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.

# H8/300H Tiny Series

## Access to the Serial EEPROM in Clock Synchronous Mode of the Serial Communication Interface

### Introduction

The H8/3687 group are single-chip microcomputers based on the high-speed H8/300H CPU, and integrate all the peripheral functions necessary for system configuration. The H8/300H CPU employs an instruction set which is compatible with the H8/300 CPU.

The H8/3687 group incorporates, as peripheral functions necessary for system configuration, four types of timers, $I^2C$ bus interface, serial communication interface, and 10-bit A/D converter. These devices can be utilized as embedded microcomputers in sophisticated control systems.

These H8/300 H Series -H8/3687- Application Notes consist of a "Basic Edition" which describes operation examples when using the on-chip peripheral functions of the H8/3687 group in isolation; they should prove useful for software and hardware design by the customer.

The operation of the programs and circuits described in these Application Notes has been verified, but in actual applications, the customer should always confirm correct operation prior to actual use.

### Target Device

H8/3687

### Contents

## 1. Overview

The SPI EEPROM is read or written to via the H8/3687 serial communication interface in clock synchronous mode.

## 2. Configuration

Figure 2.1 shows a diagram of connections between the H8/3687 and SPI EEPROM.

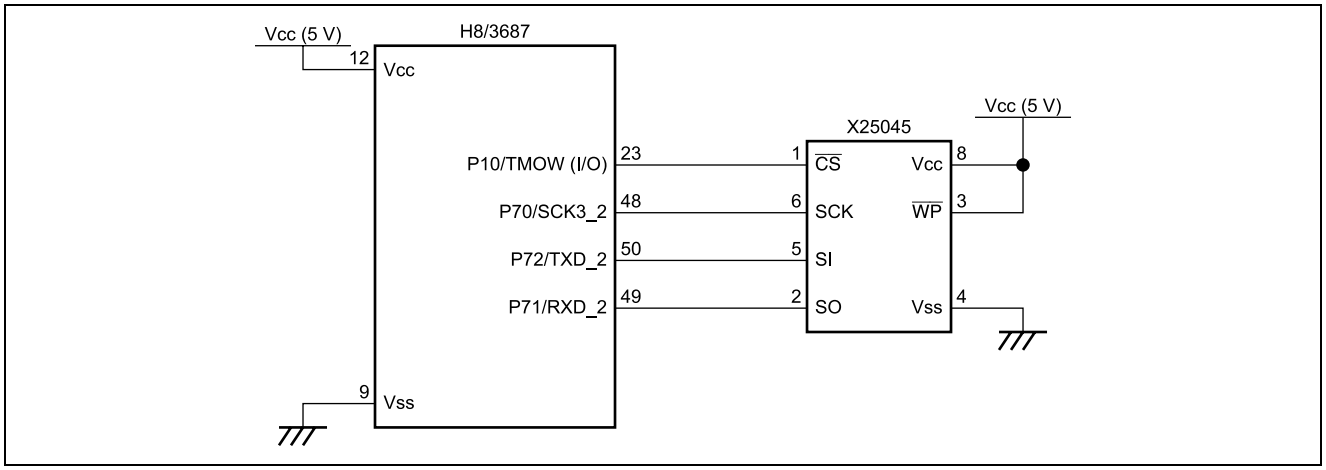



**Figure 2.1 Connection to SPI EEPROM**

Specifications:

- H8/3687 operating frequency: 16 MHz
- Table 2.1 shows the SPI EEPROM X25045 pin specifications
- SPI EEPROM specifications: 4 kbits (512 × 8 bits)

**Table 2.1 SPI EEPROM Pin Specifications**

| Symbol | Description |
|---|---|
| $\overline{CS}$ | Chip select input |
| SO | Serial output |
| SI | Serial input |
| SCK | Serial clock input |
| $\overline{WP}$ | Write protect input |
| Vss | Ground |
| Vcc | Supply voltage |
| $\overline{RESET}$/RESET | Reset output |

# 3. Sample Programs

## 3.1 Functions

1. One byte of data is written to the SPI EEPROM (Byte Write).
2. One byte of data is read from the SPI EEPROM (Random Read).
3. Data is written to the SPI EEPROM continuously (Page Write).
4. Data is read from the SPI EEPROM continuously (Sequential Read).

## 3.2 Embedding the Sample Programs

1. Sample program 3-A
   Incorporate #define directives.
2. Sample program 3-B
   Incorporate prototype declarations.
3. Sample program 3-C
   Incorporate source program.

Add setting initialization.

Add the SPI EEPROM access processing.

## 3.3 Modifications to Sample Programs

Without modifications to the sample program, the system may not run. Modifications must be made according to the customer's program and system environment.

1. A file with definitions of IO register structures can be obtained free of charge from the following Renesas Technology web site:http://www.renesas.com/eng/products/mpumcu/tool/crosstool/iodef/index.html
   The sample program can be used without further changes. When creating definitions independently, the customer should modify the IO register structures used in the sample program as appropriate.
2. In the sample program, timer Z is designed to start every 10 ms and timeout after 5 seconds, in order to monitor the state of the serial communication interface. The timer processing can be modified according to the needs of the customer, and of course can be used without modification. When using the timer processing in the sample program without modification, the following changes should be made.
   A. Sample program 3-D
      - The timer Z reset vector should be added.
      - com_timer should be added as a common variable.
      - The timer Z initial setting processing should be added.
        (The GRA setting should be changed according to the operating frequency of the microcomputer being used, so that the timer Z interrupt occurs in 10 ms. For setting values, refer to the H8/3687 Hardware Manual; for the location of the setting to be changed, refer to the program notes in the sample program.)
      - The timer Z interrupt processing should be added.
3. The serial communication interface transfer rate should be set with the BRR register according to the target device specifications and the microcomputer operating frequency. Refer to the H8/3687 Hardware Manual for setting values, and to the program notes in the sample program for the location to be changed. In this sample program, the transfer rate is set to 100 kbps.

## 3.4 Method of Use

1. One byte of data is written to the SPI EEPROM.

```
unsigned int com_spi_eeprom_write
        (unsigned int rom_addr, unsigned char rom_data)
```

| Argument | Description |
|---|---|
| rom_addr | Specifies the ROM address where data is written to. |
| rom_data | Specifies data to be written. |

| Return value | Description |
|---|---|
| 0 | Normal termination |
| 1 | Abnormal termination (transfer-preparation completion timeout) |
| 2 | Abnormal termination (transfer-completion wait timeout) |
| 3 | Abnormal termination (reception-completion wait timeout) |
| 4 | Abnormal termination (write wait timeout) |

Example of use:

```
int ret ;
unsigned char rom_data;
unsigned int rom_addr;
ret = com_spi_eeprom_write (rom_addr , rom_data)
```

2. One byte of data is read from the SPI EEPROM.

```
unsigned int com_spi_eeprom_read
        (unsigned int rom_addr, unsigned char *rom_data)
```

| Argument | Description |
|---|---|
| rom_addr | Specifies the ROM address where data is read from. |
| *rom_data | Specifies the address where read data is stored. |

| Return value | Description |
|---|---|
| 0 | Normal termination |
| 1 | Abnormal termination (transfer-preparation completion timeout) |
| 2 | Abnormal termination (transfer-completion wait timeout) |
| 3 | Abnormal termination (reception-completion wait timeout) |
| 4 | Abnormal termination (write wait timeout) |

Example of use:

```
int ret;
unsigned char *rom_data;
unsigned int rom_addr;
ret = com_spi_eeprom_read (rom_addr, rom_length, *rom_data)
```

3. Data is continuously written to the SPI EEPROM.

```
unsigned int com_spi_eeprom_page_write
(unsigned int rom_addr, unsigned int  rom_length,
unsigned char *rom_data)
```

| Argument | Description |
|---|---|
| rom_addr | Specifies the ROM address where data is written to. |
| rom_length | Specifies the write data length. On this device, up to 4 bytes of data can be written. |
| *rom_data | Specifies the start address of the area where write data is stored. |

| Return value | Description |
|---|---|
| 0 | Normal termination |
| 1 | Abnormal termination (transfer-preparation completion timeout) |
| 2 | Abnormal termination (transfer-completion wait timeout) |
| 3 | Abnormal termination (reception-completion wait timeout) |
| 4 | Abnormal termination (write wait timeout) |

Example of use:

```
int ret ;
unsigned char *rom_data;
unsigned int  rom_length , rom_addr;
ret = com_spi_eeprom_page_write (rom_addr, rom_length, *rom_data)
```

4. Data is continuously read from the SPI EEPROM.

```
unsigned int com_spi_eeprom_seq_read
(unsigned int rom_addr, unsigned int rom_length,
   unsigned char *rom_data)
```

| Argument | Description |
|---|---|
| rom_addr | Specifies the ROM address where data is read from. |
| rom_length | Specifies the read data length. |
| *rom_data | Specifies the start address of the area where read data is stored. |

| Return value | Description |
|---|---|
| 0 | Normal termination |
| 1 | Abnormal termination (transfer-preparation completion timeout) |
| 2 | Abnormal termination (transfer-completion wait timeout) |
| 3 | Abnormal termination (reception-completion wait timeout) |

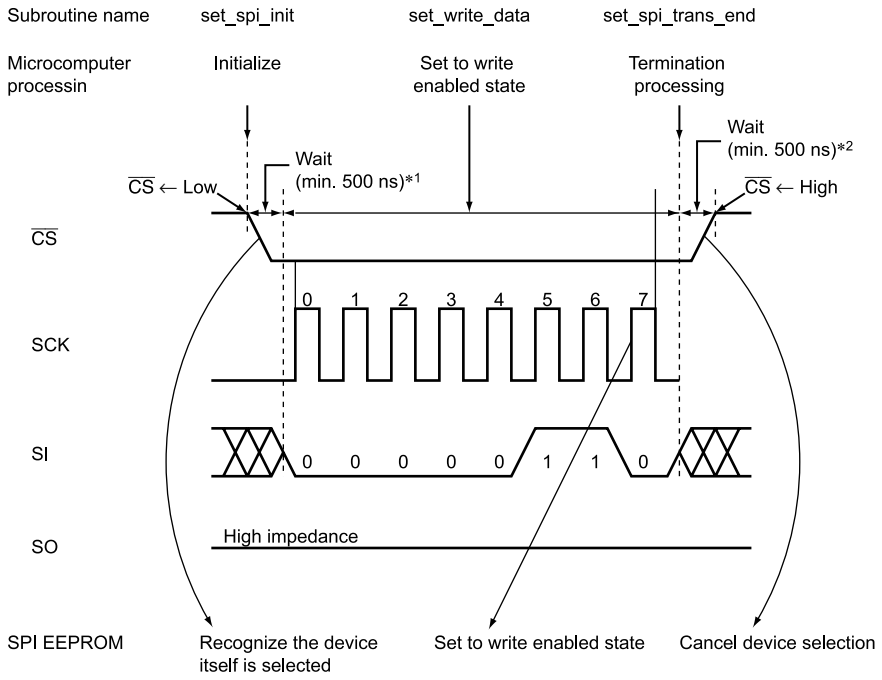
Example of use:

```
int ret;
unsigned char *rom_data;
unsigned int  rom_length, rom_addr;
ret = com_spi_eeprom_seq_read (rom_addr, rom_length, *rom_data)
```

## 3.5 Explanation of Operation

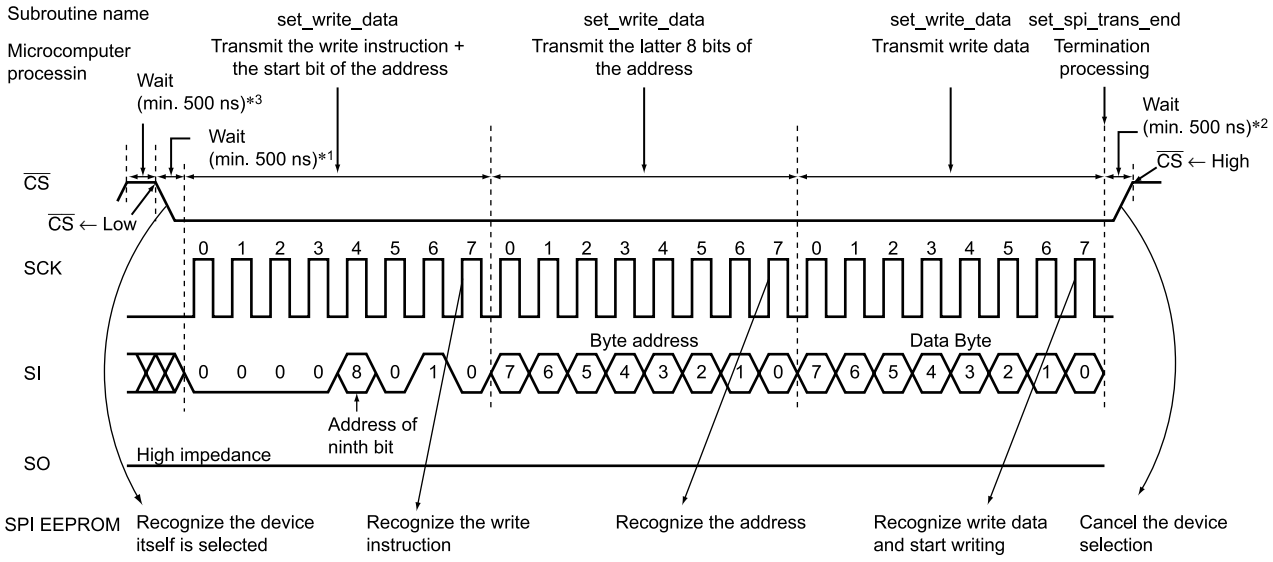
The following figure explains the operation of the H8 microcomputer and the EEPROM with respect to the interface signal state.

1. One byte of data is written to the SPI EEPROM (Byte Write).
   Procedure (a) Cancel the SPI EEPROM write disabled state.



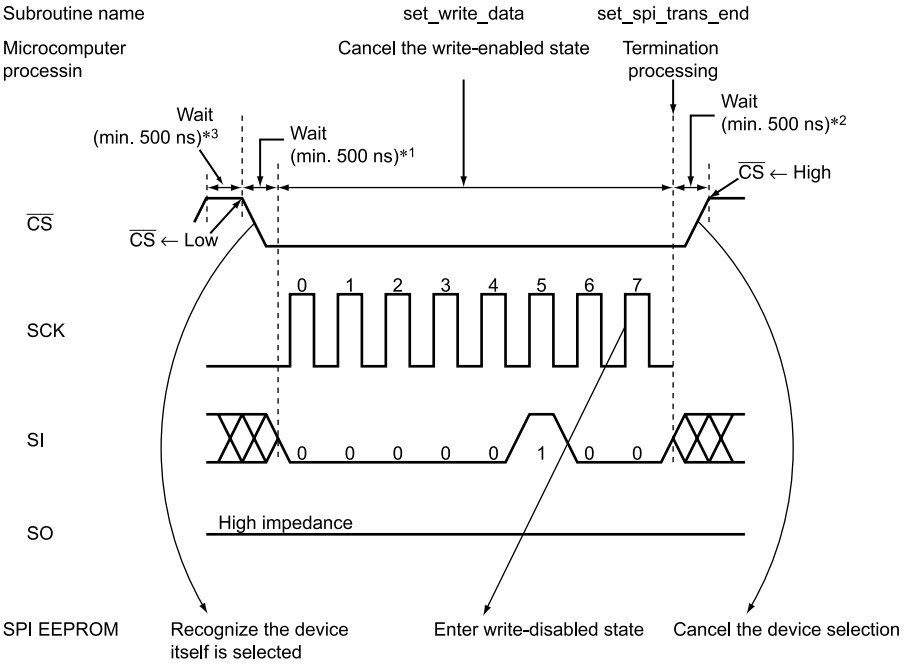

Notes: 1. A wait is inserted here for the purpose of taking at least a 500 ns interval from turning on $\overline{CS}$ to starting the data transfer. Since the waiting time is different on each device, refer to the data sheet to set the appropriate waiting time.
2. A wait is inserted here for the purpose of taking at least a 500 ns interval from terminating the data transfer to turning off $\overline{CS}$. Since the waiting time is different on each device, refer to the data sheet to set the appropriate waiting time.

Procedure (b) Write data



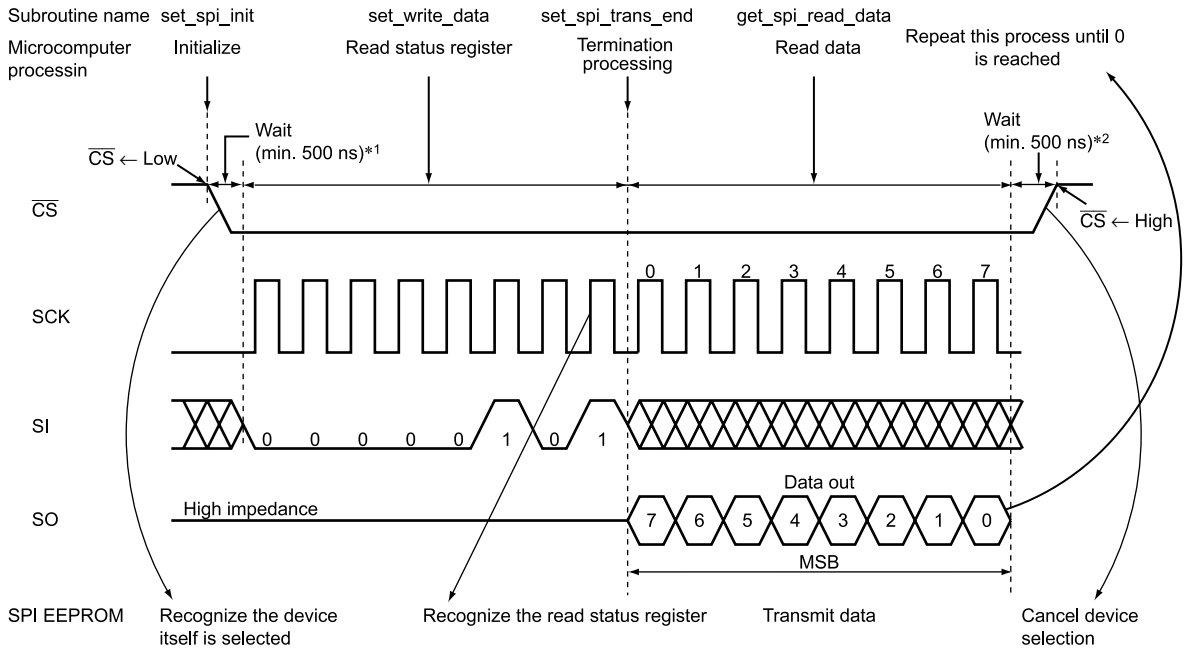

Notes: 1. A wait is inserted here for the purpose of taking at least a 500 ns interval from turning on $\overline{CS}$ to starting the data transfer. Since the waiting time is different on each device, refer to the data sheet to set the appropriate waiting time.
2. A wait is inserted here for the purpose of taking at least a 500 ns interval from terminating the data transfer to turning off $\overline{CS}$. Since the waiting time is different on each device, refer to the data sheet to set the appropriate waiting time.
3. A wait is inserted here for the purpose of taking at least a 500 ns interval from turning off $\overline{CS}$ to turning it on again. Since the waiting time is different on each device, refer to the data sheet to set the appropriate waiting time.

Procedure (c) Re-set the SPI EEPROM to write disabled state.



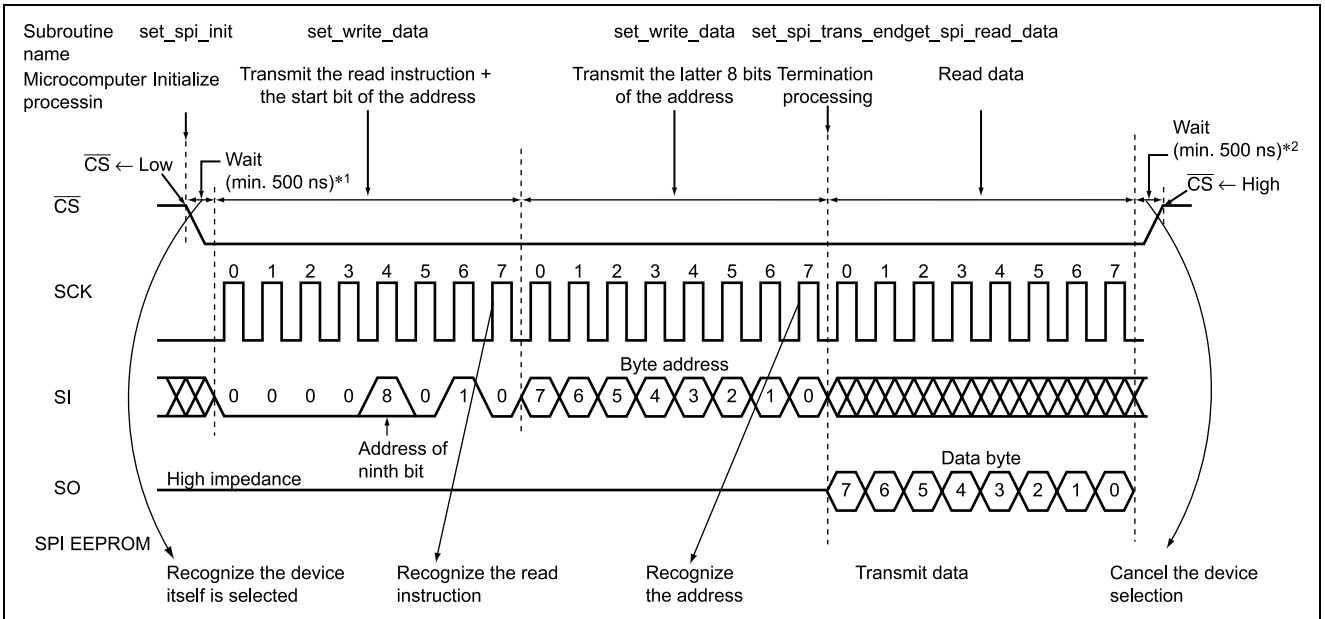

Notes: 1. A wait is inserted here for the purpose of taking at least a 500 ns interval from turning on $\overline{CS}$ to starting the data transfer. Since the waiting time is different on each device, refer to the data sheet to set the appropriate waiting time.
2. A wait is inserted here for the purpose of taking at least a 500 ns interval from terminating the data transfer to turning off $\overline{CS}$. Since the waiting time is different on each device, refer to the data sheet to set the appropriate waiting time.
3. A wait is inserted here for the purpose of taking at least a 500 ns interval from turning off $\overline{CS}$ to turning it on again. Since the waiting time is different on each device, refer to the data sheet to set the appropriate waiting time.

Procedure (d) Check write processing termination.




Notes: 1. A wait is inserted here for the purpose of taking at least a 500 ns interval from turning on $\overline{CS}$ to starting the data transfer. Since the waiting time is different on each device, refer to the data sheet to set the appropriate waiting time.
2. A wait is inserted here for the purpose of taking at least a 500 ns interval from terminating the data transfer to turning off $\overline{CS}$. Since the waiting time is different on each device, refer to the data sheet to set the appropriate waiting time.

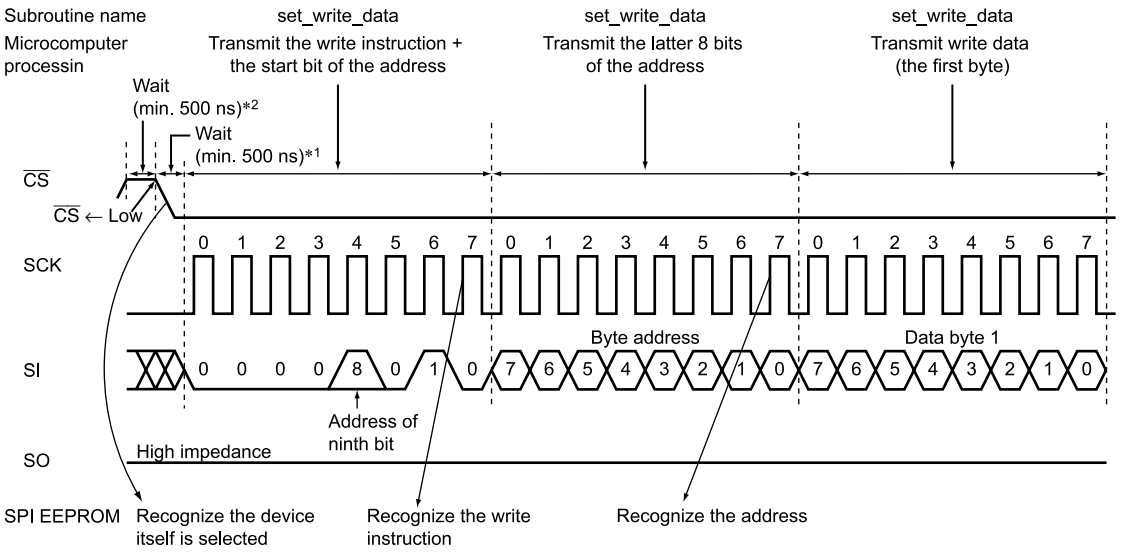
2. One byte of data is read from the SPI EEPROM (Random Read).



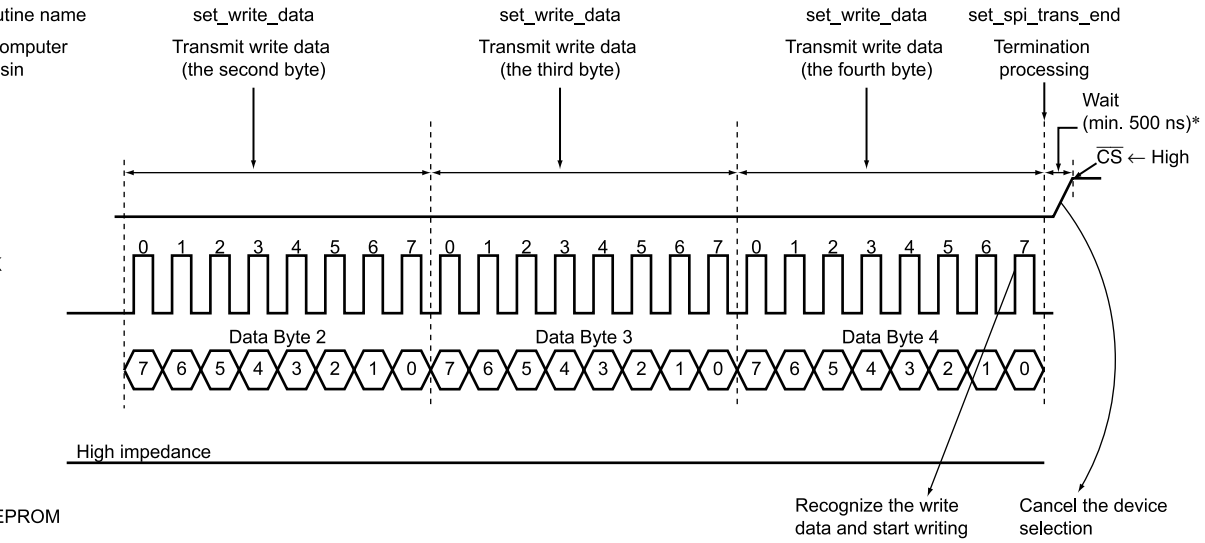

Notes: 1. A wait is inserted here for the purpose of taking at least a 500 ns interval from turning on $\overline{CS}$ to starting the data transfer. Since the waiting time is different on each device, refer to the data sheet to set the appropriate waiting time.
2. A wait is inserted here for the purpose of taking at least a 500 ns interval from terminating the data transfer to turning off $\overline{CS}$. Since the waiting time is different on each device, refer to the data sheet to set the appropriate waiting time.

3. Data is written to the SPI EEPROM continuously (Page Write).
   In this operation example, a write of 4 bytes is used.
   Procedure (a) Cancel the SPI EEPROM write disabled state.
   Same as procedure (a) described in section 3.5 (1).

   Procedure (b) Write data




Notes: 1. A wait is inserted here for the purpose of taking at least a 500 ns interval from turning on $\overline{CS}$ to starting the data transfer. Since the waiting time is different on each device, refer to the data sheet to set the appropriate waiting time.
2. A wait is inserted here for the purpose of taking at least a 500 ns interval from turning off $\overline{CS}$ to turning it on again. Since the waiting time is different on each device, refer to the data sheet to set the appropriate waiting time.

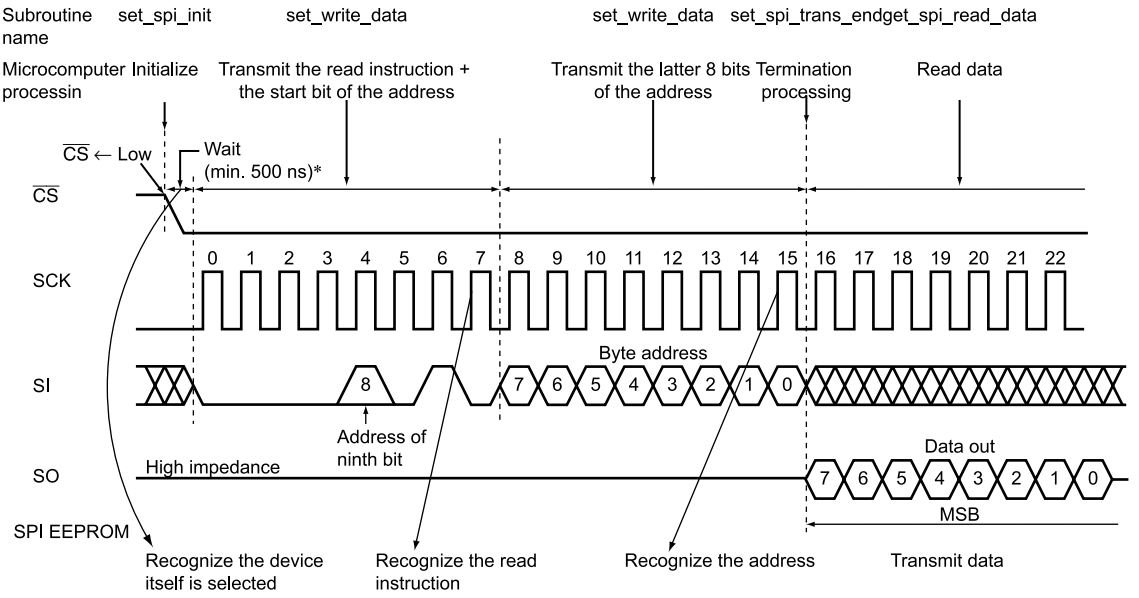



Notes: * A wait is inserted here for the purpose of taking at least a 500 ns interval from terminating the data transfer to turning off $\overline{CS}$. Since the waiting time is different on each device, refer to the data sheet to set the appropriate waiting time.

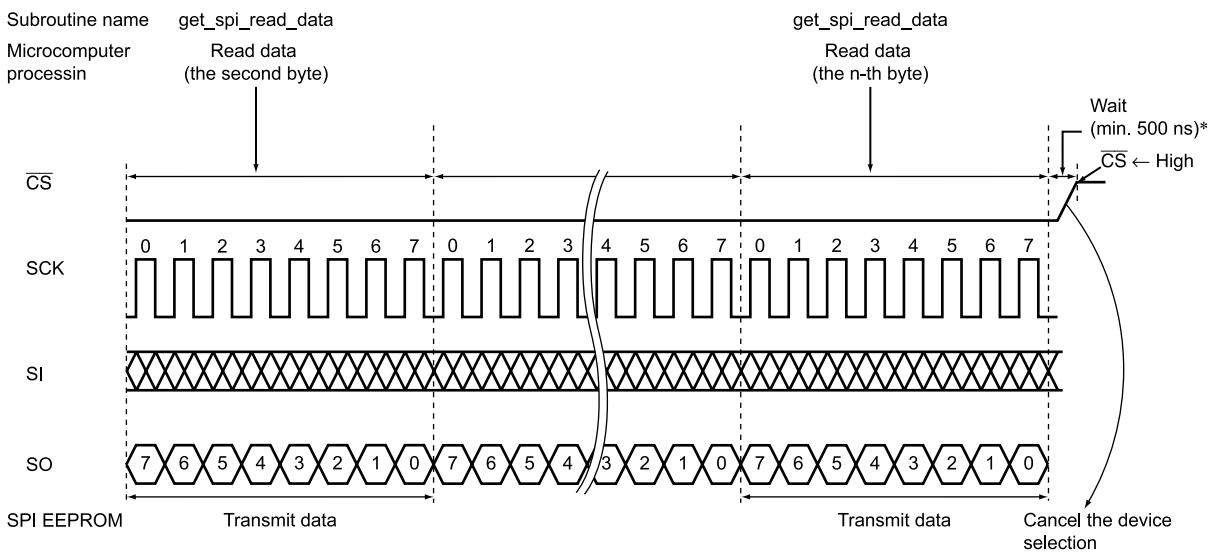
Procedure (c) Re-set the SPI EEPROM to write disabled state.Same as procedure (c) described in section 3.5 (1).

Procedure (d) Check write processing termination.Same as procedure (d) described in section 3.5 (1).

4. Data is continuously read from the SPI EEPROM (Sequential Read).




Notes: * A wait is inserted here for the purpose of taking at least a 500 ns interval from turning on $\overline{CS}$ to starting the data transfer. Since the waiting time is different on each device, refer to the data sheet to set the appropriate waiting time.




Notes: * A wait is inserted here for the purpose of taking at least a 500 ns interval from terminating the data transfer to turning off $\overline{CS}$. Since the waiting time is different on each device, refer to the data sheet to set the appropriate waiting time.

## 3.6 List of Registers Used

The internal registers of the H8 microcomputer used in the sample program are listed below. For detailed information, refer to the H8/3687 Group Hardware Manual.

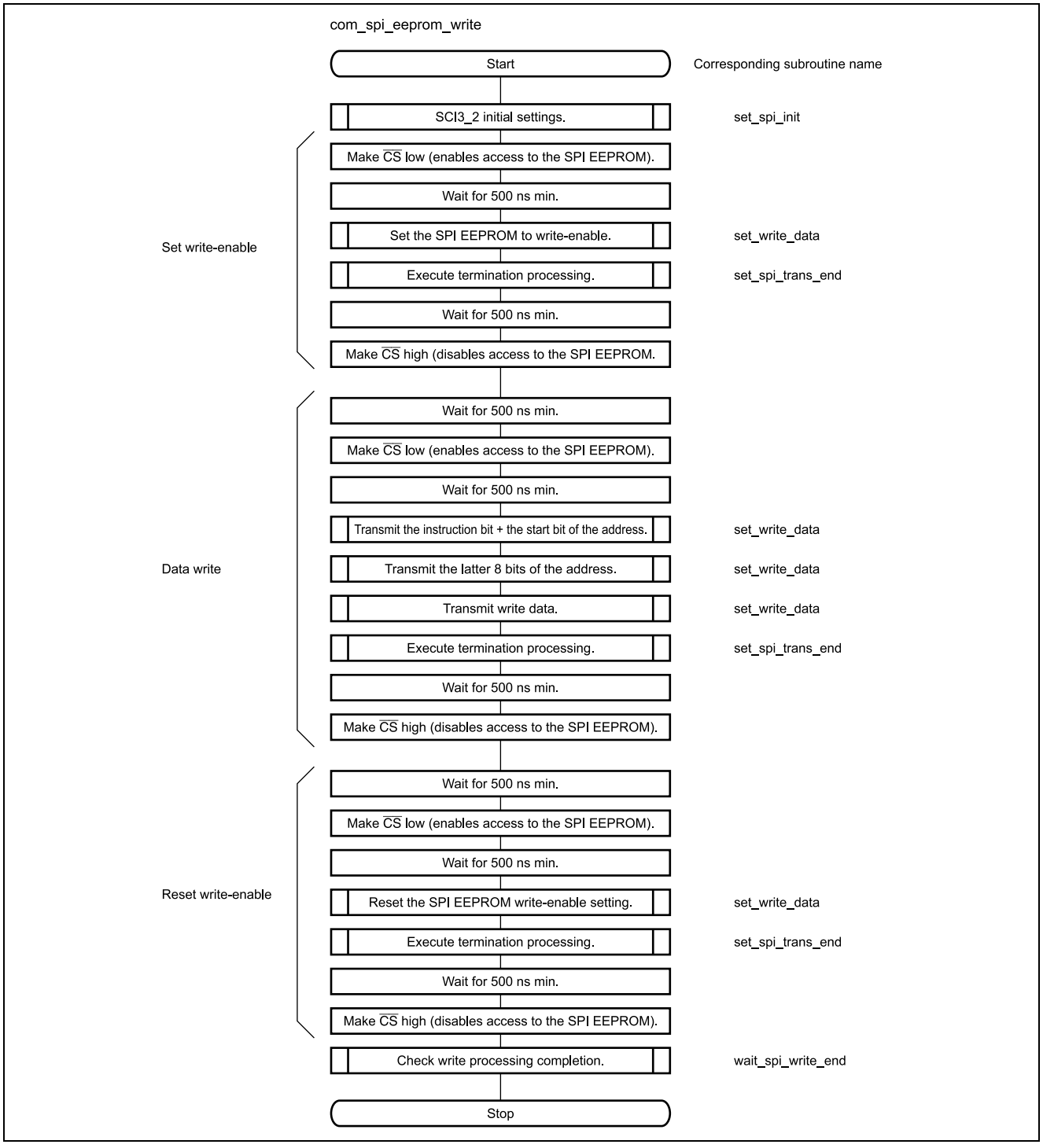
1. SCI3_2-related registers

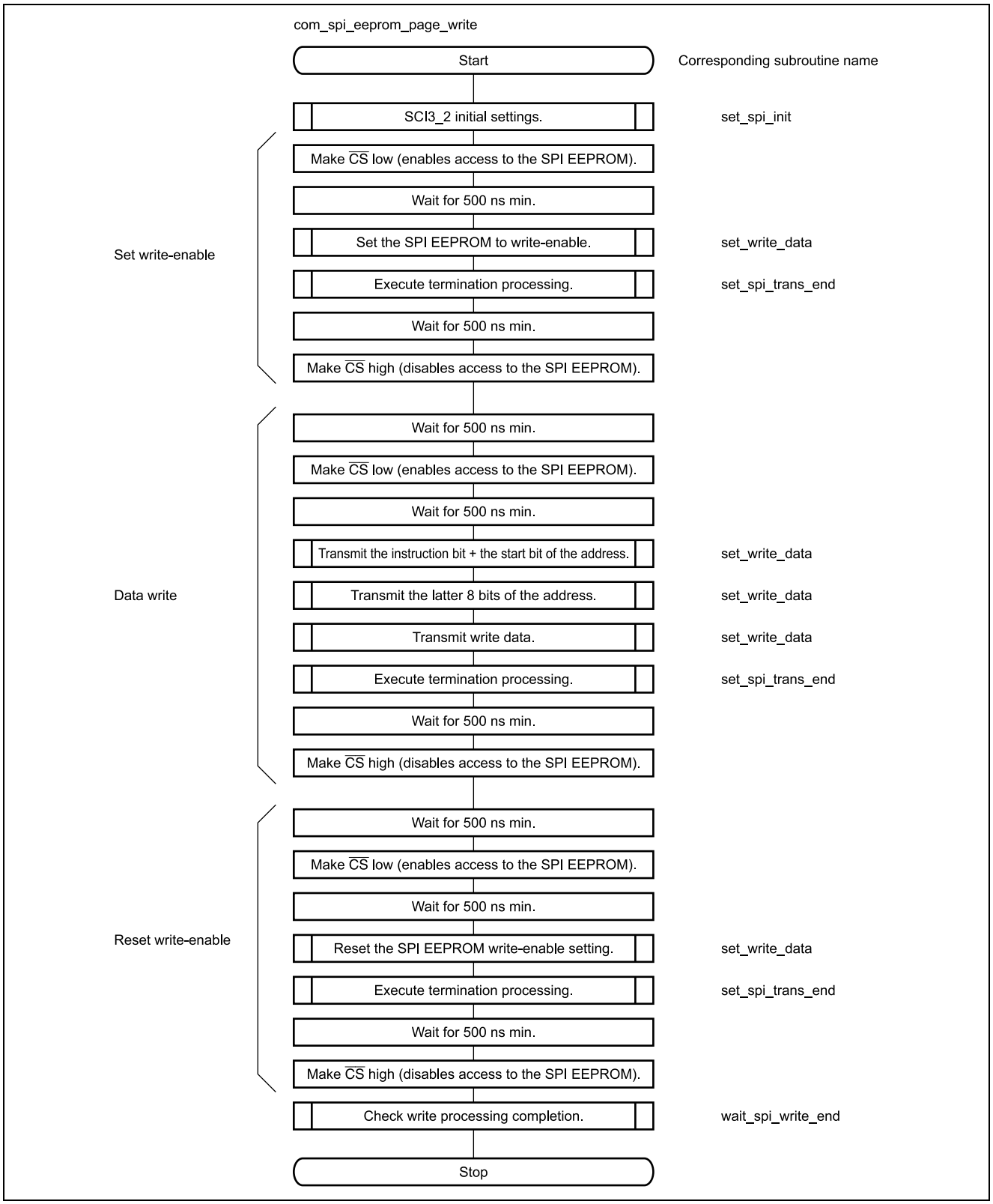
| Name | Summary |
|---|---|
| Receive data register (RDR) | 8-bit register to store receive data. |
| Transmit data register (TDR) | 8-bit register to store data to be transmitted. |
| Serial mode register (SMR) | Selects the clock source for the on-chip baud rate generator and sets the serial data communication format. |
| Serial control register 3 (SCR3) | Controls transmission/reception operation, interrupts, and selects transmission/reception clock source. |
| Serial status register (SSR) | SCI3 status flags and transmission/reception multiprocessor bits. |
| Bit rate register (BRR) | 8-bit register to set the bit rate. |

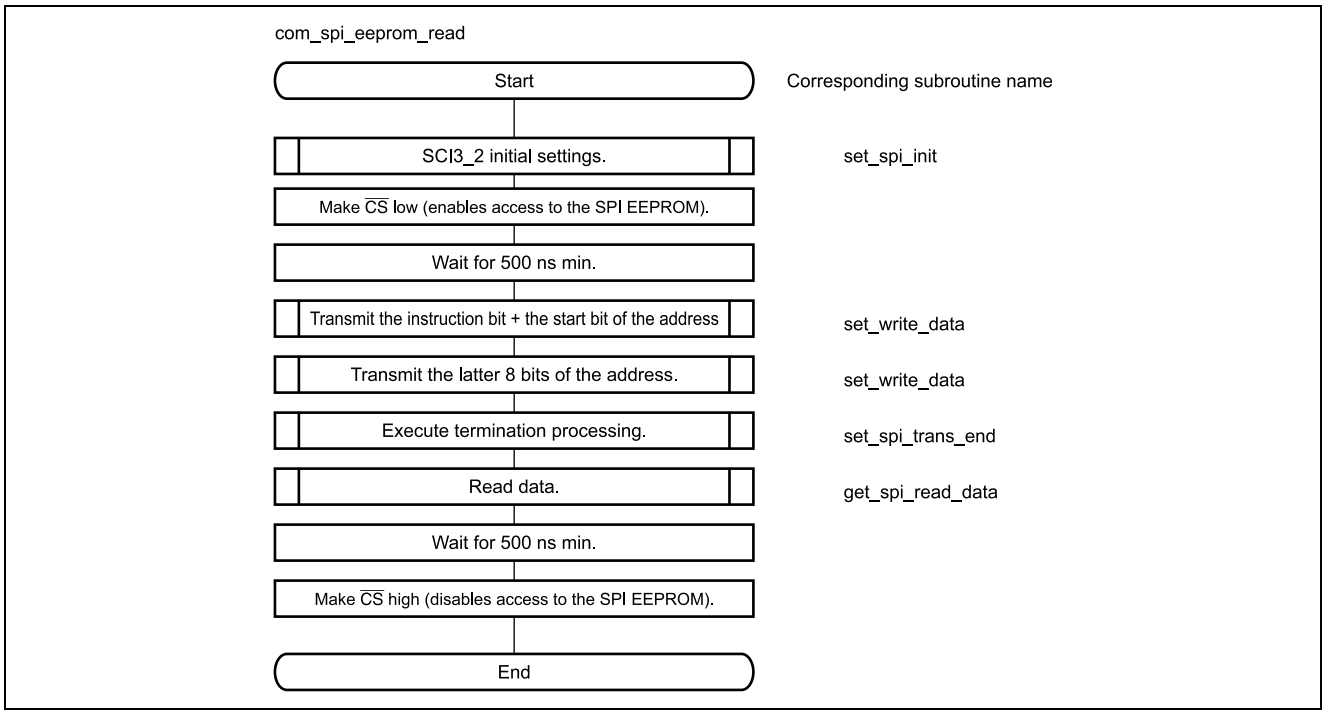
2. Timer Z-related registers
   Timer Z has various functions, but in the sample program it uses the GRA register compare-match function to generate an interrupt every 10 ms.

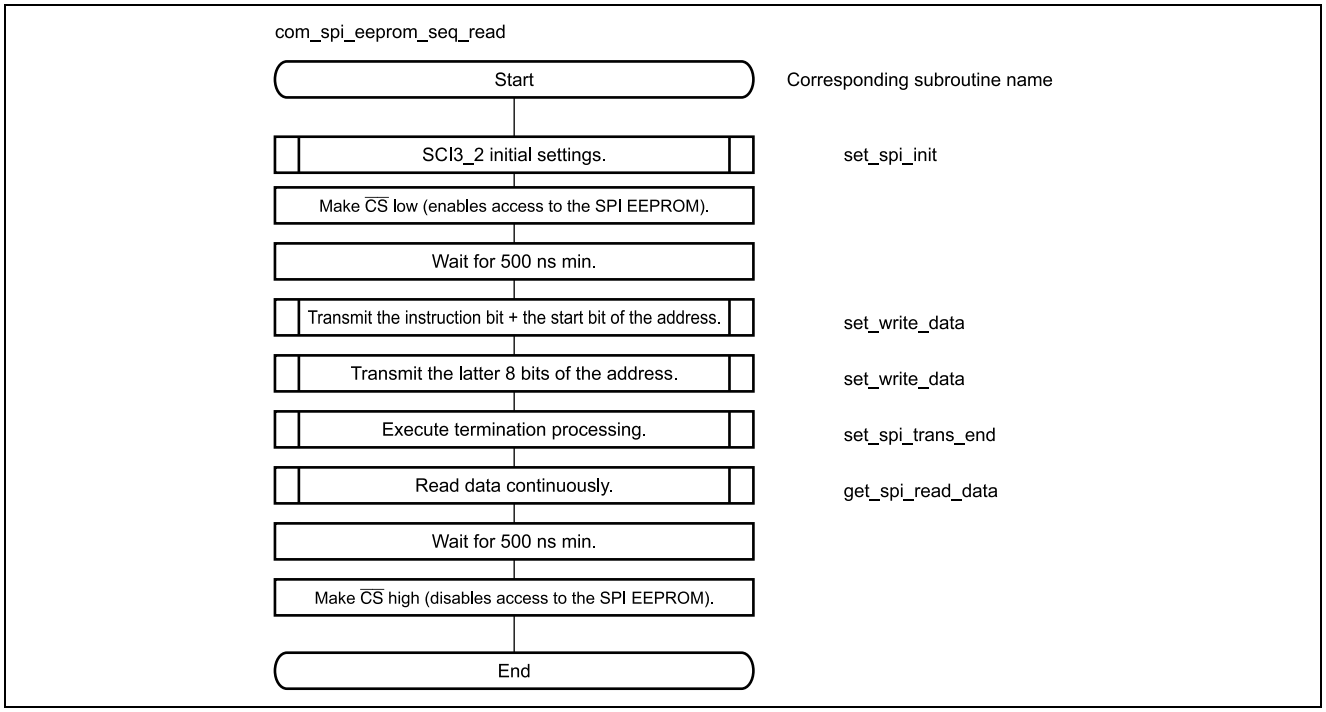
| Name | Summary |
|---|---|
| Timer start register (TSTR) | Starts or stops TCNT operation. |
| Timer mode register (TMDR) | Sets buffer operation and selects synchronous operation. |
| Timer PWM mode register (TPMR) | Sets pins for PWM mode. Not used in this sample program. |
| Timer function control register (TFCR) | Selects the operating mode and output level. Not used in this sample program. |
| Timer output master enable register (TOER) | Enables/disables channel 0 and channel 1 output. |
| Timer output control register (TOCR) | Selects initial output settings before the first compare match occurs. |
| Timer counter (TCNT) | 16-bit read/write register which counts up with the input clock. |
| General registers A, B, C, D (GRA, GRB, GRC, GRD) | GR is a 16-bit read/write register. Each channel has four GR registers, therefore, a total of eight registers are provided. These registers can be used as either output-compare registers or as input-capture registers, according to the TIORA and TIORC settings. |
| Timer control register (TCR) | Selects the TCNT counter clock, edge for an external clock, and counter clear conditions. |
| Timer I/O control register (TIORA) | Selects the functions of the GRA and GRB to be used as output-compare registers or as input-capture registers. |
| Timer status register (TSR) | Indicates the TCNT overflow/underflow generation and GRA/GRB/GRC/GRD compare match or input capture generation. |
| Timer interrupt enable register (TIER) | Enables/disables overflow interrupt requests or GR compare-match/input-capture interrupt. requests. |
| PWM mode output level control register (POCR) | Controls the active level in PWM mode. Not used in this sample program. |

## 3.7 Flowcharts

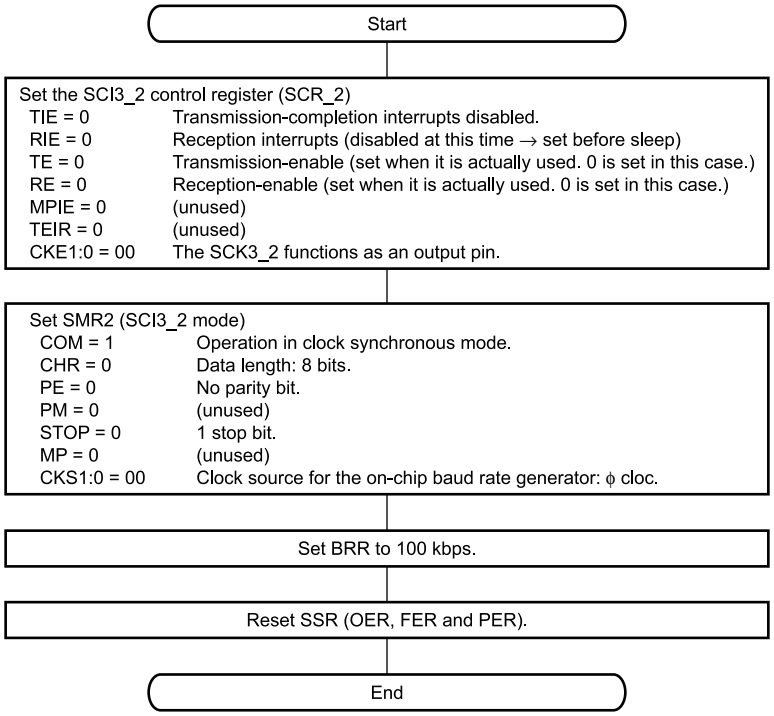



com_spi_eeprom_page_write
Start
Corresponding subroutine name
SCI3_2 initial settings.
set_spi_init
Make CS low (enables access to the SPI EEPROM).
Wait for 500 ns min.
Set write-enable
Set the SPI EEPROM to write-enable.
set_write_data
Execute termination processing.
set_spi_trans_end
Wait for 500 ns min.
Make CS high (disables access to the SPI EEPROM).
Wait for 500 ns min.
Make CS low (enables access to the SPI EEPROM).
Wait for 500 ns min.
Transmit the instruction bit + the start bit of the address.
set_write_data
Data write
Transmit the latter 8 bits of the address.
set_write_data
Transmit write data.
set_write_data
Execute termination processing.
set_spi_trans_end
Wait for 500 ns min.
Make CS high (disables access to the SPI EEPROM).
Wait for 500 ns min.
Make CS low (enables access to the SPI EEPROM).
Wait for 500 ns min.
Reset write-enable
Reset the SPI EEPROM write-enable setting.
set_write_data
Execute termination processing.
set_spi_trans_end
Wait for 500 ns min.
Make CS high (disables access to the SPI EEPROM).
Check write processing completion.
wait_spi_write_end
Stop

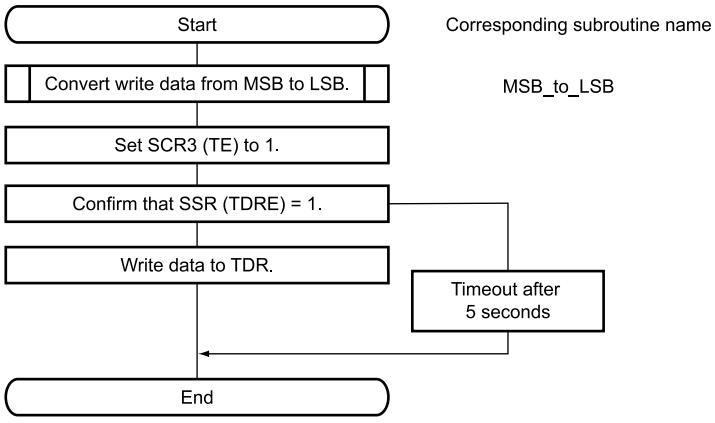

com_spi_eeprom_read
Start
Corresponding subroutine name
SCI3_2 initial settings.
set_spi_init
Make CS low (enables access to the SPI EEPROM).
Wait for 500 ns min.
Transmit the instruction bit + the start bit of the address
set_write_data
Transmit the latter 8 bits of the address.
set_write_data
Execute termination processing.
set_spi_trans_end
Read data.
get_spi_read_data
Wait for 500 ns min.
Make CS high (disables access to the SPI EEPROM).
End


com_spi_eeprom_seq_read
Start
Corresponding subroutine name
SCI3_2 initial settings.
set_spi_init
Make CS low (enables access to the SPI EEPROM).
Wait for 500 ns min.
Transmit the instruction bit + the start bit of the address.
set_write_data
Transmit the latter 8 bits of the address.
set_write_data
Execute termination processing.
set_spi_trans_end
Read data continuously.
get_spi_read_data
Wait for 500 ns min.
Make CS high (disables access to the SPI EEPROM).
End

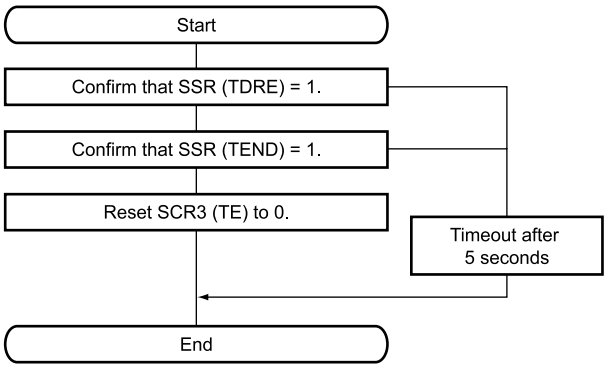
set_spi_init
: Initial settings prior to accessing the SPI.



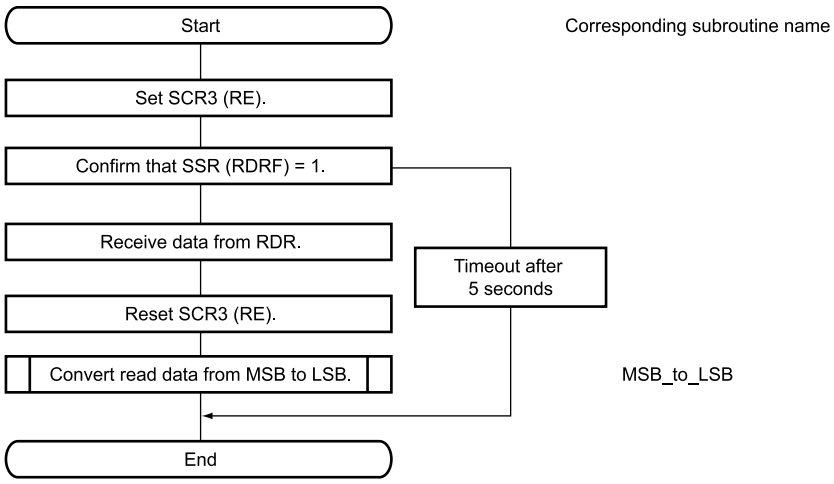

set_write_data (unsigned char write_data)
: Transmits data.
write_data: Write data



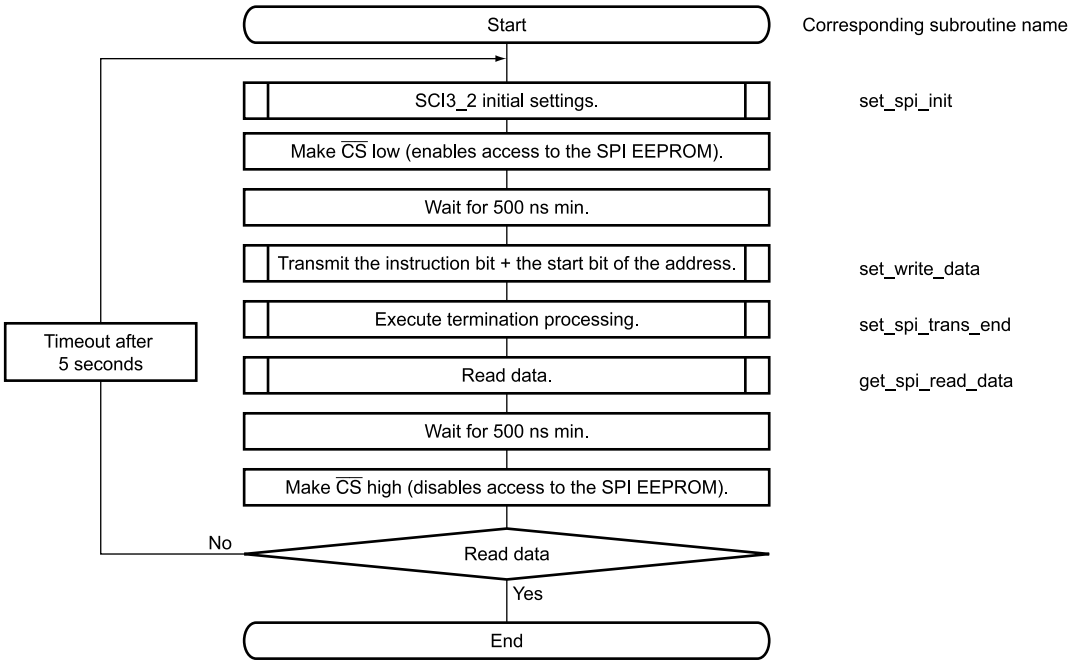
set_spi_trans_end
: Executes data transmission termination processing.




get_spi_read_data (unsigned char *read_data)
: Receives data.
*read_data: Address to store read data.



wait_spi_write_end
: Checks SPI write processing termination.
com_spi_eeprom_read




com_delay (int_delaytime)
: Set desired delay time
delaytime: 1 for approx. 0.5 µs

MSB_to_LSB (unsigned char in_data)
: Reverse the bit order
in_data: MSB data

## 3.8 Program Listing

```
/* ------------------------------------------------------------------------------------------------------------------------ */
/* ------------------------------------------------------------------------------------------------------------------------ */
/* 1. Sample Program 3-A  #define directives ------------------------------------------------------------------------------- */
/* ------------------------------------------------------------------------------------------------------------------------ */
/* ------------------------------------------------------------------------------------------------------------------------ */
/************************************************************************************************************************/
/*   For SPIEEPROM access                                                                                                   */
/************************************************************************************************************************/
#define   SET_WRITE_MODE      0x02
#define   SET_READ_MODE       0x03
#define   RESET_WRITE_ENABLE  0x04
#define   READ_STATUS         0x05
#define   SET_WRITE_ENABLE    0x06

/************************************************************************************************************************/
/*   SPIEEPROM access error code (other than 0)                                                                             */
/************************************************************************************************************************/
#define   SPI_TDRE_TOUT       1
#define   SPI_TEND_TOUT       2
#define   SPI_RDRF_TOUT       3
#define   WRITE_TOUT          4

/* ------------------------------------------------------------------------------------------------------------------------ */
/* ------------------------------------------------------------------------------------------------------------------------ */
/* 2. Sample program 3-B Prototype declaration ------------------------------------------------------------------------------ */
/* ------------------------------------------------------------------------------------------------------------------------ */
/* ------------------------------------------------------------------------------------------------------------------------ */
/************************************************************************************************************************/
/************************************************************************************************************************/
/*   SPI BUS access processing                                                                                              */
/************************************************************************************************************************/
/************************************************************************************************************************/
void com_delay( int delaytime ) ;
void set_spi_init ( ) ;
unsigned char MSB_to_LSB (unsigned char in_data) ;
unsigned int set_write_data (unsigned char write_data);
unsigned int set_spi_trans_end ();
unsigned int wait_spi_write_end () ;
unsigned int get_spi_read_data (unsigned char *read_data);
unsigned int com_spi_eeprom_read ( unsigned int rom_addr , unsigned char *rom_data );
unsigned int com_spi_eeprom_write (unsigned int rom_addr , unsigned char rom_data );
unsigned int com_spi_eeprom_seq_read ( unsigned int rom_addr , unsigned int rom_length , unsigned char *rom_data ) ;
unsigned int com_spi_eeprom_page_write ( unsigned int rom_addr , unsigned int rom_length , unsigned char *rom_data ) ;
```

```
/* ------------------------------------------------------------------------------------------------------------------------ */
/* ------------------------------------------------------------------------------------------------------------------------ */
/* 3. Sample program 3-C Source codes ------------------------------------------------------------------------------------- */
/* ------------------------------------------------------------------------------------------------------------------------ */
/* ------------------------------------------------------------------------------------------------------------------------ */

/* ------------------------------------------------------------------------------------------------------------------------ */
/* 3.1. Setting initialization -------------------------------------------------------------------------------------------- */
/* ------------------------------------------------------------------------------------------------------------------------ */
/* Add the following to the initial settings in the H8 start-up processing                                                  */

    /************************************************************************************************************************/
    /*   PCR1      Defines input or output for the IO port 1.                                                               */
    /*         PCR17     = 1  Unused (defined as an output pin)                                                             */
    /*         PCR16     = 1  Unused (defined as an output pin)                                                             */
    /*         PCR15     = 1  Unused (defined as an output pin)                                                             */
    /*         PCR14     = 1  Unused (defined as an output pin)                                                             */
    /*         PCR12     = 1  Unused (defined as an output pin)                                                             */
    /*         PCR11     = 1  Unused (defined as an output pin)                                                             */
    /*         PCR10     = 1  Used as the CS pin of SPIEEPROM                                                               */
    /************************************************************************************************************************/
    IO.PCR1             = 0xFF ;

    /************************************************************************************************************************/
    /*   PDR1      Specifies the output data of the IO port 1.                                                              */
    /*         PDR17     = 0  Unused (defined as an output pin)                                                             */
    /*         PDR16     = 0  Unused (defined as an output pin)                                                             */
    /*         PDR15     = 0  Unused (defined as an output pin)                                                             */
    /*         PDR14     = 0  Unused (defined as an output pin)                                                             */
    /*         PDR12     = 0  Unused (defined as an output pin)                                                             */
    /*         PDR11     = 0  Unused (defined as an output pin)                                                             */
    /*         PDR10     = 1  Sets CS = high (SPIEEPRON not active).                                                        */
    /************************************************************************************************************************/
    IO.PDR1.BYTE            = 0x01 ;
        /* ##(program note)################################################################################################ */
        /* ## Since the CS pin of SPIEEPROMk is an active low signal, the CS pin should be initialized to high (not active).  ## */
        /* ## The CS level is not guaranteed before this setting,                                                          ## */
        /* ## but the SPIEEPROM cannot be written to illegally because it is write protected.                             ## */
        /* ################################################################################################################ */
```

```
/* ------------------------------------------------------------------------------------------------------------------------ */
/* 3.2 SPIEEPROM access processing ----------------------------------------------------------------------------------------- */
/* ------------------------------------------------------------------------------------------------------------------------ */

/************************************************************************************************************************/
/************************************************************************************************************************/
/************************************************************************************************************************/
/*                                                                                                                      */
/*                                SPI EEPROM control                                                                    */
/*                                                                                                                      */
/************************************************************************************************************************/
/************************************************************************************************************************/
/************************************************************************************************************************/
/************************************************************************************************************************/
/*   1. Module name: com_delay                                                                                          */
/*   2. Function overview: Set delay time as desired                                                                    */
/*   3. History of revisions: REV  Date created/revised    Created/revised by     Revision contents                     */
/*                            000  2002.12.14                    Ueda                    New                             */
/************************************************************************************************************************/
void com_delay( int delaytime )
{
     register int i,a;

     for(i=0;i<delaytime;i++)
          a++;
}

/************************************************************************************************************************/
/*   1. Module name: set_spi_init                                                                                       */
/*   2. Function overview: Sets initial settings prior to SPI access                                                    */
/************************************************************************************************************************/
void set_spi_init( )
{
     /*********************************************************************************************************************/
     /*   SCR3_2          Sets the SCI3_2 control register                                                               */
     /*        TIE        = 0 Transmit end interrupts disabled                                                           */
     /*        RIE        = 0 Receive interrupts (disabled at this time; set before sleep)                               */
     /*        TE         = 0 Transmission enabled (set when it is actually used. 0 is set in this case)                 */
     /*        RE         = 0 Reception enabled (set when it is actually used. 0 is set in this case)                    */
     /*        MPIE       = 0 (Unused)                                                                                   */
     /*        TEIR       = 0 (Unused)                                                                                   */
     /*        CKE1:0     = 00 Uses SCK3_2 as an output pin                                                              */
     /*********************************************************************************************************************/
     SCI3_2.SCR3.BYTE    = 0x00 ;

     /*********************************************************************************************************************/
     /*   SMR_2               Sets SCI3_2 mode                                                                           */
     /*        COM        = 1  Operates in clock synchronous mode                                                        */
     /*        CHR        = 0  Data length: 8 bits                                                                       */
     /*        PE         = 0  No parity bit                                                                             */
     /*        PM         = 0   (Unused)                                                                                 */
     /*        STOP       = 0  1 stop bit                                                                                */
     /*        MP         = 0   (Unused)                                                                                 */
     /*        CKS1:0     = 00 On-chip baud rate generator clock source: ϕ clock                                         */
     /*********************************************************************************************************************/
     SCI3_2.SMR.BYTE     = 0x80 ;
```

```
    /*****************************************************************************************************************************/
    /*   BRR            Set to 100 kbps-                                                                                         */
    /*****************************************************************************************************************************/
    SCI3_2.BRR = 0x27;
        /* ##(program note)############################################################################################### */
        /* ## The value set for BRR should be modified depending on the necessary transfer rate.                        ## */
        /* ## For details, refer to the H8/3687 Hardware Manual.                                                        ## */
        /* ############################################################################################################### */

    /*****************************************************************************************************************************/
    /*   Resets SSR (OER, FER, and PER)                                                                                           */
    /*****************************************************************************************************************************/
    SCI3_2.SSR.BIT.OER = 0 ;                                        /* Overrun error reset                                   */
    SCI3_2.SSR.BIT.FER = 0 ;                                        /* Framing error reset                                   */
    SCI3_2.SSR.BIT.PER = 0 ;                                        /* Parity error reset                                    */

}

/*****************************************************************************************************************************/
/*   1. Module name: MSB_to_LSB                                                                                               */
/*   2. Function overview: Reverses the bit order                                                                             */
/*****************************************************************************************************************************/
unsigned char MSB_to_LSB (unsigned char in_data)
{
    int i ;
    unsigned char out_data ;

    out_data = 0 ;
    for (i=0; i<8; i++){                                            /* Clears the receive buffer                             */

        switch (i){
            case 0 :
                out_data = out_data | ((in_data & 0x01) << 7) ;
                break ;
            case 1 :
                out_data = out_data | ((in_data & 0x02) << 5) ;
                break ;
            case 2 :
                out_data = out_data | ((in_data & 0x04) << 3) ;
                break ;
            case 3 :
                out_data = out_data | ((in_data & 0x08) << 1) ;
                break ;
            case 4 :
                out_data = out_data | ((in_data & 0x10) >> 1) ;
                break ;
            case 5 :
                out_data = out_data | ((in_data & 0x20) >> 3) ;
                break ;
            case 6 :
                out_data = out_data | ((in_data & 0x40) >> 5) ;
                break ;
            case 7 :
                out_data = out_data | ((in_data & 0x80) >> 7) ;
                break ;
        }
    }

    return (out_data) ;
}
```

```
/************************************************************************************************************************/
/*   1. Module name: set_write_data                                                                                     */
/*   2. Function overview: Transmits data                                                                               */
/************************************************************************************************************************/
unsigned int set_write_data (unsigned char write_data)
{
     int ret , Timer_wk , i ;
     unsigned char buf ;

     ret = NORMAL_END ;

     /*******************************************************************************************************************/
     /*   Converts write data from MSB to LSB                                                                           */
     /*******************************************************************************************************************/
     buf = MSB_to_LSB(write_data) ;
          /* ##(program note)################################################################################################ */
          /* ## The SCI interface on the H8 microcomputer treats data with LSB first (inputs/outputs data from bit 0),       ## */
          /* ## while EEPROM in this sample program treats data with MSB first (inputs/outputs data from bit 7).             ## */
          /* ## Therefore, the bit order of transmit data is changed here.                                                   ## */
          /* ################################################################################################################ */

     /*******************************************************************************************************************/
     /*   Sets SCR3 (TE)                                                                                                */
     /*******************************************************************************************************************/
     SCI3_2.SCR3.BIT.TE =     1    ;                                    /* Enables transmission                          */

     /*******************************************************************************************************************/
     /*   Confirms that SSR (TDRE) = 1                                                                                  */
     /*******************************************************************************************************************/
     com_timer.wait_100ms_spi = 50 ;
     while(SCI3_2.SSR.BIT.TDRE == 0){                                   /* Waits until data transfer is possible         */
          Timer_wk = com_timer.wait_100ms_spi ;
          if (Timer_wk == 0){                                           /* Timeout after 5 seconds                       */
               ret = SPI_TDRE_TOUT ;                                    /* Performs no operation even if an error occurs.*/
               goto exit ;
          }
     #ifdef UT
          SCI3_2.SSR.BIT.TDRE = 1 ;
     #endif
     }

     /*******************************************************************************************************************/
     /*   Writes data                                                                                                   */
     /*******************************************************************************************************************/
     SCI3_2.TDR = buf ;                                                 /* Transmits data; this resets SSR (TDRE).       */

exit :
     return (ret) ;

}
```

```
/*****************************************************************************************************************************************/
/*   1. Module name: set_spi_trans_end                                                                                                   */
/*   2. Function overview: Executes data transmit exit processing.                                                                      */
/*****************************************************************************************************************************************/
unsigned int set_spi_trans_end ()
{
     int ret , Timer_wk;

     ret = NORMAL_END ;
     /*****************************************************************************************************************************************/
     /*   Confirms that SSR (TDRE) = 1                                                                                                    */
     /*****************************************************************************************************************************************/
     com_timer.wait_100ms_spi = 50 ;
     while(SCI3_2.SSR.BIT.TDRE == 0){                                    /* Waits until data transfer is possible            */
          Timer_wk = com_timer.wait_100ms_spi ;
          if (Timer_wk == 0){                                            /* Timeout after 5 seconds                          */
               ret = SPI_TDRE_TOUT ;                                     /* Performs no operation even if an error occurs.   */
               goto exit ;
          }
     #ifdef UT
          SCI3_2.SSR.BIT.TDRE = 1 ;
     #endif
     }

     /*****************************************************************************************************************************************/
     /*   Confirms that SSR (TDRE) = 1                                                                                                    */
     /*****************************************************************************************************************************************/
     com_timer.wait_100ms_spi = 50 ;
     while(SCI3_2.SSR.BIT.TEND == 0){                                    /* Waits until data transfer has                    */
          Timer_wk = com_timer.wait_100ms_spi ;
          if (Timer_wk == 0){                                            /* Timeout after 5 seconds                          */
               ret = SPI_TEND_TOUT ;                                     /* Performs no operation even if an error occurs.   */
               goto exit ;
          }
     #ifdef UT
          SCI3_2.SSR.BIT.TEND = 1 ;
     #endif
     }

exit :
     com_delay(10) ;

     /*****************************************************************************************************************************************/
     /*   Resets SCR3 (TE)                                                                                                                */
     /*****************************************************************************************************************************************/
     SCI3_2.SCR3.BIT.TE =     0    ;                                     /* Disables transmission                            */

     return (ret) ;

}
```

```
/*****************************************************************************************************************************************************/
/*   1. Module name: get_spi_read_data                                                                                                               */
/*   2. Function overview: Receives data.                                                                                                            */
/*****************************************************************************************************************************************************/
unsigned int get_spi_read_data (unsigned char *read_data)
{
     int ret , Timer_wk ;
     unsigned char buf ;

     ret = NORMAL_END ;

     /*****************************************************************************************************************************************************/
     /*   Resets SCR3 (RE)                                                                                                                              */
     /*****************************************************************************************************************************************************/
     SCI3_2.SCR3.BIT.RE =     1    ;                                        /* Enables reception                                                    */


     /*****************************************************************************************************************************************************/
     /*   Confirms that SSR (RDRF) = 1                                                                                                                  */
     /*****************************************************************************************************************************************************/
     com_timer.wait_100ms_spi = 50 ;

     while (SCI3_2.SSR.BIT.RDRF == 0){                                     /* Waits for received data (max. 5 seconds)                             */
          Timer_wk = com_timer.wait_100ms_spi ;
          if (Timer_wk == 0){                                              /* Timeout after 5 seconds                                              */
               ret =SPI_RDRF_TOUT ;                                        /* Performs no operation even if an error occurs.                       */
               goto exit ;
          }
          #ifdef UT
               SCI3_2.SSR.BIT.RDRF = 1 ;
          #endif
     }

exit :

     buf = SCI3_2.RDR ;                                                    /* Receives data; this resets SSR (RDRF).                               */

     /*****************************************************************************************************************************************************/
     /*   Resets SCR3 (RE)                                                                                                                              */
     /*****************************************************************************************************************************************************/
     SCI3_2.SCR3.BIT.RE =     0    ;                                        /* Disables receive operation                                           */

     /*****************************************************************************************************************************************************/
     /* Converts write data from MSB first to LSB first                                                                                                 */
     /*****************************************************************************************************************************************************/
     *read_data = MSB_to_LSB(buf) ;
          /* ##(program note)################################################################################################################ */
          /* ## The SCI interface on the H8 microcomputer treats data with LSB first (inputs/outputs data from bit 0),       ## */
          /* ## while EEPROM in this sample program treats data with MSB first (inputs/outputs data from bit 7).             ## */
          /* ## Therefore, the bit order of transmit data is changed here.                                                   ## */
          /* ################################################################################################################ */

     return (ret) ;

}
```

```
/*****************************************************************************************************************************************/
/*   1. Module name: wait_spi_write_end                                                                                                  */
/*   2. Function overview: Checks write completion of SPI                                                                                */
/*****************************************************************************************************************************************/
unsigned int wait_spi_write_end ()
{

     int ret , i;
      unsigned char      status;
     union {
           unsigned int       d_int ;
           unsigned char      d_byte[2];
     } buf;


     ret = NORMAL_END ;

     com_timer.wait_100ms = 50 ;
     do{
          /*****************************************************************************************************************************/
          /*   Initializes SCI3_2                                                                                                      */
          /*****************************************************************************************************************************/
          set_spi_init( ) ;

          /*****************************************************************************************************************************/
          /*   Brings CS to low (enables SPIEEPROM access).                                                                            */
          /*****************************************************************************************************************************/
          IO.PDR1.BYTE = 0x00 ;
          com_delay(10) ;
          /* ##(program note)############################################################################################################ */
          /* ## Inserts a wait here for the purpose of making at least a 500 ns interval from                                      ## */
          /* ## turning on CS to starting the data transfer.                                                                       ## */
          /* ## Since the waiting time differs depending on the device to be controlled,                                           ## */
          /* ## refer to the data sheet to set the appropriate waiting time.                                                       ## */
          /* ############################################################################################################################ */

          /*****************************************************************************************************************************/
          /*****************************************************************************************************************************/
          /*   Reads data.                                                                                                             */
          /*****************************************************************************************************************************/
          /*****************************************************************************************************************************/

          /*****************************************************************************************************************************/
          /*   Transmits instruction data (read status).                                                                               */
          /*****************************************************************************************************************************/
          ret = set_write_data (READ_STATUS) ;
          if (ret !=0) { goto exit ;}

          /*****************************************************************************************************************************/
          /*   Executes exit processing.                                                                                               */
          /*****************************************************************************************************************************/
          ret = set_spi_trans_end () ;
          if (ret !=0) { goto exit ;}

          /*****************************************************************************************************************************/
          /*   Reads data.                                                                                                             */
          /*****************************************************************************************************************************/
          ret = get_spi_read_data (&status) ;
          if (ret !=0) { goto exit ;}
```

```
        /*****************************************************************************************************************/
        /*   Brings CS to high (disables SPIEEPROM access).                                                              */
        /*****************************************************************************************************************/
        com_delay(10) ;
        /* ##(program note)############################################################################################ */
        /* ## Inserts a wait here for the purpose of making at least a 500 ns interval from                          ## */
        /* ## terminating the data transfer to turning off CS.                                                       ## */
        /* ## Since the waiting time differs depending on the device to be controlled,                               ## */
        /* ## refer to the data sheet to set the appropriate waiting time.                                           ## */
        /* ############################################################################################################ */
        IO.PDR1.BYTE = 0x01 ;

        #ifdef UT
             status = 0x01 ;
        #endif

        if (com_timer.wait_100ms == 0){                          /* Timeout after 5 seconds                         */
             ret = WRITE_TOUT;                                   /* Abnormal termination (timeout)                  */
             goto exit ;
        }

    } while ((status & 0x01) == 1) ;                             /* Write is in progress.                           */

exit :                                                           /* Error processing                                */
    /*****************************************************************************************************************/
    /*   Brings CS to high (disables SPIEEPROM access).                                                              */
    /*****************************************************************************************************************/
    com_delay(10) ;
        /* ##(program note)############################################################################################ */
        /* ## Inserts a wait here for the purpose of making at least a 500 ns interval from                          ## */
        /* ## terminating the data transfer to turning off CS.                                                       ## */
        /* ## Since the waiting time differs depending on the device to be controlled,                               ## */
        /* ## refer to the data sheet to set the appropriate waiting time.                                           ## */
        /* ############################################################################################################ */
    IO.PDR1.BYTE = 0x01 ;

    return (ret) ;

}

/*****************************************************************************************************************************/
/*   1. Module name: com_spi_eeprom_read                                                                                     */
/*   2. Function overview: Reads 1-byte data from SPIEEPROM.                                                                 */
/*****************************************************************************************************************************/
unsigned int com_spi_eeprom_read ( unsigned int rom_addr , unsigned char *rom_data )
{
    int ret ;
    union {
          unsigned int       d_int ;
          unsigned char      d_byte[2];
    } buf;


    ret = NORMAL_END ;

    /*****************************************************************************************************************/
    /*   Initializes SCI3_2.                                                                                         */
    /*****************************************************************************************************************/
    set_spi_init( ) ;
```

```
    /*****************************************************************************************************************************/
    /*   Brings CS to low (enables SPIEEPROM access).                                                                            */
    /*****************************************************************************************************************************/
    IO.PDR1.BYTE = 0x00 ;
    com_delay(10) ;
        /* ##(program note)########################################################################################################## */
        /* ## Inserts a wait here for the purpose of making at least a 500 ns interval from                                    ## */
        /* ## turning on CS to starting the data transfer.                                                                     ## */
        /* ## Since the waiting time differs depending on the device to be controlled,                                         ## */
        /* ## refer to the data sheet to set the appropriate waiting time.                                                     ## */
        /* ########################################################################################################################## */

    /*****************************************************************************************************************************/
    /*****************************************************************************************************************************/
    /*   Reads data                                                                                                              */
    /*****************************************************************************************************************************/
    /*****************************************************************************************************************************/

    /*****************************************************************************************************************************/
    /*   Transmits the instruction data + address start bit                                                                      */
    /*****************************************************************************************************************************/
    buf.d_int = rom_addr ;
    buf.d_byte[0] = (buf.d_byte[0] && 0x01) << 3 ;
    buf.d_byte[0] |= SET_READ_MODE ;

    ret = set_write_data (buf.d_byte[0]) ;
    if (ret !=0) { goto exit ;}
    /*****************************************************************************************************************************/
    /*   Transmits the latter 8 bits of the address                                                                              */
    /*****************************************************************************************************************************/
    ret = set_write_data (buf.d_byte[1]) ;
    if (ret !=0) { goto exit ;}

    /*****************************************************************************************************************************/
    /*   Executes exit processing.                                                                                               */
    /*****************************************************************************************************************************/
    ret = set_spi_trans_end () ;
    if (ret !=0) { goto exit ;}

    /*****************************************************************************************************************************/
    /*   Reads data                                                                                                              */
    /*****************************************************************************************************************************/
    ret = get_spi_read_data (&buf.d_byte[0]) ;
    if (ret !=0) { goto exit ;}


    *rom_data = buf.d_byte[0] ;

exit :
    /*****************************************************************************************************************************/
    /*   Brings CS to high (disables SPIEEPROM access).                                                                          */
    /*****************************************************************************************************************************/
    com_delay(10) ;
        /* ##(program note)########################################################################################################## */
        /* ## Inserts a wait here for the purpose of making at least a 500 ns interval from                                    ## */
        /* ## terminating the data transfer to turning off CS.                                                                 ## */
        /* ## Since the waiting time differs depending on the device to be controlled,                                         ## */
        /* ## refer to the data sheet to set the appropriate waiting time.                                                     ## */
        /* ########################################################################################################################## */

    IO.PDR1.BYTE = 0x01 ;

    return (ret);

}
```

```
/*****************************************************************************************************************************************************/
/*   1. Module name: com_spi_eeprom_seq_read                                                                                                         */
/*   2. Function overview: Reads 1-byte data from SPIEEPROM.                                                                                         */
/*****************************************************************************************************************************************************/
unsigned int com_spi_eeprom_seq_read ( unsigned int rom_addr , unsigned int rom_length , unsigned char *rom_data )
{
     int ret , i;
     union {
           unsigned int       d_int ;
           unsigned char      d_byte[2];
     } buf;


     ret = NORMAL_END ;

     /*****************************************************************************************************************************************************/
     /*   Initializes SCI3_2.                                                                                                                           */
     /*****************************************************************************************************************************************************/
     set_spi_init( ) ;


     /*****************************************************************************************************************************************************/
     /*   Brings CS to low (enables SPIEEPROM access).                                                                                                  */
     /*****************************************************************************************************************************************************/
     IO.PDR1.BYTE = 0x00 ;
     com_delay(10) ;
          /* ##(program note)################################################################################################################ */
          /* ## Inserts a wait here for the purpose of making at least a 500 ns interval from                                                ## */
          /* ## turning on CS to starting the data transfer.                                                                                  ## */
          /* ## Since the waiting time differs depending on the device to be controlled,                                                      ## */
          /* ## refer to the data sheet to set the appropriate waiting time.                                                                  ## */
          /* ################################################################################################################################ */

     /*****************************************************************************************************************************************************/
     /*****************************************************************************************************************************************************/
     /*   Reads data                                                                                                                                    */
     /*****************************************************************************************************************************************************/
     /*****************************************************************************************************************************************************/

     /*****************************************************************************************************************************************************/
     /*   Transmits the instruction data + address start bit                                                                                            */
     /*****************************************************************************************************************************************************/
     buf.d_int = rom_addr ;
     buf.d_byte[0] = (buf.d_byte[0] && 0x01) << 3 ;
     buf.d_byte[0] |= SET_READ_MODE ;

     ret = set_write_data (buf.d_byte[0]) ;
     if (ret !=0) { goto exit ;}

     /*****************************************************************************************************************************************************/
     /*   Transmits the latter 8 bits of the address                                                                                                    */
     /*****************************************************************************************************************************************************/
     ret = set_write_data (buf.d_byte[1]) ;
     if (ret !=0) { goto exit ;}

     /*****************************************************************************************************************************************************/
     /*   Executes exit processing.                                                                                                                     */
     /*****************************************************************************************************************************************************/
     ret = set_spi_trans_end () ;
     if (ret !=0) { goto exit ;}
```

```
      /*****************************************************************************************************************************************/
      /*   Reads data continuously.                                                                                                            */
      /*****************************************************************************************************************************************/
      for (i=0; i< (rom_length) ; i++){
           ret = get_spi_read_data (&buf.d_byte[0]) ;
           if (ret !=0) { goto exit ;}

           *rom_data = buf.d_byte[0] ;
           *rom_data ++ ;
      }

exit :

      /*****************************************************************************************************************************************/
      /*   Brings CS to high (disables SPIEEPROM access).                                                                                      */
      /*****************************************************************************************************************************************/
      com_delay(10) ;
           /* ##(program note)################################################################################################################ */
           /* ## Inserts a wait here for the purpose of making at least a 500 ns interval from                                            ## */
           /* ## terminating the data transfer to turning off CS.                                                                         ## */
           /* ## Since the waiting time differs depending on the device to be controlled,                                                 ## */
           /* ## refer to the data sheet to set the appropriate waiting time.                                                             ## */
           /* ################################################################################################################################ */

      IO.PDR1.BYTE = 0x01 ;

      return (ret);

}
/*****************************************************************************************************************************************/
/*   1. Module name: com_spi_eeprom_write                                                                                                    */
/*   2. Function overview: Writes 1-byte data to SPIEEPROM.                                                                                  */
/*****************************************************************************************************************************************/
unsigned int com_spi_eeprom_write (unsigned int rom_addr , unsigned char rom_data )
{
      int ret ;
      union {
            unsigned int       d_int ;
            unsigned char      d_byte[2];
      } buf;


      ret = NORMAL_END ;

      /*****************************************************************************************************************************************/
      /*   Initializes SCI3_2.                                                                                                                 */
      /*****************************************************************************************************************************************/
      set_spi_init( ) ;
```

```
/*****************************************************************************************************************************/
/*****************************************************************************************************************************/
/*   Cancels SPI EEPROM write enable                                                                                         */
/*****************************************************************************************************************************/
/*****************************************************************************************************************************/
/*****************************************************************************************************************************/
/*   Brings CS to low (enables SPIEEPROM access).                                                                            */
/*****************************************************************************************************************************/
IO.PDR1.BYTE = 0x00 ;
com_delay(10) ;
     /* ##(program note)############################################################################################### */
     /* ## Inserts a wait here for the purpose of making at least a 500 ns interval from                                ## */
     /* ## turning on CS to starting the data transfer.                                                                 ## */
     /* ## Since the waiting time differs depending on the device to be controlled,                                     ## */
     /* ## refer to the data sheet to set the appropriate waiting time.                                                  ## */
     /* ################################################################################################################ */

/*****************************************************************************************************************************/
/*   Specifies the SPI EEPROM write enable.                                                                                  */
/*****************************************************************************************************************************/
ret = set_write_data (SET_WRITE_ENABLE) ;
if (ret !=0) { goto exit ;}


/*****************************************************************************************************************************/
/*   Executes exit processing.                                                                                               */
/*****************************************************************************************************************************/
ret = set_spi_trans_end () ;
if (ret !=0) { goto exit ;}

/*****************************************************************************************************************************/
/*   Brings CS to high (disables SPIEEPROM access).                                                                          */
/*****************************************************************************************************************************/
com_delay(10) ;
     /* ##(program note)############################################################################################### */
     /* ## Inserts a wait here for the purpose of making at least a 500 ns interval from                              ## */
     /* ## terminating the data transfer to turning off CS.                                                           ## */
     /* ## Since the waiting time differs depending on the device to be controlled,                                   ## */
     /* ## refer to the data sheet to set the appropriate waiting time.                                               ## */
     /* ################################################################################################################ */

IO.PDR1.BYTE = 0x01 ;


/*****************************************************************************************************************************/
/*****************************************************************************************************************************/
/*   Writes data.                                                                                                            */
/*****************************************************************************************************************************/
/*****************************************************************************************************************************/
/*****************************************************************************************************************************/
/*   Brings CS to low (enables SPIEEPROM access).                                                                            */
/*****************************************************************************************************************************/
com_delay(10) ;
     /* ##(program note)############################################################################################### */
     /* ## Inserts a wait here for the purpose of making at least a 500 ns interval from                              ## */
     /* ## turning off CS to turning it on again.                                                                     ## */
     /* ## Since the waiting time differs depending on the device to be controlled,                                   ## */
     /* ## refer to the data sheet to set the appropriate waiting time.                                               ## */
     /* ################################################################################################################ */
```

```
IO.PDR1.BYTE = 0x00 ;
     /* ##(program note)################################################################################################ */
     /* ## Inserts a wait here for the purpose of making at least a 500 ns interval from                              ## */
     /* ## turning on CS to starting the data transfer.                                                               ## */
     /* ## Since the waiting time differs depending on the device to be controlled,                                   ## */
     /* ## refer to the data sheet to set the appropriate waiting time.                                               ## */
     /* ################################################################################################################ */
com_delay(10) ;

/*****************************************************************************************************************************/
/*   Transmits the instruction data + address start bit                                                                      */
/*****************************************************************************************************************************/
buf.d_int = rom_addr ;
buf.d_byte[0] = (buf.d_byte[0] && 0x01) << 3 ;
buf.d_byte[0] |= SET_WRITE_MODE ;

ret = set_write_data (buf.d_byte[0]) ;
if (ret !=0) { goto exit ;}

/*****************************************************************************************************************************/
/*   Transmits the latter 8 bits of the address                                                                              */
/*****************************************************************************************************************************/
ret = set_write_data (buf.d_byte[1]) ;
if (ret !=0) { goto exit ;}

/*****************************************************************************************************************************/
/*   Transmits write data                                                                                                    */
/*****************************************************************************************************************************/
ret = set_write_data (rom_data) ;
if (ret !=0) { goto exit ;}

/*****************************************************************************************************************************/
/*   Executes exit processing.                                                                                               */
/*****************************************************************************************************************************/
ret = set_spi_trans_end () ;
if (ret !=0) { goto exit ;}

/*****************************************************************************************************************************/
/*   Brings CS to high (disables SPIEEPROM access).                                                                          */
/*****************************************************************************************************************************/
com_delay(10) ;
IO.PDR1.BYTE = 0x01 ;

/*****************************************************************************************************************************/
/*****************************************************************************************************************************/
/*   Cancels SPI EEPROM write enable                                                                                         */
/*****************************************************************************************************************************/
/*****************************************************************************************************************************/
/*****************************************************************************************************************************/
/*   Brings CS to low (enables SPIEEPROM access).                                                                            */
/*****************************************************************************************************************************/
com_delay(10) ;
     /* ##(program note)################################################################################################ */
     /* ## Inserts a wait here for the purpose of making at least a 500 ns interval from                              ## */
     /* ## turning off CS to turning it on again.                                                                     ## */
     /* ## Since the waiting time differs depending on the device to be controlled,                                   ## */
     /* ## refer to the data sheet to set the appropriate waiting time.                                               ## */
     /* ################################################################################################################ */

IO.PDR1.BYTE = 0x00 ;
```

```
    com_delay(10) ;
        /* ##(program note)################################################################################################ */
        /* ## Inserts a wait here for the purpose of making at least a 500 ns interval from                             ## */
        /* ## turning on CS to starting the data transfer.                                                              ## */
        /* ## Since the waiting time differs depending on the device to be controlled,                                  ## */
        /* ## refer to the data sheet to set the appropriate waiting time.                                              ## */
        /* ################################################################################################################ */

    /*****************************************************************************************************************/
    /*   Cancels SPI EEPROM write enable                                                                             */
    /*****************************************************************************************************************/
    ret = set_write_data (RESET_WRITE_ENABLE) ;
    if (ret !=0) { goto exit ;}

    /*****************************************************************************************************************/
    /*   Executes exit processing.                                                                                   */
    /*****************************************************************************************************************/
    ret = set_spi_trans_end () ;
    if (ret !=0) { goto exit ;}

    /*****************************************************************************************************************/
    /*   Brings CS to high (disables SPIEEPROM access).                                                              */
    /*****************************************************************************************************************/
    com_delay(10) ;
        /* ##(program note)################################################################################################ */
        /* ## Inserts a wait here for the purpose of making at least a 500 ns interval from                             ## */
        /* ## terminating the data transfer to turning off CS.                                                          ## */
        /* ## Since the waiting time differs depending on the device to be controlled,                                  ## */
        /* ## refer to the data sheet to set the appropriate waiting time.                                              ## */
        /* ################################################################################################################ */

    IO.PDR1.BYTE = 0x01 ;

    /*****************************************************************************************************************/
    /*   Checks write completion.                                                                                    */
    /*****************************************************************************************************************/
    ret = wait_spi_write_end () ;
        if (ret !=0) { goto exit ;}
        /* ##(program note)################################################################################################ */
        /* ## SPIEEPROM starts write operation by CS = high. The write completion is checked by checking the SPIEEPROM     ## */
        /* ## internal status register since the write operation takes some time                                         ## */
        /* ################################################################################################################ */

    return (ret);

exit :                                                          /* Error processing                                 */
    /*****************************************************************************************************************/
    /*   Brings CS to high (disables SPIEEPROM access).                                                              */
    /*****************************************************************************************************************/
    com_delay(10) ;
        /* ##(program note)################################################################################################ */
        /* ## Inserts a wait here for the purpose of making at least a 500 ns interval from                             ## */
        /* ## terminating the data transfer to turning off CS.                                                          ## */
        /* ## Since the waiting time differs depending on the device to be controlled,                                  ## */
        /* ## refer to the data sheet to set the appropriate waiting time.                                              ## */
        /* ################################################################################################################ */

    IO.PDR1.BYTE = 0x01 ;

    return (ret);

}
```

```
/*****************************************************************************************************************************************************/
/*   1. Module name: com_spi_eeprom_page_write                                                                                                       */
/*   2. Function overview: Writes 4-byte data to SPIEEPROM.                                                                                          */
/*****************************************************************************************************************************************************/
unsigned int com_spi_eeprom_page_write ( unsigned int rom_addr , unsigned int rom_length , unsigned char *rom_data )
{
     int ret , i ;
     union {
           unsigned int       d_int ;
           unsigned char      d_byte[2];
     } buf;

     union {
           unsigned long      d_long ;
           unsigned char      d_byte[4];
     } write_data;


     ret = NORMAL_END ;
     /*****************************************************************************************************************************************************/
     /*   Initializes SCI3_2.                                                                                                                             */
     /*****************************************************************************************************************************************************/
     set_spi_init( ) ;

     /*****************************************************************************************************************************************************/
     /*****************************************************************************************************************************************************/
     /*   Cancels SPI EEPROM write enable                                                                                                                 */
     /*****************************************************************************************************************************************************/
     /*****************************************************************************************************************************************************/
     /*****************************************************************************************************************************************************/
     /*   Brings CS to low (enables SPIEEPROM access).                                                                                                    */
     /*****************************************************************************************************************************************************/
     IO.PDR1.BYTE = 0x00 ;
     com_delay(10) ;
          /* ##(program note)################################################################################################################ */
          /* ## Inserts a wait here for the purpose of making at least a 500 ns interval from                                                  ## */
          /* ## turning on CS to starting the data transfer.                                                                                   ## */
          /* ## Since the waiting time differs depending on the device to be controlled,                                                       ## */
          /* ## refer to the data sheet to set the appropriate waiting time.                                                                   ## */
          /* ################################################################################################################################ */


     /*****************************************************************************************************************************************************/
     /*   Cancels SPI EEPROM write enable                                                                                                                 */
     /*****************************************************************************************************************************************************/
     ret = set_write_data (SET_WRITE_ENABLE) ;
     if (ret !=0) { goto exit ;}

     /*****************************************************************************************************************************************************/
     /*   Executes exit processing.                                                                                                                       */
     /*****************************************************************************************************************************************************/
     ret = set_spi_trans_end () ;
     if (ret !=0) { goto exit ;}
```

```
/*****************************************************************************************************************************/
/*   Brings CS to high (disables SPIEEPROM access).                                                                          */
/*****************************************************************************************************************************/
com_delay(10) ;
     /* ##(program note)################################################################################################## */
     /* ## Inserts a wait here for the purpose of making at least a 500 ns interval from                                ## */
     /* ## terminating the data transfer to turning off CS.                                                             ## */
     /* ## Since the waiting time differs depending on the device to be controlled,                                     ## */
     /* ## refer to the data sheet to set the appropriate waiting time.                                                  ## */
     /* ################################################################################################################## */


IO.PDR1.BYTE = 0x01 ;


/*****************************************************************************************************************************/
/*****************************************************************************************************************************/
/*   Writes data.                                                                                                            */
/*****************************************************************************************************************************/
/*****************************************************************************************************************************/
/*****************************************************************************************************************************/
/*   Brings CS to low (enables SPIEEPROM access).                                                                            */
/*****************************************************************************************************************************/
com_delay(10) ;
     /* ##(program note)################################################################################################## */
     /* ## Inserts a wait here for the purpose of making at least a 500 ns interval from                                ## */
     /* ## turning off CS to turning it on again.                                                                       ## */
     /* ## Since the waiting time differs depending on the device to be controlled,                                     ## */
     /* ## refer to the data sheet to set the appropriate waiting time.                                                  ## */
     /* ################################################################################################################## */
IO.PDR1.BYTE = 0x00 ;
com_delay(10) ;
     /* ##(program note)################################################################################################## */
     /* ## Inserts a wait here for the purpose of making at least a 500 ns interval from                                ## */
     /* ## turning on CS to starting the data transfer.                                                                  ## */
     /* ## Since the waiting time differs depending on the device to be controlled,                                     ## */
     /* ## refer to the data sheet to set the appropriate waiting time.                                                  ## */
     /* ################################################################################################################## */


/*****************************************************************************************************************************/
/*   Transmits the instruction data + address start bit                                                                      */
/*****************************************************************************************************************************/
buf.d_int = rom_addr ;
buf.d_byte[0] = (buf.d_byte[0] && 0x01) << 3 ;
buf.d_byte[0] |= SET_WRITE_MODE ;

ret = set_write_data (buf.d_byte[0]) ;
if (ret !=0) { goto exit ;}

/*****************************************************************************************************************************/
/*   Transmits the latter 8 bits of the address                                                                              */
/*****************************************************************************************************************************/
ret = set_write_data (buf.d_byte[1]) ;
if (ret !=0) { goto exit ;}
```

```
/*****************************************************************************************************************************/
/*   Transmits write data                                                                                                    */
/*****************************************************************************************************************************/
for (i=0; i< rom_length ; i++){
     buf.d_byte[0] = *rom_data  ;
     ret = set_write_data (buf.d_byte[0]) ;
          if (ret !=0) { goto exit ;}
     *rom_data ++ ;
}

/*****************************************************************************************************************************/
/*   Executes exit processing.                                                                                               */
/*****************************************************************************************************************************/
ret = set_spi_trans_end () ;
if (ret !=0) { goto exit ;}

/*****************************************************************************************************************************/
/*   Brings CS to high (disables SPIEEPROM access).                                                                          */
/*****************************************************************************************************************************/
com_delay(10) ;
     /* ##(program note)############################################################################################### */
     /* ## Inserts a wait here for the purpose of making at least a 500 ns interval from                                ## */
     /* ## terminating the data transfer to turning off CS.                                                             ## */
     /* ## Since the waiting time differs depending on the device to be controlled,                                     ## */
     /* ## refer to the data sheet to set the appropriate waiting time.                                                  ## */
     /* ############################################################################################################### */


IO.PDR1.BYTE = 0x01 ;

/*****************************************************************************************************************************/
/*****************************************************************************************************************************/
/*   Specifies SPI EEPROM write enable.                                                                                      */
/*****************************************************************************************************************************/
/*****************************************************************************************************************************/
/*****************************************************************************************************************************/
/*   Brings CS to low (enables SPIEEPROM access).                                                                            */
/*****************************************************************************************************************************/
com_delay(10) ;
     /* ##(program note)############################################################################################### */
     /* ## Inserts a wait here for the purpose of making at least a 500 ns interval from                                ## */
     /* ## turning off CS to turning it on again.                                                                       ## */
     /* ## Since the waiting time differs depending on the device to be controlled,                                     ## */
     /* ## refer to the data sheet to set the appropriate waiting time.                                                  ## */
     /* ############################################################################################################### */
IO.PDR1.BYTE = 0x00 ;
com_delay(10) ;
     /* ##(program note)############################################################################################### */
     /* ## Inserts a wait here for the purpose of making at least a 500 ns interval from                                ## */
     /* ## turning on CS to starting the data transfer.                                                                  ## */
     /* ## Since the waiting time differs depending on the device to be controlled,                                     ## */
     /* ## refer to the data sheet to set the appropriate waiting time.                                                  ## */
     /* ############################################################################################################### */


/*****************************************************************************************************************************/
/*   Cancels SPI EEPROM write enable                                                                                         */
/*****************************************************************************************************************************/
ret = set_write_data (RESET_WRITE_ENABLE) ;
if (ret !=0) { goto exit ;}
```

```
    /*****************************************************************************************************************************/
    /*   Executes exit processing.                                                                                               */
    /*****************************************************************************************************************************/
    ret = set_spi_trans_end () ;
    if (ret !=0) { goto exit ;}

    /*****************************************************************************************************************************/
    /*   Brings CS to high (disables SPIEEPROM access).                                                                          */
    /*****************************************************************************************************************************/
    com_delay(10) ;
        /* ##(program note)########################################################################################################## */
        /* ## Inserts a wait here for the purpose of making at least a 500 ns interval from                                    ## */
        /* ## terminating the data transfer to turning off CS.                                                                  ## */
        /* ## Since the waiting time differs depending on the device to be controlled,                                          ## */
        /* ## refer to the data sheet to set the appropriate waiting time.                                                      ## */
        /* ########################################################################################################################## */

    IO.PDR1.BYTE = 0x01 ;

    /*****************************************************************************************************************************/
    /*   Checks write completion.                                                                                                */
    /*****************************************************************************************************************************/
    ret = wait_spi_write_end () ;
        if (ret !=0) { goto exit ;}
        /* ##(program note)########################################################################################################## */
        /* ## SPIEEPROM starts write operation by CS = high. The write completion is checked by checking the SPIEEPROM     ## */
        /* ## internal status register since the write operation takes some time                                                 ## */
        /* ########################################################################################################################## */

    return (ret);

exit :                                                          /* Error processing                                            */
    /*****************************************************************************************************************************/
    /*   Brings CS to high (disables SPIEEPROM access).                                                                          */
    /*****************************************************************************************************************************/
    com_delay(10) ;
        /* ##(program note)########################################################################################################## */
        /* ## Inserts a wait here for the purpose of making at least a 500 ns interval from                                    ## */
        /* ## terminating the data transfer to turning off CS.                                                                  ## */
        /* ## Since the waiting time differs depending on the device to be controlled,                                          ## */
        /* ## refer to the data sheet to set the appropriate waiting time.                                                      ## */
        /* ########################################################################################################################## */

    IO.PDR1.BYTE = 0x01 ;

    return (ret);

}
```

```
/* ---------------------------------------------------------------------------------------------------------------------- */
/* ---------------------------------------------------------------------------------------------------------------------- */
/* 4. Sample Program 3-D TimerZ Processing ------------------------------------------------------------------------------ */
/* ---------------------------------------------------------------------------------------------------------------------- */
/* ---------------------------------------------------------------------------------------------------------------------- */

/* ---------------------------------------------------------------------------------------------------------------------- */
/* 4.1 Addition of reset vectors ---------------------------------------------------------------------------------------- */
/* ---------------------------------------------------------------------------------------------------------------------- */
/*   Set the jump destination to h8_timerz.                                                                               */

/* ---------------------------------------------------------------------------------------------------------------------- */
/* 4.2 Common variable definitions for TimerZ --------------------------------------------------------------------------- */
/* ---------------------------------------------------------------------------------------------------------------------- */
     struct    {
          int counter;                                          /* 100 ms counter                                      */
          int wait_10ms;                                        /* Sets a wait time of 10 ms                           */
          int wait_100ms;                                       /* Sets the wait time in 100 ms units (common)         */
          int wait_100ms_scan;                                  /* Sets the wait time in 100 ms units (for I2C)        */
     }com_timer;

/* ---------------------------------------------------------------------------------------------------------------------- */
/* 4.3 TimerZ initial settings ------------------------------------------------------------------------------------------ */
/* ---------------------------------------------------------------------------------------------------------------------- */
     /* ################################################################################################################## */
     /* ################################################################################################################## */
     /*                                                                                                                    */
     /*        Sets TimerZ                                                                                                 */
     /*                                                                                                                    */
     /* ################################################################################################################## */
     /* ################################################################################################################## */
     /**********************************************************************************************************************/
     /*   Sets TimerZ initial settings                                                                                     */
     /**********************************************************************************************************************/
     TZ.TSTR.BYTE = 0x00 ;
     TZ.TMDR.BYTE = 0x00 ;
     TZ.TPMR.BYTE = 0x00 ;
     TZ.TFCR.BYTE = 0x00 ;
     TZ.TOER.BYTE = 0xFF ;
     TZ.TOCR.BYTE = 0x00 ;

     TZ0.TCR.BYTE   = 0x23 ;
                                                           /* Clears the counter when a GRA compare matchoccurs. */
                                                           /* CKEG[1:0] = 00 Counts at the rising edge           */
                                                           /* TPSC[2:0] = 011 Counts using internal clock ϕ/8    */
     TZ0.TIORA.BYTE = 0x00 ;
                                                           /* IOA[2:0] = 000 RA is used as the output compare register    */
     TZ0.TIER.BYTE  = 0x01 ;
                                                           /* Enables MFA                                        */

     TZ0.GRA        = 20000 ;                              /* Issues an interrupt every 10 ms                    */
          /* ##(program note)############################################################################################## */
          /* ## The set values differ depending on the operating frequency of the microcomputer.                       ## */
          /* ## Refer to the H8/3687 Hardware Manual.                                                                   ## */
          /* ################################################################################################################## */

     TZ0.TCNT       = 0 ;                                  /* Clears the timer counter                           */
```

```
     /************************************************************************************************************************/
     /*   Starts timerZ                                                                                                      */
     /************************************************************************************************************************/
     TZ.TSTR.BYTE   = 0x01 ;                                          /* timer start                                       */
                                                                      /* STR0 = 1 TCNT_0□start                             */


/* ------------------------------------------------------------------------------------------------------------------------ */
/* 4.4 TimerZ interrupt processing ---------------------------------------------------------------------------------------- */
/* ------------------------------------------------------------------------------------------------------------------------ */
/************************************************************************************************************************/
/*   1. Module name: h8_TimerZ                                                                                          */
/*   Function overview: Interval timer processing every 10 msec                                                         */
/*   3. History of revisions: REV  Date created/revised    Created/revised by     Revision contents                     */
/*           000  2002.02.11         Ueda            New                                                               */
/************************************************************************************************************************/
#pragma interrupt( h8_timerz )
void h8_timerz( void )
{

     /************************************************************************************************************************/
     /*   Clears the source                                                                                                  */
     /************************************************************************************************************************/
     com_global.dummy = TZ0.TSR.BYTE;                                 /* dummy read                                        */

     TZ0.TSR.BIT.IMFA = 0;                                            /* IMFA clear                                        */

     /************************************************************************************************************************/
     /*   -1 in units of 10 ms
     */
     /************************************************************************************************************************/
     if( com_timer.wait_10ms>0 )
          com_timer.wait_10ms --;

     /************************************************************************************************************************/
     /*   Increments the counter                                                                                             */
     /************************************************************************************************************************/
     com_timer.counter++;
     if( com_timer.counter >= 10 ){
          /************************************************************************************************************************/
          /*   -1 in units of 100 ms                                                                                          */
          /************************************************************************************************************************/
          if( com_timer.wait_100ms>0 )
               com_timer.wait_100ms --;
          if( com_timer.wait_100ms_scan>0 )
               com_timer.wait_100ms_scan --;

          com_timer.counter = 0;
     }

}
```

## 4. Reference Documents

- H8/3687 Group Hardware Manual (published by Renesas Technology Corp.)
- X25043/45 Application Notes (published by Xicor, Inc.)

## Revision Record

| | | Description | |
|---|---|---|---|
| Rev. | Date | Page | Summary |
| 1.00 | Sep.29.03 | — | First edition issued |
| | | | |
| | | | |
| | | | |
| | | | |

## Keep safety first in your circuit designs!

1. Renesas Technology Corporation puts the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage.
   Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of nonflammable material or (iii) prevention against any malfunction or mishap.

## Notes regarding these materials

1. These materials are intended as a reference to assist our customers in the selection of the Renesas Technology Corporation product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Renesas Technology Corporation or a third party.
2. Renesas Technology Corporation assumes no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts, programs, algorithms, or circuit application examples contained in these materials.
3. All information contained in these materials, including product data, diagrams, charts, programs and algorithms represents information on products at the time of publication of these materials, and are subject to change by Renesas Technology Corporation without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corporation or an authorized Renesas Technology Corporation product distributor for the latest product information before purchasing a product listed herein.
   The information described here may contain technical inaccuracies or typographical errors.
   Renesas Technology Corporation assumes no responsibility for any damage, liability, or other loss rising from these inaccuracies or errors.
   Please also pay attention to information published by Renesas Technology Corporation by various means, including the Renesas Technology Corporation Semiconductor home page (http://www.renesas.com).
4. When using any or all of the information contained in these materials, including product data, diagrams, charts, programs, and algorithms, please be sure to evaluate all information as a total system before making a final decision on the applicability of the information and products. Renesas Technology Corporation assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
5. Renesas Technology Corporation semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Renesas Technology Corporation or an authorized Renesas Technology Corporation product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.
6. The prior written approval of Renesas Technology Corporation is necessary to reprint or reproduce in whole or in part these materials.
7. If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination.
   Any diversion or reexport contrary to the export control laws and regulations of Japan and/or the country of destination is prohibited.
8. Please contact Renesas Technology Corporation for further details on these materials or the products contained therein.