To our customers,

Old Company Name in Catalogs and Other Documents

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

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

April 1st, 2010
Renesas Electronics Corporation

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

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

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

SH7262/SH7264 Group

Renesas Serial Peripheral Interface

Read/Write EEPROM

Summary

This application note describes examples of reading/writing EEPROM using the SH7262/SH7264 Renesas Serial Peripheral Interface (RSPI).

Target Device

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

Contents

1. Introduction

1.1 Specifications

- Use the EEPROM of 16 KB (128 Kbit) to connect with the SH7264 MCU.
- Use the channel 0 of the RSPI to access EEPROM.

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

2. Applications

Connect the SH7264 MCU (Master) with the SPI-compatible EEPROM (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

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

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

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

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

2.2 EEPROM Pin Connection

The following table lists the specifications of the SPI-compliant EEPROM (N58X25128I, Renesas Technology) used in this application.

Table 1 EEPROM Specifications

Item	Description
Power Supply	Single supply 1.8 V to 5.5 V
SPI mode	Supports SPI modes 0 and 3
Clock frequency	5 MHz (2.5 V to 5.5 V), 3 MHz (1.8 V to 5.5 V)
Capacity	16 KB (128 Kbit)
Page Write	64 bytes/page
Write Cycle Time	5 ms (2.5 V min.), 8 ms (1.8 V min.)
Endurance	1,000,000 Erase/Write cycles

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

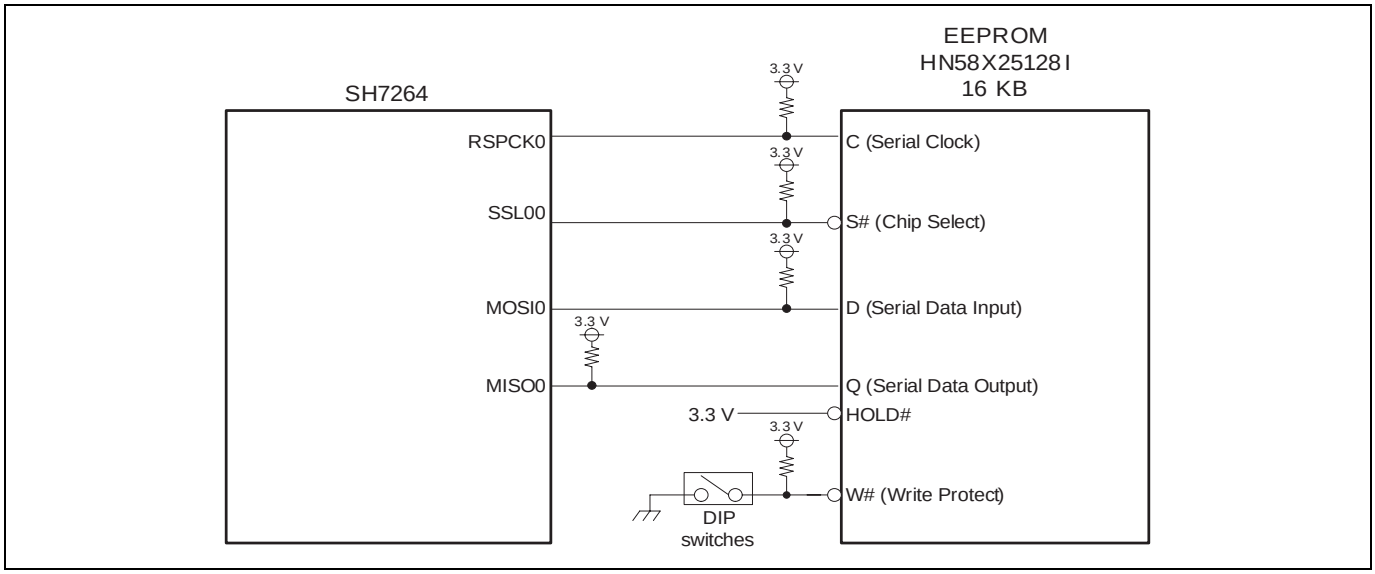

Figure 1 EEPROM Circuit

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

Table 2 Multiplexed Output

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

Note: SH7264 Multiplexed Pins
MISO0, MOSI0, SSL00, and RSPCK0 pins are multiplexed, and set to general-purpose I/O ports as default. Before accessing EEPROM, use the general-purpose I/O port control register to set the multiplexed pins to RSPI pins.

2.3 Interface Timing Example

This section describes an example of the interface timing between the SH7264 and EEPROM. Initialize the RSPI and specify the clock frequency according to the timing condition of the EEPROM (slave device).

Figure 2 shows an example of the data transfer timing. As the EEPROM used in this application latches data at the rising edge of the clock and outputs data at the falling edge of the clock, specify 1 to the CPOL and CPHA bits in the command register (SPCMD). By this setting, RSPCK is specified to 1 when it is idling, and the timing to vary the data in the RSPI can be set to the odd edge (falling edge). Initialize the RSPI to satisfy the timing conditions shown in Table 3 and Table 4.

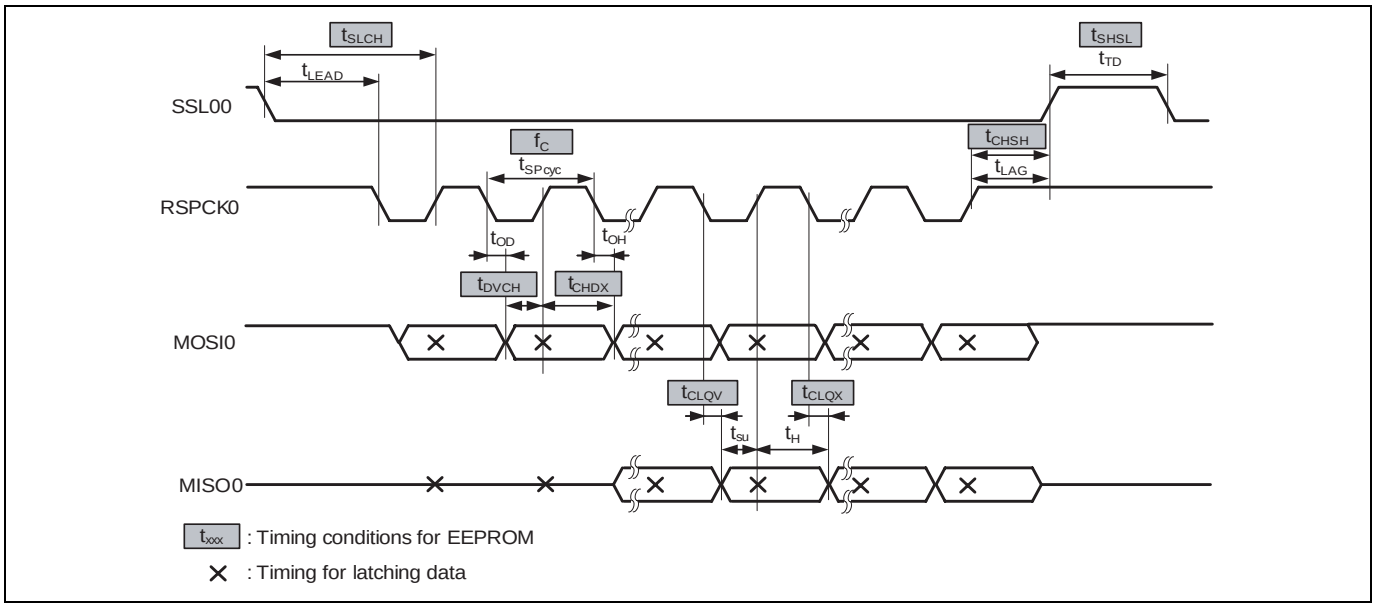

Figure 2 Data Transfer Timing Example (CPOL =1, CPHA = 1)

Table 3 Timing Conditions for EEPROM when Transferring Data

Symbol	Item	Description	Related Registers
t_{SLCH}	Chip Select Low Setup Time	Time required for the slave device to latch data from asserting SSL to the RSPCK rising: The following formula must be fulfilled: t_{LEAD}(=RSPCK delay) + 1/2 x t_{SPcyc} > t_{SLCH} (min)	SPCKD register SPCMD register
t_{SHSL}	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_C	Serial Clock Frequency	The maximum operating frequency supported by the slave device. The following formula must be fulfilled: f_C (max) > 1/ t_{SPcyc}	SPBR register SPCMD register
t_{CHSH}	Chip select Low Hold Time	Time required from the last RSPCK rising to the SSL negation. The following formula must be fulfilled: t_{LAG}(=SSL negation delay) > t_{CHSH} (min)	SSLND register SPCMD register
t_{DVCH}	Data Input Setup Time	Time required for the master device from outputting data to latching data. The following formula must be fulfilled: 1/2 x t_{SPcyc} - t_{OD}(max) > t_{DVCH} (min)	
t_{CHDX}	Data Input Hold Time	Time required for the master device from latching data to stop the data output. The following formula must be fulfilled: t_{OH}(min) + 1/2 x t_{SPcyc} > t_{CHDX} (min)	

Table 4 Timing Conditions for the SH7264 MCU when Transferring Data

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

2.4 Sample Program Operation

2.4.1 RSPI Initialization Example

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

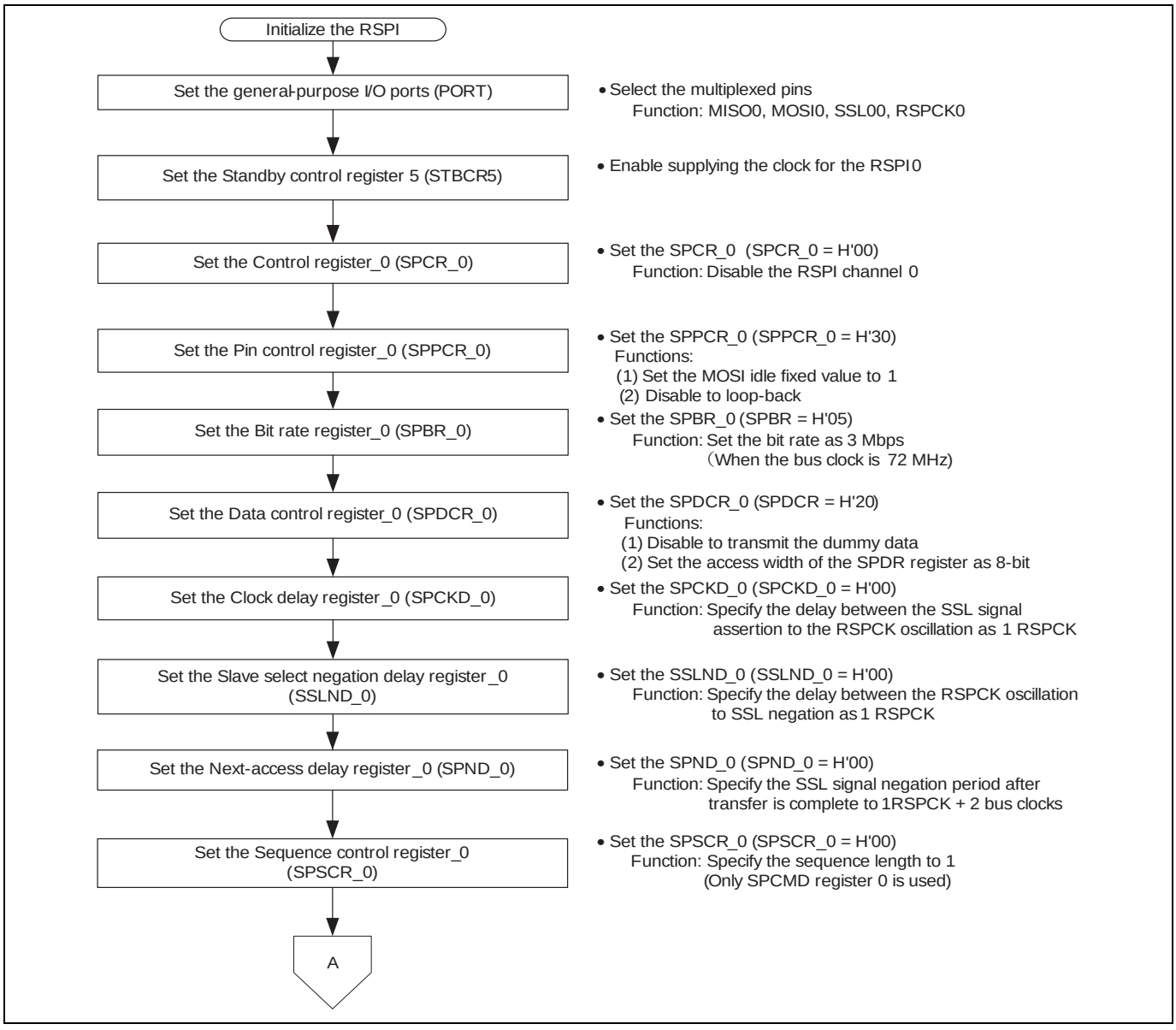

Figure 3 RSPI Initialization Flow Chart (1/2)

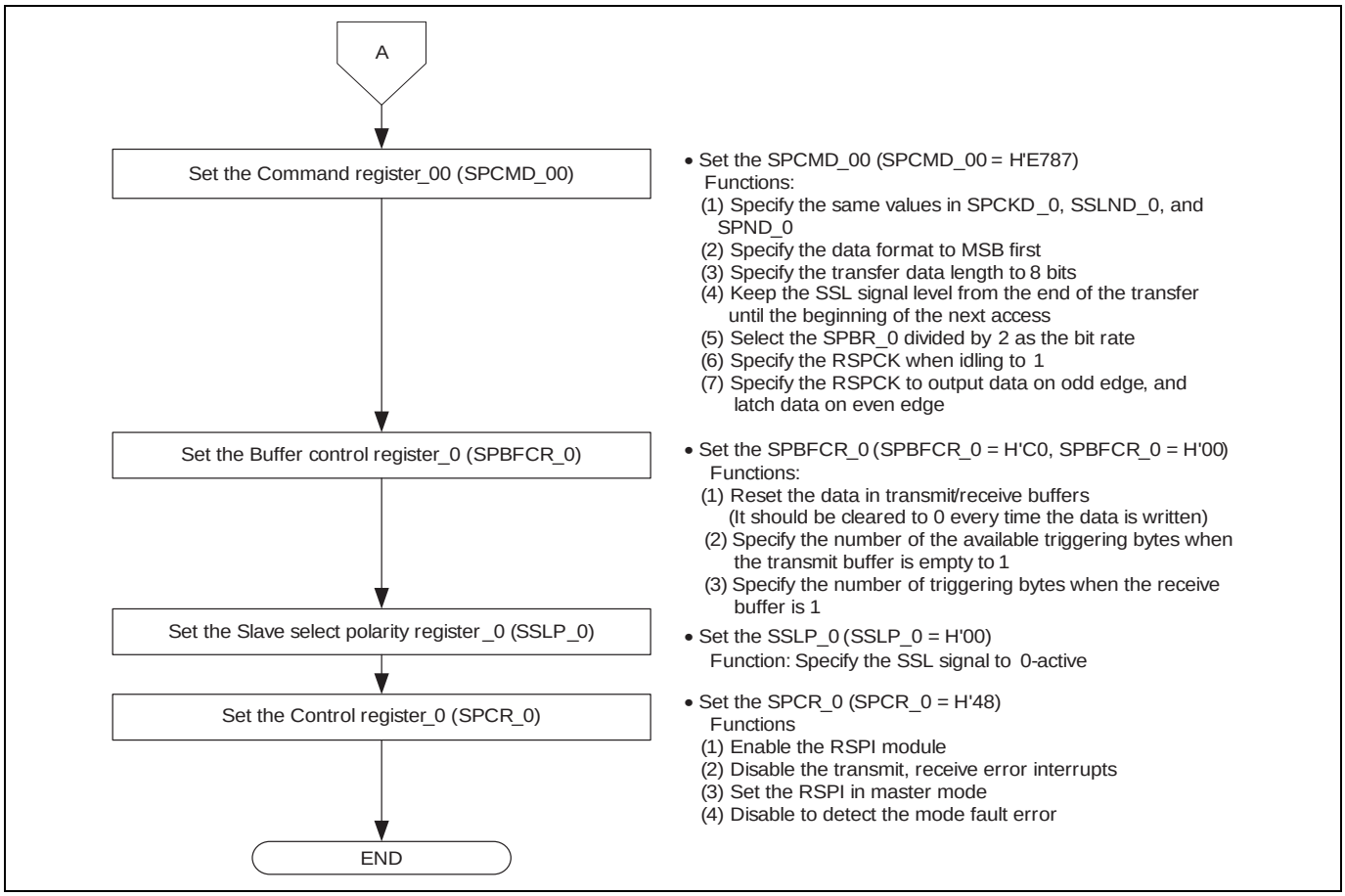

Figure 4 RSPI Initialization Flow Chart (2/2)

2.4.2 Command Transfer Example

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

This application refers to the commands of the Renesas Technology, HN58X25128I. For details on commands, refer to the datasheet provided by the EEPROM manufacturer.

A. Major Commands
The following table lists the major commands for the HN58X25128I.

Table 5 HN58X25128I Major Commands

Command Name	Opcode	Address Bytes	Data Bytes	Function
WREN	H'06	0	0	Write Enable
WRDI	H'04	0	0	Write Disable
RDSR	H'05	0	1	Read Status Register
WRSR	H'01	0	1	Write Status Register
READ	H'03	2	1+ (1)	Read from Memory Array
WRITE	H'02	2	1 to 64 (2)	Write to 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 address in the same page (When the device goes beyond the end of the page, it will wrap around back to the beginning of the same page).

B. Command Sequence Example
Figure 5 shows the sequence example of the READ command.

When issuing the READ command, the master device transfers the opcode (H'03) and two 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 be clearing the SPE bit in the SPCR register to 0 after the all data transfer is complete.

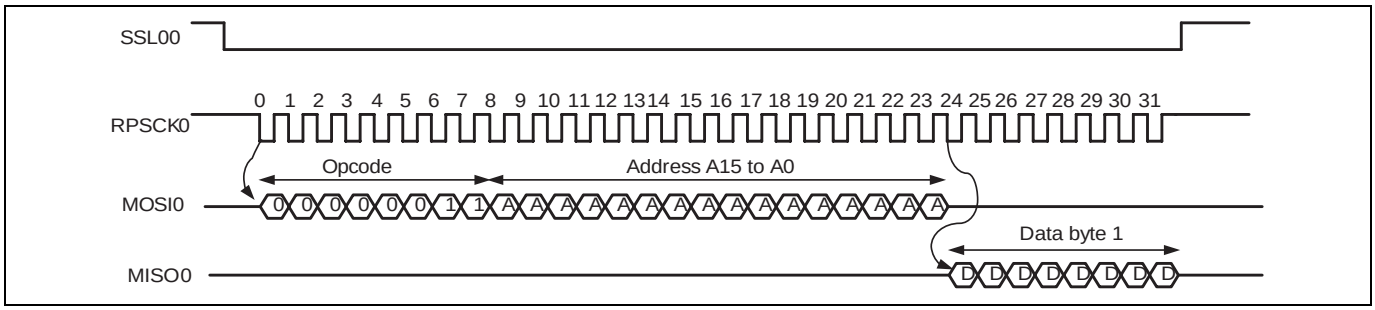

Figure 5 READ Command Sequence (Opcode: H'03)

C. Command Transfer Example in the Sample Program

The READ command that uses both master output and slave output, and the WRITE command that uses the master output are supported by the sample program. Figure 6 shows the flow chart of the READ command transfer. The READ command follows this flow chart. Figure 7 shows the flow chart of the WRITE command transfer.

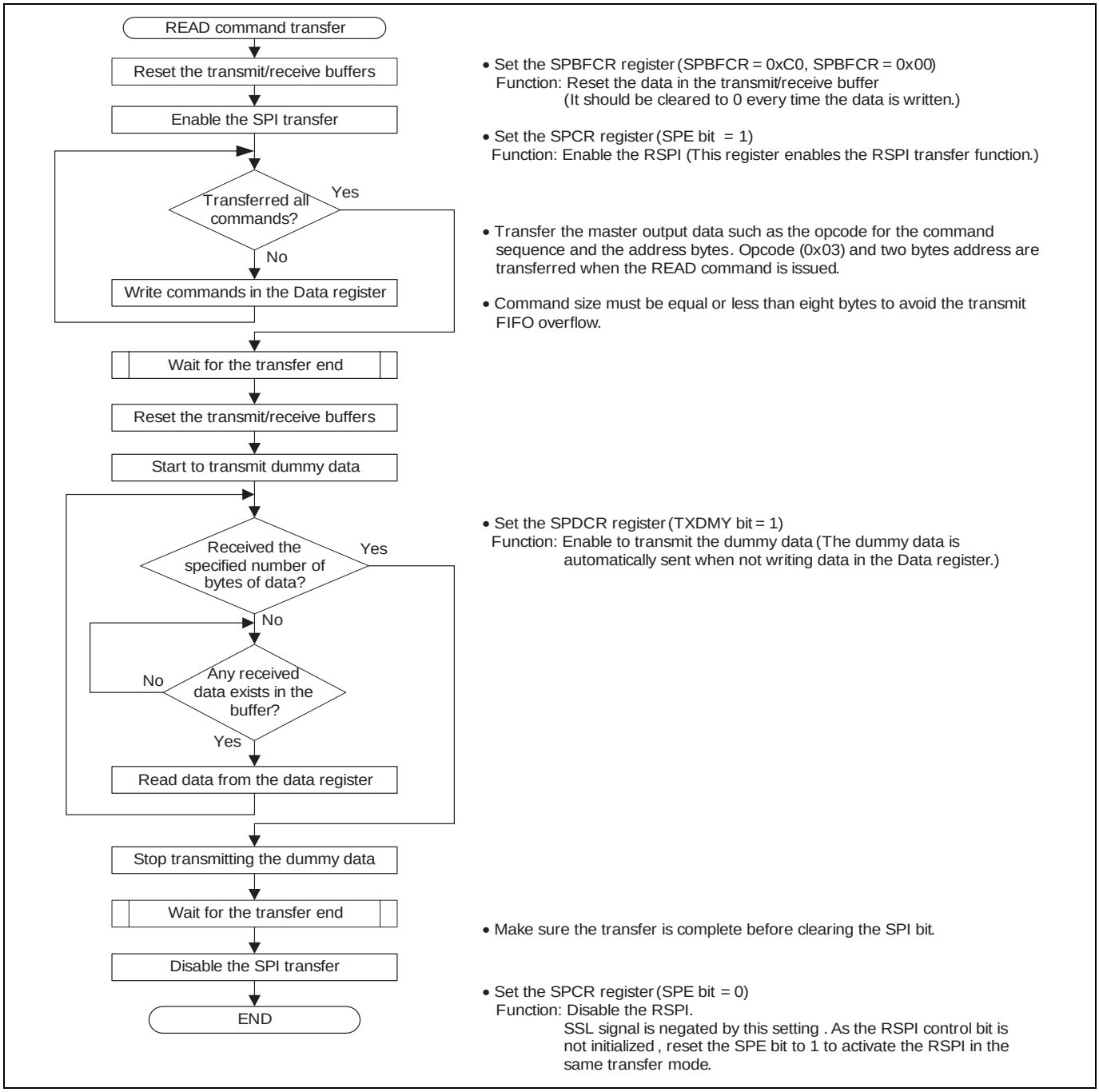

Figure 6 Flow Chart of the READ Command Transfer

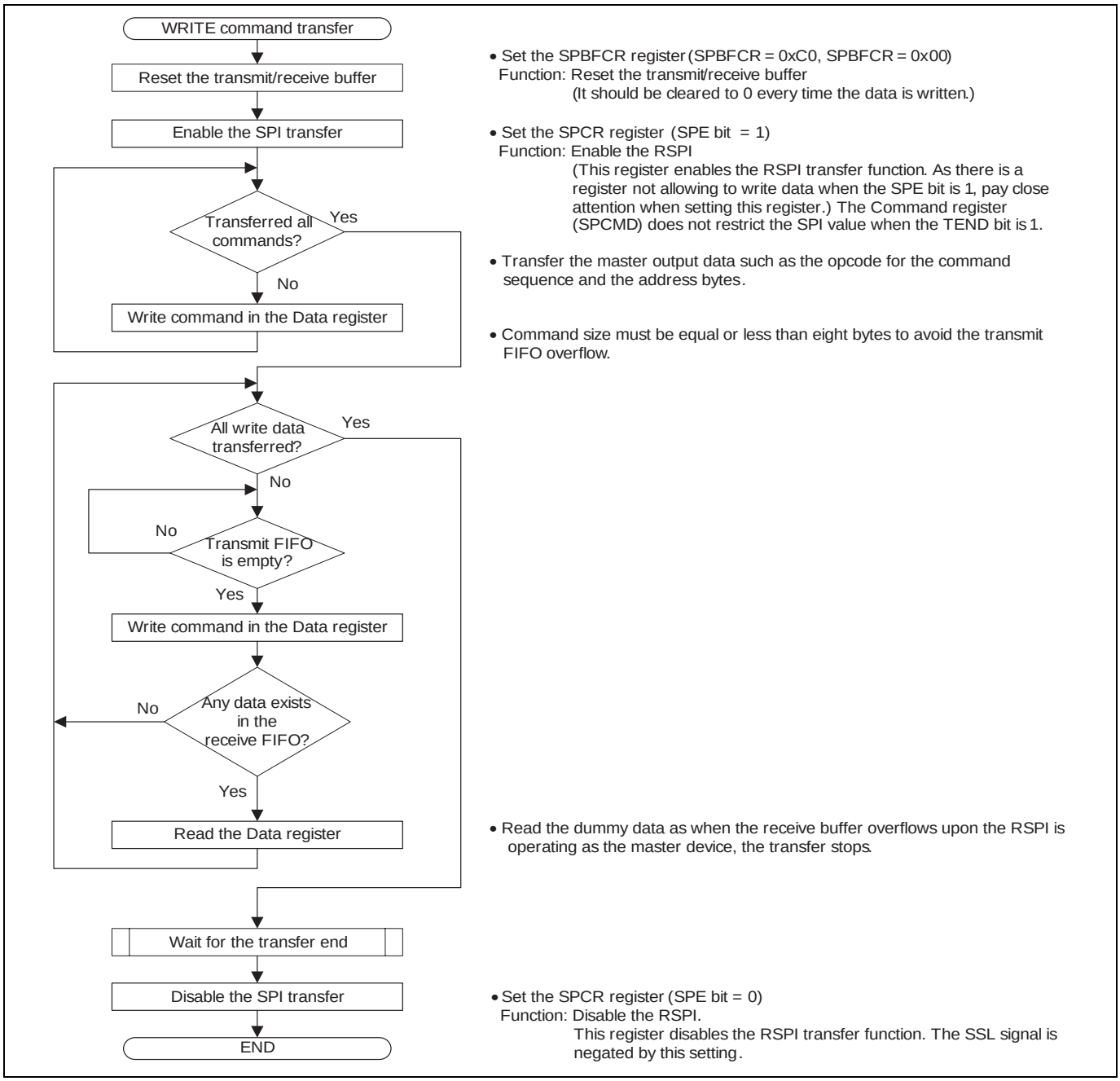

Figure 7 Flow Chart of the WRITE Command Transfer

2.4.3 Main Function

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

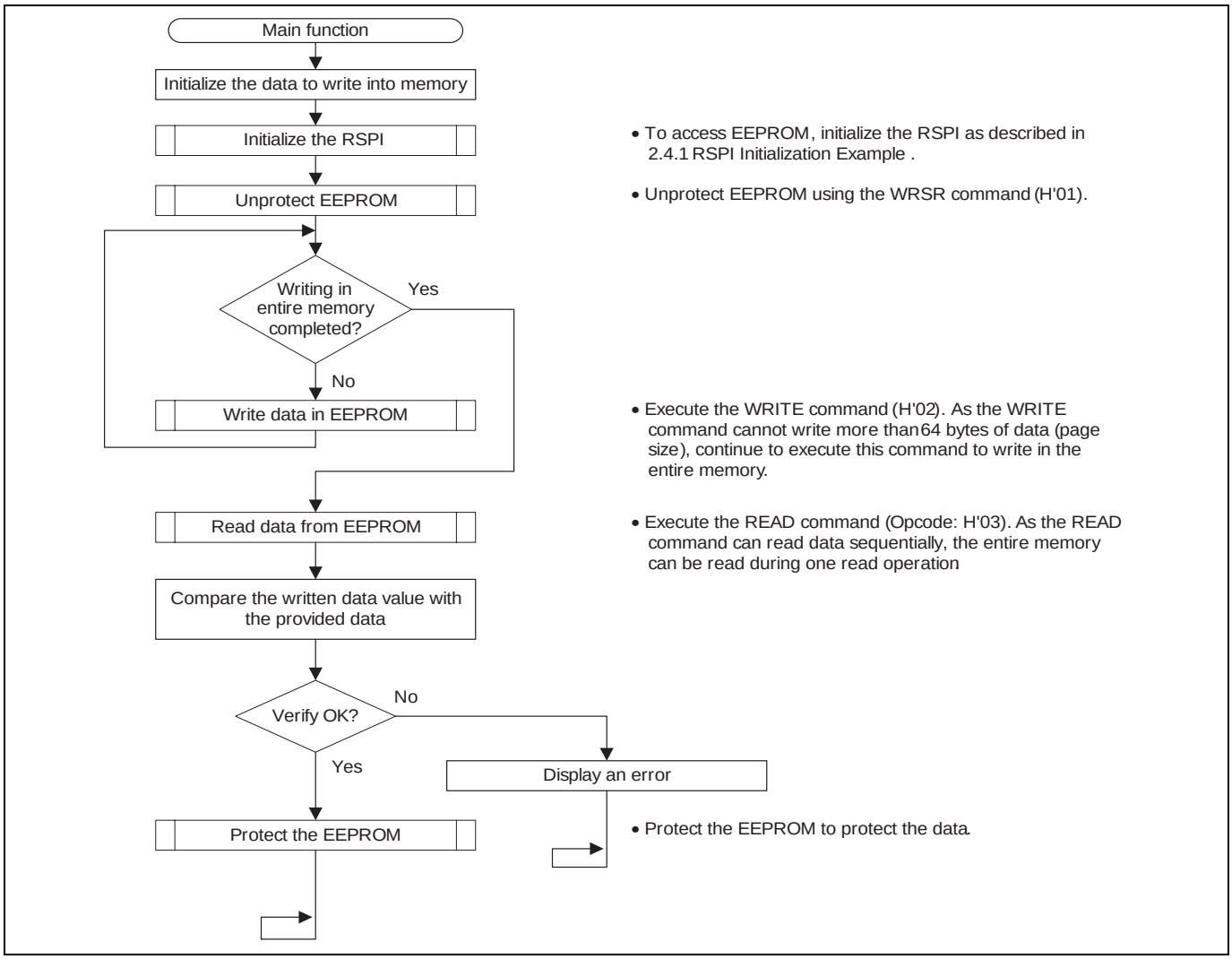

Figure 8 Main Function Flow Chart in the Sample Program

3. Sample Program Listing

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

```
1       /*""FILE COMMENT""*********** Technical reference data ************************
2       *
3       *       System Name : SH7264 Sample Program
4       *       File Name   : main.c
5       *       Abstract    : Renesas Serial Peripheral Interface Read/Write EEPROM
6       *       Version     : 1.00.00
7       *       Device      : SH7262/SH7264
8       *       Tool-Chain  : High-performance Embedded Workshop (Version 4.04.01)
9       *                   : C/C++ compiler package for the SuperH RISC engine family
10      *                   :                             (Ver.9.02 Release00).
11      *       OS          : None
12      *       H/W Platform: M3A-HS64G50 (CPU board)
13      *       Disclaimer  :
14      *
15      *       The information described here may contain technical inaccuracies or
16      *       typographical errors. Renesas Technology Corporation and Renesas Solutions
17      *       assume no responsibility for any damage, liability, or other loss rising
18      *       from these inaccuracies or errors.
19      *
20      *       Copyright (C) 2009 Renesas Technology Corp. All Rights Reserved
21      *       AND Renesas Solutions Corp. All Rights Reserved
22      *
23      *       History     : Feb.20,2009 Ver.1.00.00
24      *""FILE COMMENT END""*************************************************************/
25      #include <stdio.h>
26      #include "eeprom.h"
27
28      /* ==== Macro definition ==== */
29      #define TOP_ADDRESS    0              /* EEPROM start address */
30
31      /* ==== Function prototype declaration ==== */
32      void main(void);
33
34      /* ==== Variable definition ==== */
35      #pragma section DEBUG_32K_BYTES
36      static unsigned char data[EEP_MEM_SIZE];
37      static unsigned char rbuf[EEP_MEM_SIZE];
38      #pragma section
39
```

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

```
40    /*""FUNC COMMENT""*****************************************************************
41     * ID          :
42     * Outline     : EEPROM access main
43     *-------------------------------------------------------------------------------
44     * Include     :
45     *-------------------------------------------------------------------------------
46     * Declaration : void main(void);
47     *-------------------------------------------------------------------------------
48     * Function    : Writes or reads EEPROM.
49     *             : Initializes the RSPI channel 0, writes data in the entire memory
50     *             : array from the beginning. Then, it reads and verifies the data.
51     *-------------------------------------------------------------------------------
52     * Argument    : void
53     *-------------------------------------------------------------------------------
54     * Return Value: void
55     *""FUNC COMMENT END""*************************************************************/
56    void main(void)
57    {
58      int i;
59      static unsigned long addr;
60
61      /* ==== Initializes the data==== */
62      for(i = 0; i < EEP_MEM_SIZE; i++){
63        data[i] = i % 100;
64        rbuf[i] = 0;
65      }
66      /* ==== Initializes the RSPI ==== */
67      eep_init_eeprom();
68
69      /* ==== Unprotects EEPROM ==== */
70      eep_protect_ctrl( EEP_REQ_UNPROTECT );
71
72      /* ==== Writes data ==== */
73      for(addr = TOP_ADDRESS; addr < (TOP_ADDRESS + EEP_MEM_SIZE); addr += EEP_PAGE_SIZE ){
74        eep_byte_write( addr, data+addr, EEP_PAGE_SIZE );
75      }
76      /* ==== Reads data ==== */
77      eep_byte_read( TOP_ADDRESS, rbuf, EEP_MEM_SIZE );
78
79      /* ==== Verifies data ==== */
80      for(i = 0; i < EEP_MEM_SIZE; i++){
81        if( data[i] != rbuf[i] ){
82            puts("Error: verify error\n");
83            fflush(stdout);
84            while(1);
85        }
86      }
```

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

```
87      /* ==== Protects EEPROM ==== */
88      eep_protect_ctrl( EEP_REQ_PROTECT );
89
90      while(1){
91            /* loop */
92        }
93    }
94
95    /* End of File */
```

3.4 Sample Program Listing "eeprom.c" (1/11)

```
1    /*""FILE COMMENT""************ Technical reference data ************************
2    *
3    *       System Name : SH7264 Sample Program
4    *       File Name   : eeprom.c
5    *       Abstract    : Renesas Serial Peripheral Interface Read/Write EEPROM
6    *       Version     : 1.00.00
7    *       Device      : SH7262/SH7264
8    *       Tool-Chain  : High-performance Embedded Workshop (Version 4.04.01)
9    *                   : C/C++ compiler package for the SuperH RISC engine family
10   *                   :                              (Ver.9.02 Release00).
11   *       OS          : None
12   *       H/W Platform: M3A-HS64G50 (CPU board)
13   *       Disclaimer  :
14   *
15   *       The information described here may contain technical inaccuracies or
16   *       typographical errors. Renesas Technology Corporation and Renesas Solutions
17   *       assume no responsibility for any damage, liability, or other loss rising
18   *       from these inaccuracies or errors.
19   *
20   *       Copyright (C) 2009 Renesas Technology Corp. All Rights Reserved
21   *       AND Renesas Solutions Corp. All Rights Reserved
22   *
23   *       History     : Feb.20,2009 Ver.1.00.00
24   *""FILE COMMENT END""*************************************************************/
25   #include <stdio.h>
26   #include <machine.h>
27   #include "iodefine.h"
28   #include "eeprom.h"
29
30   /* ==== Macro definition ==== */
31   #define EEPROMCMD_WRITE_ENABLE 0x06
32   #define EEPROMCMD_WRITE_DISABLE   0x04
33   #define EEPROMCMD_READ_STATUS  0x05
34   #define EEPROMCMD_WRITE_STATUS 0x01
35   #define EEPROMCMD_READ_ARRAY   0x03
36   #define EEPROMCMD_WRITE_ARRAY  0x02
37   #define UNPROTECT_WR_STATUS       0x00
38   #define PROTECT_WR_STATUS     0x0C
39   #define EEP_BUSY_BIT          0x01
40
```

3.5 Sample Program Listing "eeprom.c" (2/11)

```
41    /* ==== Function prototype declaration ==== */
42    /*** Local function ***/
43    static void write_enable(void);
44    static void write_disable(void);
45    static void busy_wait(void);
46    static unsigned char read_status(void);
47    static void write_status(unsigned char status);
48    static void io_init_rspi(void);
49    static void io_cmd_exe(unsigned char *ope, int ope_sz, unsigned char *data, int data_sz);
50    static void io_cmd_exe_rdmode(unsigned char *ope, int ope_sz, unsigned char *rd, int rd_sz);
51    static void io_wait_tx_end(void);
52
53    /* ==== Variable definition ==== */
54
55    /*""FUNC COMMENT""*****************************************************************
56     * ID          :
57     * Outline     : EEPROM initialization
58     *-----------------------------------------------------------------------------
59     * Include     :
60     *-----------------------------------------------------------------------------
61     * Declaration : void eep_init_eeprom(void);
62     *-----------------------------------------------------------------------------
63     * Function    : Initializes EEPROM for being accessed. Initializes the channel 0
64     *             : of the Renesas Serial Peripheral Interface (RSPI).
65     *-----------------------------------------------------------------------------
66     * Argument    : void
67     *-----------------------------------------------------------------------------
68     * Return Value: void
69     *""FUNC COMMENT END""*************************************************************/
70    void eep_init_eeprom(void)
71    {
72      /* ==== Initializes the RSPI0 ==== */
73      io_init_rspi();
74    }
75    /*""FUNC COMMENT""*****************************************************************
76     * ID          :
77     * Outline     : Protect/unprotect operation
78     *-----------------------------------------------------------------------------
79     * Include     :
80     *-----------------------------------------------------------------------------
81     * Declaration : void eep_protect_ctrl(enum eep_req req);
82     *-----------------------------------------------------------------------------
83     * Function    : Protects or unprotects EEPROM.
84     *             : Use the argument req to specify. Default setting and unprotecting
85     *             : method depends on the specifications of the EEPROM.
86     *-----------------------------------------------------------------------------
87     * Argument    : enum eep_req req ; I : EEP_REQ_UNPROTECT -> Write-enable entire memory array
88     *             :                        EEP_REQ_PROTECT   -> Write-protect entire memory array
89     *-----------------------------------------------------------------------------
90     * Return Value: void
91     *""FUNC COMMENT END""*************************************************************/
```

3.6 Sample Program Listing "eeprom.c" (3/11)

```
92    void eep_protect_ctrl(enum eep_req req)
93    {
94      if( req == EEP_REQ_UNPROTECT ){
95        write_status( UNPROTECT_WR_STATUS);       /* Unprotect entire area */
96      }
97      else{
98        write_status( PROTECT_WR_STATUS );    /* Protect entire area */
99      }
100   }
101
102   /*""FUNC COMMENT""*****************************************************************
103    * ID          :
104    * Outline     : Write data
105    *-------------------------------------------------------------------------------
106    * Include     :
107    *-------------------------------------------------------------------------------
108    * Declaration : void eep_byte_write(unsigned long addr, unsigned char *buf, int size);
109    *-------------------------------------------------------------------------------
110    * Function    : Writes the specified data into EEPROM.
111    *             : Before writing, issue the Write Enable command. After writing,
112    *             : make sure to check the status of EEPROM is not busy.
113    *             : The maximum write data size depends on the type of the device.
114    *-------------------------------------------------------------------------------
115    * Argument    : unsigned long addr ; I : Address in EEPROM to write
116    *             : unsigned char *buf ; I : Buffer address to store the write data
117    *             : int size           ; I : Number of bytes to write
118    *-------------------------------------------------------------------------------
119    * Return Value: void
120    *""FUNC COMMENT END""*************************************************************/
121   void eep_byte_write(unsigned long addr, unsigned char *buf, int size)
122   {
123     unsigned char cmd[3];
124
125     cmd[0] = EEPROMCMD_WRITE_ARRAY;
126     cmd[1] = (unsigned char)((addr >>  8) & 0xff);
127     cmd[2] = (unsigned char)( addr        & 0xff);
128     write_enable();
129     io_cmd_exe(cmd, 3, buf, size);
130     busy_wait();
131   }
132
```

3.7 Sample Program Listing "eeprom.c" (4/11)

```
133   /*""FUNC COMMENT""*****************************************************************
134    * ID          :
135    * Outline     : Read data
136    *-------------------------------------------------------------------------------
137    * Include     :
138    *-------------------------------------------------------------------------------
139    * Declaration : void eep_byte_read(unsigned long addr, unsigned char *buf, int size);
140    *-------------------------------------------------------------------------------
141    * Function    : Reads the specified number of bytes of data from EEPROM.
142    *-------------------------------------------------------------------------------
143    * Argument    : unsigned long addr ; I : Address in EEPROM to read
144    *             : unsigned char *buf ; I : Buffer address to store the read data
145    *             : int size           ; I : Number of bytes to read
146    *-------------------------------------------------------------------------------
147    * Return Value: void
148    *""FUNC COMMENT END""*************************************************************/
149   void eep_byte_read(unsigned long addr, unsigned char *buf, int size)
150   {
151     unsigned char cmd[3];
152
153     cmd[0] = EEPROMCMD_READ_ARRAY;
154     cmd[1] = (unsigned char)((addr >>  8) & 0xff);
155     cmd[2] = (unsigned char)( addr        & 0xff);
156     io_cmd_exe_rdmode(cmd, 3, buf, size);
157   }
158   /*""FUNC COMMENT""*****************************************************************
159    * ID          :
160    * Outline     : Write-enable
161    *-------------------------------------------------------------------------------
162    * Include     :
163    *-------------------------------------------------------------------------------
164    * Declaration : static void write_enable(void);
165    *-------------------------------------------------------------------------------
166    * Function    : Issues the Write Enable command to enable writing into EEPROM.
167    *-------------------------------------------------------------------------------
168    * Argument    : void
169    *-------------------------------------------------------------------------------
170    * Return Value: void
171    *""FUNC COMMENT END""*************************************************************/
172   static void write_enable(void)
173   {
174     unsigned char cmd[1];
175     cmd[0] = EEPROMCMD_WRITE_ENABLE;
176     io_cmd_exe(cmd, 1, NULL, 0);
177   }
178
```

3.8 Sample Program Listing "eeprom.c" (5/11)

```
179   /*""FUNC COMMENT""*****************************************************************
180    * ID          :
181    * Outline     : Write disable
182    *-------------------------------------------------------------------------------
183    * Include     :
184    *-------------------------------------------------------------------------------
185    * Declaration : static void write_disable(void);
186    *-------------------------------------------------------------------------------
187    * Function    : Issues the Write Disable command to disable writing into EEPROM.
188    *-------------------------------------------------------------------------------
189    * Argument    : void
190    *-------------------------------------------------------------------------------
191    * Return Value: void
192    *""FUNC COMMENT END""*************************************************************/
193   static void write_disable(void)
194   {
195     unsigned char cmd[1];
196     cmd[0] = EEPROMCMD_WRITE_DISABLE;
197     io_cmd_exe(cmd, 1, NULL, 0);
198   }
199
200   /*""FUNC COMMENT""*****************************************************************
201    * ID          :
202    * Outline     : Busy waiting
203    *-------------------------------------------------------------------------------
204    * Include     :
205    *-------------------------------------------------------------------------------
206    * Declaration : static void busy_wait(void);
207    *-------------------------------------------------------------------------------
208    * Function    : Loops internally when the EEPROM status is busy.
209    *-------------------------------------------------------------------------------
210    * Argument    : void
211    *-------------------------------------------------------------------------------
212    * Return Value: void
213    *""FUNC COMMENT END""*************************************************************/
214   static void busy_wait(void)
215   {
216     while ((read_status() & EEP_BUSY_BIT) != 0) {  /* RDY/BSY */
217       /* serial flash is busy */
218     }
219   }
220
```

3.9 Sample Program Listing "eeprom.c" (6/11)

```
221   /*""FUNC COMMENT""*****************************************************************
222    * ID          :
223    * Outline     : Read status
224    *-------------------------------------------------------------------------------
225    * Include     :
226    *-------------------------------------------------------------------------------
227    * Declaration : static unsigned char read_status(void);
228    *-------------------------------------------------------------------------------
229    * Function    : Reads the status of EEPROM.
230    *-------------------------------------------------------------------------------
231    * Argument    : void
232    *-------------------------------------------------------------------------------
233    * Return Value: Status register value
234    *""FUNC COMMENT END""*************************************************************/
235   static unsigned char read_status(void)
236   {
237     unsigned char buf;
238     unsigned char cmd[1];
239
240     cmd[0] = EEPROMCMD_READ_STATUS;
241     io_cmd_exe_rdmode(cmd, 1, &buf, 1);
242     return buf;
243   }
244
245   /*""FUNC COMMENT""*****************************************************************
246    * ID          :
247    * Outline     : Write status
248    *-------------------------------------------------------------------------------
249    * Include     :
250    *-------------------------------------------------------------------------------
251    * Declaration : static void write_status(unsigned char status);
252    *-------------------------------------------------------------------------------
253    * Function    : Writes the status of EEPROM.
254    *-------------------------------------------------------------------------------
255    * Argument    : unsigned char status ; I : status register value
256    *-------------------------------------------------------------------------------
257    * Return Value: void
258    *""FUNC COMMENT END""*************************************************************/
259   static void write_status(unsigned char status)
260   {
261     unsigned char cmd[2];
262
263     cmd[0] = EEPROMCMD_WRITE_STATUS;
264     cmd[1] = status;
265
266     write_enable();
267     io_cmd_exe(cmd, 2, NULL, 0);
268     busy_wait();
269   }
```

3.10 Sample Program Listing "eeprom.c" (7/11)

```
270    /*""FUNC COMMENT""*****************************************************************
271     * ID          :
272     * Outline     : RSPI initialization
273     *-------------------------------------------------------------------------------
274     * Include     :
275     *-------------------------------------------------------------------------------
276     * Declaration : static void io_init_rspi(void);
277     *-------------------------------------------------------------------------------
278     * Function    : Initializes channel 0 of the RSPI.
279     *             : Sets the RSPI in master mode to set parameters required to transfer
280     *             : according to the specifications of EEPROM.
281     *-------------------------------------------------------------------------------
282     * Argument    : void
283     *-------------------------------------------------------------------------------
284     * Return Value: void
285     *""FUNC COMMENT END""*************************************************************/
286    static void io_init_rspi(void)
287    {
288      /* ==== PORT ==== */
289      PORT.PFCR3.BIT.PF12MD = 3;  /* PF12:MISO0 */
290      PORT.PFCR2.BIT.PF11MD = 3;  /* PF11:MOSI0 */
291      PORT.PFCR2.BIT.PF10MD = 3;  /* PF10:SSL00 */
292      PORT.PFCR2.BIT.PF9MD  = 3;  /* PF9:RSPCK0 */
293
294      /* ==== CPG ==== */
295      CPG.STBCR5.BIT.MSTP51 = 0;  /* RSPI0 active */
296
```

3.11 Sample Program Listing "eeprom.c" (8/11)

```
297      /* ==== RSPI ==== */
298      RSPI0.SPCR.BYTE   =  0x00;   /* Disables channel 0 of the RSPI */
299      RSPI0.SPPCR.BYTE  =  0x30;   /* MOSI idle fixed value = 1 */
300      RSPI0.SPBR.BYTE   = 0x05;    /* Specifies the base bit rate as 3 MHz
301                                                    (Bus clock = 72 MHz) */
302      RSPI0.SPDCR.BYTE  =  0x20;   /* Disables to send the dummy data */
303                               /* Access width of the SPDR register: 8-bit */
304      RSPI0.SPCKD.BYTE  =  0x00;   /* RSPCK delay: 1 RSPCK */
305      RSPI0.SSLND.BYTE  =  0x00;   /* SSL negate delay: 1 RSPCK */
306      RSPI0.SPND.BYTE   =  0x00;   /* Next access delay: 1 RSPCK + 2 Bus clocks */
307      RSPI0.SPSCR.BYTE  =  0x00;   /* Sequence length: 1 (SPCMD0 is only used) */
308      RSPI0.SPCMD0.WORD = 0xE787; /* MSB first */
309                               /* Data length: 8-bit */
310                               /* Keeps the SSL signal level after transfer
311                                   is completed */
312                               /* Bit rate: Base bit rate is divided by 2 */
313                               /* RSPCK when it is idling is 1 */
314                               /* Latches data on odd edge, outputs data on
315                                   even edge */
316      RSPI0.SPBFCR.BYTE =  0xC0;   /* Enables to reset data in the transmit/receive buffer */
317      RSPI0.SPBFCR.BYTE =  0x00;   /* Disables to reset data in the transmit/receive buffer */
318                               /* Number of triggers in transmit buffer:
319                                   more than one byte available */
320                               /* Number of triggers in receive buffer:
321                                   more than one byte received */
322      RSPI0.SSLP.BYTE   =  0x00;   /* SSLP = b'0 SSL signal 0-active */
323      RSPI0.SPCR.BYTE   =  0x48;   /* Master mode */
324                               /* Disables interrupts */
325                               /* Enables channel 0 of the RSPI */
326    }
327
```

3.12 Sample Program Listing "eeprom.c" (9/11)

```
328    /*""FUNC COMMENT""*****************************************************************
329     * ID          :
330     * Outline     : Execute command (No read data).
331     *-------------------------------------------------------------------------------
332     * Include     :
333     *-------------------------------------------------------------------------------
334     * Declaration : static void io_cmd_exe(unsigned char *ope, int ope_sz,
335     *             :                        unsigned char *data,int data_sz)
336     *-------------------------------------------------------------------------------
337     * Function    : Executes the specified command.
338     *             : Transmits the argument ope, and then transmits the argument data.
339     *             : Discards the received data.
340     *             : Set one of the values between 0 and 8 in the ope_sz.
341     *             : Set one of the values between 0 and 256 in the data_sz.
342     *-------------------------------------------------------------------------------
343     * Argument    : unsigned char *ope ; I : Start address of the opcode block and
344     *                                      : address block to transmit
345     *             : int ope_sz         ; I : Number of bytes in the opcode block and
346     *                                      : address block
347     *             : unsigned char *data; I : Start address of the data block to transmit
348     *             : int data_sz        ; I : Number of bytes in the data block
349     *-------------------------------------------------------------------------------
350     * Return Value: void
351     *""FUNC COMMENT END""*************************************************************/
352    static void io_cmd_exe(unsigned char *ope, int ope_sz, unsigned char *data, int data_sz)
353    {
354      unsigned char tmp;
355
356      /* ==== Reset buffer ==== */
357      RSPI0.SPBFCR.BYTE  = 0xC0u;
358      RSPI0.SPBFCR.BYTE  = 0x00u;
359
360      /* ---- Enables the SPI transfer ---- */
361      RSPI0.SPCR.BIT.SPE = 1;
362
363      /* ==== MOSI (command, address, write data) ==== */
364      while(ope_sz--){
365        RSPI0.SPDR.BYTE = *ope++;  /* Command size must be equal or less than 8 bytes */
366      }
367      while(data_sz--){
368        while( RSPI0.SPSR.BIT.SPTEF == 0 ){
369            /* wait */
370        }
371        RSPI0.SPDR.BYTE = *data++;
372        if( RSPI0.SPSR.BIT.SPRF == 1 ){
373            tmp = RSPI0.SPDR.BYTE; /* Dummy read to avoid an overflow of data */
374        }
375      }
376      io_wait_tx_end();              /* Waits for transfer end */
377
```

3.13 Sample Program Listing "eeprom.c" (10/11)

```
378       /* ---- SPI transfer end (SSL negation) ---- */
379       RSPI0.SPCR.BIT.SPE = 0;
380     }
381     /*""FUNC COMMENT""*****************************************************************
382      * ID          :
383      * Outline     : Execute command (With read data).
384      *-------------------------------------------------------------------------------
385      * Include     :
386      *-------------------------------------------------------------------------------
387      * Declaration : static void io_cmd_exe_rdmode(unsigned char *ope, int ope_sz,
388      *             :                               unsigned char *rd,  int rd_sz)
389      *-------------------------------------------------------------------------------
390      * Function    : Executes the specified command.
391      *             : Transmits the argument ope, and then receives data in the argument rd.
392      *             : Set one of the values between 0 and 8 in the ope_sz.
393      *             : More than 0 can be set in the rd_sz.
394      *-------------------------------------------------------------------------------
395      * Argument    : unsigned char *ope ; I : Start address of the opcode block and
396      *                                      : address block to transmit
397      *             : int ope_sz         ; I : Number of bytes in the opcode block and
398      *                                      : address block
399      *             : unsigned char *rd  ; I : Buffer address to store the received data
400      *             : int rd_sz          ; I : Number of bytes in the data block
401      *-------------------------------------------------------------------------------
402      * Return Value: void
403      *""FUNC COMMENT END""*************************************************************/
404     static void io_cmd_exe_rdmode(unsigned char *ope, int ope_sz, unsigned char *rd, int rd_sz)
405     {
406       /* ==== Resets buffer ==== */
407       RSPI0.SPBFCR.BYTE  = 0xC0u;
408       RSPI0.SPBFCR.BYTE  = 0x00u;
409
410       /* ---- Enables the SPI transfer ---- */
411       RSPI0.SPCR.BIT.SPE = 1;
412
413       /* ---- MOSI(command, address, dummy) ---- */
414       while(ope_sz--){
415         RSPI0.SPDR.BYTE = *ope++;  /* Command size must be equal or less than 8 bytes */
416       }
417       io_wait_tx_end();              /* Waits for transfer end */
418
```

3.14 Sample Program Listing "eeprom.c" (11/11)

```
419      /* ---- MISO(Read data) ---- */
420      RSPI0.SPBFCR.BYTE  = 0xC0u;     /* Resets buffer */
421      RSPI0.SPBFCR.BYTE  = 0x00u;
422
423      RSPI0.SPDCR.BIT.TXDMY = 1;      /* Enables to transmit the dummy data */
424      while(rd_sz--){
425        while( RSPI0.SPSR.BIT.SPRF == 0){
426            /* wait */
427        }
428        *rd++ = RSPI0.SPDR.BYTE;
429      }
430      RSPI0.SPDCR.BIT.TXDMY = 0;      /* Disables to transmit the dummy data */
431      io_wait_tx_end();               /* Waits for transfer end */
432
433      /* ---- SPI transfer end (SSL negation) ---- */
434      RSPI0.SPCR.BIT.SPE = 0;
435    }
436
437    /*""FUNC COMMENT""*****************************************************************
438     * ID          :
439     * Outline     : Transmit end waiting
440     *-------------------------------------------------------------------------------
441     * Include     :
442     *-------------------------------------------------------------------------------
443     * Declaration : static void io_wait_tx_end(void);
444     *-------------------------------------------------------------------------------
445     * Function    : Loops internally until the transmission is completed.
446     *-------------------------------------------------------------------------------
447     * Argument    : void
448     *-------------------------------------------------------------------------------
449     * Return Value: void
450     *""FUNC COMMENT END""*************************************************************/
451    static void io_wait_tx_end(void)
452    {
453      while(RSPI0.SPSR.BIT.TEND == 0){
454        /* wait */
455      }
456    }
457
458    /* End of File */
```

3.15 Sample Program Listing "eeprom.h" (1/1)

```
1       /*""FILE COMMENT""************ Technical reference data ************************
2       *
3       *       System Name : SH7264 Sample Program
4       *       File Name   : eeprom.h
5       *       Abstract    : Renesas Serial Peripheral Interface Read/Write EEPROM
6       *       Version     : 1.00.00
7       *       Device      : SH7262/SH7264
8       *       Tool-Chain  : High-performance Embedded Workshop (Version 4.04.01)
9       *                   : C/C++ compiler package for the SuperH RISC engine family
10      *                   :                               (Ver.9.02 Release00).
11      *       OS          : None
12      *       H/W Platform: M3A-HS64G50 (CPU board)
13      *       Disclaimer  :
14      *
15      *       The information described here may contain technical inaccuracies or
16      *       typographical errors. Renesas Technology Corporation and Renesas Solutions
17      *       assume no responsibility for any damage, liability, or other loss rising
18      *       from these inaccuracies or errors.
19      *
20      *       Copyright (C) 2009 Renesas Technology Corp. All Rights Reserved
21      *       AND Renesas Solutions Corp. All Rights Reserved
22      *
23      *       History     : Feb.20,2009 Ver.1.00.00
24      *""FILE COMMENT END""*************************************************************/
25
26      #ifndef _EEPROM_H_
27      #define _EEPROM_H_
28
29      /* ==== Macro definition ==== */
30      #define EEP_PAGE_SIZE      64          /* Page size of EEPROM */
31      #define EEP_MEM_SIZE       0x4000      /* EEPROM size = 16 KB */
32      enum eep_req{
33        EEP_REQ_PROTECT = 0,                 /* Requests to protect */
34        EEP_REQ_UNPROTECT                    /* Requests to unprotect */
35      };
36      /* ==== Function prototype declaration ==== */
37      void eep_init_serial_flash(void);
38      void eep_protect_ctrl(enum eep_req req);
39      void eep_byte_write(unsigned long addr, unsigned char *buf, int size);
40      void eep_byte_read(unsigned long addr, unsigned char *buf, int size);
41
42      /* ==== Variable definition ==== */
43
44      #endif /* _EEPROM_H_ */
45      /* End of File */
```

4. References

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

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

Website and Support

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

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

Revision History

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

Notes regarding these materials

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