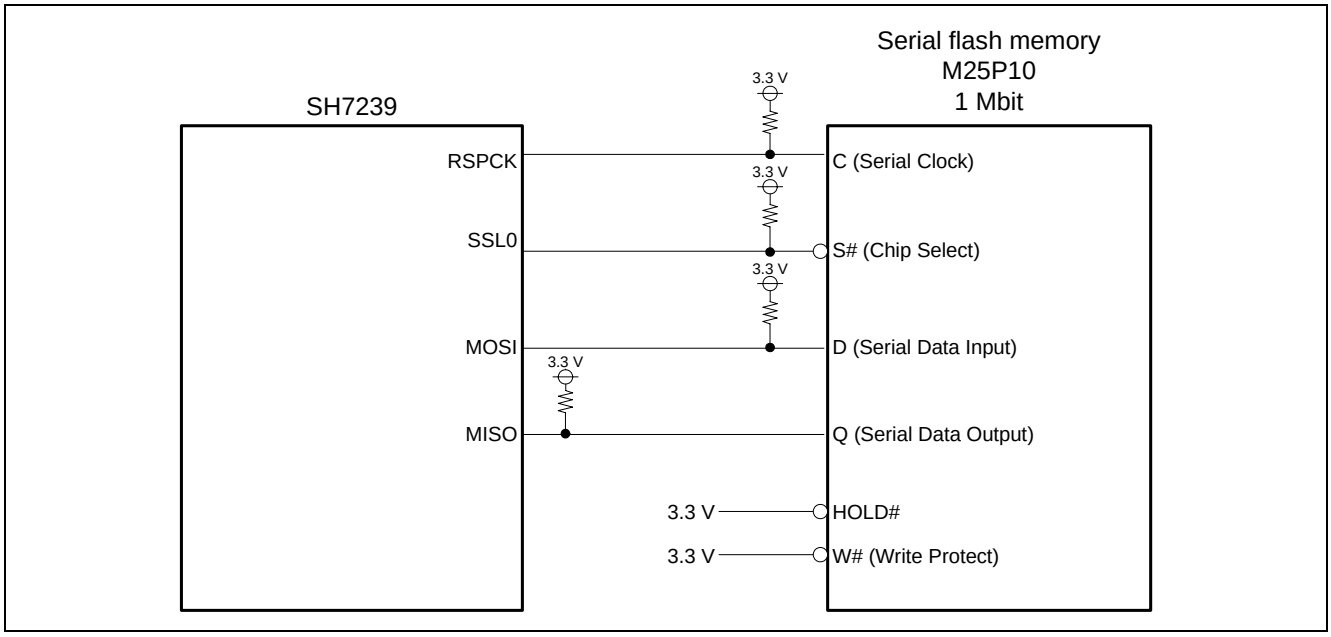


**Figure 1 Serial Flash Memory Circuit**

Note: Pull up or pull down the control signal pins using the external resistors
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 high-impedance. SSL0 pin is pulled up by an external resistor to high level. Pull up or down the RSPCK and MOSI pins. As the MISO pin is configured as input, pull-up or pull-down is recommended to avoid floating to the midpoint voltage.

**Table 2 Multiplexed Output**

| Peripheral Functions | Pin Name | SH7239 Port Control Register | | SH7239 Multiplexed Pin Name |
|---|---|---|---|---|
| | | Register Name | MD bit Setting | |
| RSPI | RSPCK | PACRL2 | PA6MD[2:0] = B'101 | PA6/IRQ6/TCLKA/CS6#/RSPCK/SCK1 |
| | MOSI | PACRL2 | PA7MD[2:0] = B'101 | PA7/IRQ5/TCLKB/CS5#/MOSI/TXD1 |
| | MISO | PACRL3 | PA8MD[2:0] = B'101 | PA8/IRQ4/TCLKC/CS4#/MISO/RXD1 |
| | SSL0 | PACRL3 | PA9MD[2:0] = B'101 | PA9/IRQ3/TCLKD/CS3#/SSL0/SCK0 |

Note: SH7239 Multiplexed pins
RSPCK, MOSI, MISO, and SSL0 pins are multiplexed, and set to general-purpose I/O ports by 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 SH7239 and serial flash memory. Initialize the RSPI and the clock frequency according to the 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 RSPI 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 listed in Table 3 and Table 4.

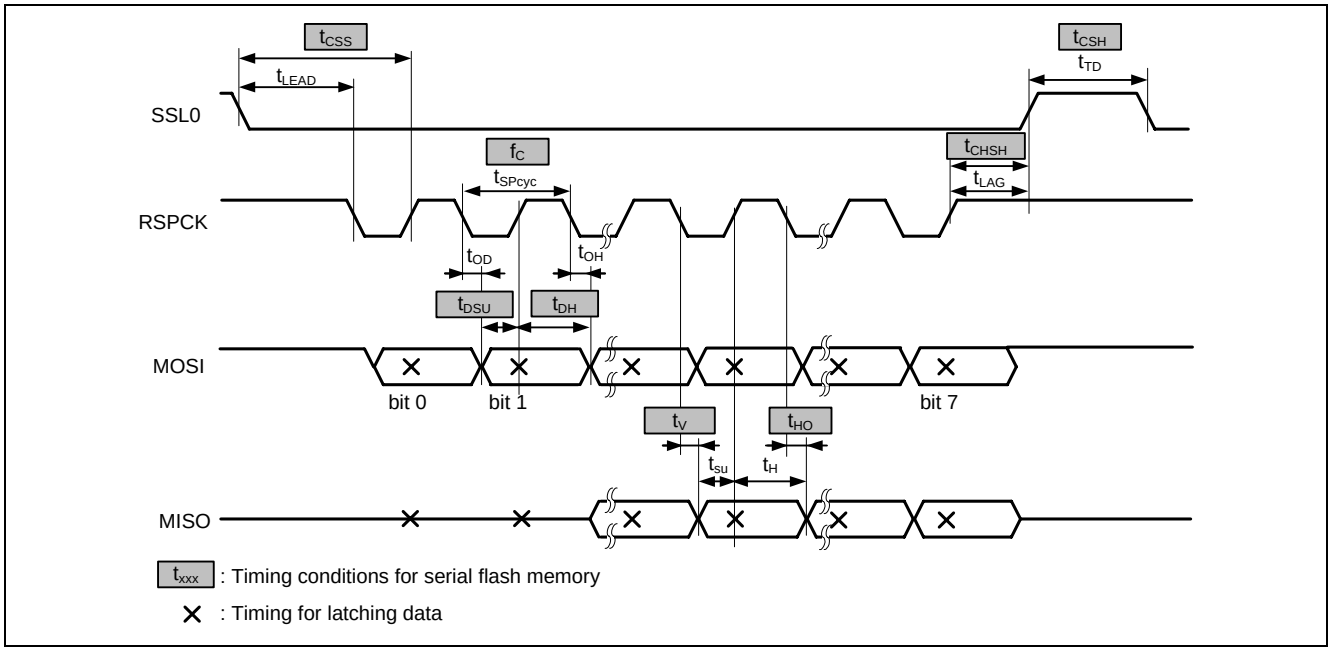


**Figure 2 Data Transfer Timing Example (CPOL = 1, CPHA = 1)**

**Table 3 Timing Conditions for Serial Flash Memory when Transferring Data**

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

**Table 4 Timing Conditions for the SH7239 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.<br>The following formula must be fulfilled:<br>1/2 × $t_{SPcyc}$ − $t_V$ (max.) ≥ $t_{SU}$ (min.) | SPBR register<br>SPCMD register |
| $t_H$ | Data Input Hold Time | Time required for the slave device from latching data to stop the data output.<br>The following formula must be fulfilled:<br>$t_{HO}$ (min.) + 1/2 × $t_{SPcyc}$ ≥ $t_H$(min.) | SPBR register<br>SPCMD register |

## 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.

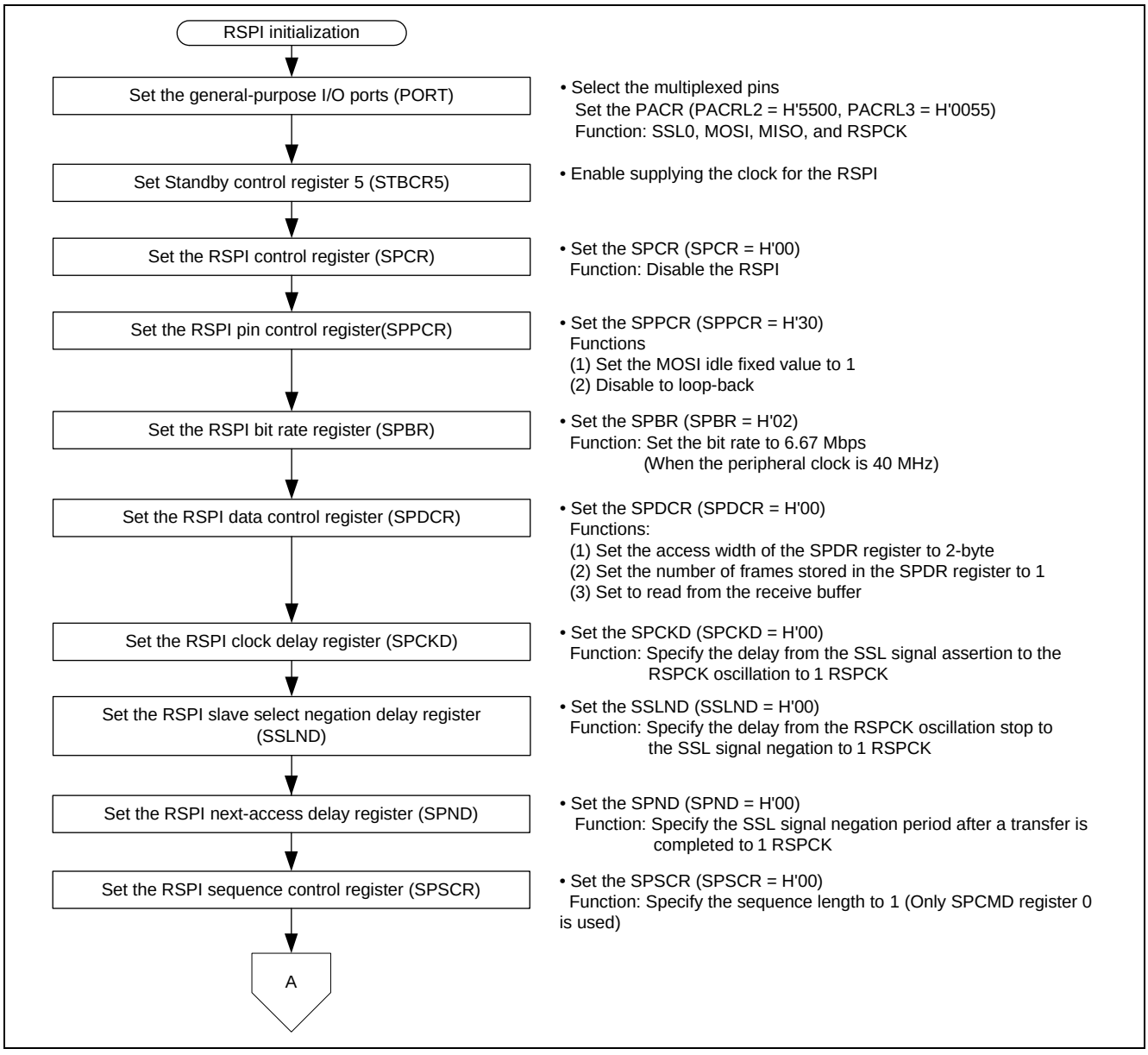


**Figure 3 Flow Chart for Initializing the RSPI (1/2)**

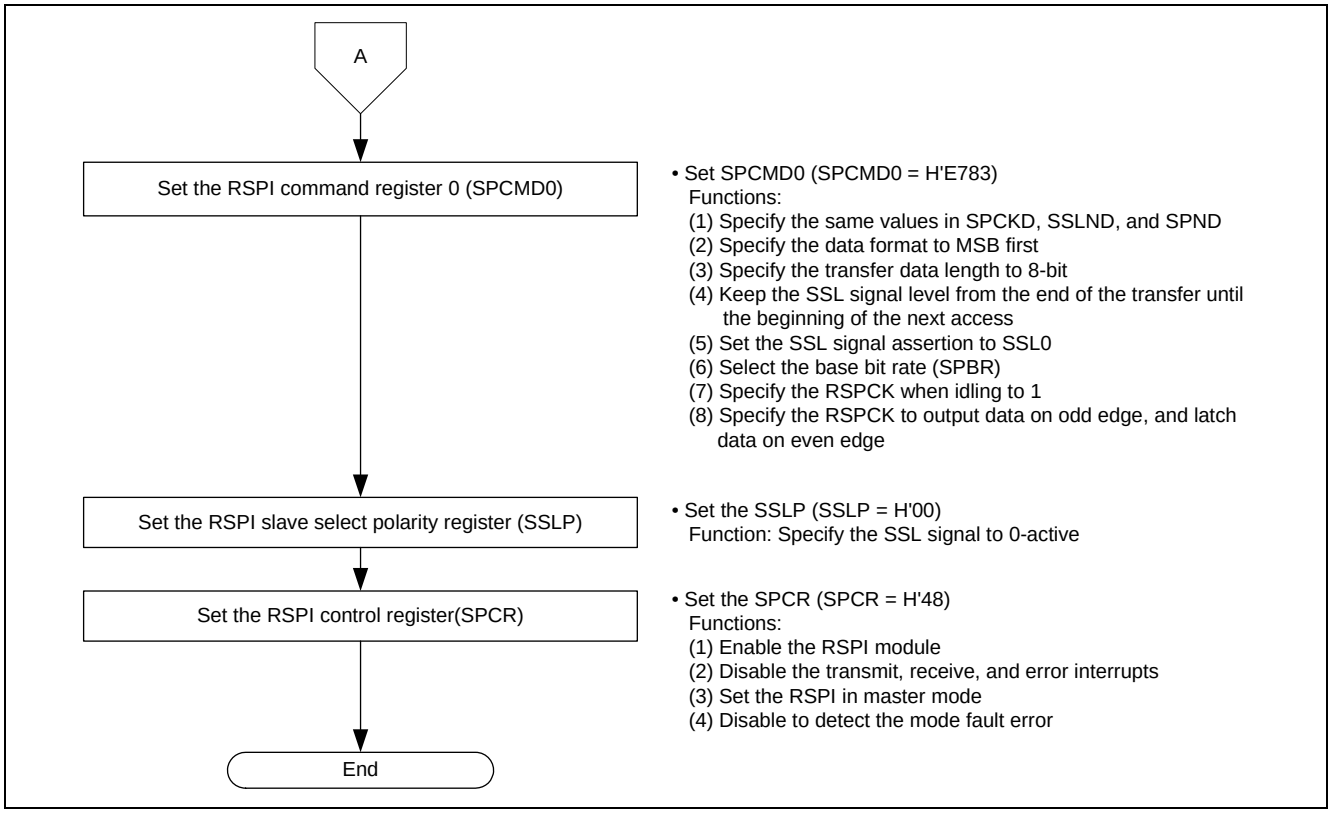


**Figure 4 Flow Chart for Initializing the RSPI (2/2)**

### 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 charts in the sample program.

This application refers to the commands of the STMicroelectronics M25P10. 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 M25P10.

**Table 5 M25P10 Commands**

| Command Name | Opcode | Address Bytes | Data Bytes | Function |
|---|---|---|---|---|
| Write Enable (WREN) | H'06 | 0 | 0 | Enables the program/erase command |
| Write Disable (WRDI) | H'04 | 0 | 0 | Disables the program/erase command |
| Read Status Register (RDSR) | H'05 | 0 | 1 or more | Reads the status |
| Write Status Register (WRSR) | H'01 | 0 | 1 | Writes the status |
| Read Data Bytes (READ) | H'03 | 3 | 1 or more (1) | Reads the data |
| Page Program (PP) | H'02 | 3 | 1 to 128 (2) | Writes the data |
| Sector Erase (SE) | H'D8 | 3 | 0 | Erases data in sectors (256 Kbit) |
| Bulk Erase (BE) | H'C7 | 0 | 0 | Erases the entire memory array |

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 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 Data Bytes (READ) command.

When issuing the Read Data Bytes (READ) 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 completed.

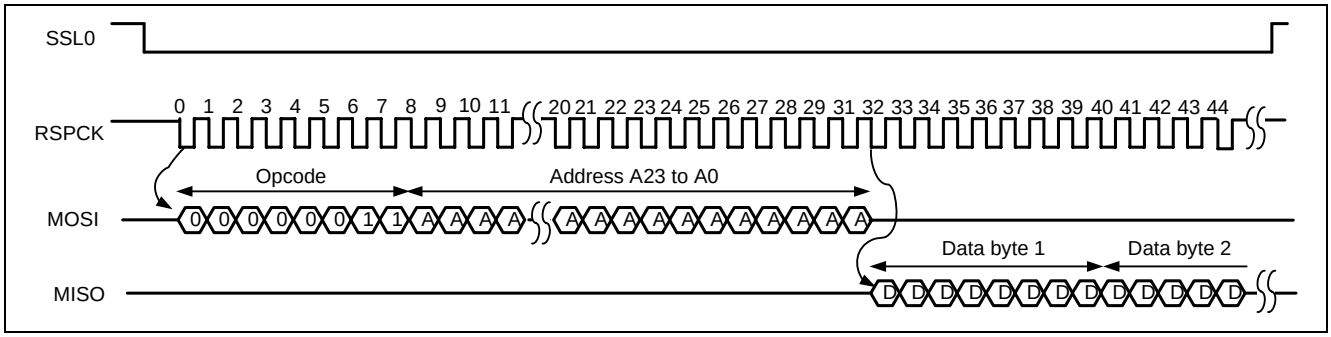


**Figure 5 Read Command Sequence (Opcode: H'03)**

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/write commands transfer. Figure 7 shows the flow chart of the data transfer.

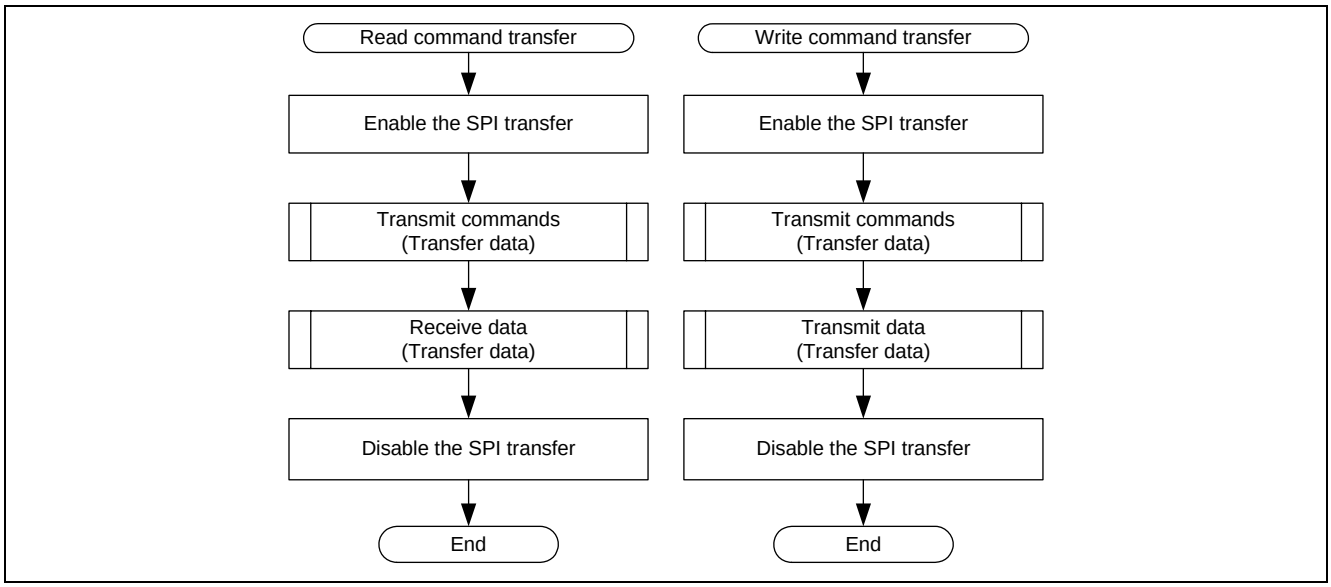


**Figure 6 Flow Chart of Read/Write Commands Transfer**

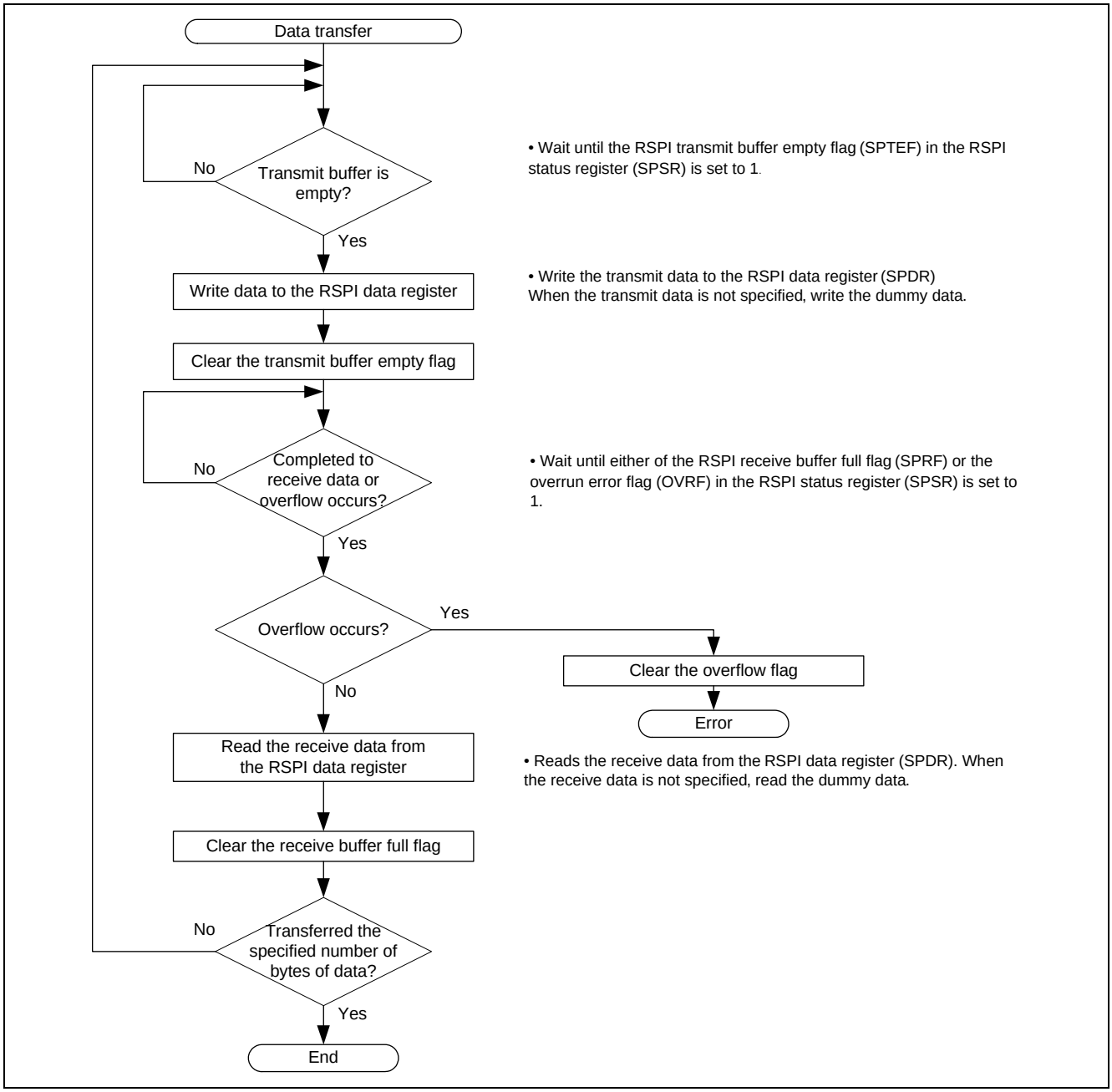


**Figure 7 Flow Chart of the Data Transfer**

### 2.4.3 Main Function

Figure 8 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.

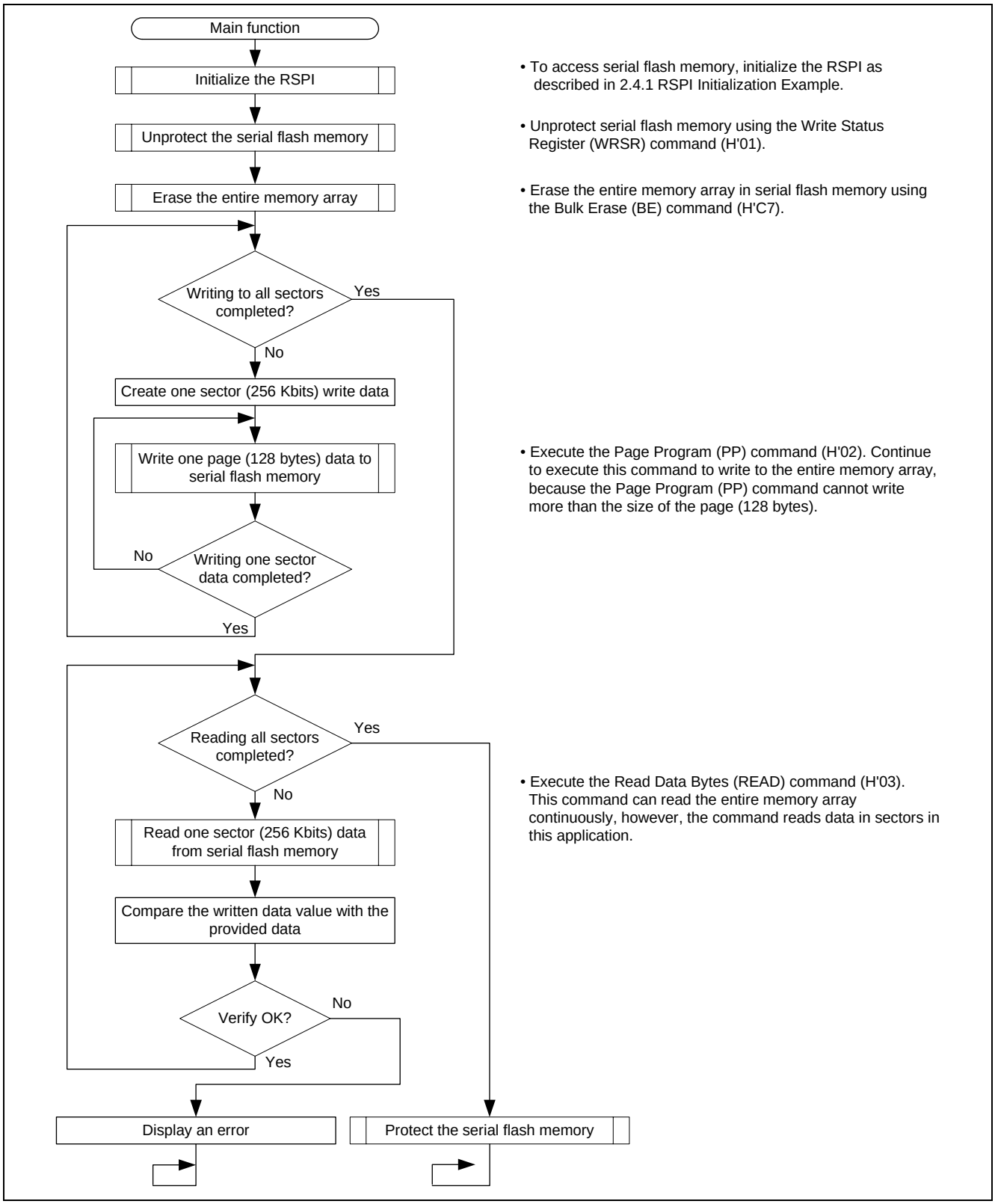


**Figure 8 Main Function Flow Chart**

# 3. Sample Program Listing

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

```
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 : SH7239 Sample Program
31  *   File Name   : main.c
32  *   Abstract    : Interfacing Serial Flash Memory Using the Renesas Serial
33  *               : Peripheral Interface
34  *   Version     : 1.00.00
35  *   Device      : SH7239
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: R0K572390 (CPU board)
41  *   Description : Connects the serial flash memory with the MCU using the Renesas
42  *               : Serial Peripheral Interface.
43  ***********************************************************************
44  *   History     : Aug.17,2010 Ver.1.00.00
45  *""FILE COMMENT END""*****************************************************/
46  #include <stdio.h>
47  #include "serial_flash.h"
```

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

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

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

```
94      /* ==== Writes data (1 Mbits) ==== */
95      addr = TOP_ADDRESS;
96      for(i = 0; i < SF_NUM_OF_SECTOR; i++){
97        /* ---- Initializes the data (4 KB) ---- */
98        for(j = 0; j < SF_BUFF_SIZE; j++){
99            data[j] = (i + j) % 100;
100       }
101       /* ---- Writes one sector (32 KB) data ---- */
102       for(k = 0; k < SF_SECTOR_SIZE/SF_BUFF_SIZE; k++){
103           /* ---- Writes one buffer (4 KB) data ---- */
104           for(j = 0; j < ( SF_BUFF_SIZE / SF_PAGE_SIZE ); j++){
105               /* ---- Writes one page (128 bytes) data ---- */
106               sf_byte_program( addr, data+(j*SF_PAGE_SIZE), SF_PAGE_SIZE );
107               addr += SF_PAGE_SIZE;          /* Updates the destination address to write */
108           }
109       }
110     }
111     /* ==== Reads data (1 Mbits) ==== */
112     addr = TOP_ADDRESS;
113     for(i = 0; i < SF_NUM_OF_SECTOR; i++){
114       /* ---- Reads one sector (32 KB) data ---- */
115       for(k = 0; k < SF_SECTOR_SIZE/SF_BUFF_SIZE; k++){
116           /* ---- Reads one buffer (4 KB) data ---- */
117           sf_byte_read( addr, rbuf, SF_BUFF_SIZE );
118           addr += SF_BUFF_SIZE;              /* Updates the destination address to read */
119           /* ---- Verifies data ---- */
120           for(j = 0; j < SF_BUFF_SIZE; j++){
121               data[j] = (i + j) % 100;       /* Outputs the written data */
122               if( data[j] != rbuf[j] ){
123                   puts("Error: verify error\n");
124                   fflush(stdout);
125                   while(1);
126               }
127           }
128       }
129     }
130     /* ==== Protects serial flash memory ==== */
131     sf_protect_ctrl( SF_REQ_PROTECT );
132
133     while(1){
134          /* loop */
135       }
136   }
137
138   /* End of File */
```

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

```
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 : SH7239 Sample Program
31  *   File Name   : serial_flash.c
32  *   Abstract    : Interfacing Serial Flash Memory Using the Renesas Serial
33  *               : Peripheral Interface
34  *   Version     : 1.00.00
35  *   Device      : SH7239
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: R0K572390 (CPU board)
41  *   Description : Connects the serial flash memory with the MCU using the Renesas
42  *               : Serial Peripheral Interface.
43  ***********************************************************************
44  *   History     : Aug.17,2010 Ver.1.00.00
45  *""FILE COMMENT END""*****************************************************/
46  #include <stdio.h>
47  #include <machine.h>
48  #include "iodefine.h"
49  #include "serial_flash.h"
50
```

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

```
51  /* ==== Macro definition ==== */
52  #define SFLASHCMD_CHIP_ERASE  0xc7
53  #define SFLASHCMD_SECTOR_ERASE0xd8
54  #define SFLASHCMD_BYTE_PROGRAM0x02
55  #define SFLASHCMD_BYTE_READ       0x0B
56  #define SFLASHCMD_BYTE_READ_LOW   0x03
57  #define SFLASHCMD_WRITE_ENABLE0x06
58  #define SFLASHCMD_WRITE_DISABLE   0x04
59  #define SFLASHCMD_READ_STATUS 0x05
60  #define SFLASHCMD_WRITE_STATUS0x01
61  #define UNPROTECT_WR_STATUS       0x00
62  #define PROTECT_WR_STATUS     0x3C
63
64  /* ==== Function prototype declaration ==== */
65  /*** Local function ***/
66  static void write_enable(void);
67  static void write_disable(void);
68  static void busy_wait(void);
69  static unsigned char read_status(void);
70  static void write_status(unsigned char status);
71  static void io_init_rspi(void);
72  static void io_cmd_exe(unsigned char *ope, int ope_sz, unsigned char *data, int data_sz);
73  static void io_cmd_exe_rdmode(unsigned char *ope, int ope_sz, unsigned char *rd, int rd_sz);
74  static int io_rspi_transfer(unsigned char *write_data, unsigned char *read_data, int data_sz);
75
76  /* ==== Variable definition ==== */
77
78  /*""FUNC COMMENT""*****************************************************************
79   * ID           :
80   * Outline      : Serial flash memory initialization
81   *-----------------------------------------------------------------------------
82   * Include      :
83   *-----------------------------------------------------------------------------
84   * Declaration  : void sf_init_serial_flash(void);
85   *-----------------------------------------------------------------------------
86   * Description  : Initializes serial flash memory for being accessed.
87   *              : Initializes the Renesas Serial Peripheral Interface (RSPI).
88   *-----------------------------------------------------------------------------
89   * Argument     : void
90   *-----------------------------------------------------------------------------
91   * Return Value : void
92   *-----------------------------------------------------------------------------
93   * Note         : None
94   *""FUNC COMMENT END""*************************************************************/
95  void sf_init_serial_flash(void)
96  {
97    /* ==== Initializes the RSPI ==== */
98    io_init_rspi();
99  }
```

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

```
100  /*""FUNC COMMENT""*****************************************************************
101   * ID           :
102   * Outline      : Protect/unprotect operation
103   *-------------------------------------------------------------------------------
104   * Include      :
105   *-------------------------------------------------------------------------------
106   * Declaration  : void sf_protect_ctrl(enum sf_req req);
107   *-------------------------------------------------------------------------------
108   * Description  : Protects or unprotects serial flash memory.
109   *              : Use the argument req to specify. Default setting and unprotecting
110   *              : method depends on the specifications of the serial flash memory.
111   *-------------------------------------------------------------------------------
112   * Argument     : enum sf_req req ; I : SF_REQ_UNPROTECT -> Write-enable all sectors
113   *              :                       SF_REQ_PROTECT   -> Write-protect all sectors
114   *-------------------------------------------------------------------------------
115   * Return Value : void
116   *-------------------------------------------------------------------------------
117   * Note         : None
118   *""FUNC COMMENT END""*************************************************************/
119  void sf_protect_ctrl(enum sf_req req)
120  {
121    if( req == SF_REQ_UNPROTECT ){
122      write_status( UNPROTECT_WR_STATUS);       /* Unprotects the entire memory area */
123    }
124    else{
125      write_status( PROTECT_WR_STATUS );    /* Protects the entire memory area */
126    }
127  }
128  /*""FUNC COMMENT""*****************************************************************
129   * ID           :
130   * Outline      : Chip erase
131   *-------------------------------------------------------------------------------
132   * Include      :
133   *-------------------------------------------------------------------------------
134   * Declaration  : void sf_chip_erase(void);
135   *-------------------------------------------------------------------------------
136   * Description  : Erases all bits in serial flash memory.
137   *              : Before erasing or programming, issue the Write Enable command.
138   *              : After erasing or programming, make sure to check the status of
139   *              : serial flash memory if it is not busy.
140   *-------------------------------------------------------------------------------
141   * Argument     : void
142   *-------------------------------------------------------------------------------
143   * Return Value : void
144   *-------------------------------------------------------------------------------
145   * Note         : None
146   *""FUNC COMMENT END""*************************************************************/
147  void sf_chip_erase(void)
148  {
```

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

```
149     unsigned char cmd[1];
150     cmd[0] = SFLASHCMD_CHIP_ERASE;
151
152     write_enable();
153     io_cmd_exe(cmd, 1, NULL, 0);
154     busy_wait();
155   }
156   /*""FUNC COMMENT""*****************************************************************
157    * ID           :
158    * Outline      : Sector erase
159    *-------------------------------------------------------------------------------
160    * Include      :
161    *-------------------------------------------------------------------------------
162    * Declaration  : void sf_sector_erase(int sector_no);
163    *-------------------------------------------------------------------------------
164    * Description  : Erases the specified sector in serial flash memory.
165    *              : Before erasing or programming, issue the Write Enable command.
166    *              : After erasing or programming, make sure to check the status of
167    *              : serial flash memory if it is not busy.
168    *-------------------------------------------------------------------------------
169    * Argument     : int sector_no ; I : Sector number
170    *-------------------------------------------------------------------------------
171    * Return Value : void
172    *-------------------------------------------------------------------------------
173    * Note         : None
174    *""FUNC COMMENT END""*************************************************************/
175   void sf_sector_erase(int sector_no)
176   {
177     unsigned char cmd[4];
178     unsigned long addr = sector_no * SF_SECTOR_SIZE;
179
180     cmd[0] = SFLASHCMD_SECTOR_ERASE;
181     cmd[1] = (addr >> 16) & 0xff;
182     cmd[2] = (addr >>  8) & 0xff;
183     cmd[3] =  addr        & 0xff;
184
185     write_enable();
186     io_cmd_exe(cmd, 4, NULL, 0);
187     busy_wait();
188   }
```

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

```
/*""FUNC COMMENT""*****************************************************************
 * ID           :
 * Outline      : Program data
 *-----------------------------------------------------------------------------
 * Include      :
 *-----------------------------------------------------------------------------
 * Declaration  : void sf_byte_program(unsigned long addr, unsigned char *buf, int size);
 *-----------------------------------------------------------------------------
 * Description  : Programs the specified data in serial flash memory.
 *              : Before erasing or programming, issue the Write Enable command.
 *              : After erasing or programming, make sure to check the status of
 *              : serial flash memory if it is not busy.
 *              : The maximum write data size depends on the type of the device.
 *-----------------------------------------------------------------------------
 * Argument     : unsigned long addr ; I : Address in serial flash memory to write
 *              : unsigned char *buf ; I : Buffer address to store the write data
 *              : int size           ; I : Number of bytes to write
 *-----------------------------------------------------------------------------
 * Return Value : void
 *-----------------------------------------------------------------------------
 * Note         : None
 *""FUNC COMMENT END""*************************************************************/
void sf_byte_program(unsigned long addr, unsigned char *buf, int size)
{
  unsigned char cmd[4];

  cmd[0] = SFLASHCMD_BYTE_PROGRAM;
  cmd[1] = (unsigned char)((addr >> 16) & 0xff);
  cmd[2] = (unsigned char)((addr >>  8) & 0xff);
  cmd[3] = (unsigned char)( addr        & 0xff);
  write_enable();
  io_cmd_exe(cmd, 4, buf, size);
  busy_wait();
}
```

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

```
/*""FUNC COMMENT""*****************************************************************
 * ID           :
 * Outline      : Read data
 *-----------------------------------------------------------------------------
 * Include      :
 *-----------------------------------------------------------------------------
 * Declaration  : void sf_byte_read(unsigned long addr, unsigned char *buf, int size);
 *-----------------------------------------------------------------------------
 * Description  : Reads the specified number of bytes from serial flash memory.
 *-----------------------------------------------------------------------------
 * Argument     : unsigned long addr ; I : Address in serial flash memory to read
 *              : unsigned char *buf ; I : Buffer address to store the read data
 *              : int size           ; I : Number of bytes to read
 *-----------------------------------------------------------------------------
 * Return Value : void
 *-----------------------------------------------------------------------------
 * Note         : None
 *""FUNC COMMENT END""*************************************************************/
void sf_byte_read(unsigned long addr, unsigned char *buf, int size)
{
  unsigned char cmd[4];

  cmd[0] = SFLASHCMD_BYTE_READ_LOW;
  cmd[1] = (unsigned char)((addr >> 16) & 0xff);
  cmd[2] = (unsigned char)((addr >>  8) & 0xff);
  cmd[3] = (unsigned char)( addr        & 0xff);
  io_cmd_exe_rdmode(cmd, 4, buf, size);
}
/*""FUNC COMMENT""*****************************************************************
 * ID           :
 * Outline      : Write enable
 *-----------------------------------------------------------------------------
 * Include      :
 *-----------------------------------------------------------------------------
 * Declaration  : static void write_enable(void);
 *-----------------------------------------------------------------------------
 * Description  : Issues the Write Enable command to enable erasing or programming
 *              : serial flash memory.
 *-----------------------------------------------------------------------------
 * Argument     : void
 *-----------------------------------------------------------------------------
 * Return Value : void
 *-----------------------------------------------------------------------------
 * Note         : None
 *""FUNC COMMENT END""*************************************************************/
static void write_enable(void)
{
  unsigned char cmd[1];
  cmd[0] = SFLASHCMD_WRITE_ENABLE;
  io_cmd_exe(cmd, 1, NULL, 0);
}
```

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

```
/*""FUNC COMMENT""*************************************************************
 * ID           :
 * Outline      : Write disable
 *-----------------------------------------------------------------------------
 * Include      :
 *-----------------------------------------------------------------------------
 * Declaration  : static void write_disable(void);
 *-----------------------------------------------------------------------------
 * Description  : Issues the Write Disable command to disable erasing or programming
 *              : serial flash memory.
 *-----------------------------------------------------------------------------
 * Argument     : void
 *-----------------------------------------------------------------------------
 * Return Value : void
 *-----------------------------------------------------------------------------
 * Note         : None
 *""FUNC COMMENT END""*********************************************************/
static void write_disable(void)
{
  unsigned char cmd[1];
  cmd[0] = SFLASHCMD_WRITE_DISABLE;
  io_cmd_exe(cmd, 1, NULL, 0);
}
/*""FUNC COMMENT""*************************************************************
 * ID           :
 * Outline      : Busy waiting
 *-----------------------------------------------------------------------------
 * Include      :
 *-----------------------------------------------------------------------------
 * Declaration  : static void busy_wait(void);
 *-----------------------------------------------------------------------------
 * Description  : Loops internally when the serial flash memory is busy.
 *-----------------------------------------------------------------------------
 * Argument     : void
 *-----------------------------------------------------------------------------
 * Return Value : void
 *-----------------------------------------------------------------------------
 * Note         : None
 *""FUNC COMMENT END""*********************************************************/
static void busy_wait(void)
{
  while ((read_status() & 0x01) != 0) {
    /* serial flash is busy */
  }
}
```

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

```
/*""FUNC COMMENT""*****************************************************************
 * ID           :
 * Outline      : Read status
 *-----------------------------------------------------------------------------
 * Include      :
 *-----------------------------------------------------------------------------
 * Declaration  : static unsigned char read_status(void);
 *-----------------------------------------------------------------------------
 * Description  : Reads the status of serial flash memory.
 *-----------------------------------------------------------------------------
 * Argument     : void
 *-----------------------------------------------------------------------------
 * Return Value : Status register value
 *-----------------------------------------------------------------------------
 * Note         : None
 *""FUNC COMMENT END""*************************************************************/
static unsigned char read_status(void)
{
  unsigned char buf;
  unsigned char cmd[1];

  cmd[0] = SFLASHCMD_READ_STATUS;
  io_cmd_exe_rdmode(cmd, 1, &buf, 1);
  return buf;
}
/*""FUNC COMMENT""*****************************************************************
 * ID           :
 * Outline      : Write status
 *-----------------------------------------------------------------------------
 * Include      :
 *-----------------------------------------------------------------------------
 * Declaration : static void write_status(unsigned char status);
 *-----------------------------------------------------------------------------
 * Description : Writes the status of serial flash memory.
 *-----------------------------------------------------------------------------
 * Argument    : unsigned char status ; I : status register value
 *-----------------------------------------------------------------------------
 * Return Value: void
 *-----------------------------------------------------------------------------
 * Note         : None
 *""FUNC COMMENT END""*************************************************************/
static void write_status(unsigned char status)
{
  unsigned char cmd[2];

  cmd[0] = SFLASHCMD_WRITE_STATUS;
  cmd[1] = status;

  write_enable();
  io_cmd_exe(cmd, 2, NULL, 0);
  busy_wait();
}
```

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

```
/*""FUNC COMMENT""*************************************************************
 * ID          :
 * Outline     : RSPI initialization
 *-----------------------------------------------------------------------------
 * Include     :
 *-----------------------------------------------------------------------------
 * Declaration : static void io_init_rspi(void);
 *-----------------------------------------------------------------------------
 * Description : Initializes the RSPI.
 *             : Sets the RSPI in master mode to set parameters required to transfer
 *             : according to the specifications of serial flash memory.
 *-----------------------------------------------------------------------------
 * Argument    : void
 *-----------------------------------------------------------------------------
 * Return Value: void
 *-----------------------------------------------------------------------------
 * Note         : None
 *""FUNC COMMENT END""*********************************************************/
static void io_init_rspi(void)
{
  /* ==== PFC ==== */
  PFC.PACRL3.BIT.PA9MD = 5;   /* SSL0 */
  PFC.PACRL3.BIT.PA8MD = 5;   /* MISO */
  PFC.PACRL2.BIT.PA7MD = 5;   /* MOSI */
  PFC.PACRL2.BIT.PA6MD = 5;   /* RSPCK */

  /* ==== CPG ==== */
  STB.CR5.BIT._RSPI = 0;  /* RSPI active */

  /* ==== RSPI ==== */
  RSPI.SPCR.BYTE   =  0x00;   /* Disables the RSPI */
  RSPI.SPPCR.BYTE  =  0x30;   /* MOSI idle fixed value = 1 */
  RSPI.SPBR.BYTE   = 0x02;/* Specifies the base bit rate as 6.67 MHz */
                          /* (P clock = 40 MHz) */
  RSPI.SPDCR.BYTE  =  0x00;   /* Access width of the SPDR register: 16-bit */
  RSPI.SPCKD.BYTE  =  0x00;   /* RSPCK delay: 1 RSPCK */
  RSPI.SSLND.BYTE  =  0x00;   /* SSL negation delay: 1 RSPCK */
  RSPI.SPND.BYTE   =  0x00;   /* Next access delay: 1 RSPCK */
  RSPI.SPSCR.BYTE  =  0x00;   /* Sequence length: 1 (Only SPCMD0 is used) */
  RSPI.SPCMD0.WORD = 0xE783;  /* MSB first */
                          /* Data length: 8-bit */
                          /* Keeps the SSL signal level after a transfer */
                          /* is completed */
                          /* Bit rate: Base bit rate is not divided */
                          /* RSPCK when idling is 1 */
                          /* Outputs data on odd edge, latches data on even edge */
  RSPI.SSLP.BYTE   =  0x00;   /* SSLP = b'0 SSL signal 0-active */
  RSPI.SPCR.BYTE   =  0x48;   /* Master mode */
                          /* Disables interrupts */
                          /* Enables the RSPI */
}
```

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

```
/*""FUNC COMMENT""*****************************************************************
 * ID           :
 * Outline      : Execute command (No read data).
 *-----------------------------------------------------------------------------
 * Include      :
 *-----------------------------------------------------------------------------
 * Declaration  : static int io_cmd_exe(unsigned char *ope, int ope_sz,
 *              :                       unsigned char *data,int data_sz)
 *-----------------------------------------------------------------------------
 * Description  : Executes the specified command.
 *              : Transmits the argument ope, and then transmits the argument
 *              : data. Discards the received data.
 *              : Set one of the values between 0 and 8 to the ope_sz.
 *              : Set one of the values between 0 and 256 to the data_sz.
 *-----------------------------------------------------------------------------
 * Argument     : unsigned char *ope ; I : Start address of the opcode block
 *                                     : address block to transmit
 *              : int ope_sz         ; I : Number of bytes in the opcode block and
 *                                     : address block
 *              : unsigned char *data; I : Start address of the data block to transmit
 *              : int data_sz        ; I : Number of bytes in the data block
 *-----------------------------------------------------------------------------
 * Return Value : void
 *-----------------------------------------------------------------------------
 * Note         : None
 *""FUNC COMMENT END""*************************************************************/
static void io_cmd_exe(unsigned char *ope, int ope_sz, unsigned char *data, int data_sz)
{
  /* ---- Enables the SPI transfer ---- */
  RSPI.SPCR.BIT.SPE = 1;

  /* ---- MOSI ---- */
  io_rspi_transfer(ope, NULL, ope_sz);
  io_rspi_transfer(data, NULL, data_sz);

  /* ---- SPI transfer is completed (SSL negation) ---- */
  RSPI.SPCR.BIT.SPE = 0;
}
```

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

```
460  /*""FUNC COMMENT""*****************************************************************
461   * ID           :
462   * Outline      : Execute command (With read data).
463   *-------------------------------------------------------------------------------
464   * Include      :
465   *-------------------------------------------------------------------------------
466   * Declaration  : static void io_cmd_exe_rdmode(unsigned char *ope, int ope_sz,
467   *              :                               unsigned char *rd,  int rd_sz)
468   *-------------------------------------------------------------------------------
469   * Description  : Executes the specified command.
470   *              : Transmits the argument ope, and then receives data in the
471   *              : argument rd. Set one of the values between 0 and 8 in the ope_sz.
472   *              : More than 0 can be set in the rd_sz.
473   *-------------------------------------------------------------------------------
474   * Argument     : unsigned char *ope ; I : Start address of the opcode block and
475   *                                       : address block to transmit
476   *              : int ope_sz         ; I : Number of bytes in the opcode block and
477   *                                       : address block
478   *              : unsigned char *rd  ; I : Buffer address to store the received data
479   *              : int rd_sz          ; I : Number of bytes in the data block
480   *-------------------------------------------------------------------------------
481   * Return Value : void
482   *-------------------------------------------------------------------------------
483   * Note         : None
484   *""FUNC COMMENT END""*************************************************************/
485  static void io_cmd_exe_rdmode(unsigned char *ope, int ope_sz, unsigned char *rd, int rd_sz)
486  {
487    /* ---- Enables the SPI transfer ---- */
488    RSPI.SPCR.BIT.SPE = 1;
489
490    /* ---- MISO ---- */
491    io_rspi_transfer(ope, NULL, ope_sz);
492    io_rspi_transfer(NULL, rd, rd_sz);
493
494    /* ---- SPI transfer is completed (SSL negation) ---- */
495    RSPI.SPCR.BIT.SPE = 0;
496  }
```

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

```
/*""FUNC COMMENT""*****************************************************************
 * ID           :
 * Outline      : RSPI data transfer
 *-----------------------------------------------------------------------------
 * Include      :
 *-----------------------------------------------------------------------------
 * Declaration  : int io_rspi_transfer(unsigned char *write_data,
 *              :           unsigned char *read_data, int data_sz);
 *-----------------------------------------------------------------------------
 * Description  : Transfers commands and data in bytes. Transmits the opcode or
 *              : data from the argument write_data, and receives the data in the
 *              : argument read_data.
 *              : When the argument write_data is NULL, this function transmits
 *              : the dummy data (0xff). When the argument read_data is NULL,
 *              : this function does not receive the data.
 *-----------------------------------------------------------------------------
 * Argument     : unsigned char *write_data : I : Start address of the transmit data
 *              : unsinged char *read_data : O : Buffer address to store the
 *                                             : received data
 *              : int data_sz : I : Number of bytes of the transmit and received data
 *-----------------------------------------------------------------------------
 * Return Value : 0 : Succeeded to transfer data
 *              : -1: Overrun error occurs
 *-----------------------------------------------------------------------------
 * Note         : None
 *""FUNC COMMENT END""*************************************************************/
static int io_rspi_transfer(unsigned char *write_data, unsigned char *read_data, int data_sz)
{
  unsigned short tmp;

  while(data_sz--){
    while(RSPI.SPSR.BIT.SPTEF == 0){
        /* wait */
    }
    /* Writes the transmit data to the data register */
    if(write_data != (unsigned char *)0){
        tmp = (unsigned short)*write_data++;
    }
    else{
        tmp = 0x00ff;  /* Dummy write data */

    }

    RSPI.SPDR.WORD = 0x00ff & tmp;

    RSPI.SPSR.BIT.SPTEF = 0;   /* Clears the bit to 0 to transmit data */

```

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

```
544       /* Waits until the reception is completed */
545       while((RSPI.SPSR.BYTE & 0x81) == 0x00){
546           /* Waits until the receive buffer is full or an overrun error occurs */
547       }
548
549       /* Overrun error occurs? */
550       if(RSPI.SPSR.BIT.OVRF == 1){
551           RSPI.SPSR.BIT.OVRF = 0;
552           return -1;  /* Overrun error occurred */
553       }
554
555       /* Reads the received data */
556       tmp = RSPI.SPDR.WORD;
557       if(read_data != (unsigned char *)0){
558           *read_data++ = (unsigned char)tmp;
559       }
560       RSPI.SPSR.BIT.SPRF = 0;
561     }
562
563     return 0;
564   }
565
566   /* End of File */
```

## 3.17 Sample Program Listing "serial_flash.h" (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 : SH7239 Sample Program
31  *   File Name   : serial_flash.h
32  *   Abstract    : Interfacing Serial Flash Memory Using the Renesas Serial
33  *               : Peripheral Interface
34  *   Version     : 1.00.00
35  *   Device      : SH7239
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: R0K572390 (CPU board)
41  *   Description : Connects the serial flash memory with the MCU using the Renesas
42  *               : Serial Peripheral Interface.
43  ***********************************************************************
44  *   History     : Aug.17,2010 Ver.1.00.00
45  *""FILE COMMENT END""*****************************************************/
46  #ifndef _SERIAL_FLASH_H_
47  #define _SERIAL_FLASH_H_
48
```

## 3.18 Sample Program Listing "serial_flash.h" (2/2)

```
49  /* ==== Macro definition ==== */
50  #define SF_PAGE_SIZE       128         /* Page size of serial flash memory */
51  #define SF_SECTOR_SIZE     0x8000      /* Sector size = 32 KB */
52  #define SF_NUM_OF_SECTOR   4           /* Number of sectors: 4  */
53  #define SF_BUFF_SIZE       0x1000      /* Buffer area to verify the serial */
54                                         /* flash data = 4 KB */
55  enum sf_req{
56    SF_REQ_PROTECT = 0,                  /* Requests to protect */
57    SF_REQ_UNPROTECT                     /* Requests to unprotect */
58  };
59  /* ==== Function prototype declaration ==== */
60  void sf_init_serial_flash(void);
61  void sf_protect_ctrl(enum sf_req req);
62  void sf_chip_erase(void);
63  void sf_sector_erase(int sector_no);
64  void sf_byte_program(unsigned long addr, unsigned char *buf, int size);
65  void sf_byte_read(unsigned long addr, unsigned char *buf, int size);
66
67  #endif /* _SERIAL_FLASH_H_ */
68  /* End of File */
```

## 4. References

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

- Hardware Manual
  SH7239 Group, SH7237 Group Hardware User's Manual Rev.1.00
- The latest version of the hardware user's manual 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 | Nov.24.10 | — | 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