SH7239 Group

Reading/Writing EEPROM

Using the Renesas Serial Peripheral Interface

R01AN0058EJ0100
Rev. 1.00
Nov. 24, 2010

Summary

This application note describes examples of reading/writing EEPROM using the SH7239 Microcomputers (MCUs) Renesas Serial Peripheral Interface (RSPI).

Target Device

SH7239 MCU

Contents

1. Introduction 2
2. Applications 3
3. Sample Program Listing 14
4. References 31

1. Introduction

1.1 Specifications

- Use an EEPROM of 64 KB (512 Kbit) to connect with the SH7239 MCU

1.2 Modules Used

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

1.3 Applicable Conditions

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

2. Applications

Connect the SH7239 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

SH7239 RSPI allows full-duplex, synchronous, serial communications with peripheral devices in SPI operation using MOSI (Master Out Slave In), MISO (Master In Slave Out), SSL (Slave Select), and RSPCK (SPI Clock) pins.

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

- Master/slave modes
- Serial transfer clock with programmable polarity and phase (change SPI modes)
- Transfer bit length selectable (8- to 16-bit, 20-, 24-, and 32-bit)

2.2 EEPROM Pin Connection

The following table lists the specifications of the SPI-compliant EEPROM (R1EX25512ATA00A, Renesas Electronics) used in this application.

Table 1 EEPROM Specifications

Item	Description
Power Supply Voltage	Single supply 1.8 V to 5.5 V
SPI Modes	Supports SPI modes 0 and 3
Clock Frequency	5 MHz (2.5 to 5.5 V), 3 MHz (1.8 to 5.5 V)
Capacity	64 KB (512 Kbit)
Page Write	128 bytes/page
Write Cycle Time	5 ms (max.)
Endurance	1,000,000 Erase/Write cycles

The figure below shows an example of EEPROM. Set the SH7239 pin functions as shown in Table 2.

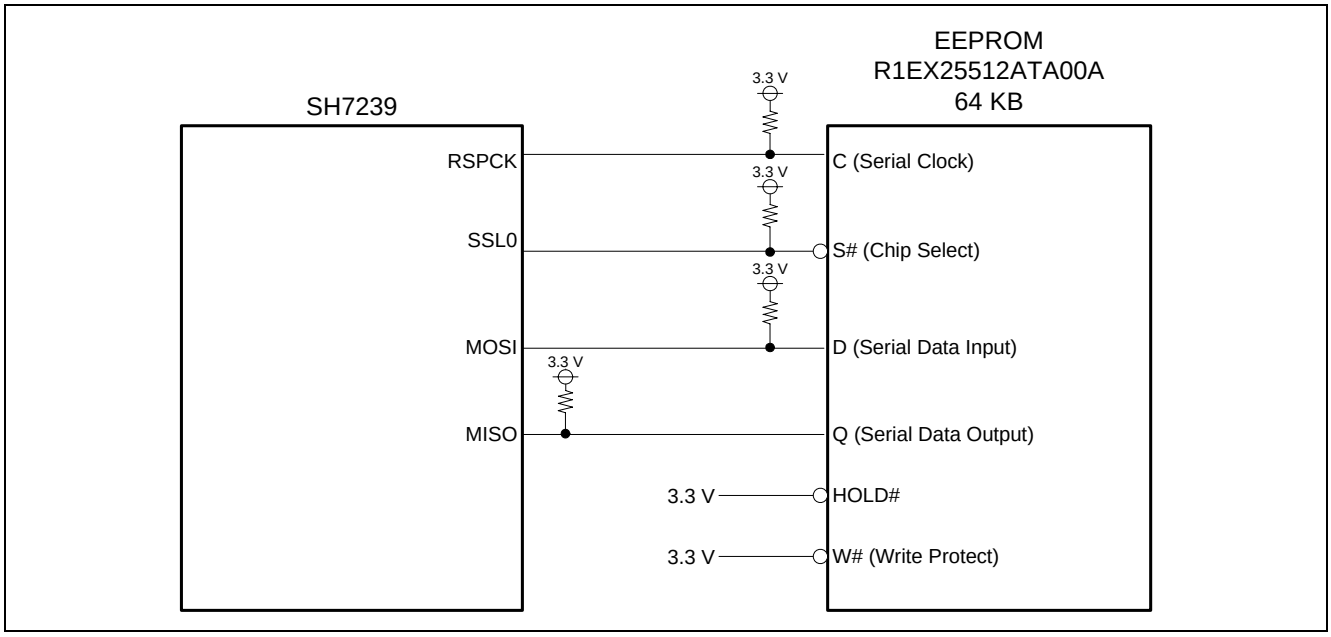

Figure 1 EEPROM Circuit

Note: Pull up or pull down the control signal pins using the external resistors
To pull up or pull down the control signal pins, determine the signal line level not to cause the external device malfunction when the MCU pin status is high-impedance. SSL0 pin is pulled up by an external resistor to high level. Pull up or down the RSPCK and MOSI pins. As the MISO pin is configured as input, pull-up or pull-down is recommended to avoid floating to the midpoint voltage.

Table 2 Multiplexed Output

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

Note: SH7239 Multiplexed pins
RSPCK, MOSI, MISO, and SSL0 pins are multiplexed, and set to general-purpose I/O ports by default. Before accessing 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 SH7239 and EEPROM. Initialize the RSPI and the clock frequency according to the EEPROM, which is used as a 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 RSPI command register (SPCMD). By this setting, RSPCK is specified to 1 when it is idling, and the timing to vary the data in the RSPI can be set to the odd edge (falling edge). Initialize the RSPI to satisfy the timing conditions listed in Table 3 and Table 4.

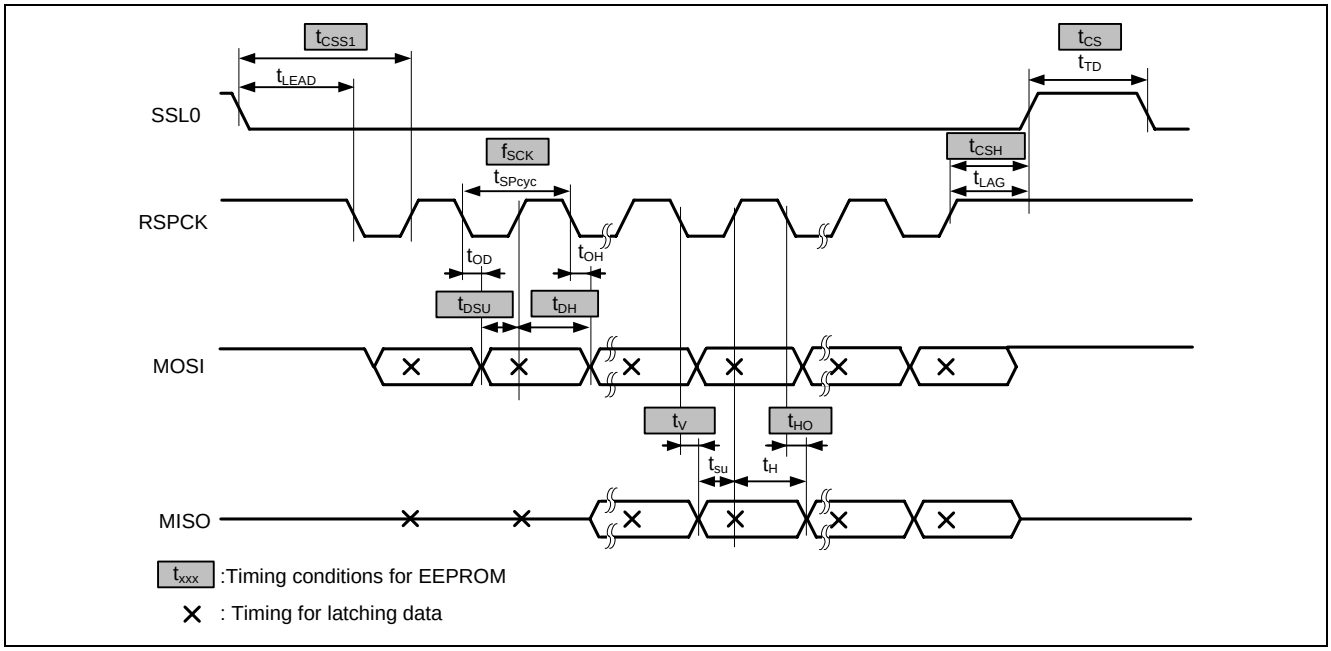

Figure 2 Data Transfer Timing Example (CPOL = 1, CPHA = 1)

Table 3 Timing Conditions for EEPROM when Transferring Data

Symbol	Item	Description	Related registers
t_{CSS1}	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 × t_{SPcyc} ≥ t_{CSS1} (min.)	SPCKD register SPCMD register SPBR register
t_{CS}	Chip Select High Time	Time required for SSL negation. The following formula must be fulfilled: t_{TD} (= next-access delay) ≥ t_{CS} (min.)	SPND register SPCMD register
f_{SCK}	Serial Clock Frequency	The maximum operating frequency supported by the slave device. The following formula must be fulfilled: f_{SCK} (max.) ≥ 1/t_{SPcyc}	SPBR register SPCMD register
t_{CSH}	Chip select Low Hold Time	Hold time required from the last RSPCK rising to the SSL negation. The following formula must be fulfilled: t_{LAG} (= SSL negation delay) ≥ t_{CSH} (min.)	SSLND register SPCMD register
t_{DSU}	Data Input Setup Time	Time required for the master device from outputting data to latching data. The following formula must be fulfilled: 1/2 × t_{SPcyc} − t_{OD}(max.) ≥ t_{DSU} (min.)	SPBR register SPCMD register
t_{DH}	Data Input Hold Time	Time required for the master device to remain the data output. The following formula must be fulfilled: t_{OH} (min.) + 1/2 × t_{SPcyc} ≥ t_{DH} (min.)	SPBR register SPCMD register

Table 4 Timing Conditions for the SH7239 MCU when Transferring Data

Symbol	Item	Description	Related registers
t_{SU}	Data Input Setup Time	Time required for the slave device from outputting data to latching data. The following formula must be fulfilled: 1/2 × t_{SPcyc} − t_{W} (max.) ≥ t_{SU} (min.)	SPBR register SPCMD register
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_{HO} (min.) + 1/2 × t_{SPcyc} ≥ t_{H} (min.)	SPBR register SPCMD register

2.4 Sample Program Operation

2.4.1 RSPI Initialization Example

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

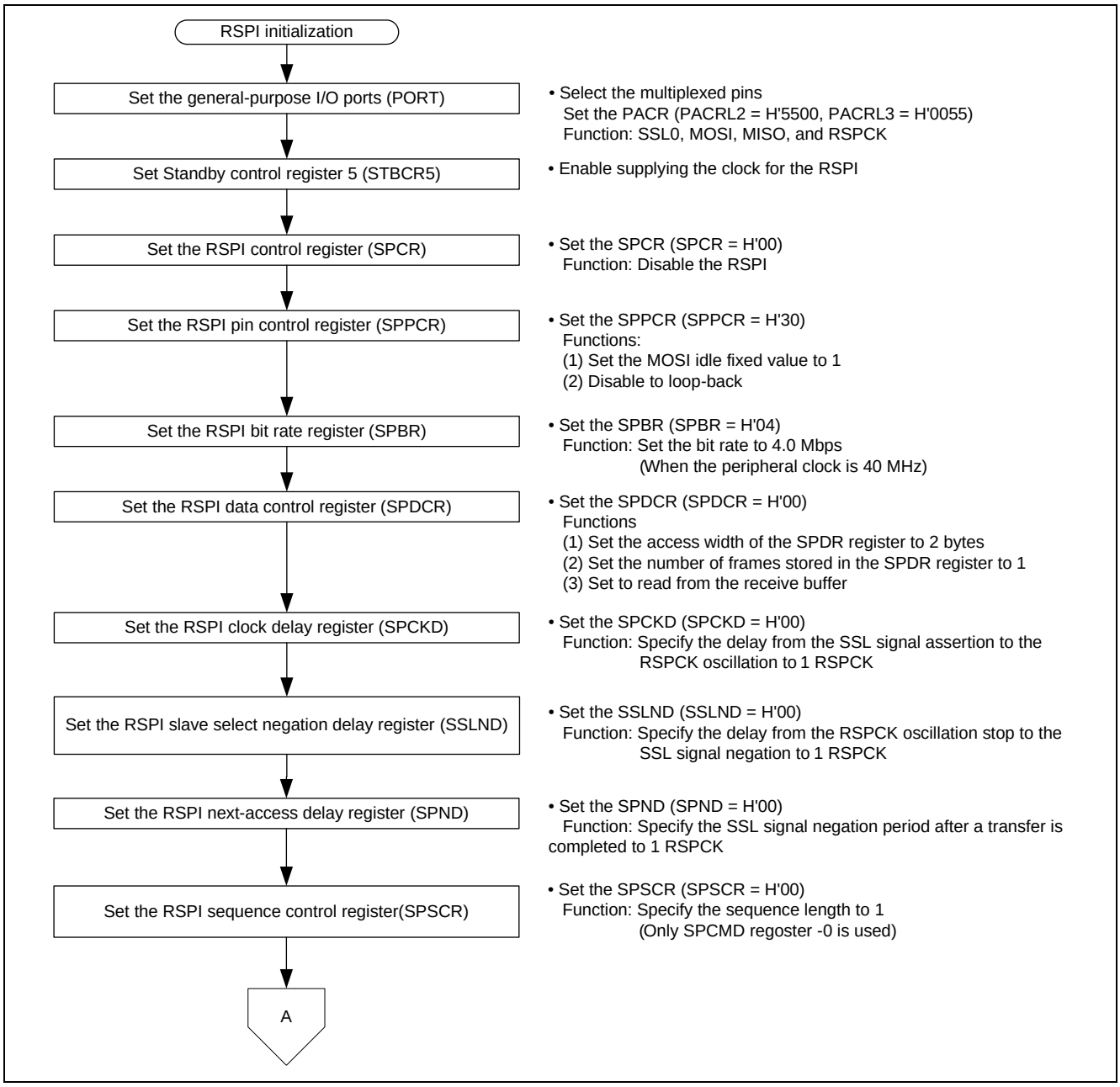

Figure 3 Flow Chart for Initializing the RSPI (1/2)

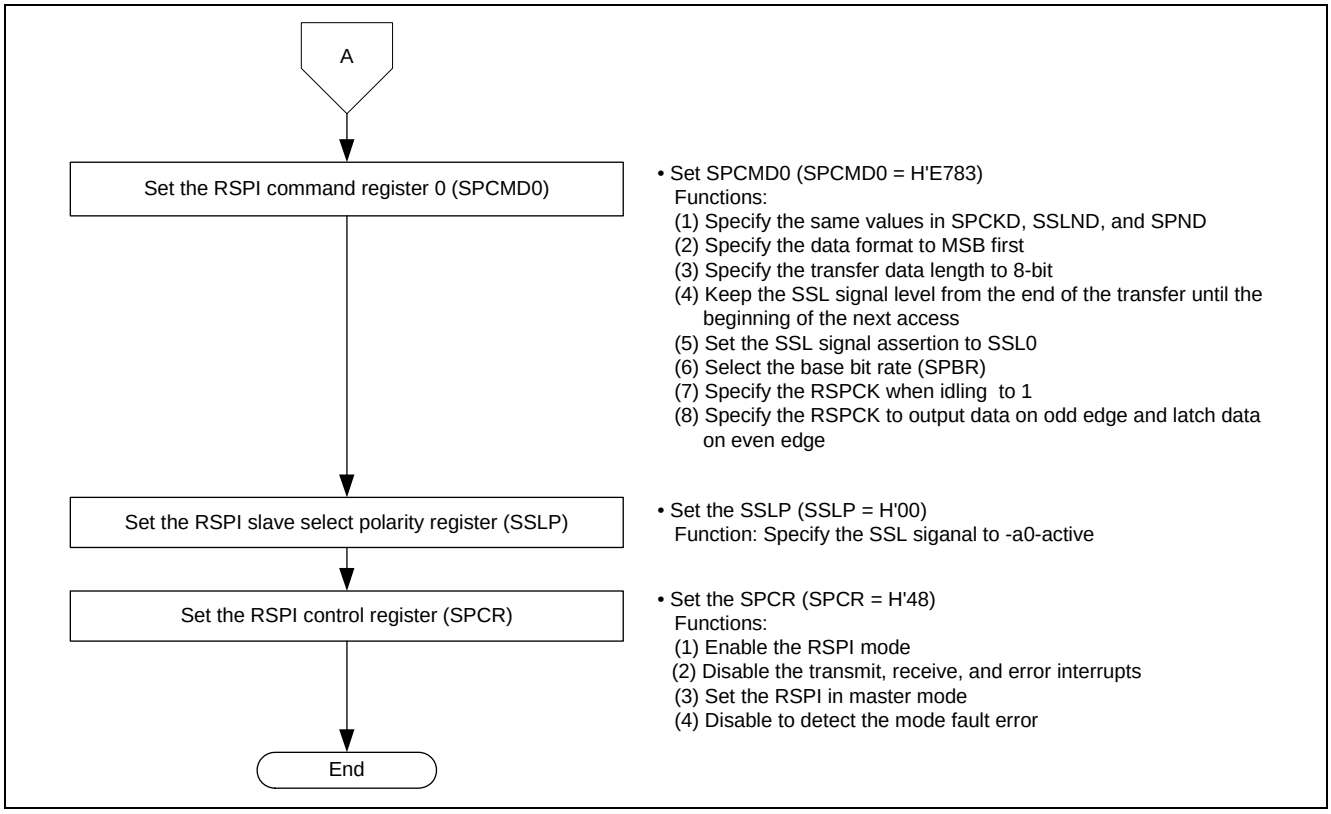

Figure 4 Flow Chart for Initializing the RSPI (2/2)

2.4.2 Command Transfer Example

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

This application refers to the commands of the Renesas Electronics R1EX25512ATA00A. 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 R1EX25512ATA00A.

Table 5 R1EX25512ATA00A Commands

Command Name	Opcode	Address Bytes	Data Bytes	Description
WREN	H'06	0	0	Write Enable
WRDI	H'04	0	0	Write Disable
RDSR	H'05	0	1	Reads the status
WRSR	H'01	0	1	Writes the status
READ	H'03	2	1 or more (1)	Reads data from the memory
WRITE	H'02	2	1 to 128 (2)	Writes data to the memory

Notes 1. Reads the address incremented from the specified address (When the last byte of the memory array has been read, the device will continue reading back at the beginning of the array).

2. Writes the data in the incremented in the same page (When the device goes beyond the end of the page, it will wrap around back to the beginning of the same page).

B. Command Sequence Example

Figure 5 shows the sequence example of the READ command.

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

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

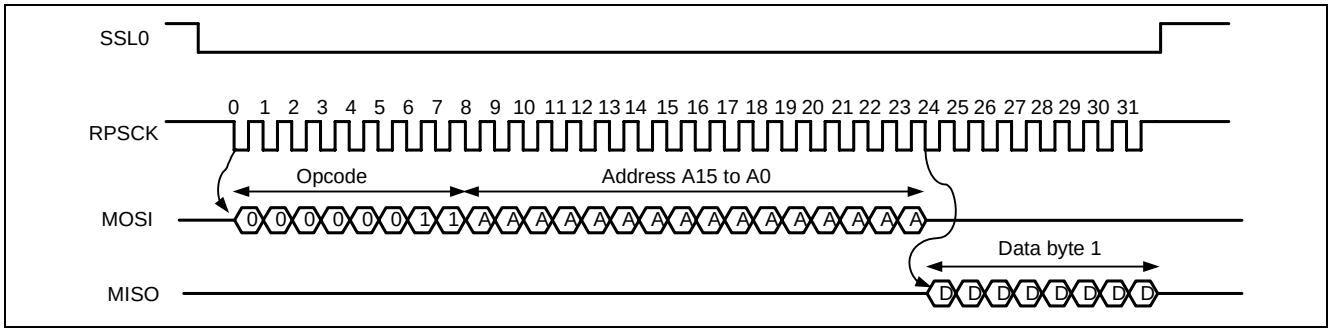

Figure 5 Read Command Sequence (Opcode: H'03)

C. Command Transfer Example in the Sample Program

The Read command that uses both master output and slave output, and the Write command that uses the master output are supported by the sample program. Figure 6 shows the flow chart of the read/write commands transfer. Figure 7 shows the flow chart of the data transfer.

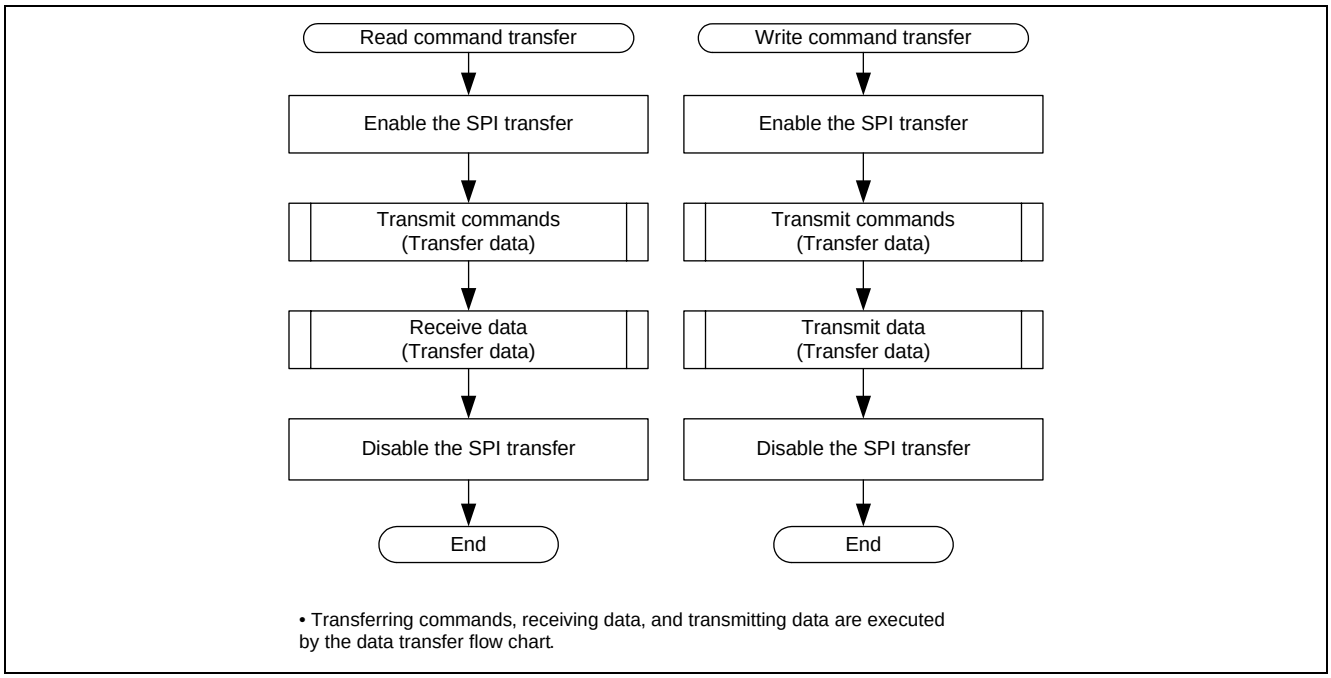

Figure 6 Flow Chart of Read/Write Commands Transfer

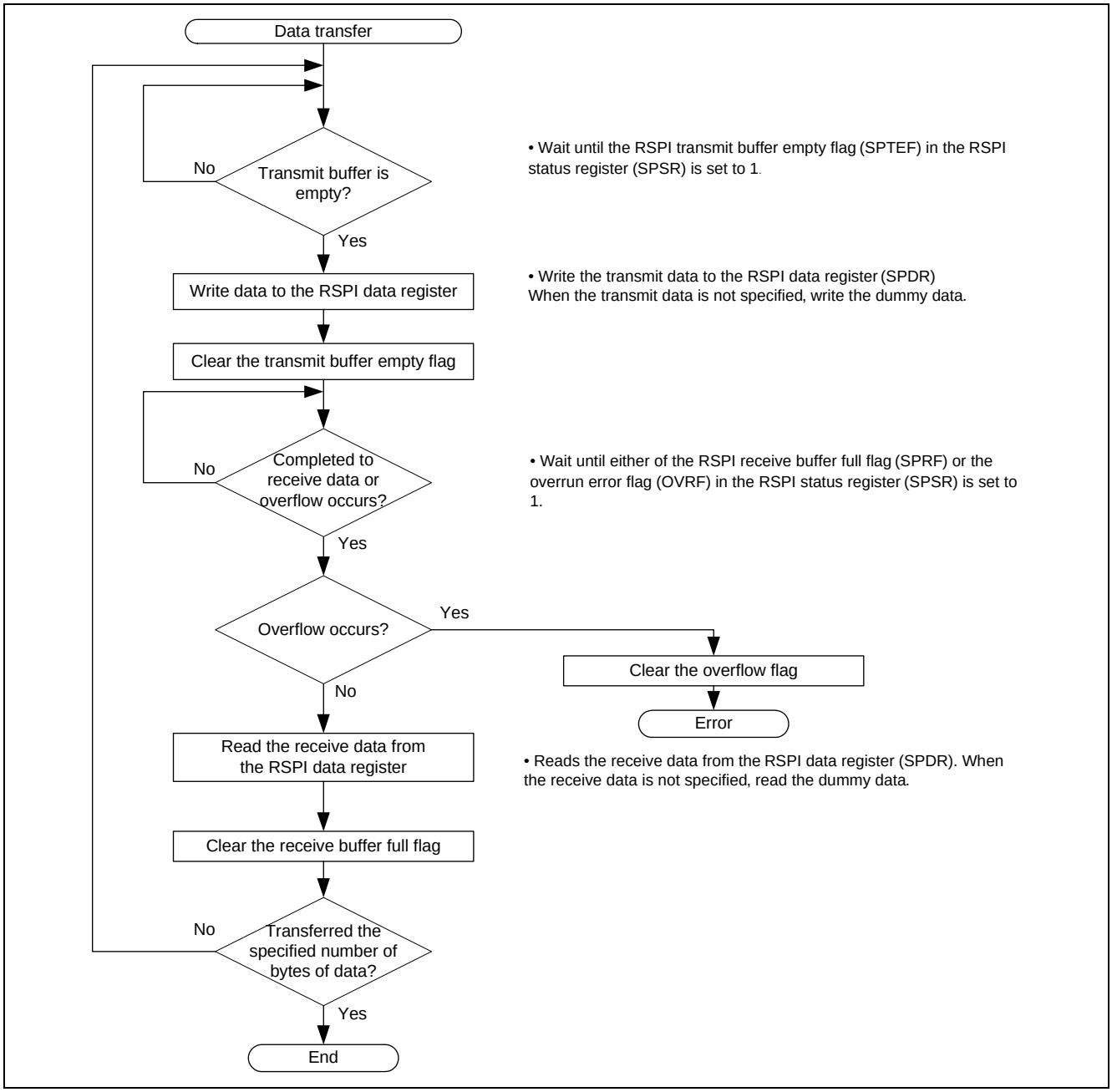

Figure 7 Flow Chart of the Data Transfer

2.4.3 Main Function

Figure 8 shows the flow chart of the main function in the sample program. The sample program writes data in the entire memory array, and compares the written value to the read value.

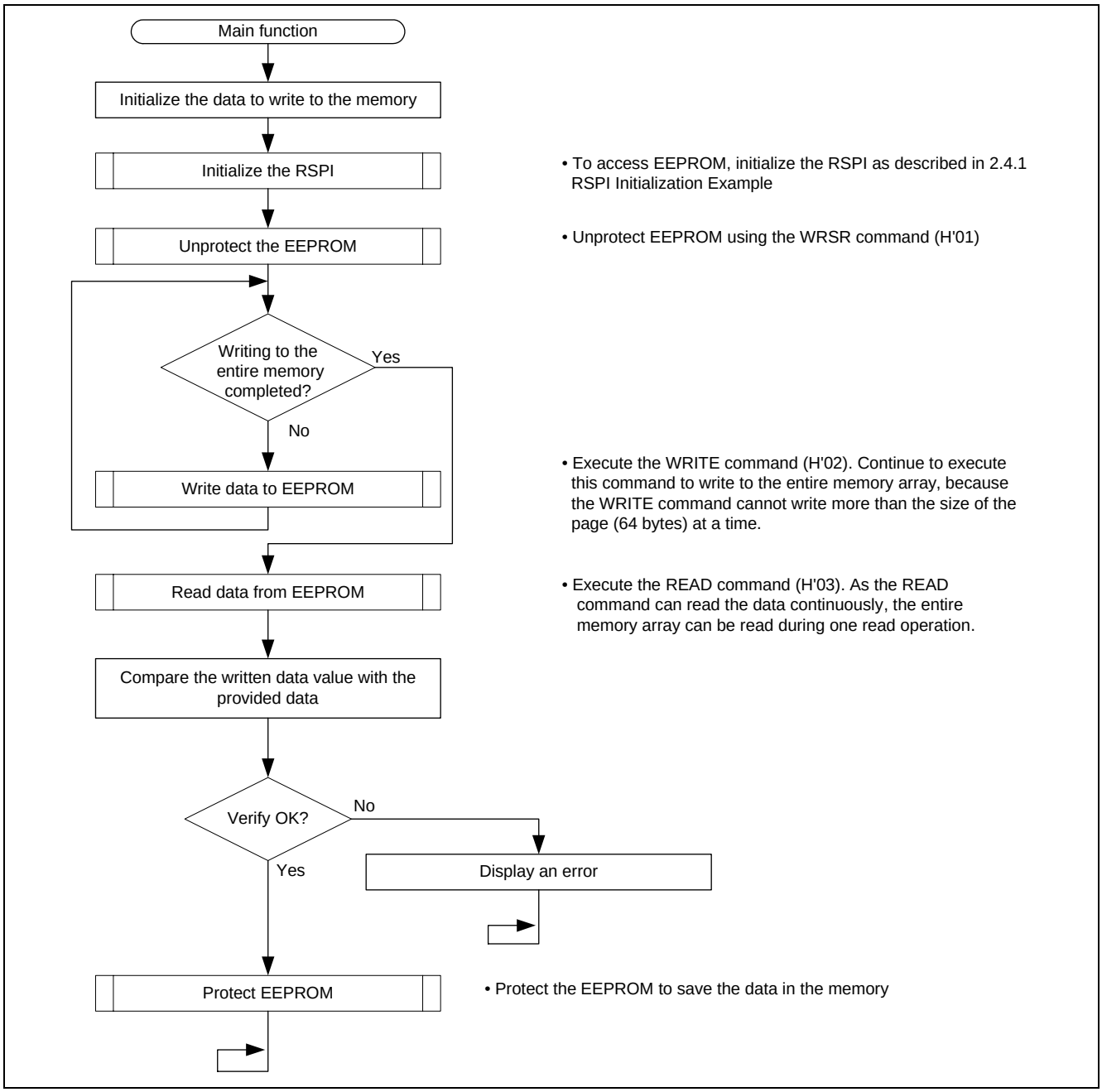

Figure 8 Main Function Flow Chart

3. Sample Program Listing

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

```
/*****************************************************************************
*   DISCLAIMER
*
*   This software is supplied by Renesas Electronics Corporation and is only
*   intended for use with Renesas products.  No other uses are authorized.
*
*   This software is owned by Renesas Electronics Corporation and is protected under
*   all applicable laws, including copyright laws.
*
*   THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
*   REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
*   INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
*   PARTICULAR PURPOSE AND NON-INFRINGEMENT.  ALL SUCH WARRANTIES ARE EXPRESSLY
*   DISCLAIMED.
*
*   TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
*   ELECTRONICS CORPORATION NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
*   FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
*   FOR ANY REASON RELATED TO THIS SOFTWARE, EVEN IF RENESAS OR ITS
*   AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
*
*   Renesas reserves the right, without notice, to make changes to this
*   software and to discontinue the availability of this software.
*   By using this software, you agree to the additional terms and
*   conditions found by accessing the following link:
*   http://www.renesas.com/disclaimer
*****************************************************************************
*   Copyright (C) 2010 Renesas Electronics Corporation. All Rights Reserved.
*""FILE COMMENT""*********** Technical reference data *************************
*   System Name : SH7239 Sample Program
*   File Name   : main.c
*   Abstract    : Reading/Writing EEPROM Using the Renesas Serial
*               : Peripheral Interface
*   Version     : 1.00.00
*   Device      : SH7239
*   Tool-Chain  : High-performance Embedded Workshop (Ver.4.07.00).
*               : C/C++ compiler package for the SuperH RISC engine family
*               :                               (Ver.9.03 Release02).
*   OS          : None
*   H/W Platform: R0K572390 (CPU board)
*   Description : Connects the EEPROM with the MCU using the Renesas Serial
*               : Peripheral Interface.
*****************************************************************************
*   History     : Aug.20,2010 Ver.1.00.00
*""FILE COMMENT END""*********************************************************/
#include <stdio.h>
#include "eeprom.h"

```

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

```
49  /* ==== Macro definition ==== */
50  #define TOP_ADDRESS    0              /* Start address of EEPROM */
51
52  /* ==== Function prototype declaration ==== */
53  void main(void);
54
55  /* ==== Variable definition ==== */
56  #pragma section DEBUG_BUFFER
57  static unsigned char data[EEP_BUFF_SIZE];
58  static unsigned char rbuf[EEP_BUFF_SIZE];
59  #pragma section
60
61  /*""FUNC COMMENT""*****************************************************************
62   * ID           :
63   * Outline      : Accessing EEPROM main
64   *-----------------------------------------------------------------------------
65   * Include      :
66   *-----------------------------------------------------------------------------
67   * Declaration  : void main(void);
68   *-----------------------------------------------------------------------------
69   * Description  : Writes or reads EEPROM.
70   *              : Initializes the RSPI, and then writes data to the entire memory
71   *              : array from the beginning. Then, it reads and verifies the data.
72   *-----------------------------------------------------------------------------
73   * Argument     : void
74   *-----------------------------------------------------------------------------
75   * Return Value : void
76   *-----------------------------------------------------------------------------
77   * Note         : None
78   *""FUNC COMMENT END""*************************************************************/
79  void main(void)
80  {
81    int i,j;
82    static unsigned long addr;
83
84    /* ==== Initializes the RSPI ==== */
85    eep_init_eeprom();
86
87    /* ==== Unprotects EEPROM ==== */
88    eep_protect_ctrl( EEP_REQ_UNPROTECT );
89
```

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

```
90      /* ==== Writes data (64 KB) ==== */
91      addr = TOP_ADDRESS;
92      /* ---- Initializes the data (16 KB) ---- */
93      for(i = 0; i < EEP_BUFF_SIZE; i++){
94        data[i] = i % 100;
95        rbuf[i] = 0;
96      }
97      /* ---- Writes one memory (64 KB) data ---- */
98      for(j = 0; j < EEP_MEM_SIZE/EEP_BUFF_SIZE; j++){
99        /* ---- Writes one buffer (4 KB) data ---- */
100       for(i = 0; i < ( EEP_BUFF_SIZE / EEP_PAGE_SIZE ); i++){
101           /* ---- Writes one page (128 bytes) data ---- */
102           eep_byte_write( addr, data+(i*EEP_PAGE_SIZE), EEP_PAGE_SIZE );
103           addr += EEP_PAGE_SIZE;            /* Updates the destination address to write */
104       }
105     }
106     /* ==== Reads data (64 KB) ==== */
107     addr = TOP_ADDRESS;
108     /* ---- Reads one memory (64 KB) data ---- */
109     for(j = 0; j < EEP_MEM_SIZE/EEP_BUFF_SIZE; j++){
110       /* ---- Reads one buffer (4 KB) data ---- */
111       eep_byte_read( addr, rbuf, EEP_BUFF_SIZE );
112       addr += EEP_BUFF_SIZE;                /* Updates the destination address to read */
113       /* ---- Verifies data ---- */
114       for(i = 0; i < EEP_BUFF_SIZE; i++){
115           data[i] = i % 100;        /* Outputs the written data */
116           if( data[i] != rbuf[i] ){
117               puts("Error: verify error\n");
118               fflush(stdout);
119               while(1);
120           }
121       }
122     }
123     /* ==== Protects EEPROM ==== */
124     eep_protect_ctrl( EEP_REQ_PROTECT );
125
126     while(1){
127          /* loop */
128       }
129   }
130
131   /* End of File */
```

3.4 Sample Program Listing "eeprom.c" (1/12)

```
1   /***********************************************************************
2   *   DISCLAIMER
3   *
4   *   This software is supplied by Renesas Electronics Corporation and is only
5   *   intended for use with Renesas products.  No other uses are authorized.
6   *
7   *   This software is owned by Renesas Electronics Corporatio nand is protected under
8   *   all applicable laws, including copyright laws.
9   *
10  *   THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
11  *   REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
12  *   INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
13  *   PARTICULAR PURPOSE AND NON-INFRINGEMENT.  ALL SUCH WARRANTIES ARE EXPRESSLY
14  *   DISCLAIMED.
15  *
16  *   TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
17  *   ELECTRONICS CORPORATION NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
18  *   FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
19  *   FOR ANY REASON RELATED TO THIS SOFTWARE, EVEN IF RENESAS OR ITS
20  *   AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
21  *
22  *   Renesas reserves the right, without notice, to make changes to this
23  *   software and to discontinue the availability of this software.
24  *   By using this software, you agree to the additional terms and
25  *   conditions found by accessing the following link:
26  *   http://www.renesas.com/disclaimer
27  ***********************************************************************
28  *   Copyright (C) 2010 Renesas Electronics Corporation. All Rights Reserved.
29  *""FILE COMMENT""*********** Technical reference data *************************
30  *   System Name : SH7239 Sample Program
31  *   File Name   : eeprom.c
32  *   Abstract    : Reading/Writing EEPROM Using the Renesas Serial
33  *               : Peripheral Interface
34  *   Version     : 1.00.00
35  *   Device      : SH7239
36  *   Tool-Chain  : High-performance Embedded Workshop (Ver.4.07.00).
37  *               : C/C++ compiler package for the SuperH RISC engine family
38  *               :                             (Ver.9.03 Release02).
39  *   OS          : None
40  *   H/W Platform: R0K572390 (CPU board)
41  *   Description : Connects the EEPROM with the MCU using the Renesas Serial
42  *               : Peripheral Interface.
43  ***********************************************************************
44  *   History     : Aug.20,2010 Ver.1.00.00
45  *""FILE COMMENT END""*****************************************************/
46  #include <stdio.h>
47  #include <machine.h>
48  #include "iodefine.h"
49  #include "eeprom.h"
50
```

3.5 Sample Program Listing "eeprom.c" (2/12)

```
/* ==== Macro definition ==== */
#define EEPROMCMD_WRITE_ENABLE0x06
#define EEPROMCMD_WRITE_DISABLE   0x04
#define EEPROMCMD_READ_STATUS 0x05
#define EEPROMCMD_WRITE_STATUS0x01
#define EEPROMCMD_READ_ARRAY  0x03
#define EEPROMCMD_WRITE_ARRAY 0x02
#define UNPROTECT_WR_STATUS       0x00
#define PROTECT_WR_STATUS     0x0C
#define EEP_BUSY_BIT          0x01

/* ==== Function prototype declaration ==== */
/*** Local function ***/
static void write_enable(void);
static void write_disable(void);
static void busy_wait(void);
static unsigned char read_status(void);
static void write_status(unsigned char status);
static void io_init_rspi(void);
static void io_cmd_exe(unsigned char *ope, int ope_sz, unsigned char *data, int data_sz);
static void io_cmd_exe_rdmode(unsigned char *ope, int ope_sz, unsigned char *rd, int rd_sz);
static int io_rspi_transfer(unsigned char *write_data, unsigned char *read_data, int data_sz);

/* ==== Variable definition ==== */

/*""FUNC COMMENT""*****************************************************************
 * ID           :
 * Outline      : EEPROM initialization
 *-----------------------------------------------------------------------------
 * Include      :
 *-----------------------------------------------------------------------------
 * Declaration  : void eep_init_eeprom(void);
 *-----------------------------------------------------------------------------
 * Description  : Initializes EEPROM for being accessed.
 *              : Initializes the Renesas Serial Peripheral Interface (RSPI).
 *-----------------------------------------------------------------------------
 * Argument     : void
 *-----------------------------------------------------------------------------
 * Return Value : void
 *-----------------------------------------------------------------------------
 * Note         : None
 *""FUNC COMMENT END""*************************************************************/
void eep_init_eeprom(void)
{
  /* ==== Initializes RSPI0 ==== */
  io_init_rspi();
}
```

3.6 Sample Program Listing "eeprom.c" (3/12)

```
/*""FUNC COMMENT""*****************************************************************
 * ID           :
 * Outline      : Protect/unprotect operation
 *-----------------------------------------------------------------------------
 * Include      :
 *-----------------------------------------------------------------------------
 * Declaration  : void eep_protect_ctrl(enum eep_req req);
 *-----------------------------------------------------------------------------
 * Description  : Protects or unprotects EEPROM. Use the argument req to specify.
 *              : Default setting and unprotecting method depends on the
 *              : specifications of EEPROM.
 *-----------------------------------------------------------------------------
 * Argument     : enum eep_req req ; I : EEP_REQ_UNPROTECT -> Write-enable entire memory array
 *              :                      EEP_REQ_PROTECT   -> Write-protect entire memory array
 *-----------------------------------------------------------------------------
 * Return Value : void
 *-----------------------------------------------------------------------------
 * Note         : None
 *""FUNC COMMENT END""*************************************************************/
void eep_protect_ctrl(enum eep_req req)
{
  if( req == EEP_REQ_UNPROTECT ){
    write_status( UNPROTECT_WR_STATUS);       /* Unprotects the entire memory area */
  }
  else{
    write_status( PROTECT_WR_STATUS );    /* Protects the entire memory area */
  }
}
/*""FUNC COMMENT""*****************************************************************
 * ID           :
 * Outline      : Write data
 *-----------------------------------------------------------------------------
 * Include      :
 *-----------------------------------------------------------------------------
 * Declaration  : void eep_byte_write(unsigned long addr, unsigned char *buf, int size);
 *-----------------------------------------------------------------------------
 * Description  : Writes the specified data to EEPROM.
 *              : Issue the Write Enable command before writing. After writing,
 *              : make sure to check the status of EEPROM if it is not busy.
 *              : The maximum write data size depends on the type of the device.
 *-----------------------------------------------------------------------------
 * Argument     : unsigned long addr ; I : Address in EEPROM to write
 *              : unsigned char *buf ; I : Buffer address to store the write data
 *              : int size           ; I : Number of bytes to write
 *-----------------------------------------------------------------------------
 * Return Value : void
 *-----------------------------------------------------------------------------
 * Note         : None
 *""FUNC COMMENT END""*************************************************************/
void eep_byte_write(unsigned long addr, unsigned char *buf, int size)
```

3.7 Sample Program Listing "eeprom.c" (4/12)

```
148  {
149    unsigned char cmd[3];
150
151    cmd[0] = EEPROMCMD_WRITE_ARRAY;
152    cmd[1] = (unsigned char)((addr >>  8) & 0xff);
153    cmd[2] = (unsigned char)( addr        & 0xff);
154    write_enable();
155    io_cmd_exe(cmd, 3, buf, size);
156    busy_wait();
157  }
158  /*""FUNC COMMENT""*****************************************************************
159   * ID           :
160   * Outline      : Read data
161   *-------------------------------------------------------------------------------
162   * Include      :
163   *-------------------------------------------------------------------------------
164   * Declaration  : void eep_byte_read(unsigned long addr, unsigned char *buf, int size);
165   *-------------------------------------------------------------------------------
166   * Description  : Reads the specified number of bytes from EEPROM.
167   *-------------------------------------------------------------------------------
168   * Argument     : unsigned long addr ; I : Address in EEPROM to read
169   *              : unsigned char *buf ; I : Buffer address to store the read data
170   *              : int size           ; I : Number of bytes to read
171   *-------------------------------------------------------------------------------
172   * Return Value : void
173    *------------------------------------------------------------------------------
174   * Note         : None
175   *""FUNC COMMENT END""*************************************************************/
176  void eep_byte_read(unsigned long addr, unsigned char *buf, int size)
177  {
178    unsigned char cmd[3];
179
180    cmd[0] = EEPROMCMD_READ_ARRAY;
181    cmd[1] = (unsigned char)((addr >>  8) & 0xff);
182    cmd[2] = (unsigned char)( addr        & 0xff);
183    io_cmd_exe_rdmode(cmd, 3, buf, size);
184  }
```

3.8 Sample Program Listing "eeprom.c" (5/12)

```
185  /*""FUNC COMMENT""*****************************************************************
186   * ID           :
187   * Outline      : Write enable
188   *-------------------------------------------------------------------------------
189   * Include      :
190   *-------------------------------------------------------------------------------
191   * Declaration  : static void write_enable(void);
192   *-------------------------------------------------------------------------------
193   * Description  : Issues the Write Enable command to enable writing data to
194   *              : EEPROM.
195   *-------------------------------------------------------------------------------
196   * Argument     : void
197   *-------------------------------------------------------------------------------
198   * Return Value : void
199   *-------------------------------------------------------------------------------
200   * Note         : None
201   *""FUNC COMMENT END""*************************************************************/
202  static void write_enable(void)
203  {
204    unsigned char cmd[1];
205    cmd[0] = EEPROMCMD_WRITE_ENABLE;
206    io_cmd_exe(cmd, 1, NULL, 0);
207  }
208  /*""FUNC COMMENT""*****************************************************************
209   * ID           :
210   * Outline      : Write disable
211   *-------------------------------------------------------------------------------
212   * Include      :
213   *-------------------------------------------------------------------------------
214   * Declaration  : static void write_disable(void);
215   *-------------------------------------------------------------------------------
216   * Description  : Issues the Write Disable command to disable writing data to
217   *              : EEPROM.
218   *-------------------------------------------------------------------------------
219   * Argument     : void
220   *-------------------------------------------------------------------------------
221   * Return Value : void
222   *-------------------------------------------------------------------------------
223   * Note         : None
224   *""FUNC COMMENT END""*************************************************************/
225  static void write_disable(void)
226  {
227    unsigned char cmd[1];
228    cmd[0] = EEPROMCMD_WRITE_DISABLE;
229    io_cmd_exe(cmd, 1, NULL, 0);
230  }
```

3.9 Sample Program Listing "eeprom.c" (6/12)

```
231  /*""FUNC COMMENT""*****************************************************************
232   * ID           :
233   * Outline      : Busy waiting
234   *-------------------------------------------------------------------------------
235   * Include      :
236   *-------------------------------------------------------------------------------
237   * Declaration  : static void busy_wait(void);
238   *-------------------------------------------------------------------------------
239   * Description  : Loops internally when the EEPROM status is busy.
240   *-------------------------------------------------------------------------------
241   * Argument     : void
242   *-------------------------------------------------------------------------------
243   * Return Value : void
244   *-------------------------------------------------------------------------------
245   * Note         : None
246   *""FUNC COMMENT END""*************************************************************/
247  static void busy_wait(void)
248  {
249    while ((read_status() & EEP_BUSY_BIT) != 0) {  /* RDY/BSY */
250      /* EEPROM is busy */
251    }
252  }
253  /*""FUNC COMMENT""*****************************************************************
254   * ID           :
255   * Outline      : Read status
256   *-------------------------------------------------------------------------------
257   * Include      :
258   *-------------------------------------------------------------------------------
259   * Declaration  : static unsigned char read_status(void);
260   *-------------------------------------------------------------------------------
261   * Description  : Reads the status of EEPROM.
262   *-------------------------------------------------------------------------------
263   * Argument     : void
264   *-------------------------------------------------------------------------------
265   * Return Value : Status register value
266   *-------------------------------------------------------------------------------
267   * Note         : None
268   *""FUNC COMMENT END""*************************************************************/
269  static unsigned char read_status(void)
270  {
271    unsigned char buf;
272    unsigned char cmd[1];
273
274    cmd[0] = EEPROMCMD_READ_STATUS;
275    io_cmd_exe_rdmode(cmd, 1, &buf, 1);
276    return buf;
277  }
```

3.10 Sample Program Listing "eeprom.c" (7/12)

```
278  /*""FUNC COMMENT""*****************************************************************
279   * ID           :
280   * Outline      : Write status
281   *-------------------------------------------------------------------------------
282   * Include      :
283   *-------------------------------------------------------------------------------
284   * Declaration  : static void write_status(unsigned char status);
285   *-------------------------------------------------------------------------------
286   * Description  : Writes the status of EEPROM.
287   *-------------------------------------------------------------------------------
288   * Argument     : unsigned char status ; I : status register value
289   *-------------------------------------------------------------------------------
290   * Return Value : void
291   *-------------------------------------------------------------------------------
292   * Note         : None
293   *""FUNC COMMENT END""*************************************************************/
294  static void write_status(unsigned char status)
295  {
296    unsigned char cmd[2];
297
298    cmd[0] = EEPROMCMD_WRITE_STATUS;
299    cmd[1] = status;
300
301    write_enable();
302    io_cmd_exe(cmd, 2, NULL, 0);
303    busy_wait();
304  }
```

3.11 Sample Program Listing "eeprom.c" (8/12)

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

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

  /* ==== RSPI ==== */
  RSPI.SPCR.BYTE   =  0x00;   /* Disables the RSPI */
  RSPI.SPPCR.BYTE  =  0x30;   /* MOSI idle fixed value = 1 */
  RSPI.SPBR.BYTE   = 0x04;/* Specifies the base bit rate as 4.0 MHz
                          /* (P clock = 40 MHz) */
  RSPI.SPDCR.BYTE  =  0x00;   /* Access width of the SPDR register: 16-bit */
  RSPI.SPCKD.BYTE  =  0x00;   /* RSPCK delay: 1 RSPCK */
  RSPI.SSLND.BYTE  =  0x00;   /* SSL negation delay: 1 RSPCK */
  RSPI.SPND.BYTE   =  0x00;   /* Next-access delay: 1 RSPCK */
  RSPI.SPSCR.BYTE  =  0x00;   /* Sequence length: 1 (Only SPCMD0 is used) */
  RSPI.SPCMD0.WORD = 0xE783;  /* MSB first */
                          /* Data length: 8-bit */
                          /* Keeps the SSL signal level after a transfer */
                          /* is completed */
                          /* Bit rate: Base bit rate is not divided */
                          /* RSPCK  when idling is 1 */
                          /* Outputs data on odd edge, latches data on even edge */
  RSPI.SSLP.BYTE   =  0x00;   /* SSLP = b'0 SSL signal 0-active */
  RSPI.SPCR.BYTE   =  0x48;   /* Master mode */
                          /* Disables interrupts */
                          /* Enables the RSPI */
}
```

3.12 Sample Program Listing "eeprom.c" (9/12)

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

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

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

3.13 Sample Program Listing "eeprom.c" (10/12)

```
/*""FUNC COMMENT""*************************************************************
 * ID           :
 * Outline      : Execute command (With read data).
 *-----------------------------------------------------------------------------
 * Include      :
 *-----------------------------------------------------------------------------
 * Declaration  : static void io_cmd_exe_rdmode(unsigned char *ope, int ope_sz,
 *              :                               unsigned char *rd,  int rd_sz)
 *-----------------------------------------------------------------------------
 * Description  : Executes the specified command.
 *              : Transmits the argument ope, and then receives data in the
 *              : argument rd. Set one of the values between 0 and 8 to the ope_sz.
 *              : More than 0 can be set to the rd_sz.
 *-----------------------------------------------------------------------------
 * Argument     : unsigned char *ope ; I : Start address of the opcode block and
 *                                       : address block to transmit
 *              : int ope_sz         ; I : Number of bytes in the opcode block and
 *                                       : address block
 *              : unsigned char *rd  ; I : Buffer address to store the received data
 *              : int rd_sz          ; I : Number of bytes in the data block
 *-----------------------------------------------------------------------------
 * Return Value : void
 *-----------------------------------------------------------------------------
 * Note         : None
 *""FUNC COMMENT END""*********************************************************/
static void io_cmd_exe_rdmode(unsigned char *ope, int ope_sz, unsigned char *rd, int rd_sz)
{
  /* ---- Enables the SPI transfer ---- */
  RSPI.SPCR.BIT.SPE = 1;

  /* ---- MISO ---- */
  io_rspi_transfer(ope, NULL, ope_sz);
  io_rspi_transfer(NULL, rd, rd_sz);

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

3.14 Sample Program Listing "eeprom.c" (11/12)

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

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

    }

    RSPI.SPDR.WORD = 0x00ff & tmp;

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

```

3.15 Sample Program Listing "eeprom.c" (12/12)

```
478       /* Waits until the reception is completed */
479       while((RSPI.SPSR.BYTE & 0x81) == 0x00){
480           /* Waits until the receive buffer is full or an overrun error occurs */
481       }
482
483       /* Overrun error occurs? */
484       if(RSPI.SPSR.BIT.OVRF == 1){
485           RSPI.SPSR.BIT.OVRF = 0;
486           return -1;  /* Overrun error occurred */
487       }
488
489       /* Reads the received data */
490       tmp = RSPI.SPDR.WORD;
491       if(read_data != (unsigned char *)0){
492           *read_data++ = (unsigned char)tmp;
493       }
494       RSPI.SPSR.BIT.SPRF = 0;
495     }
496
497     return 0;
498   }
499
500   /* End of File */
```

3.16 Sample Program Listing "eeprom.h" (1/2)

```
/***********************************************************************
*   DISCLAIMER
*
*   This software is supplied by Renesas Electronics Corporation and is only
*   intended for use with Renesas products.  No other uses are authorized.
*
*   This software is owned by Renesas Electronics Corporation and is protected under
*   all applicable laws, including copyright laws.
*
*   THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
*   REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
*   INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
*   PARTICULAR PURPOSE AND NON-INFRINGEMENT.  ALL SUCH WARRANTIES ARE EXPRESSLY
*   DISCLAIMED.
*
*   TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
*   ELECTRONICS CORPORATION NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
*   FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
*   FOR ANY REASON RELATED TO THIS SOFTWARE, EVEN IF RENESAS OR ITS
*   AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
*
*   Renesas reserves the right, without notice, to make changes to this
*   software and to discontinue the availability of this software.
*   By using this software, you agree to the additional terms and
*   conditions found by accessing the following link:
*   http://www.renesas.com/disclaimer
***********************************************************************
*   Copyright (C) 2010 Renesas Electronics Corporation. All Rights Reserved.
*""FILE COMMENT""*********** Technical reference data *************************
*   System Name : SH7239 Sample Program
*   File Name   : eeprom.h
*   Abstract    : Reading/Writing EEPROM Using the Renesas Serial
*               : Peripheral Interface
*   Version     : 1.00.00
*   Device      : SH7239
*   Tool-Chain  : High-performance Embedded Workshop (Ver.4.07.00).
*               : C/C++ compiler package for the SuperH RISC engine family
*               :                             (Ver.9.03 Release02).
*   OS          : None
*   H/W Platform: R0K572390 (CPU board)
*   Description : Connects the EEPROM with the MCU using the Renesas Serial
*               : Peripheral Interface.
***********************************************************************
*   History     : Aug.20,2010 Ver.1.00.00
*""FILE COMMENT END""*****************************************************/
#ifndef _EEPROM_H_
#define _EEPROM_H_

```

3.17 Sample Program Listing "eeprom.h" (2/2)

```
49  /* ==== Macro definition ==== */
50  #define EEP_PAGE_SIZE     128        /* Page size of EEPROM */
51  #define EEP_MEM_SIZE      0x10000    /* EEPROM size (64 KB) */
52  #define EEP_BUFF_SIZE     0x1000     /* Buffer area to verify the EEPROM data */
53                                     /* = 4 KB  */
54  enum eep_req{
55    EEP_REQ_PROTECT = 0,               /* Requests to protect */
56    EEP_REQ_UNPROTECT                  /* Requests to unprotect */
57  };
58  /* ==== Function prototype declaration ==== */
59  void eep_init_serial_flash(void);
60  void eep_protect_ctrl(enum eep_req req);
61  void eep_byte_write(unsigned long addr, unsigned char *buf, int size);
62  void eep_byte_read(unsigned long addr, unsigned char *buf, int size);
63
64  /* ==== Variable definition ==== */
65
66  #endif /* _EEPROM_H_ */
67
68  /* End of File */
```

4. References

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

- Hardware Manual
  SH7239 Group, SH7237 Group Hardware User's Manual Rev. 1.00
- The latest version of the hardware manual can be downloaded from the Renesas Electronics website.

Website and Support

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

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

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Nov.24.10	—	First edition issued

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this manual, refer to the relevant sections of the manual. If the descriptions under General Precautions in the Handling of MPU/MCU Products and in the body of the manual differ from each other, the description in the body of the manual takes precedence.

1. Handling of Unused Pins
   Handle unused pins in accord with the directions given under Handling of Unused Pins in the manual.
   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.
2. Processing at Power-on
   The state of the product is undefined at the moment when power is supplied.
   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.
3. Prohibition of Access to Reserved Addresses
   Access to reserved addresses is prohibited.
   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.
4. Clock Signals
   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.
   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.
5. Differences between Products
   Before changing from one product to another, i.e. to one with a different type number, confirm that the change will not lead to problems.
   — The characteristics of MPU/MCU in the same group but having different type numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different type numbers, implement a system-evaluation test for each of the products.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.
   - "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.
   - "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.
   - "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

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

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

RENESAS

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
7F, No. 363 Fu Shing North Road Taipei, Taiwan, R.O.C.
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
1 harbourFront Avenue, #06-10, keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 1.0