SH7262/SH7264 Group

High-speed Read/Write Serial Flash Memory Using the Renesas Serial Peripheral Interface

Summary

This application note describes how to read or write serial flash memory in high-speed 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 2
2. Applications 3
3. Sample Program Listing 19
4. References 43

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.
- Procedures are optimized to access the large volume data in high-speed.

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 Interfacing Serial Flash Memory Using the Renesas Serial Peripheral Interface
- 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 application accesses serial flash memory in high-speed for large volume data.

This chapter describes the pin connection example and flow charts of the sample program.

2.1 RSPI Operation

SH7264 RSPI supports 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)

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

Figure 1 shows an example of serial flash memory circuit. Set the SH7264 pin functions as shown in Table 2.

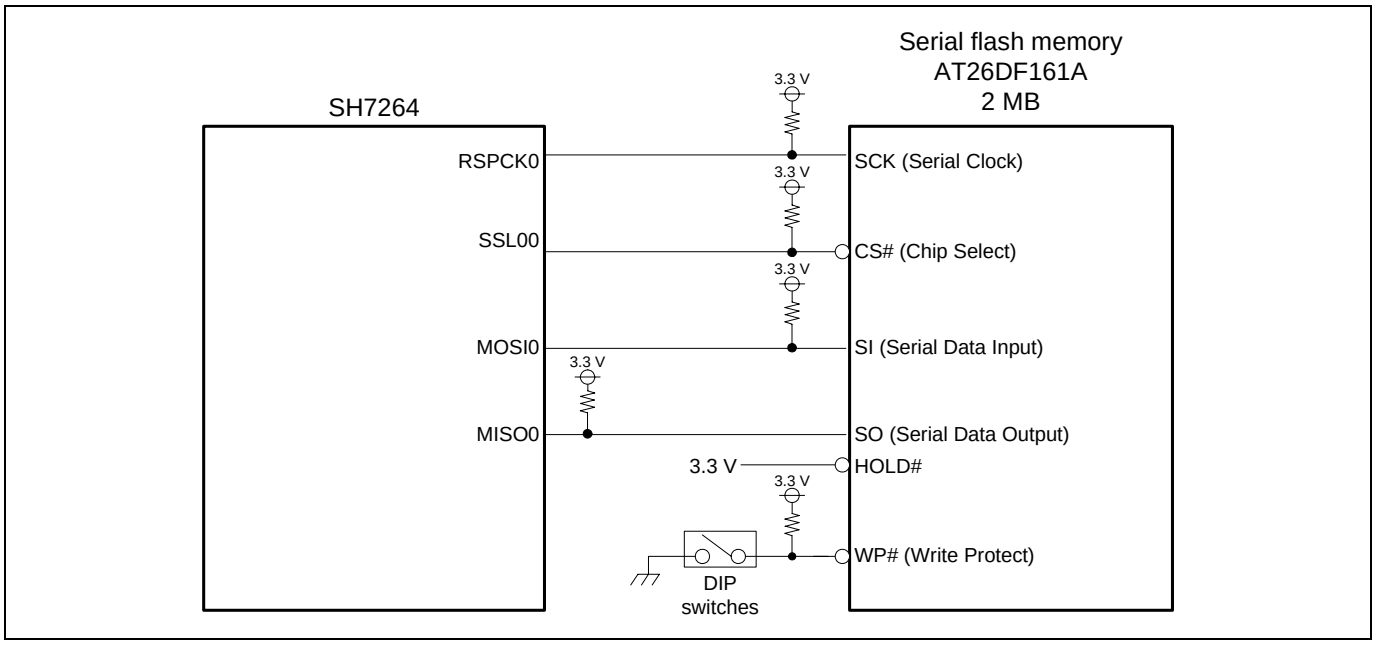


Figure 1 Serial Flash Memory Circuit

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 Pins

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 When Accessing in High-speed

This section describes an example of the interface timing when accessing serial flash memory in high-speed. The interface timing by the typical procedure to control the SPI is explained, as well as the procedure to read/write serial flash memory in high-speed.

2.3.1 Interface Timing by the Typical Procedure to Control SPI

Figure 2 shows an example of the data transfer timing by the typical procedure to control SPI. According to the specifications of the serial flash memory used in this application, both master and slave output data on the falling edge of the clock, and latch data on the rising edge of the clock after a half cycle later. This procedure supports full-duplex communication.

For details on this procedure, refer to the application note "SH7262/SH7264 Group Interfacing Serial Flash Memory Using the Renesas Serial Peripheral Interface".

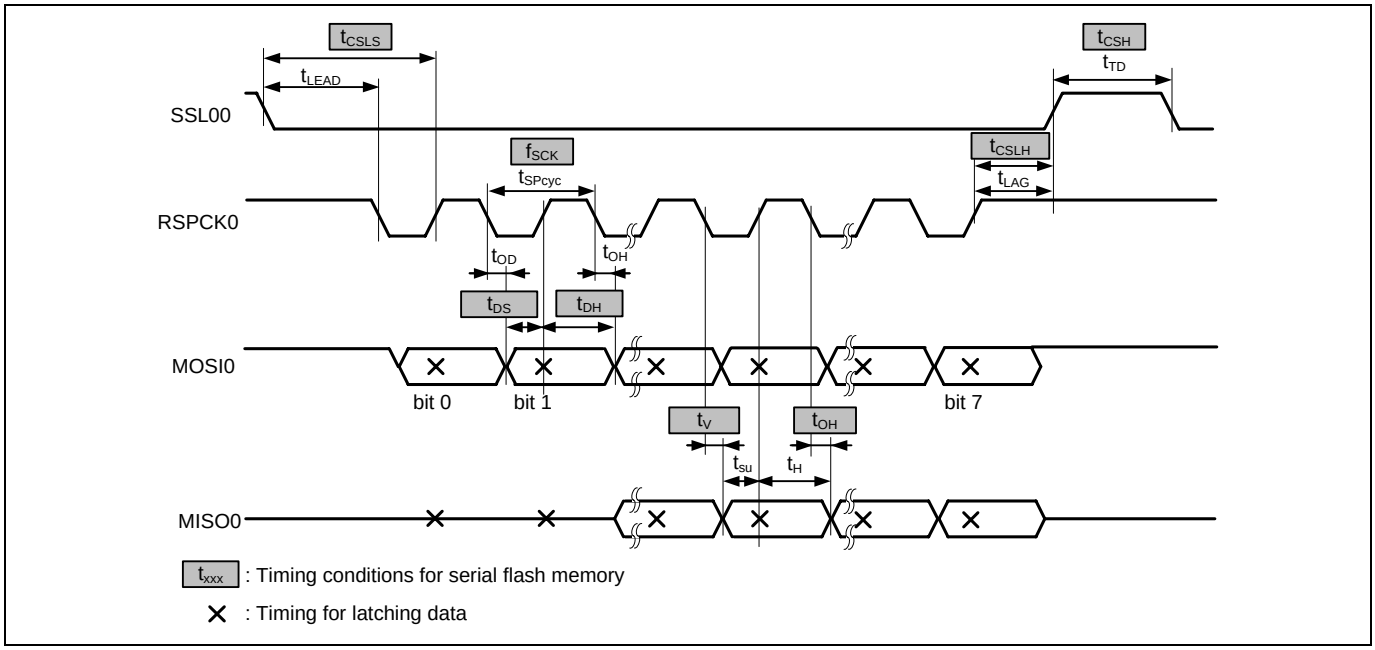


Figure 2 Data Transfer Timing Example by the Typical Procedure to Control SPI (CPOL = 1, CPHA =1)

2.3.2 Extending the Setup Time and the Access Width

This section describes RSPI setting and the interface timing when accessing serial flash memory in high-speed.

This example extends the setup time to one cycle, to specify the RSPCK as 36 MHz, and extends the access width to the data register (SPDR) on data transfer to the longword-wide (32-bit). As this procedure requires a complex control, however, it allows the SPI to transfer data efficiently.

(1) Extending the Setup Time

The setup time by the typical control procedure described in 2.3.1 is less than half a cycle of the RSPCK. The SH7264 data input setup time (t_{SU}) is 15 ns at minimum. When setting the RSPCK frequency at 36 MHz at maximum (when the bus clock is 72 MHz), the half cycle is approximately 13 ns at minimum. As it does not satisfy the timing condition, extend the setup time to allow the RSPCK frequency at 36 MHz.

Following example describes how to extend the setup time when using the Read Array command.

The figure below shows the command sequence for the Read Array command (Opcode: H'0B). The former part of the transfer is MOSI, the SH7264 (Master) outputs commands and addresses. The latter part of the transfer is MISO, the serial flash memory (Master) outputs data. To extend the setup time, change CPOL and CPHA bits settings in the SPCMD register in the former part and latter part of the transfer. Table 3 describes the CPOL bit and the CPHA bit.

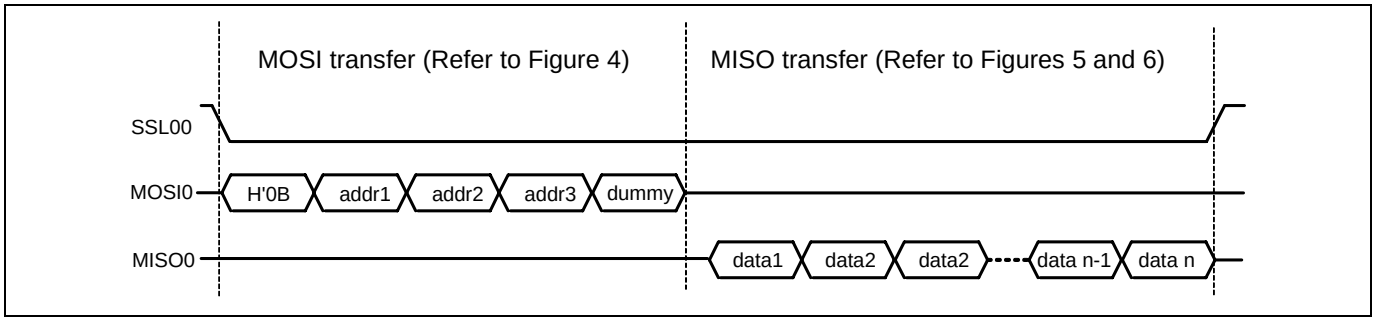


Figure 3 Command Sequence When Extending the Setup Time (Read Array Command)

Table 3 CPOL Bit and CPHA bit

Register Name	Bit	Bit Name	R/W	Description
Command register (SPCMD)	1	CPOL	R/W	RSPCK Polarity Setting Specifies the RSPCK polarity in master or slave mode. When transferring/receiving data between the RSPI and the other module, set the polarity of the RSPCK at the same level. 0: RSPCK = 0 when idle 1: RSPCK = 1 when idle
	0	CPHA	R/W	RSPCK Phase Setting Specifies the RSPCK phase in master or slave mode. When transferring/receiving data between the RSPI and the other module, set the phase of the RSPCK at the same level. 0: Latches the data on odd edge, and outputs data on even edge 1: Outputs data on odd edge, and latches on even edge

This section describes the MOSI transfer.

To extend the setup time, set the timing between the master output and slave input (latch data) as one cycle of the RSPCK. As the serial flash memory used in this application latches data on the rising edge, the SH7264 must outputs data on the preceding rising edge.

There are two combinations of options for bit setting as (CPOL = 1, CPHA =0) or (CPOL = 0, CPHA = 1) for the master to output data on the rising edge. This example uses (CPOL = 1, CPHA = 0) for the following reason.

When setting the CPHA bit to 1, the master (SH7264) outputs the first data bit on the first RSPCK edge (on the rising edge when the CPOL bit is 0), not upon asserting SSL signal. And the slave (serial flash memory) latches data on the first rising edge. Therefore, when setting the CPOL bit to 0, and the CPHA bit to 1, the slave latches data when the master outputs the first bit of data. This setting does not satisfy the setup condition.

When using the setting (CPOL =1, CPHA = 0), the master outputs the first data bit upon asserting SSL signal. There is more than one cycle before the first rising edge of the RSPCK, the timing when the slave latches data. This setting satisfies the setup condition. From the second data bit, the master outputs data on the rising edge of the RSPCK, and the slave latches data on the next rising edge to satisfy the timing condition. The following figure shows the MOSI transfer timing when setting (CPOL = 1, CPHA = 0).

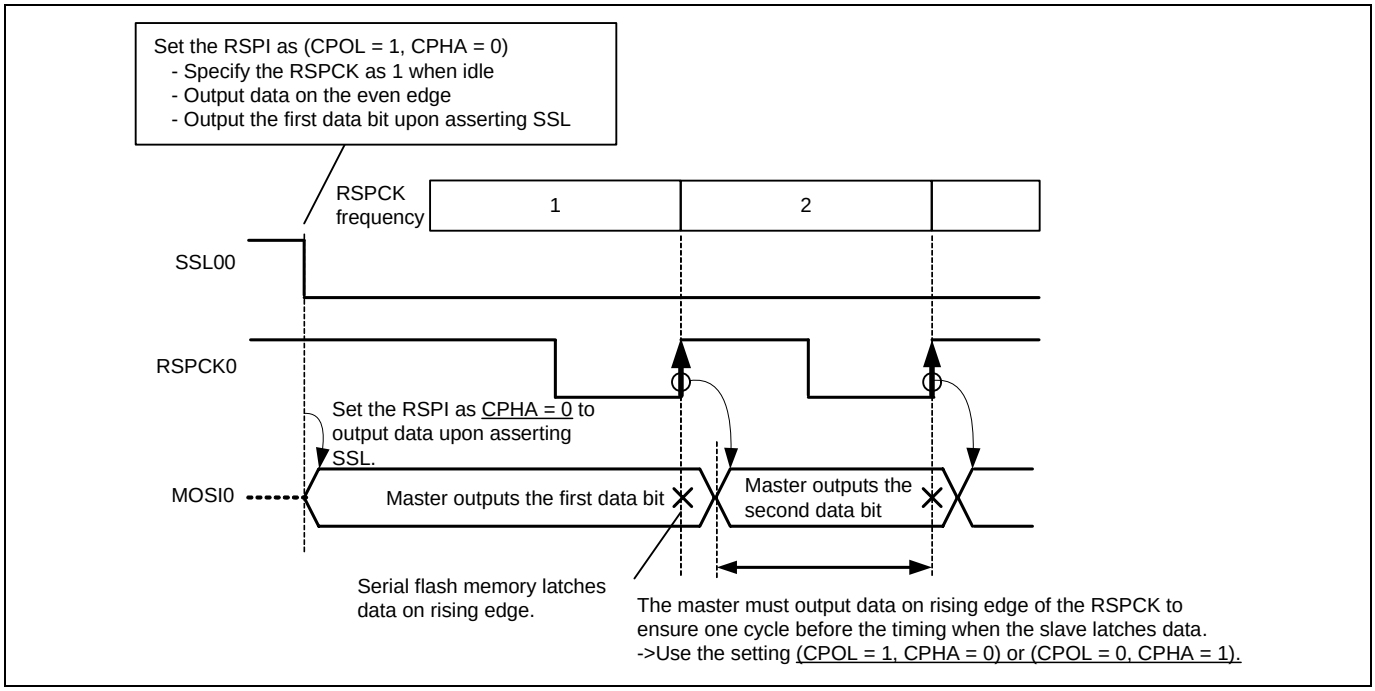


Figure 4 Interface Timing on MOSI Transfer

This section describes the MISO transfer.

As the master latches data in the MISO transfer, set CPOL and CPHA bits so that the master latches data one cycle after the slave outputs data. As the serial flash memory used in this application outputs data on the falling edge, the SH7264 must latches data on the preceding edge (falling edge). (CPOL =1, CPHA = 0) setting is already used in the MOSI transfer, however, change the setting to (CPOL 0, CPHA = 1) for the following reason.

Figure 5 shows the timing without changing the settings of CPOL and CPHA bits. As the master latches data when the slave outputs data on the falling edge of the RSPCK falling edge, this setting does not satisfy the timing condition.

Figure 6 shows the timing for (CPOL = 0, CPHA = 1). As the RSPCK falls when changing the RSPI setting, the slave outputs data at the same timing. Then, the master latches data one cycle after the falling edge of the RSPCK. This setting satisfies the timing condition.

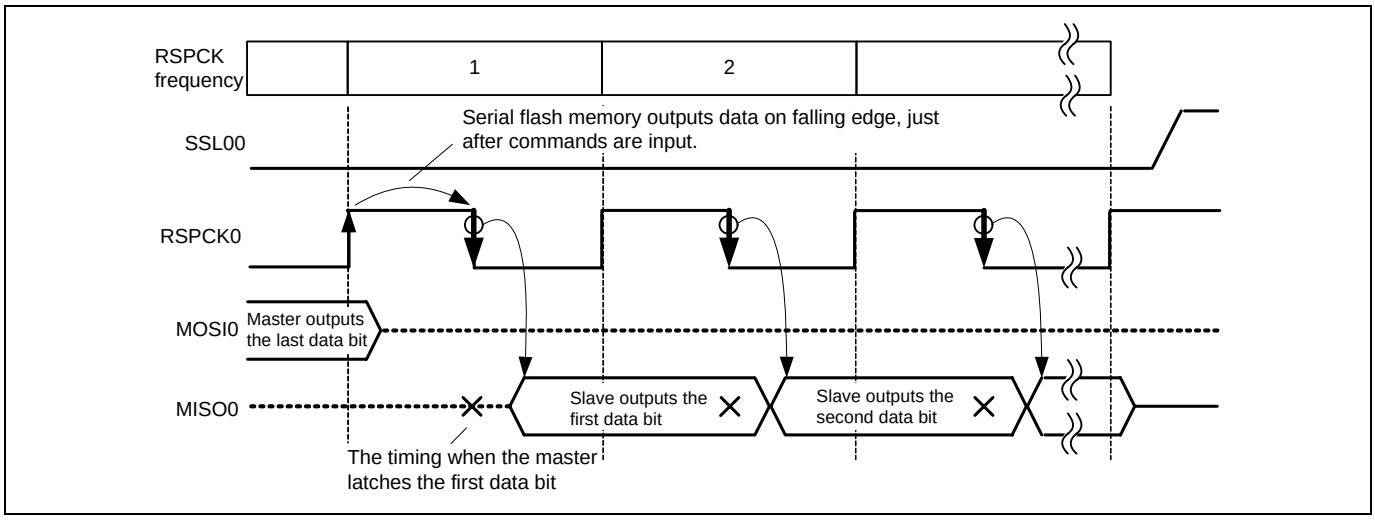


Figure 5 Interface Timing on MISO Transfer (CPOL and CPHA bits are not changed)

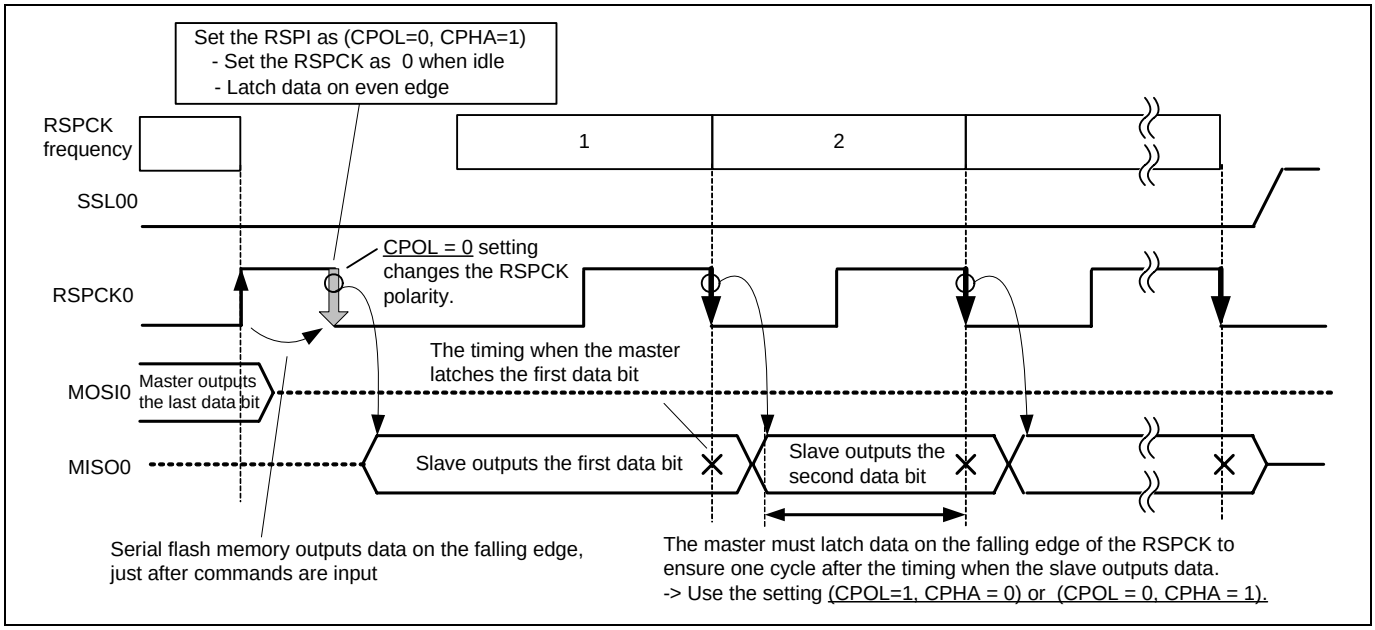


Figure 6 Interface Timing on MISO Transfer (CPOL and CPHA bits are changed)

Figure 7 shows the interface timing when extending the setup time. Table 4 and Table 5 list the timing conditions for serial flash memory and the SH7264. Set the RSPI to satisfy these conditions.

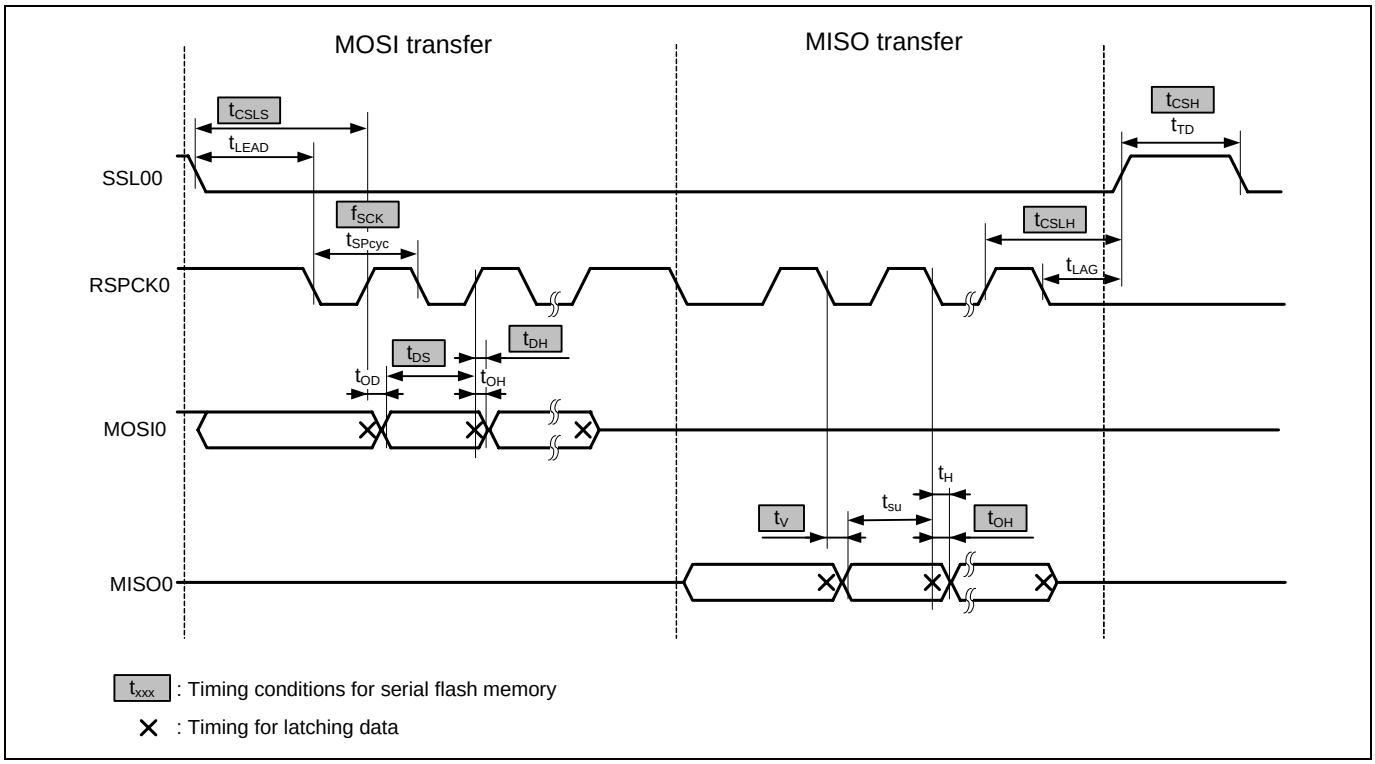


Figure 7 Interface Timing When Extending the Setup Time

Table 4 Timing Conditions for Serial Flash Memory When Extending the Setup Time

Symbol	Item	Description	Related registers
t_{CSLS}	Chip Select Low Setup Time	Time required for the slave 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. 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) + 1/2 t_{SPcyc} > t_{CSLH} (min)	SSLND register SPCMD register
t_{DS}	Data Input Setup Time	Time required for the master from outputting data to latching data. The following formula must be fulfilled: t_{SPcyc} – t_{OD}(max) > t_{DS} (min)	
t_{DH}	Data Input Hold Time	Time required for the master from latching data to stop the data output. The following formula must be fulfilled: t_{OH}(min) > t_{DH} (min)	

Table 5 Timing Conditions for the SH7264 MCU when Extending the Setup Time

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

(2) Extending the Access Width

Specifying the longword-wide access to the Data register (SPDR) reduces the number of times to insert waits (RSPCK delay, SSL negation delay, the next access delay) before and after the transfer to transfer data effectively.

When issuing the read command (Opcode: H'0B), the number of bytes output by master (command, address, and dummy data) is five. Therefore, the master outputs and transfers data in byte-wide length, and the slave outputs and transfers data in longword-wide length. The figure below shows an example of the command sequence of the extended access width.

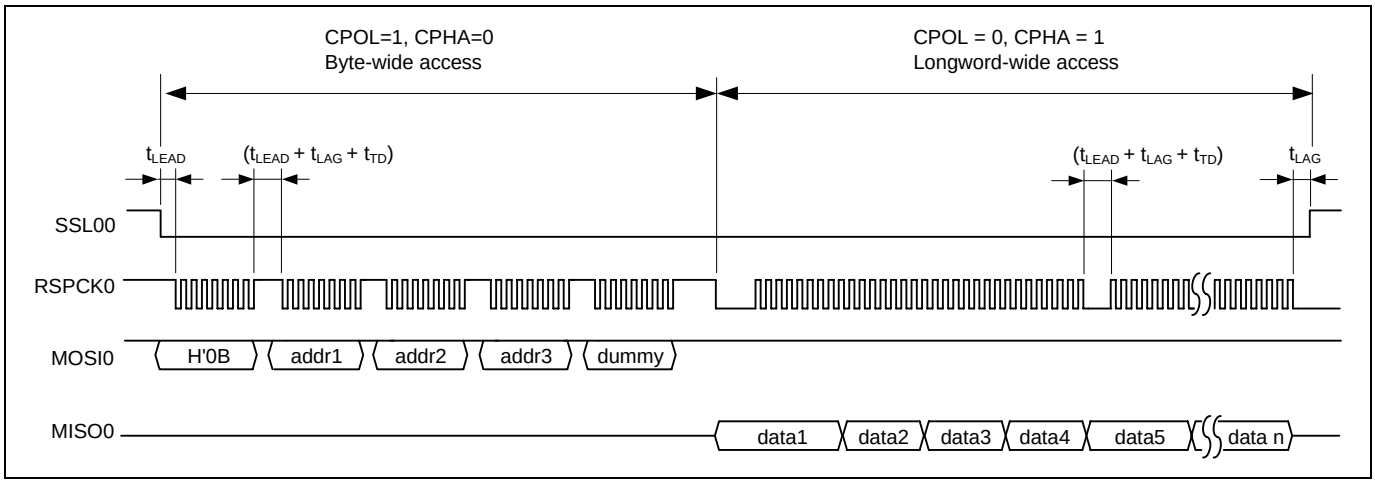


Figure 8 Command Sequence for Longword-wide Access (Opcode: H'0B)

2.4 Sample Program Operation

2.4.1 RSPI Initialization Example

Figure 9 and Figure 10 show flow charts of initializing the RSPI in the sample program. This setting enables the SPI operation in master mode.

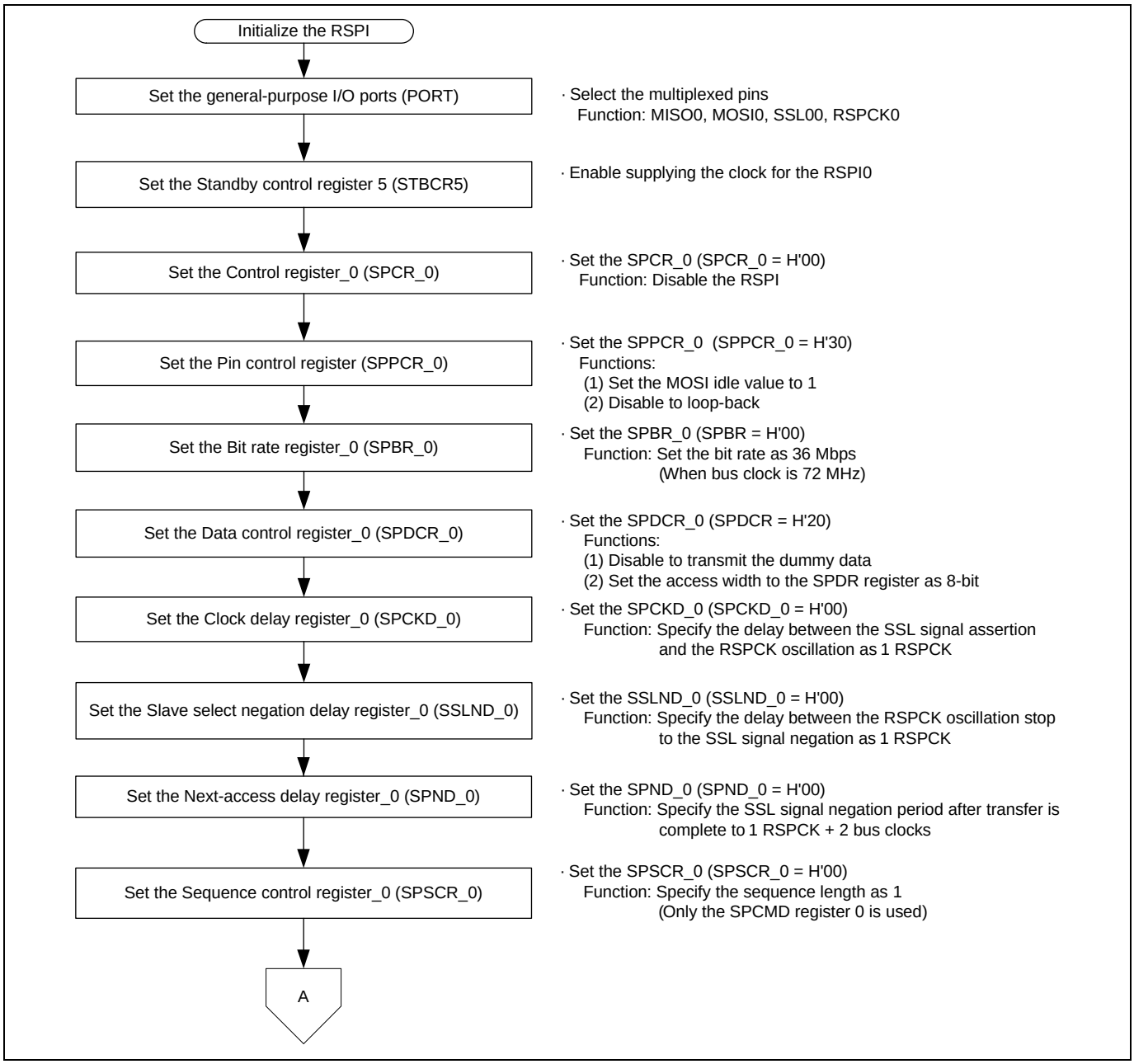


Figure 9 RSPI Initialization Flow Chart (1/2)

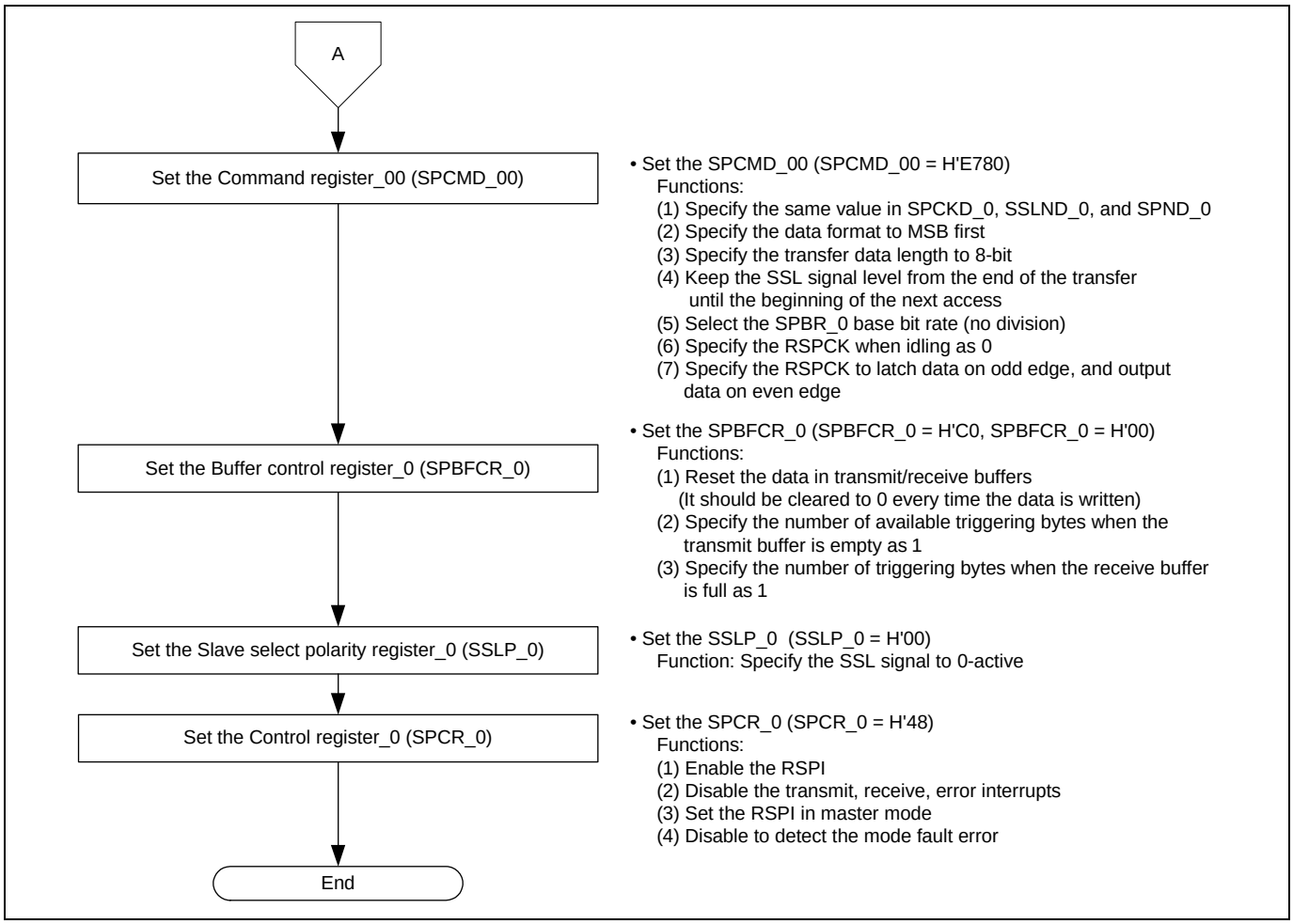


Figure 10 RSPI Initialization Flow Chart (2/2)

2.4.2 Command Transfer Example

The sample program supports two types of command, the Read command that uses both the master output and slave output, and the Write command that uses the master output only. Figure 11 to Figure 13 show the flow charts of the read command transfer. The access width when reading data is specified in longword (32-bit). Use the DMA transfer to store data in memory.

Figure 14 shows the flow chart of the write command transfer. As the busy time is longer than the time to transfer commands, the access width is specified in byte-wide in this example.

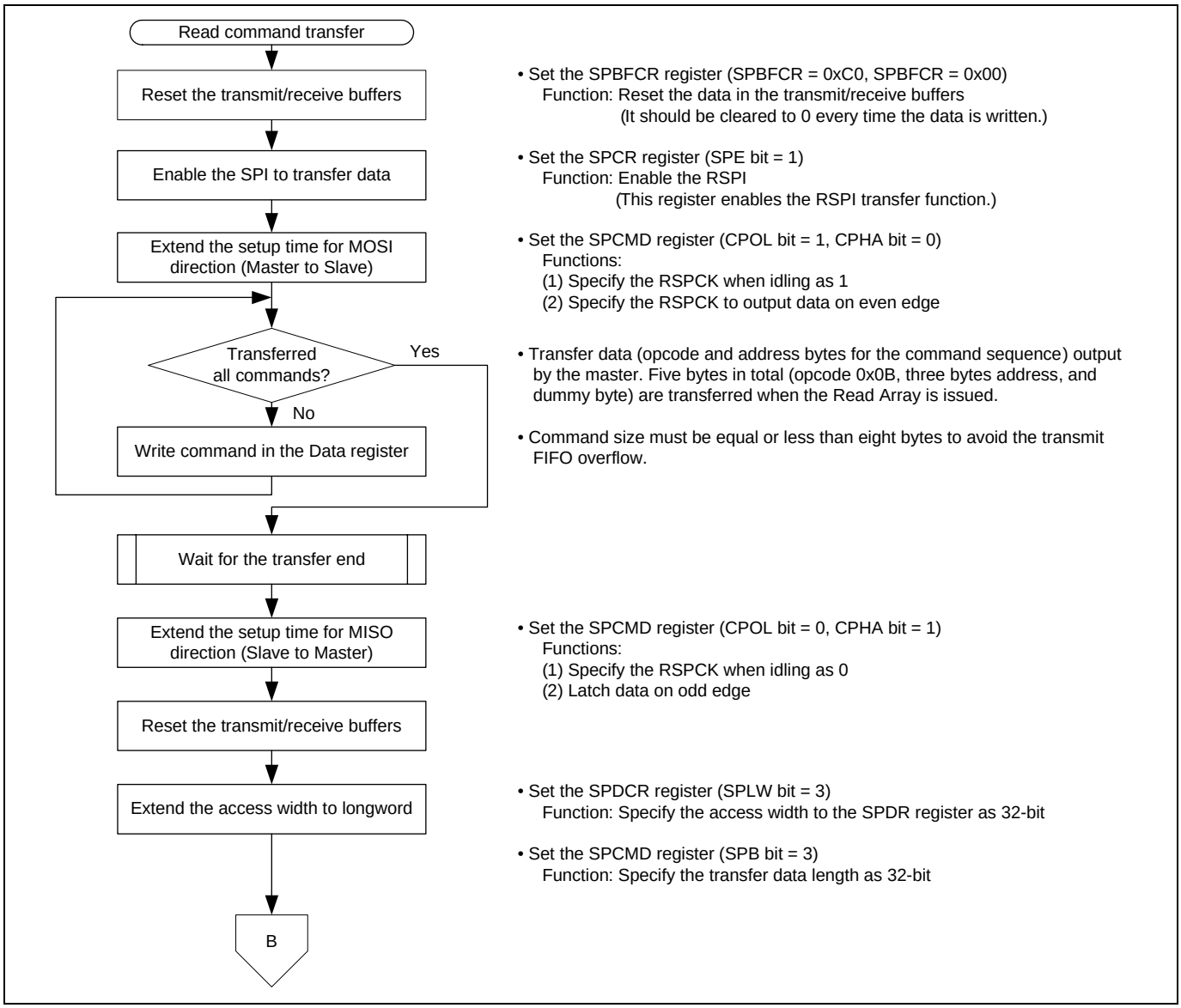


Figure 11 Flow Chart of the Read Command Transfer (1/3)

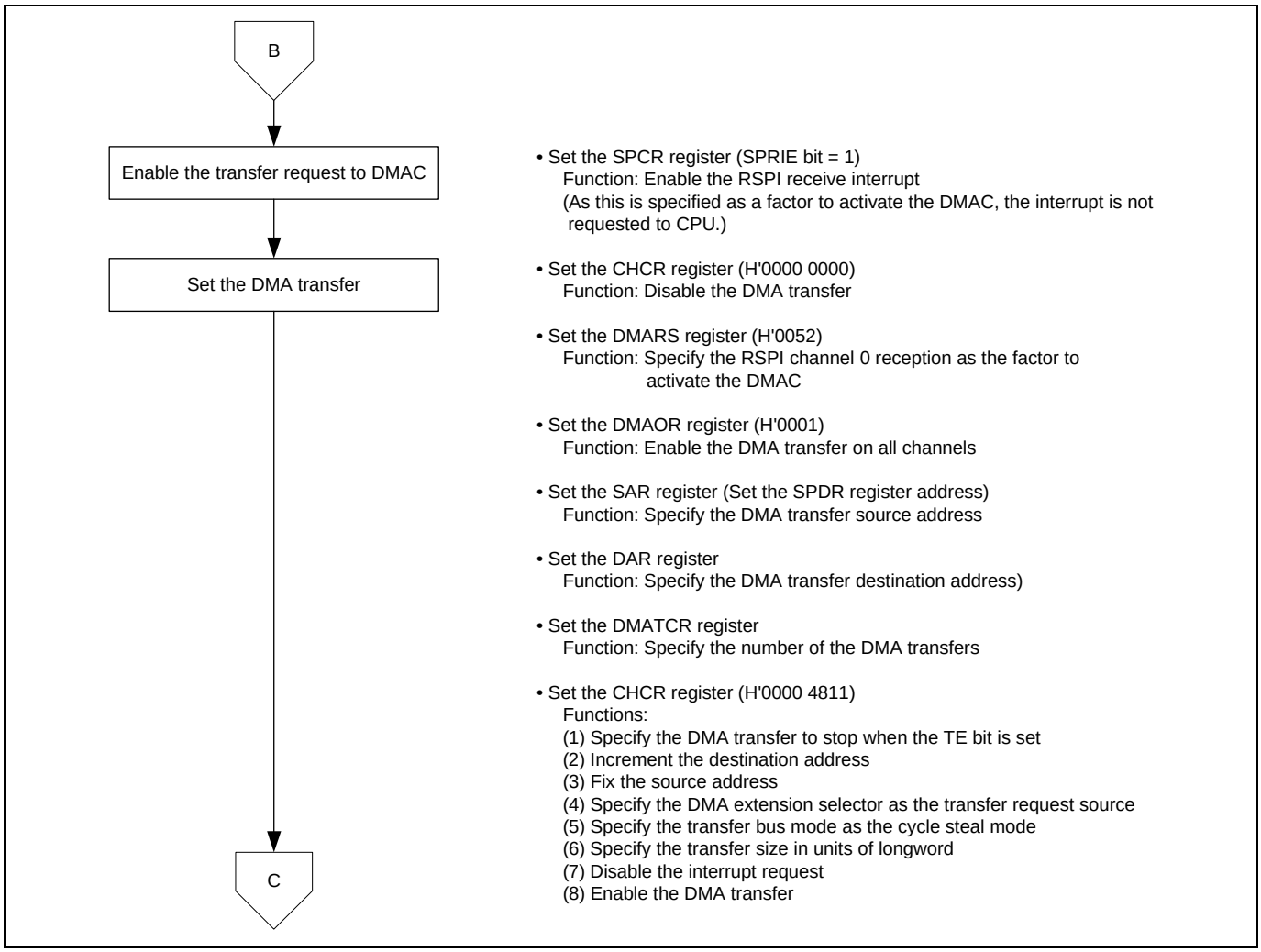
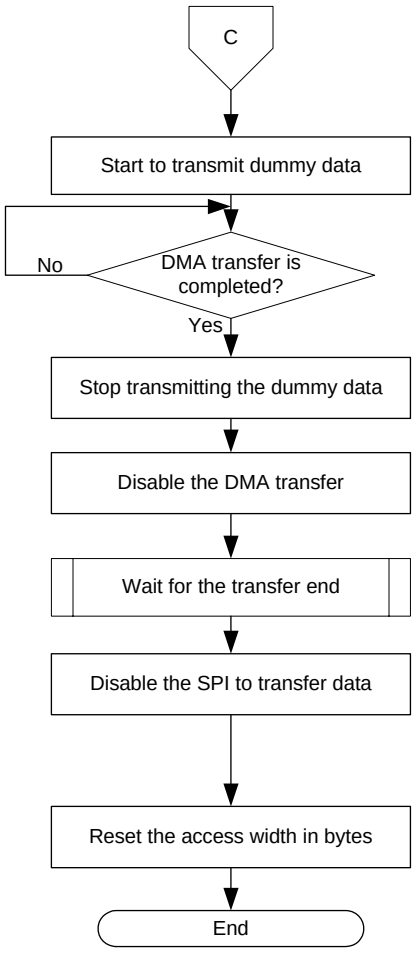


Figure 12 Flow Chart of the Read Command Transfer (2/3)



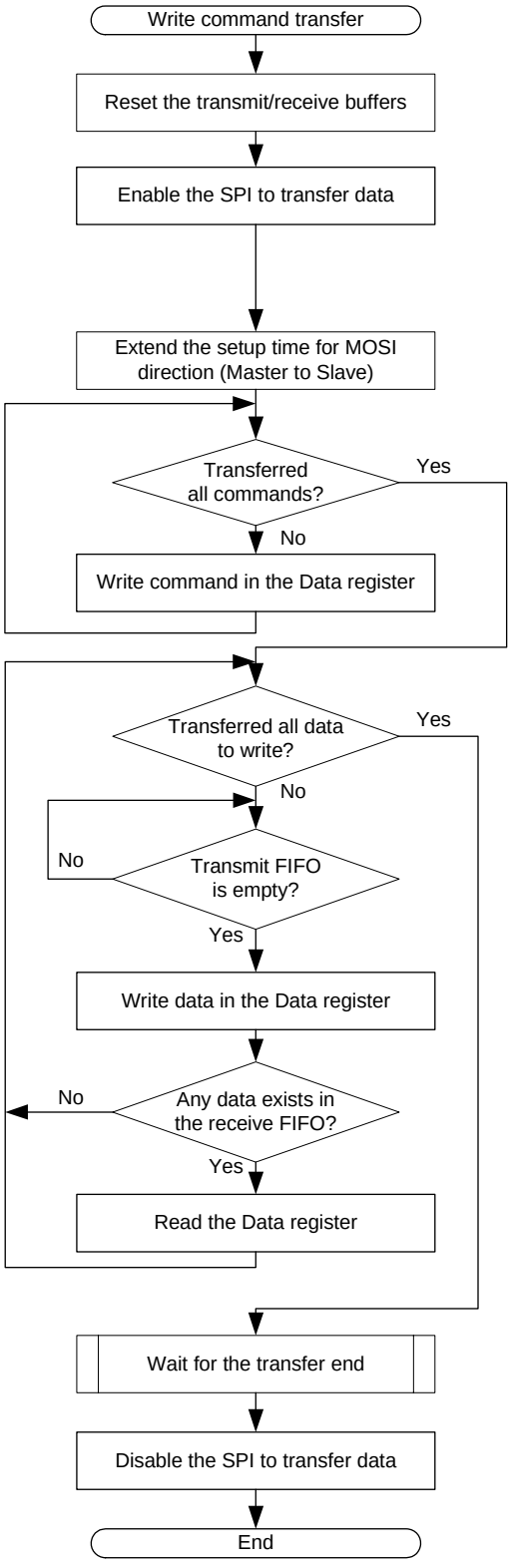
• Set the SPDCR register (TXDMY bit = 1)
Function: Enable the SPI to transmit dummy data (The dummy data is automatically sent when not writing data in the Data register.)

• Make sure the transfer is complete before clearing the SPE bit.

• Set the SPCR register (SPE bit = 0)
Function: Disable the RSPI
SSL signal is negated by this setting. As the RSPI control bit is not initialized, reset the SPE bit to 1 to activate the RSPI in the same transfer mode.

• Set the SPDCR register (SPLW bit = 1)
Function: Specify the access width to the SPDR register as 8-bit
• Set the SPCMD register (SPB bit = 7)
Function: Specify the transfer data length as 8-bit

Figure 13 Flow Chart of the Read Command Transfer (3/3)



• Set the SPBFCR register (SPBFCR = H'C0, SPBFCR = H'00)
Function: Reset the data in 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 rewrite data when SP bit is 1, pay close attention when setting this register. The Command register (SPCMD) does not restrict the the SPE bit value when the TEND bit is 1.

• Set the SPCMD register (CPOL bit = 1, CPHA bit = 0)
Functions:
(1) Specify the RSPCK when idling as 1
(2) Output data on even edge

• Transfer data (opcode and address bytes for command sequence) output by the master

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

• Read the dummy data as the RSPI transfer stops when the receive FIFO overflows upon the RSPI is operating as the master.

• 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 14 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.

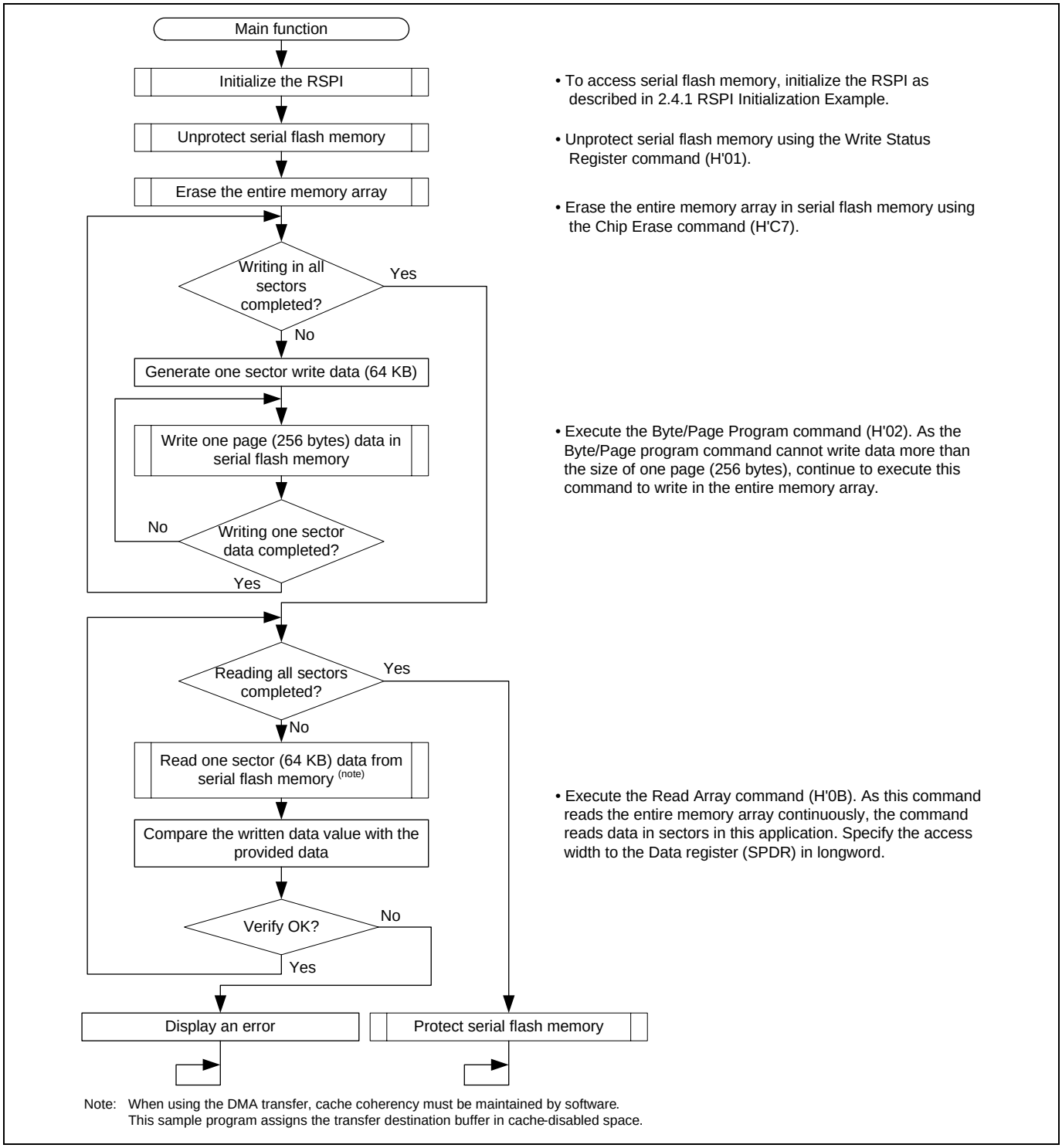


Figure 15 Main Function Flow Chart

3. Sample Program Listing

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

```
1       /***********************************************************************
2       *   DISCLAIMER
3       *
4       *   This software is supplied by Renesas Technology Corp. and is only
5       *   intended for use with Renesas products. No other uses are authorized.
6       *
7       *   This software is owned by Renesas Technology Corp. 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      *   TECHNOLOGY CORP. 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 THE 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) 2009. Renesas Technology Corp., All Rights Reserved.
29      *""FILE COMMENT""*********** Technical reference data *************************
30      *   System Name : SH7264 Sample Program
31      *   File Name   : main.c
32      *   Abstract    : High-speed Read/Write Serial Flash Memory
33      *               : Using the Renesas Serial Peripheral Interface
34      *   Version     : 1.00.00
35      *   Device      : SH7262/SH7264
36      *   Tool-Chain  : High-performance Embedded Workshop (Ver.4.04.01).
37      *               : C/C++ compiler package for the SuperH RISC engine family
38      *               :                              (Ver.9.02 Release00).
39      *   OS          : None
40      *   H/W Platform: M3A-HS64G50 (CPU board)
41      *   Description :
42      ***********************************************************************
43      *   History     : Apr.21,2009 Ver.1.00.00
44      *""FILE COMMENT END""*****************************************************/
```

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

```
45    #include <stdio.h>
46    #include "serial_flash.h"
47
48    /* ==== Macro definition ==== */
49    #define TOP_ADDRESS    0              /* Start address of serial flash memory */
50
51    /* ==== Function prototype declaration ==== */
52    void main(void);
53
54    /* ==== Variable definition ==== */
55    #pragma section LARGE_ONCHIP_RAM
56    static unsigned char data[SF_SECTOR_SIZE];
57    static unsigned long rbuf[SF_SECTOR_SIZE/sizeof(long)];
58    #pragma section
59
60    /*""FUNC COMMENT""*****************************************************************
61     * ID           :
62     * Outline      : Accessing serial flash memory main
63     *-------------------------------------------------------------------------------
64     * Include      : "serial_flash.h"
65     *-------------------------------------------------------------------------------
66     * Declaration  : void main(void);
67     *-------------------------------------------------------------------------------
68     * Description  : Erases, programs, and reads serial flash memory.
69     *              : After initializing the RSPI channel 0, erases the entire memory
70     *              : array, and writes data from the start address. Reads the
71     *              : written data to compare to the provided data.
72     *-------------------------------------------------------------------------------
73     * Argument     : void
74     *-------------------------------------------------------------------------------
75     * Return Value : void
76     *-------------------------------------------------------------------------------
77     * Note         : None
78     *""FUNC COMMENT END""*************************************************************/
79    void main(void)
80    {
81      int i, j;
82      unsigned char *p;
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
```

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

```
91      /* ==== Chip erase (2 MB, it takes about 10 seconds to complete) ==== */
92      sf_chip_erase();
93
94      /* ==== Writes data (2 MB, it takes about 10 seconds to complete) ==== */
95      addr = TOP_ADDRESS;
96      for(i = 0; i < SF_NUM_OF_SECTOR; i++){
97        /* ---- Initializes the data (64 KB) ---- */
98        for(j = 0; j < SF_SECTOR_SIZE; j++){
99            data[j] = (i + j) % 100;
100       }
101       /* ---- Writes one sector (64 KB) data ---- */
102       for(j = 0; j < ( SF_SECTOR_SIZE / SF_PAGE_SIZE ); j++){
103           /* ---- Writes one page (256 bytes) data ---- */
104           sf_byte_program( addr, data+(j*SF_PAGE_SIZE), SF_PAGE_SIZE );
105           addr += SF_PAGE_SIZE;             /* Updates the destination address to write */
106       }
107     }
108     /* ==== Reads data (2 MB) ==== */
109     addr = TOP_ADDRESS;
110     for(i = 0; i < SF_NUM_OF_SECTOR; i++){
111
112       /* ---- Reads one sector (64 KB) data ---- */
113       sf_byte_read_long( addr, rbuf, SF_SECTOR_SIZE );
114       addr += SF_SECTOR_SIZE;                   /* Updates the source address to read */
115
116       /* ---- Verifies data ---- */
117       p = (unsigned char *)rbuf;
118       for(j = 0; j < SF_SECTOR_SIZE; j++){
119           data[j] = (i + j) % 100;          /* Outputs the written data */
120           if( data[j] != *(p+j) ){
121               puts("Error: verify error\n");
122               fflush(stdout);
123               while(1);
124           }
125       }
126     }
127     /* ==== Protects serial flash memory ==== */
128     sf_protect_ctrl( SF_REQ_PROTECT );
129
130     while(1){
131          /* loop */
132       }
133   }
134
135   /* End of File */
```

3.4 Sample Program Listing "serial_flash.c" (1/19)

```
1    /***********************************************************************
2    *   DISCLAIMER
3    *
4    *   This software is supplied by Renesas Technology Corp. and is only
5    *   intended for use with Renesas products. No other uses are authorized.
6    *
7    *   This software is owned by Renesas Technology Corp. 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   *   TECHNOLOGY CORP. 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 THE 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) 2009. Renesas Technology Corp., All Rights Reserved.
29   *""FILE COMMENT""*********** Technical reference data *********************
30   *   System Name : SH7264 Sample Program
31   *   File Name   : serial_flash.c
32   *   Abstract    : High-speed Read/Write Serial Flash Memory
33   *               : Using the Renesas Serial Peripheral Interface
34   *   Version     : 1.00.00
35   *   Device      : SH7262/SH7264
36   *   Tool-Chain  : High-performance Embedded Workshop (Ver.4.04.01).
37   *               : C/C++ compiler package for the SuperH RISC engine family
38   *               :                             (Ver.9.02 Release00).
39   *   OS          : None
40   *   H/W Platform: M3A-HS64G50 (CPU board)
41   *   Description :
42   ************************************************************************
43   *   History     : Mar.09,2009 Ver.1.00.00
44   *""FILE COMMENT END""*****************************************************/
45   #include <stdio.h>
46   #include <machine.h>
47   #include "iodefine.h"
48   #include "serial_flash.h"
```

3.5 Sample Program Listing "serial_flash.c" (2/19)

```
49
50   /* ==== Macro definition ==== */
51   #define SFLASHCMD_CHIP_ERASE  0xc7
52   #define SFLASHCMD_SECTOR_ERASE0xd8
53   #define SFLASHCMD_BYTE_PROGRAM0x02
54   #define SFLASHCMD_BYTE_READ       0x0B
55   #define SFLASHCMD_BYTE_READ_LOW   0x03
56   #define SFLASHCMD_WRITE_ENABLE0x06
57   #define SFLASHCMD_WRITE_DISABLE   0x04
58   #define SFLASHCMD_READ_STATUS 0x05
59   #define SFLASHCMD_WRITE_STATUS0x01
60   #define UNPROTECT_WR_STATUS       0x00
61   #define PROTECT_WR_STATUS     0x3C
62
63   #define SF_USE_DMAC                     /* Define this macro when using the function
64                                     (sf_byte_read_long)in the DMA transfer */
65
66   /* ==== Function prototype declaration ==== */
67   /*** Local function ***/
68   static void write_enable(void);
69   static void write_disable(void);
70   static void busy_wait(void);
71   static unsigned char read_status(void);
72   static void write_status(unsigned char status);
73   static void io_init_rspi(void);
74   static void io_cmd_exe(unsigned char *ope, int ope_sz, unsigned char *data, int data_sz);
75   static void io_cmd_exe_rdmode(unsigned char *ope, int ope_sz, unsigned char *rd, int rd_sz);
76   static void io_cmd_exe_rdmode_cpu_l(unsigned char *ope, int ope_sz, unsigned long *rd, int rd_sz);
77   static void io_cmd_exe_rdmode_dma_l(unsigned char *ope, int ope_sz, unsigned long *rd, int rd_sz);
78   static void io_wait_tx_end(void);
79
80   /* ==== Variable definition ==== */
81
82   /*""FUNC COMMENT""*****************************************************************
83    * ID           :
84    * Outline      : Serial flash memory initialization
85    *-----------------------------------------------------------------------------
86    * Include      :
87    *-----------------------------------------------------------------------------
88    * Declaration  : void sf_init_serial_flash(void);
89    *-----------------------------------------------------------------------------
90    * Description  : Initializes serial flash memory for being accessed.
91    *              : Initializes channel 0 of the Renesas Serial Peripheral
92    *              : Interface (RSPI).
93    *-----------------------------------------------------------------------------
94    * Argument     : void
95    *-----------------------------------------------------------------------------
96    * Return Value : void
97    *-----------------------------------------------------------------------------
98    * Note         : None
99    *""FUNC COMMENT END""*************************************************************/
```

3.6 Sample Program Listing "serial_flash.c" (3/19)

```
100   void sf_init_serial_flash(void)
101   {
102     /* ==== Initializes the RSPI0 ==== */
103     io_init_rspi();
104   }
105   /*""FUNC COMMENT""*****************************************************************
106    * ID           :
107    * Outline      : Protect/unprotect operation
108    *-------------------------------------------------------------------------------
109    * Include      : "serial_flash.h"
110    *-------------------------------------------------------------------------------
111    * Declaration  : void sf_init_serial_flash(void);
112    *-------------------------------------------------------------------------------
113    * Description  : Protects or unprotects serial flash memory.
114    *              : Use the argument req to specify. Default setting and unprotecting
115    *              : method depends on the specifications of the serial flash memory.
116    *-------------------------------------------------------------------------------
117    * Argument     : enum sf_req req ; I : SF_REQ_UNPROTECT -> Write-enable all sectors
118    *              :                       SF_REQ_PROTECT   -> Write-protect all sectors
119    *-------------------------------------------------------------------------------
120    * Return Value : void
121    *-------------------------------------------------------------------------------
122    * Note         : None
123    *""FUNC COMMENT END""*************************************************************/
124   void sf_protect_ctrl(enum sf_req req)
125   {
126     if( req == SF_REQ_UNPROTECT ){
127       write_status( UNPROTECT_WR_STATUS);       /* Unprotects total area */
128     }
129     else{
130       write_status( PROTECT_WR_STATUS );    /* Protects total area */
131     }
132   }
```

3.7 Sample Program Listing "serial_flash.c" (4/19)

```
133  /*""FUNC COMMENT""*****************************************************************
134   * ID           :
135   * Outline      : Chip erase
136   *-------------------------------------------------------------------------------
137   * Include      :
138   *-------------------------------------------------------------------------------
139   * Declaration  : void sf_chip_erase(void);
140   *-------------------------------------------------------------------------------
141   * Description  : Erases all bits in serial flash memory.
142   *              : Before erasing or programming, issue the Write Enable command.
143   *              : After erasing or programming, make sure to check the status of
144   *              : serial flash memory if it is not busy.
145   *-------------------------------------------------------------------------------
146   * Argument     : void
147   *-------------------------------------------------------------------------------
148   * Return Value : void
149   *-------------------------------------------------------------------------------
150   * Note         : None
151   *""FUNC COMMENT END""*************************************************************/
152  void sf_chip_erase(void)
153  {
154    unsigned char cmd[1];
155    cmd[0] = SFLASHCMD_CHIP_ERASE;
156
157    write_enable();
158    io_cmd_exe(cmd, 1, NULL, 0);
159    busy_wait();
160  }
161  /*""FUNC COMMENT""*****************************************************************
162   * ID           :
163   * Outline      : Sector erase
164   *-------------------------------------------------------------------------------
165   * Include      :
166   *-------------------------------------------------------------------------------
167   * Declaration  : void sf_sector_erase(void);
168   *-------------------------------------------------------------------------------
169   * Description  : Erases the specified sector in serial flash memory.
170   *              : Before erasing or programming, issue the Write Enable command.
171   *              : After erasing or programming, make sure to check the status of
172   *              : serial flash memory if it is not busy.
173   *-------------------------------------------------------------------------------
174   * Argument     : int sector_no ; I : Sector number
175   *-------------------------------------------------------------------------------
176   * Return Value : void
177   *-------------------------------------------------------------------------------
178   * Note         : None
179   *""FUNC COMMENT END""*************************************************************/
```

3.8 Sample Program Listing "serial_flash.c" (5/19)

```
180   void sf_sector_erase(int sector_no)
181   {
182     unsigned char cmd[4];
183     unsigned long addr = sector_no * SF_SECTOR_SIZE;
184
185     cmd[0] = SFLASHCMD_SECTOR_ERASE;
186     cmd[1] = (addr >> 16) & 0xff;
187     cmd[2] = (addr >>  8) & 0xff;
188     cmd[3] =  addr        & 0xff;
189
190     write_enable();
191     io_cmd_exe(cmd, 4, NULL, 0);
192     busy_wait();
193   }
194   /*""FUNC COMMENT""*****************************************************************
195    * ID           :
196    * Outline      : Program data
197    *-------------------------------------------------------------------------------
198    * Include      :
199    *-------------------------------------------------------------------------------
200    * Declaration  : void sf_byte_program(unsigned long addr, unsigned char *buf, int size);
201    *-------------------------------------------------------------------------------
202    * Description  : Programs the specified data in serial flash memory.
203    *              : Before erasing or programming, issue the Write Enable command.
204    *              : After erasing or programming, make sure to check the status of
205    *              : serial flash memory if it is not busy.
206    *              : The maximum write data size depends on the type of the device.
207    *-------------------------------------------------------------------------------
208    * Argument     : unsigned long addr ; I : Address in serial flash memory to write
209    *              : unsigned char *buf ; I : Buffer address to store the write data
210    *              : int size           ; I : Number of bytes to write
211    *-------------------------------------------------------------------------------
212    * Return Value : void
213    *-------------------------------------------------------------------------------
214    * Note         : None
215    *""FUNC COMMENT END""*************************************************************/
216   void sf_byte_program(unsigned long addr, unsigned char *buf, int size)
217   {
218     unsigned char cmd[4];
219
220     cmd[0] = SFLASHCMD_BYTE_PROGRAM;
221     cmd[1] = (unsigned char)((addr >> 16) & 0xff);
222     cmd[2] = (unsigned char)((addr >>  8) & 0xff);
223     cmd[3] = (unsigned char)( addr        & 0xff);
224     write_enable();
225     io_cmd_exe(cmd, 4, buf, size);
226     busy_wait();
227   }
```

3.9 Sample Program Listing "serial_flash.c" (6/19)

```
228  /*""FUNC COMMENT""*****************************************************************
229   * ID           :
230   * Outline      : Read data (byte transfer).
231   *-------------------------------------------------------------------------------
232   * Include      :
233   *-------------------------------------------------------------------------------
234   * Declaration  : void sf_byte_read(unsigned long addr, unsigned char *buf, int size);
235   *-------------------------------------------------------------------------------
236   * Description  : Reads the specified number of bytes from serial flash memory.
237   *-------------------------------------------------------------------------------
238   * Argument     : unsigned long addr ; I : Address in serial flash memory to read
239   *              : unsigned char *buf ; I : Buffer address to store the read data
240   *              : int size           ; I : Number of bytes to read
241   *-------------------------------------------------------------------------------
242   * Return Value : void
243   *-------------------------------------------------------------------------------
244   * Note         : None
245   *""FUNC COMMENT END""*************************************************************/
246  void sf_byte_read(unsigned long addr, unsigned char *buf, int size)
247  {
248    unsigned char cmd[5];
249
250    cmd[0] = SFLASHCMD_BYTE_READ;
251    cmd[1] = (unsigned char)((addr >> 16) & 0xff);
252    cmd[2] = (unsigned char)((addr >>  8) & 0xff);
253    cmd[3] = (unsigned char)( addr        & 0xff);
254    cmd[4] = 0x00;
255    io_cmd_exe_rdmode(cmd, 5, buf, size);
256  }
257  /*""FUNC COMMENT""*****************************************************************
258   * ID           :
259   * Outline      : Read data (Longword transfer).
260   *-------------------------------------------------------------------------------
261   * Include      :
262   *-------------------------------------------------------------------------------
263   * Declaration  : void sf_byte_read_long(unsigned long addr, unsigned long *buf, int size);
264   *-------------------------------------------------------------------------------
265   * Description  : Reads the specified number of bytes in units of longword
266   *              : from serial flash memory.
267   *-------------------------------------------------------------------------------
268   * Argument     : unsigned long addr ; I : Address in serial flash memory to read
269   *              : unsigned long *buf ; I : Buffer address to store the read data
270   *              : int size           ; I : Number of bytes to read
271   *-------------------------------------------------------------------------------
272   * Return Value : void
273   *-------------------------------------------------------------------------------
274   * Note         : None
275   *""FUNC COMMENT END""*************************************************************/
```

3.10 Sample Program Listing "serial_flash.c" (7/19)

```
276   void sf_byte_read_long(unsigned long addr, unsigned long *buf, int size)
277   {
278     unsigned char cmd[5];
279
280     cmd[0] = SFLASHCMD_BYTE_READ;
281     cmd[1] = (unsigned char)((addr >> 16) & 0xff);
282     cmd[2] = (unsigned char)((addr >>  8) & 0xff);
283     cmd[3] = (unsigned char)( addr        & 0xff);
284     cmd[4]   = 0x00;
285   #ifdef SF_USE_DMAC
286     io_cmd_exe_rdmode_dma_l(cmd, 5, buf, size);
287   #else
288     io_cmd_exe_rdmode_cpu_l(cmd, 5, buf, size);
289   #endif
290   }
291   /*""FUNC COMMENT""*****************************************************************
292    * ID           :
293    * Outline      : Write enable
294    *-------------------------------------------------------------------------------
295    * Include      :
296    *-------------------------------------------------------------------------------
297    * Declaration  : static void write_enable(void);
298    *-------------------------------------------------------------------------------
299    * Description  : Issues the Write Enable command to enable erasing or programming
300    *              : serial flash memory.
301    *-------------------------------------------------------------------------------
302    * Argument     : void
303    *-------------------------------------------------------------------------------
304    * Return Value : void
305    *-------------------------------------------------------------------------------
306    * Note         : None
307    *""FUNC COMMENT END""*************************************************************/
308   static void write_enable(void)
309   {
310     unsigned char cmd[1];
311     cmd[0] = SFLASHCMD_WRITE_ENABLE;
312     io_cmd_exe(cmd, 1, NULL, 0);
313   }
```

3.11 Sample Program Listing "serial_flash.c" (8/19)

```
314   /*""FUNC COMMENT""*****************************************************************
315    * ID           :
316    * Outline      : Write disable
317    *-------------------------------------------------------------------------------
318    * Include      :
319    *-------------------------------------------------------------------------------
320    * Declaration  : static void write_disable(void);
321    *-------------------------------------------------------------------------------
322    * Description  : Issues the Write Disable command to disable erasing or programming
323    *              : serial flash memory.
324    *-------------------------------------------------------------------------------
325    * Argument     : void
326    *-------------------------------------------------------------------------------
327    * Return Value : void
328    *-------------------------------------------------------------------------------
329    * Note         : None
330    *""FUNC COMMENT END""*************************************************************/
331   static void write_disable(void)
332   {
333     unsigned char cmd[1];
334     cmd[0] = SFLASHCMD_WRITE_DISABLE;
335     io_cmd_exe(cmd, 1, NULL, 0);
336   }
337   /*""FUNC COMMENT""*****************************************************************
338    * ID           :
339    * Outline      : Busy wait
340    *-------------------------------------------------------------------------------
341    * Include      :
342    *-------------------------------------------------------------------------------
343    * Declaration  : static void busy_wait(void);
344    *-------------------------------------------------------------------------------
345    * Description  : Loops internally when the serial flash memory is busy.
346    *-------------------------------------------------------------------------------
347    * Argument     : void
348    *-------------------------------------------------------------------------------
349    * Return Value : void
350    *-------------------------------------------------------------------------------
351    * Note         : None
352    *""FUNC COMMENT END""*************************************************************/
353   static void busy_wait(void)
354   {
355     while ((read_status() & 0x01) != 0) {   /* RDY/BSY */
356       /* serial flash is busy */
357     }
358   }
```

3.12 Sample Program Listing "serial_flash.c" (9/19)

```
359   /*""FUNC COMMENT""*****************************************************************
360    * ID           :
361    * Outline      : Read status
362    *-------------------------------------------------------------------------------
363    * Include      :
364    *-------------------------------------------------------------------------------
365    * Declaration  : static unsigned char read_status(void);
366    *-------------------------------------------------------------------------------
367    * Description  : Reads the status of serial flash memory.
368    *-------------------------------------------------------------------------------
369    * Argument     : void
370    *-------------------------------------------------------------------------------
371    * Return Value : Status register value
372    *-------------------------------------------------------------------------------
373    * Note         : None
374    *""FUNC COMMENT END""*************************************************************/
375   static unsigned char read_status(void)
376   {
377     unsigned char buf;
378     unsigned char cmd[1];
379
380     cmd[0] = SFLASHCMD_READ_STATUS;
381     io_cmd_exe_rdmode(cmd, 1, &buf, 1);
382     return buf;
383   }
384   /*""FUNC COMMENT""*****************************************************************
385    * ID           :
386    * Outline      : Write status
387    *-------------------------------------------------------------------------------
388    * Include      :
389    *-------------------------------------------------------------------------------
390    * Declaration  : static void write_status(unsigned char status);
391    *-------------------------------------------------------------------------------
392    * Description  : Writes the status of serial flash memory.
393    *-------------------------------------------------------------------------------
394    * Argument     : unsigned char status ; I : status register value
395    *-------------------------------------------------------------------------------
396    * Return Value : void
397    *-------------------------------------------------------------------------------
398    * Note         : None
399    *""FUNC COMMENT END""*************************************************************/
400   static void write_status(unsigned char status)
401   {
402     unsigned char cmd[2];
403
404     cmd[0] = SFLASHCMD_WRITE_STATUS;
405     cmd[1] = status;
```

3.13 Sample Program Listing "serial_flash.c" (10/19)

```
406
407     write_enable();
408     io_cmd_exe(cmd, 2, NULL, 0);
409     busy_wait();
410   }
411   /*""FUNC COMMENT""*****************************************************************
412    * ID           :
413    * Outline      : RSPI initialization
414    *-------------------------------------------------------------------------------
415    * Include      :
416    *-------------------------------------------------------------------------------
417    * Declaration  : static void io_init_rspi(void);
418    *-------------------------------------------------------------------------------
419    * Description  : Initializes channel 0 of the RSPI.
420    *              : Sets the RSPI in master mode to set parameters to transfer
421    *              : according to the specifications of serial flash memory.
422    *-------------------------------------------------------------------------------
423    * Argument     : void
424    *-------------------------------------------------------------------------------
425    * Return Value : void
426    *-------------------------------------------------------------------------------
427    * Note         : None
428    *""FUNC COMMENT END""*************************************************************/
429   static void io_init_rspi(void)
430   {
431     /* ==== PORT ==== */
432     PORT.PFCR3.BIT.PF12MD = 3;  /* PF12:MISO0 */
433     PORT.PFCR2.BIT.PF11MD = 3;  /* PF11:MOSI0 */
434     PORT.PFCR2.BIT.PF10MD = 3;  /* PF10:SSL00 */
435     PORT.PFCR2.BIT.PF9MD  = 3;  /* PF9:RSPCK0 */
436
437     /* ==== CPG ==== */
438     CPG.STBCR5.BIT.MSTP51 = 0;  /* RSPI0 active */
439
```

3.14 Sample Program Listing "serial_flash.c" (11/19)

```
440     /* ==== RSPI ==== */
441     RSPI0.SPCR.BYTE   =  0x00;   /* Disables channel 0 of the RSPI */
442     RSPI0.SPPCR.BYTE  =  0x30;   /* MOSI idle fixed value = 1 */
443     RSPI0.SPBR.BYTE   = 0x00;    /* Specifies the base bit rate as 36 MHz
444                                                    (Bus clock = 72 MHz) */
445     RSPI0.SPDCR.BYTE  =  0x20;   /* Disables to transmit the dummy data */
446                               /* Access width to the SPDR register: 8-bit */
447     RSPI0.SPCKD.BYTE  =  0x00;   /* RSPCK delay: 1 RSPCK */
448     RSPI0.SSLND.BYTE  =  0x00;   /* SSL negate delay: 1 RSPCK */
449     RSPI0.SPND.BYTE   =  0x00;   /* Next access delay: 1 RSPCK + 2 Bus clocks */
450     RSPI0.SPSCR.BYTE  =  0x00;   /* Sequence length: 1 (SPCMD0 is only used) */
451     RSPI0.SPCMD0.WORD = 0xE780; /* MSB first */
452                               /* Data length: 8-bit */
453                               /* Keeps the SSL signal level after transfer
454                                   is completed */
455                               /* Bit rate: Base bit rate is not divided */
456                               /* RSPCK when idling is 0 */
457                               /* Latches data on odd edge, outputs data on even edge */
458     RSPI0.SPBFCR.BYTE =  0xC0;   /* Enables to reset data in the
459                                    transmit/receive buffer */
460     RSPI0.SPBFCR.BYTE =  0x00;   /* Disables to reset data in the
461                                    transmit/receive buffer */
462                               /* Number of triggers in transmit buffer:
463                                    more than one byte available */
464                               /* Number of triggers in receive buffer:
465                                    more than one byte received */
466     RSPI0.SSLP.BYTE   =  0x00;   /* SSLP = b'0 SSL signal 0-active */
467     RSPI0.SPCR.BYTE   =  0x48;   /* Master mode */
468                               /* Disables interrupts */
469                               /* Enables channel 0 of the RSPI */
470   }
```

3.15 Sample Program Listing "serial_flash.c" (12/19)

```
471  /*""FUNC COMMENT""*****************************************************************
472   * ID           :
473   * Outline      : Execute command (No read data).
474   *-------------------------------------------------------------------------------
475   * Include      :
476   *-------------------------------------------------------------------------------
477   * Declaration  : static void io_cmd_exe(unsigned char *ope, int ope_sz,
478   *              :                        unsigned char *data,int data_sz)
479   *-------------------------------------------------------------------------------
480   * Description  : Executes the specified command.
481   *              : Transmits the argument ope, and then transmits the argument data.
482   *              : Discards the received data.
483   *              : Set one of the values between 0 and 8 in the ope_sz.
484   *              : Set one of the values between 0 and 256 in the data_sz.
485   *-------------------------------------------------------------------------------
486   * Argument     : unsigned char *ope ; I : Start address of the opcode block and
487   *                                         address block to transmit
488   *              : int ope_sz         ; I : Number of bytes in the opcode block and
489   *                                         address block
490   *              : unsigned char *data; I : Start address of the data block to transmit
491   *              : int data_sz        ; I : Number of bytes in the data block
492   *-------------------------------------------------------------------------------
493   * Return Value : void
494   *-------------------------------------------------------------------------------
495   * Note         : None
496   *""FUNC COMMENT END""*************************************************************/
497  static void io_cmd_exe(unsigned char *ope, int ope_sz, unsigned char *data, int data_sz)
498  {
499    unsigned char tmp;
500
501    /* ==== Resets buffer ==== */
502    RSPI0.SPBFCR.BYTE  = 0xC0u;
503    RSPI0.SPBFCR.BYTE  = 0x00u;
504
505    /* ---- Enables the SPI transfer ---- */
506    RSPI0.SPCR.BIT.SPE = 1;
507
508    /* ==== MOSI (command, address, write data) ==== */
509    RSPI0.SPCMD0.BIT.CPOL= 1;       /* RSPCK when idling is 1 */
510    RSPI0.SPCMD0.BIT.CPHA= 0;       /* Outputs data on even (rising) edge */
511
512    while(ope_sz--){
513      RSPI0.SPDR.BYTE = *ope++;  /* Command size must be equal or less than 8 bytes */
514    }
```

3.16 Sample Program Listing "serial_flash.c" (13/19)

```
515     while(data_sz--){
516       while( RSPI0.SPSR.BIT.SPTEF == 0 ){
517           /* wait */
518       }
519       RSPI0.SPDR.BYTE = *data++;
520       if( RSPI0.SPSR.BIT.SPRF == 1 ){
521           tmp = RSPI0.SPDR.BYTE; /* Dummy read to avoid an overflow of data */
522       }
523     }
524     io_wait_tx_end();               /* Waits for transfer end */
525
526     /* ---- SPI transfer end (SSL negation) ---- */
527     RSPI0.SPCR.BIT.SPE = 0;
528   }
529   /*""FUNC COMMENT""*****************************************************************
530    * ID           :
531    * Outline      : Execute command (With read data, byte transfer).
532    *-------------------------------------------------------------------------------
533    * Include      :
534    *-------------------------------------------------------------------------------
535    * Declaration  : static void io_cmd_exe_rdmode(unsigned char *ope, int ope_sz,
536    *              :                               unsigned char *rd,  int rd_sz)
537    *-------------------------------------------------------------------------------
538    * Description  : Executes the specified command.
539    *              : Transmits the argument ope, and then receives data in the argument rd.
540    *              : Transfer data in unit of bytes.
541    *              : Set one of the values between 0 and 8 in the ope_sz.
542    *              : More than 0 can be set in the rd_sz.
543    *-------------------------------------------------------------------------------
544    * Argument     : unsigned char *ope ; I : Start address of the opcode block and
545    *                                         address block to transmit
546    *              : int ope_sz         ; I : Number of bytes in the opcode block and
547    *                                         address block
548    *              : unsigned char *rd  ; I : Buffer address to store the received data
549    *              : int rd_sz          ; I : Number of bytes in the data block
550    *-------------------------------------------------------------------------------
551    * Return Value : void
552    *-------------------------------------------------------------------------------
553    * Note         : None
554    *""FUNC COMMENT END""*************************************************************/
555   static void io_cmd_exe_rdmode(unsigned char *ope, int ope_sz, unsigned char *rd, int rd_sz)
556   {
557     /* ==== Resets buffer ==== */
558     RSPI0.SPBFCR.BYTE  = 0xC0u;
559     RSPI0.SPBFCR.BYTE  = 0x00u;
560
```

3.17 Sample Program Listing "serial_flash.c" (14/19)

```
561     /* ---- Enables the SPI transfer ---- */
562     RSPI0.SPCR.BIT.SPE = 1;
563
564     /* ---- MOSI (command, address, dummy) ---- */
565     RSPI0.SPCMD0.BIT.CPOL= 1;       /* RSPCK when idling is 1 */
566     RSPI0.SPCMD0.BIT.CPHA= 0;       /* Outputs data on even (rising) edge */
567
568     while(ope_sz--){
569       RSPI0.SPDR.BYTE = *ope++;  /* Command size must be equal or less than 8 bytes */
570     }
571     io_wait_tx_end();               /* Waits for transfer end */
572
573     /* ---- MISO (read data) ---- */
574     RSPI0.SPCMD0.BIT.CPOL= 0;       /* RSPCK when idling is 0 */
575     RSPI0.SPCMD0.BIT.CPHA= 1;       /* Latches data on even (falling) edge */
576
577     RSPI0.SPBFCR.BYTE  = 0xC0u;     /* Resets buffer */
578     RSPI0.SPBFCR.BYTE  = 0x00u;
579
580     RSPI0.SPDCR.BIT.TXDMY = 1;      /* Enables to transmit the dummy data */
581     while(rd_sz--){
582       while( RSPI0.SPSR.BIT.SPRF == 0){
583           /* wait */
584       }
585       *rd++ = RSPI0.SPDR.BYTE;
586     }
587     RSPI0.SPDCR.BIT.TXDMY = 0;      /* Disables to transmit the dummy data */
588     io_wait_tx_end();               /* Waits for transfer end */
589
590     /* ---- SPI transfer end (SSL negation) ---- */
591     RSPI0.SPCR.BIT.SPE = 0;
592   }
```

3.18 Sample Program Listing "serial_flash.c" (15/19)

```
593   /*""FUNC COMMENT""*****************************************************************
594    * ID           :
595    * Outline      : Execute command (With read data, longword transfer).
596    *-------------------------------------------------------------------------------
597    * Include      :
598    *-------------------------------------------------------------------------------
599    * Declaration  : static void io_cmd_exe_rdmode_cpu_l(unsigned char *ope, int ope_sz,
600    *              :                                     unsigned long *rd,  int rd_sz);
601    *-------------------------------------------------------------------------------
602    * Description  : Executes the specified command.
603    *              : Transmits the argument ope, and then receives data in the argument rd.
604    *              : Transfer the read data in units of longword.
605    *              : Set one of the values between 0 and 8 in the ope_sz.
606    *              : Although more than 0 can be set in the rd_sz, set the value
607    *              : in multiples of 4.
608    *-------------------------------------------------------------------------------
609    * Argument     : unsigned long *ope ; I : Start address of the opcode block and
610    *                                         address block to transmit
611    *              : int ope_sz         ; I : Number of bytes in the opcode block and
612    *                                       : address block
613    *              : unsigned long *rd  ; I : Buffer address to store the received data
614    *              : int rd_sz          ; I : Number of bytes in the data block
615    *-------------------------------------------------------------------------------
616    * Return Value : void
617    *-------------------------------------------------------------------------------
618    * Note         : None
619    *""FUNC COMMENT END""*************************************************************/
620   static void io_cmd_exe_rdmode_cpu_l(unsigned char *ope, int ope_sz, unsigned long *rd, int
621   rd_sz)
622   {
623     /* ==== Resets buffer ==== */
624     RSPI0.SPBFCR.BYTE  = 0xC0u;
625     RSPI0.SPBFCR.BYTE  = 0x00u;
626
627     /* ---- Enables the SPI transfer ---- */
628     RSPI0.SPCR.BIT.SPE = 1;
629
630     /* ---- MOSI (command, address, dummy) ---- */
631     RSPI0.SPCMD0.BIT.CPOL= 1;       /* RSPCK when idling is 1 */
632     RSPI0.SPCMD0.BIT.CPHA= 0;       /* Outputs data on even (rising) edge */
633
634     while(ope_sz--){
635       RSPI0.SPDR.BYTE = *ope++;  /* Command size must be equal or less than 8 bytes */
636     }
637     io_wait_tx_end();              /* Waits for transfer end */
```

3.19 Sample Program Listing "serial_flash.c" (16/19)

```
638     /* ---- MISO (read data) ---- */
639     RSPI0.SPCMD0.BIT.CPOL= 0;       /* RSPCK when idling is 0 */
640     RSPI0.SPCMD0.BIT.CPHA= 1;       /* Latches data on even (falling) edge */
641
642     RSPI0.SPBFCR.BYTE  = 0xC0u;     /* Resets buffer */
643     RSPI0.SPBFCR.BYTE  = 0x00u;
644
645     RSPI0.SPDCR.BIT.SPLW = 3;       /* Access width to the SPDR register: 32-bit */
646     RSPI0.SPCMD0.BIT.SPB = 3;       /* Transfer data length: 32-bit */
647
648     RSPI0.SPDCR.BIT.TXDMY = 1;          /* Enables to transmit the dummy data */
649     rd_sz >>= 2;                    /* Calculates the number of transfers in longword */
650     while( rd_sz-- ){
651       while( RSPI0.SPSR.BIT.SPRF == 0){
652           /* wait */
653       }
654       *rd++ = RSPI0.SPDR.LONG;
655     }
656     RSPI0.SPDCR.BIT.TXDMY = 0;          /* Disables to transmit the dummy data */
657     io_wait_tx_end();                   /* Waits for transfer end */
658
659     /* ==== Restores the SPI to default setting ==== */
660     RSPI0.SPCR.BIT.SPE = 0;
661     RSPI0.SPDCR.BIT.SPLW = 1;           /* Access width to the SPDR register: 8-bit */
662     RSPI0.SPCMD0.BIT.SPB = 7;           /* Transfer data length: 8-bit */
663   }
664   #ifdef SF_USE_DMAC
```

3.20 Sample Program Listing "serial_flash.c" (17/19)

```
665  /*""FUNC COMMENT""*****************************************************************
666   * ID           :
667   * Outline      : Execute command (With read data, longword transfer and DMA).
668   *-------------------------------------------------------------------------------
669   * Include      :
670   *-------------------------------------------------------------------------------
671   * Declaration  : static void io_cmd_exe_rdmode_dma_l(unsigned char *ope, int ope_sz,
672   *              :                                     unsigned long *rd,  int rd_sz);
673   *-------------------------------------------------------------------------------
674   * Description  : Executes the specified command.
675   *              : Transmits the argument ope, and then receives data in the argument rd.
676   *              : Transfer the read data in units of longword, by the DMA transfer.
677   *              : Set one of the values between 0 and 8 in the ope_sz.
678   *              : Although more than 0 can be set in the rd_sz, set the value
679   *              : in multiples of 4.
680   *-------------------------------------------------------------------------------
681   * Argument     : unsigned long *ope ; I : Start address of the opcode block and
682   *                                         address block
683   *              : int ope_sz         ; I : Number of bytes in the opcode block and
684   *                                         address block
685   *              : unsigned long *rd  ; I : Buffer address to store the received data
686   *              : int rd_sz          ; I : Number of bytes in the data block
687   *-------------------------------------------------------------------------------
688   * Return Value : void
689   *-------------------------------------------------------------------------------
690   * Note         : None
691   *""FUNC COMMENT END""*************************************************************/
692  static void io_cmd_exe_rdmode_dma_l(unsigned char *ope, int ope_sz, unsigned long *rd, int
693  rd_sz)
694  {
695
696    /* ==== Resets buffer ==== */
697    RSPI0.SPBFCR.BYTE  = 0xC0u;
698    RSPI0.SPBFCR.BYTE  = 0x00u;
699
700    /* ---- Enables the SPI transfer ---- */
701    RSPI0.SPCR.BIT.SPE = 1;
702
703    /* ---- MOSI (command, address, dummy) ---- */
704    RSPI0.SPCMD0.BIT.CPOL= 1;       /* RSPCK when idling is 1 */
705    RSPI0.SPCMD0.BIT.CPHA= 0;       /* Outputs data on even (rising) edge */
706
707    while(ope_sz--){
708      RSPI0.SPDR.BYTE = *ope++;  /* Command size must be equal or less than 8 bytes */
709    }
710    io_wait_tx_end();              /* Waits for transfer end */
```

3.21 Sample Program Listing "serial_flash.c" (18/19)

```
711     /* ==== MISO (read data) ==== */
712     RSPI0.SPCMD0.BIT.CPOL= 0;       /* RSPCK when idling is 0 */
713     RSPI0.SPCMD0.BIT.CPHA= 1;       /* Latches data on even (falling) edge */
714
715     RSPI0.SPBFCR.BYTE  = 0xC0u;     /* Resets buffer */
716     RSPI0.SPBFCR.BYTE  = 0x00u;
717
718     RSPI0.SPDCR.BIT.SPLW = 3;       /* Access width to the SPDR register: 32-bit */
719     RSPI0.SPCMD0.BIT.SPB = 3;       /* Transfer data length: 32-bit */
720
721     /* ---- Enables the DMA transfer ---- */
722     RSPI0.SPCR.BIT.SPRIE =  1;         /* Enable an interrupt (for DMA transfer) */
723     DMAC.CHCR0.LONG      =  0x00000000; /* Disables the DMA transfer */
724     DMAC.DMARS0.WORD =  0x0052u;    /* RSPI0 Rx */
725     DMAC.DMAOR.WORD      =  0x0001u;    /* Enables all DMA transfers */
726     DMAC.SAR0.LONG       =  (unsigned long)&(RSPI0.SPDR.BYTE);
727     DMAC.DAR0.LONG       =  (unsigned long)rd;        /* Transfer destination address */
728     DMAC.DMATCR0.LONG    =  rd_sz >> 2;               /* Transfer size */
729     DMAC.CHCR0.LONG      =  (unsigned long)0x00004811;  /* Transfer size (long),
730                                                    enables the DMA transfer */
731
732     /* ---- Receives data ---- */
733     RSPI0.SPDCR.BIT.TXDMY = 1;                       /* Enables to transmit the dummy data */
734     while(DMAC.CHCR0.BIT.TE == 0){                   /* Waits for the DMA transfer end */
735       /* wait */
736     }
737     RSPI0.SPDCR.BIT.TXDMY = 0;                       /* Disables to transmit the dummy data */
738     DMAC.CHCR0.LONG      =  0x00000000ul;            /* Disables the DMA0 */
739     io_wait_tx_end();               /* Waits for transfer end */
740
741     /* ==== Restores the SPI to default setting ==== */
742     RSPI0.SPCR.BIT.SPE = 0;
743     RSPI0.SPDCR.BIT.SPLW = 1;          /* Access width to the SPDR register: 8-bit */
744     RSPI0.SPCMD0.BIT.SPB = 7;          /* Transfer data length: 8-bit */
745   }
746   #endif  /* SF_USE_DMAC */
```

3.22 Sample Program Listing "serial_flash.c" (19/19)

```
747   /*""FUNC COMMENT""*****************************************************************
748    * ID           :
749    * Outline      : Transfer end wait
750    *-------------------------------------------------------------------------------
751    * Include      :
752    *-------------------------------------------------------------------------------
753    * Declaration  : static void io_wait_tx_end(void);
754    *-------------------------------------------------------------------------------
755    * Description  : Loops internally until the transmission is completed.
756    *-------------------------------------------------------------------------------
757    * Argument     : void
758    *-------------------------------------------------------------------------------
759    * Return Value : void
760    *-------------------------------------------------------------------------------
761    * Note         : None
762    *""FUNC COMMENT END""*************************************************************/
763   static void io_wait_tx_end(void)
764   {
765     while(RSPI0.SPSR.BIT.TEND == 0){
766       /* wait */
767     }
768   }
769
770   /* End of File */
```

3.23 Sample Program Listing "serial_flash.h" (1/2)

```
1     /***********************************************************************
2     *   DISCLAIMER
3     *
4     *   This software is supplied by Renesas Technology Corp. and is only
5     *   intended for use with Renesas products. No other uses are authorized.
6     *
7     *   This software is owned by Renesas Technology Corp. 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    *   TECHNOLOGY CORP. 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 THE 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) 2009. Renesas Technology Corp., All Rights Reserved.
29    *""FILE COMMENT""************ Technical reference data *********************
30    *   System Name : SH7264 Sample Program
31    *   File Name   : serial_flash.h
32    *   Abstract    : High-speed Read/Write Serial Flash Memory
33    *               : Using the Renesas Serial Peripheral Interface
34    *   Version     : 1.00.00
35    *   Device      : SH7262/SH7264
36    *   Tool-Chain  : High-performance Embedded Workshop (Ver.4.04.01).
37    *               : C/C++ compiler package for the SuperH RISC engine family
38    *               :                             (Ver.9.02 Release00).
39    *   OS          : None
40    *   H/W Platform: M3A-HS64G50 (CPU board)
41    *   Description :
42    ************************************************************************
43    *   History     : Mar.09,2009 Ver.1.00.00
44    *""FILE COMMENT END""*************************************************/
```

3.24 Sample Program Listing "serial_flash.h" (2/2)

```
45
46    #ifndef _SERIAL_FLASH_H_
47    #define _SERIAL_FLASH_H_
48
49    /* ==== Macro definition ==== */
50    #define SF_PAGE_SIZE      256        /* Page size of serial flash memory */
51    #define SF_SECTOR_SIZE    0x10000    /* Sector size = 64 KB   */
52    #define SF_NUM_OF_SECTOR  32         /* Number of sectors: 32   */
53    enum sf_req{
54      SF_REQ_PROTECT = 0,             /* Requests to protect */
55      SF_REQ_UNPROTECT                /* Requests to unprotect */
56    };
57    /* ==== Function prototype declaration ==== */
58    void sf_init_serial_flash(void);
59    void sf_protect_ctrl(enum sf_req req);
60    void sf_chip_erase(void);
61    void sf_sector_erase(int sector_no);
62    void sf_byte_program(unsigned long addr, unsigned char *buf, int size);
63    void sf_byte_read(unsigned long addr, unsigned char *buf, int size);
64    void sf_byte_read_long(unsigned long addr, unsigned long *buf, int size);
65
66    /* ==== Variable definition ==== */
67
68    #endif /* _SERIAL_FLASH_H_ */
69    /* 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	Jun 30, 2009	—	First edition issued

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

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.