To our customers,

---

Old Company Name in Catalogs and Other Documents

On April 1\textsuperscript{st}, 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](http://www.renesas.com)

April 1\textsuperscript{st}, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation ([http://www.renesas.com](http://www.renesas.com))
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 depend 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.
Serial EEPROM of HN58X25xxx Series
Control Using Clock Synchronous SCI of Renesas SH

Introduction
This document should be used for reference when implementing control of the HN58X25xxx Series serial EEPROM manufactured by Renesas Technology Corp., using the clock synchronous serial communication interface (hereafter referred to as SCI) of the SuperH family manufactured by Renesas Technology Corp.

The SuperH family incorporates a clock synchronous SCI. The HN58X25xxx Series serial EEPROM can be controlled through the clock synchronous SCI and software.

This document describes sample programs for controlling the HN58X25xxx Series serial EEPROM by using the clock synchronous SCI.

Target Device
The application examples described in this document are applicable when the following MCU and condition are used.

- MCU: SuperH family
- Condition: Clock synchronous SCI is used

The programs can be executed by any SuperH family MCU with the SCI. Note however that since some functions may be altered by function addition, etc., the functions should be confirmed against the MCU manual.

Be sure to perform evaluation sufficiently when using this application note.

Contents

1. Control Method for HN58X25xxx Series Serial EEPROM ................................................................. 2
2. Sample Programs .................................................................................................................................. 6
1. Control Method for HN58X25xxx Series Serial EEPROM

1.1 Overview of Operation

Control of the HN58X25xxx Series serial EEPROM is implemented by using the clock synchronous SCI in the SuperH. The connection method is described below.

The sample programs execute the following control operations.

- Connects the S# pin of the serial EEPROM to a SuperH port and controls it using output of the SuperH general port.
- Controls data input/output by the clock synchronous SCI (using the internal clock).

Refer to the data sheets of the MCU and serial EEPROM and specify a usable clock frequency.

![Figure 1.1 Serial EEPROM Connection Example](image-url)
### 1.2 Signal Timing Generation of Clock Synchronous SCI

Signals are generated at the following timing to satisfy the serial EEPROM timing.

![Figure 1.2: Timing for Clock Synchronous SCI of SuperH](image)

- Transmission from MCU to serial EEPROM: Transmit data output at fall of transfer clock
- Reception from serial EEPROM to MCU: Receive data input at rise of transfer clock
- Transfer in MSB-first

The SCK pin level is high when transfer is not taking place.

Check the data sheets of the MCU and serial EEPROM for the maximum clock frequency that can be used.

### 1.3 Control of S# Pin of Serial EEPROM

The S# pin of the serial EEPROM is connected to a SuperH port and controlled using output of the SuperH general port.

The period from the falling edge of the S# pin (port of SuperH) of the serial EEPROM to the falling edge of the C pin (SCK of SuperH) is controlled by inserting software wait cycles.

The period from the rising edge of the C pin (SCK of SuperH) to the rising edge of the S# pin (port of SuperH) is controlled by inserting software wait cycles.

Check the data sheet of the serial EEPROM and set the software wait time according to the system.

### 1.4 MCU Hardware Resources in Use

The hardware resources to be used are shown below.

### Table 1.1: Hardware Resources in Use

<table>
<thead>
<tr>
<th>Resource in Use</th>
<th>Number of Used Resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock synchronous SCI</td>
<td>One channel (essential)</td>
</tr>
<tr>
<td>Port (for control of the S# pin of serial EEPROM)</td>
<td>One port (essential)</td>
</tr>
</tbody>
</table>

MSB-first is specified for an MCU for which MSB-first is allowed.

Endian conversion is performed through software in an MCU that supports only LSB-first.
1.5 SuperH Register Setting (Clock Synchronous SCI)

Set up the clock synchronous SCI as shown below to satisfy the serial EEPROM specifications/timing.

A setting example in which the FIFO is not used is shown below.

1.5.1 SH7206

An example of setting the SCIF based on the register descriptions in the SH7206 Group Hardware Manual Rev. 1.00 is shown in the table below.

**Table 1.2 Clock Synchronous SCI Mode Settings**

<table>
<thead>
<tr>
<th>Register</th>
<th>Bit</th>
<th>Function and Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCFRDR</td>
<td>7 to 0</td>
<td>The receive data is read from these bits.</td>
</tr>
<tr>
<td>SCFTDR</td>
<td>7 to 0</td>
<td>Set the transmit data in these bits.</td>
</tr>
<tr>
<td>SCSMR</td>
<td>15 to 8</td>
<td>Reserved These bits are always read as 0. The write value should always be 0.</td>
</tr>
<tr>
<td></td>
<td>C/A#</td>
<td>Write 1 to this bit (clock synchronous mode).</td>
</tr>
<tr>
<td></td>
<td>CHR</td>
<td>Write 0 to this bit. Since clock synchronous mode is selected, the data length is fixed at 8 bits.</td>
</tr>
<tr>
<td></td>
<td>PE</td>
<td>Write 0 to this bit. Since clock synchronous mode is selected, parity bit addition and checking is disabled</td>
</tr>
<tr>
<td></td>
<td>O/E#</td>
<td>Write 0 to this bit. Since clock synchronous mode is selected, this bit setting is invalid.</td>
</tr>
<tr>
<td></td>
<td>STOP</td>
<td>Write 0 to this bit. Since clock synchronous mode is selected, this bit setting is invalid.</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Reserved This bit is always read as 0. The write value should always be 0.</td>
</tr>
<tr>
<td></td>
<td>CKS1, CKS0</td>
<td>Select the clock source in these bits.</td>
</tr>
<tr>
<td>SCSCR</td>
<td>15 to 8</td>
<td>Reserved These bits are always read as 0. The write value should always be 0.</td>
</tr>
<tr>
<td></td>
<td>TIE</td>
<td>Write 0 to this bit.</td>
</tr>
<tr>
<td></td>
<td>RIE</td>
<td>Write 0 to this bit.</td>
</tr>
<tr>
<td></td>
<td>TE</td>
<td>Write 1 to this bit at transmission and reception.</td>
</tr>
<tr>
<td></td>
<td>RE</td>
<td>Write 1 to this bit at reception.</td>
</tr>
<tr>
<td></td>
<td>REIE</td>
<td>Write 0 to this bit.</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Reserved This bit is always read as 0. The write value should always be 0.</td>
</tr>
<tr>
<td></td>
<td>CKE1, CKE0</td>
<td>Write 00 to these bits (internal clock/SCK pin functions as input pin (input signal is ignored)).</td>
</tr>
<tr>
<td>Register</td>
<td>Bit</td>
<td>Function and Setting</td>
</tr>
<tr>
<td>----------</td>
<td>-----------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td>SCFSR</td>
<td>PER3 to PER0</td>
<td>The status is read from these bits.</td>
</tr>
<tr>
<td></td>
<td>FER3 to FER0</td>
<td>The status is read from these bits.</td>
</tr>
<tr>
<td></td>
<td>ER</td>
<td>The status is read from this bit. 0 is written to this bit at initialization.</td>
</tr>
<tr>
<td></td>
<td>TEND</td>
<td>The status is read from this bit. 0 is written to this bit at initialization.</td>
</tr>
<tr>
<td></td>
<td>TDFE</td>
<td>The status is read from this bit. 1 is written to this bit at initialization but it is not for the purpose of clearing this bit.</td>
</tr>
<tr>
<td></td>
<td>BRK</td>
<td>The status is read from this bit. 0 is written to this bit at initialization.</td>
</tr>
<tr>
<td></td>
<td>FER</td>
<td>The status is read from this bit.</td>
</tr>
<tr>
<td></td>
<td>PER</td>
<td>The status is read from this bit.</td>
</tr>
<tr>
<td></td>
<td>RDF</td>
<td>The status is read from this bit. 0 is written to this bit at initialization.</td>
</tr>
<tr>
<td></td>
<td>DR</td>
<td>The status is read from this bit. 0 is written to this bit at initialization.</td>
</tr>
<tr>
<td>SCBRR</td>
<td>7 to 0</td>
<td>Set the transfer speed in these bits.</td>
</tr>
<tr>
<td>SCFCR</td>
<td>15 to 11</td>
<td>Write 00b to these bits.</td>
</tr>
<tr>
<td></td>
<td>RSTRG2 to RSTG0</td>
<td>Write 00b to these bits.</td>
</tr>
<tr>
<td></td>
<td>RTRG1, RTRG0</td>
<td>Write 00b to these bits.</td>
</tr>
<tr>
<td></td>
<td>TTRG1, TTRG0</td>
<td>Write 11b to these bits.</td>
</tr>
<tr>
<td></td>
<td>MCE</td>
<td>Write 0 to this bit.</td>
</tr>
<tr>
<td></td>
<td>TFRST</td>
<td>1 is written to this bit at initialization.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Write 0 to this bit at transmission and reception.</td>
</tr>
<tr>
<td></td>
<td>RFRST</td>
<td>1 is written to this bit at initialization.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Write 0 to this bit at reception.</td>
</tr>
<tr>
<td></td>
<td>LOOP</td>
<td>Write 0 to this bit.</td>
</tr>
<tr>
<td>SCFDR</td>
<td>15 to 13</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>T4 to T0</td>
<td>These bits indicate the number of untransmitted data bytes.</td>
</tr>
<tr>
<td></td>
<td>7 to 5</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>R4 to R0</td>
<td>These bits indicate the number of receive data bytes.</td>
</tr>
<tr>
<td>SCSPTR</td>
<td>15 to 8</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>RTSIO</td>
<td>0 is written to this bit at initialization.</td>
</tr>
<tr>
<td></td>
<td>RTSDT</td>
<td>0 is written to this bit at initialization.</td>
</tr>
<tr>
<td></td>
<td>CTSIO</td>
<td>0 is written to this bit at initialization.</td>
</tr>
<tr>
<td></td>
<td>CTSDT</td>
<td>0 is written to this bit at initialization.</td>
</tr>
<tr>
<td></td>
<td>SCKIO</td>
<td>0 is written to this bit at initialization.</td>
</tr>
<tr>
<td></td>
<td>SCKDT</td>
<td>0 is written to this bit at initialization.</td>
</tr>
<tr>
<td></td>
<td>SPB2IO</td>
<td>0 is written to this bit at initialization.</td>
</tr>
<tr>
<td></td>
<td>SPB2DT</td>
<td>0 is written to this bit at initialization.</td>
</tr>
</tbody>
</table>
2. **Sample Programs**

Two or more of the same devices can be connected to the serial bus and controlled.

The sample programs execute the following:

- Data read processing
- Data write processing
- Write-protection processing through software protection
- Status read processing

2.1 **Overview of Software Operations**

The operations roughly described below are performed.

(1) The driver initialization processing acquires the resources to be used by the driver and initializes them.
   - At this point, the pins connected to the serial EEPROM are as follows:
     - S pin: The MCU is set to input mode
     - SCK pin: The MCU is set to input mode
     - TxD pin: The MCU is set to input mode
     - RxD pin: The MCU is set to input mode; high-level input by an external pull-up resistor

(2) Function calls perform the following operations.
   1. Execute the processing of each function.
   2. The pins connected to the serial EEPROM are as follows:
      - S pin: The MCU is set to input mode
      - SCK pin: The MCU is set to input mode
      - TxD pin: The MCU is set to input mode
      - RxD pin: The MCU is set to input mode; high-level input by an external pull-up resistor
## Detailed Description of Functions

### 2.2.1 Driver Initialization Processing

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EEPROM driver initialization processing</td>
<td>void eep_Init_Driver(void)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Arguments</th>
<th>Non</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Return Values</th>
<th>Non</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Initializes the EEPROM driver.</td>
</tr>
<tr>
<td>• Initializes the SFR for EEPROM control.</td>
</tr>
<tr>
<td>• Performs the following processing for each device.</td>
</tr>
<tr>
<td>(a) Opens the EEPROM control ports.</td>
</tr>
<tr>
<td>(b) Initializes the EEPROM control RAM.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
</tr>
</tbody>
</table>

### Diagram

```
Start
  ↓
  eep_Init_Sfr(): Initialize SCI-related registers
  ↓
  EEPROM_IO_INIT(): Initialize the ports to be used
                   (No operation takes place though the operation is for shared use with another MCU)
  ↓
  eep_Open_Port(DevNo): Open the ports
  ↓
  eep_Init_Ram(DevNo): Clear the used RAM
End
```
2.2.2 Write-Protection Setting Processing

### Function Name
Write-protection setting processing
signed short eep_Write_Protect(unsigned char DevNo, unsigned char WpSts)

<table>
<thead>
<tr>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned char DevNo ; Device number</td>
</tr>
<tr>
<td>unsigned char WpSts ; Write-protection setting data</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Return Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns the write-protection setting result.</td>
</tr>
<tr>
<td>EEP_OK ; Successful operation</td>
</tr>
<tr>
<td>EEP_ERR_PARAM ; Parameter error</td>
</tr>
<tr>
<td>EEP_ERR_OTHER ; Other error</td>
</tr>
</tbody>
</table>

### Operations
- Makes the write-protection setting.
- Set the write-protection setting data (WpSts) as follows:
  - EEP_WP_NONE ; No protection
  - EEP_WP_UPPER_QUART ; Upper-quarter protection setting
  - EEP_WP_UPPER_HALF ; Upper-half protection setting
  - EEP_WP_WHOLE_MEM ; Whole memory protection setting

### Notes
None

---

**Flowchart**

Start

- eep_Init_Port(DevNo): Initialize the ports

- EEP_UART_El(): Enable the SCI and set SCI parameters

- eep_Write_StsReg(DevNo,&StsReg): Write to the status register

- eep_Init_Sfr(): Initialize SCI-related registers

- eep_Open_Port(DevNo): Open the ports

End
2.2.3 Data Read Processing

Function Name
Data read processing
signed short eep_Read_Data(unsigned char DevNo, unsigned short RAddr, unsigned short RCnt, unsigned char * pData)

Arguments
<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DevNo</td>
<td>Device number</td>
</tr>
<tr>
<td>RAddr</td>
<td>Read start address</td>
</tr>
<tr>
<td>RCnt</td>
<td>Number of bytes to be read</td>
</tr>
<tr>
<td>pData</td>
<td>Read data storage buffer pointer</td>
</tr>
</tbody>
</table>

Return Values
Returns the read result.
EEP_OK         : Successful operation
EEP_ERR_PARAM  : Parameter error
EEP_ERR_HARD   : Hardware error
EEP_ERR_OTHER  : Other error

Operations
- Reads data from EEPROM in bytes.
- Reads data from the specified address for the specified number of bytes.

Notes
- The maximum write address is EEPROM size – 1.
## 2.2.4 Data Write Processing

### Function Name

Data write processing

```c
signed short eep_Write_Data(unsigned char DevNo, unsigned short WAddr, unsigned short WCnt, unsigned char FAR* pData)
```

### Arguments

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned char</td>
<td>DevNo</td>
<td>Device number</td>
</tr>
<tr>
<td>unsigned short</td>
<td>WAddr</td>
<td>Write start address</td>
</tr>
<tr>
<td>unsigned short</td>
<td>WCnt</td>
<td>Number of bytes to be written</td>
</tr>
<tr>
<td>unsigned char</td>
<td>FAR*</td>
<td>Write data storage buffer pointer</td>
</tr>
</tbody>
</table>

### Return Values

Returns the write result.

- `EEP_OK` : Successful operation
- `EEP_ERR_PARAM` : Parameter error
- `EEP_ERR_HARD` : Hardware error
- `EEP_ERR_WP` : Write-protection error
- `EEP_ERR_OTHER` : Other error

### Operations

- **Writes data to EEPROM in bytes.**
- **Writes data from the specified address for the specified number of bytes.**

### Notes

- EEPROM can be written to only when write-protection has been canceled.
- The maximum write address is EEPROM size – 1.

In a write to the serial EEPROM, the page rewrite method is used. The original data is divided into the page-unit data and then written to the EEPROM.

```
Start

  eep_Init_Port(DevNo): Initialize the ports

  EEP_UART_EI(): Enable the SCI and set SCI parameters

  eep_Read_StsReg(Dev, &StsReg): Confirm write-protection

  Write page calculation processing

  eep_Write_Page(DevNo, Waddr, AbyteCnt, pData): Write
eep_mtl_wait_lp(): Software wait

  eep_Init_Sfr(): Initialize SCI-related registers
  EEP_SET_CS(Dev, EEP_HI) : S#=H

  eep_Open_Port(DevNo): Open the ports

End
```
### 2.2.5 Status Read Processing

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Status read processing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>signed short eep_Read_Status(unsigned char DevNo, unsigned char * pStatus)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned char DevNo ; Device number</td>
</tr>
<tr>
<td>unsigned char FAR* pStatus ; Read status storage buffer</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Return Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns the status register acquisition result.</td>
</tr>
<tr>
<td>EEP_OK ; Successful operation</td>
</tr>
<tr>
<td>EEP_ERR_PARAM ; Parameter error</td>
</tr>
<tr>
<td>EEP_ERR_HARD ; Hardware error</td>
</tr>
<tr>
<td>EEP_ERR_OTHER ; Other error</td>
</tr>
</tbody>
</table>

### Operations
- Reads the status.  
  Reads from the status register.
- The following information is stored in the read status storage buffer (pStatus).
  - **Memory size ≤ 512 bytes**
    - Bits 7 to 4: Reserved (All 1)
    - Bits 3, 2: BP1, BP0  
      - 00: No protection
      - 01: Upper-quarter protection
      - 10: Upper-half protection
      - 11: Whole memory protection
    - Bit 1: WEL  
      - 0: Write disabled
      - 1: Write enabled
    - Bit 0: WIP  
      - 0: During write operation
  - **Memory size > 512 bytes**
    - Bit 7: SRWD  
      - 0: Status register can be changed
      - 1: Status register cannot be changed
    - Bits 6 to 4: Reserved (All 0)
    - Bits 3, 2: BP1, BP0  
      - 00: No protection
      - 01: Upper-quarter protection
      - 10: Upper-half protection
      - 11: Whole memory protection
    - Bit 1: WEL  
      - 0: Write disabled
      - 1: Write enabled
    - Bit 0: WIP  
      - 0: During write operation

### Notes
- None
2.3 Return Value Definition

```c
#define EEP_OK (signed short) (0) /* Successful operation */
#define EEP_ERR_PARAM (signed short) (-1) /* Parameter error */
#define EEP_ERR_HARD (signed short) (-2) /* Hardware error */
#define EEP_ERR_WP (signed short) (-3) /* Write-protection error */
#define EEP_ERR_OTHER (signed short) (-4) /* Other error */
```
2.4 User Setting Examples

Setting examples when using the Renesas Technology MCU SH7206 are shown below.

The location where a setting should be made is indicated by the comment of /** SET **/ in each file.

2.4.1 eep.h

(1) Definition of the number of devices used and device numbers

Specify the number of devices to be used and assign a number for each device.

In the example below, one device is used and 0 is assigned as the device number.

When using three or more, eep_io.h needs to be modified in addition to this file.

```c
/*----------------------------------------------------------------------------------*/
/* Define the number of the required serial EEPROM devices.(1 to N devices) */
/* Define the device number in accordance with the number of serial EEPROM */
/* devices to be connected. */
/* Define number of devices */
#define EEPROM_DEV_NUM 1 /* 1 device */
/* Define No. of slots */
#define EEPROM_DEV0 0 /* Device 0 */
#define EEPROM_DEV1 1 /* Device 1 */

(2) Definition of size of device used

Specify the size of the device to be used.

In the example below, a 256-Kbit device is used.

```c
/*----------------------------------------------------------------------------------*/
/* Define the serial EEPROM device. */
/*----------------------------------------------------------------------------------*/
/*#define EEPROM_SIZE_002K /* 2 Kbits (256 bytes) */
/*#define EEPROM_SIZE_004K /* 4 Kbits (512 bytes) */
/*#define EEPROM_SIZE_008K /* 8 Kbits (1 Kbyte) */
/*#define EEPROM_SIZE_016K /* 16 Kbits (2 Kbytes) */
/*#define EEPROM_SIZE_032K /* 32 Kbits (4 Kbytes) */
/*#define EEPROM_SIZE_064K /* 64 Kbits (8 Kbytes) */
/*#define EEPROM_SIZE_128K /* 128 Kbits (16 Kbytes) */
#define EEPROM_SIZE_256K /* 256 Kbits (32 Kbytes) */
```
2.4.2 eep_sfr.h

(1) Definition of header

Specify the header corresponding to the MCU to be used.

In the example below, the SH7206 is used.

If the header is not included, add the header and also create eep_sfr.h.xxx for each MCU with reference to the provided program.

```c
#include "Eep_sfr.h.3029" /* EEPROM driver SFR common definitions */
#include "Eep_sfr.h.36049" /* EEPROM driver SFR common definitions */
#include "Eep_sfr.h.36064" /* EEPROM driver SFR common definitions */
#include "Eep_sfr.h.38024" /* EEPROM driver SFR common definitions */
#include "Eep_sfr.h.38076" /* EEPROM driver SFR common definitions */
#include "Eep_sfr.h.2378" /* EEPROM driver SFR common definitions */
#include "Eep_sfr.h.1657" /* EEPROM driver SFR common definitions */
#include "Eep_sfr.h.1650" /* EEPROM driver SFR common definitions */
#include "Eep_sfr.h.7149" /* EEPROM driver SFR common definitions */
#include "Eep_sfr.h.7206" /* EEPROM driver SFR common definitions */
```

2.4.3 eep_sfr.h.xxx (File Prepared for Each Group)

The sample program shows a description example in which channel 0 is used as the resource of the clock synchronous SCI.

No setting needs to be modified when the above resource is used.

(1) UART resource

```c
#ifdef EEP_UART_USED
#define EEP_UART_MSTP CPG.STBCR4.BIT.MSTP44 /* UART module stop control flag */
#define EEP_UART_SMR SCIF3.SCSMR.WORD /* UART serial mode register */
#define EEP_UART_SCR SCIF3.SCSMR.WORD /* UART serial control register */
#define EEP_UART_FSR SCIF3.SCFSR.WORD /* UART serial status register */
#define EEP_UART_BRR SCIF3.SCBRR.BYTE /* UART bit rate register */
#define EEP_UART_FCR SCIF3.SCFCR.WORD /* UART FIFO control register */
#define EEP_UART_SPTR SCIF3.SCrPTR.WORD /* UART serial port register */
#define EEP_UART_LSR SCIF3.SCLSR.WORD /* UART line status register */
#define EEP_UART_TXBUF SCIF3.SCFTRD.BYTE /* UART transmit FIFO data register */
#define EEP_UART_RXBUF SCIF3.SCFDRD.BYTE /* UART receive FIFO data register */
#define EEP_UART_ORER SCIF3.SCLS.BIT.ORER /* UART overrun error flag */
#define EEP_UART_TXEND SCIF3.SCFSR.BIT.TEND /* UART transmit end flag */
#define EEP_UART_TXNEXT SCIF3.SCFSR.BIT.TDFE /* UART transmit FIFO data empty */
#define EEP_UART_RXNEXT SCIF3.SCFSR.BIT.RDF /* UART receive FIFO data full */
```

If another resource is used, make additions or modify the above program. Accordingly, also make additions or modify the /* UART setting */ definition with reference to section 1.5, SuperH Register Setting (Clock Synchronous SCI).
2.4.4  eep_io.h

(1) Definition of header

Specify the header corresponding to the MCU to be used.
In the example below, the SH7206 is used.
If the header is not included, add the header and also create eep_io.h.xxx for each MCU with reference to the provided program.

```c
#include "Eep_io.h.3029"   /* EEPROM driver I/O module common definitions */
#include "Eep_io.h.36049"   /* EEPROM driver I/O module common definitions */
#include "Eep_io.h.36064"   /* EEPROM driver I/O module common definitions */
#include "Eep_io.h.38024"   /* EEPROM driver I/O module common definitions */
#include "Eep_io.h.38076"   /* EEPROM driver I/O module common definitions */
#include "Eep_io.h.2378"    /* EEPROM driver I/O module common definitions */
#include "Eep_io.h.1657"    /* EEPROM driver I/O module common definitions */
#include "Eep_io.h.1650"    /* EEPROM driver I/O module common definitions */
#include "Eep_io.h.7149"    /* EEPROM driver I/O module common definitions */
#include "Eep_io.h.7206"    /* EEPROM driver I/O module common definitions */
```
2.4.5  eep_io.h.xxx (File Prepared for Each Group)

(1) Definition of resources used by UART of MCU used

Specify the resources of the MCU to be used.
In the example below, the clock synchronous SCI is used.

```
/* Define the combination of the MCU's resources. */
#define EEP_OPTION_1 /* Low speed */ /* UART */
```

(2) Definition of control ports of MCU used

Specify the control ports of the MCU to be used.
In the example below, RxD, TxD, and SCK of the clock synchronous SCI and CS# are assigned.
When two devices are connected, make a definition regarding CS1.
When using three or more, eep.h needs to be modified in addition to this file.

```
#define EEP_P_DATAO PORT.PEDRL.BIT.PE12DR /* EEP DataOut */
#define EEP_P_DATAI PORT.PEDRL.BIT.PE11DR /* EEP DataIn */
#define EEP_P_CLK PORT.PEDRL.BIT.PE9DR /* EEP CLK */
#define EEP_D_DATAO PORT.PEIORL.BIT.PE12IOR /* EEP DataOut */
#define EEP_D_DATAI PORT.PEIORL.BIT.PE11IOR /* EEP DataIn */
#define EEP_D_CLK PORT.PEIORL.BIT.PE91IOR /* EEP CLK */
#define EEP_PCR_DATAO PORT.PECRL4.BIT.PE12MD /* EEP DataOut */
#define EEP_PCR_DATAI PORT.PECRL3.BIT.PE11MD /* EEP DataIn */
#define EEP_PCR_CLK PORT.PECRL3.BIT.PE9MD /* EEP CLK */
#define EEP_P_CS0 PORT.PEDRL.BIT.PE15DR /* EEP CS0 (Negative-true logic) */
#define EEP_D_CS0 PORT.PEIORL.BIT.PE151OR /* EEP CS0 (Negative-true logic) */
#define EEP_PCR_CS0 PORT.PECRL4.BIT.PE15MD /* EEP CS0 (Negative-true logic) */
#if (EEP_DEV_NUM > 1)
#define EEP_P_CS1 /* EEP CS1 (Negative-true logic) */
#define EEP_D_CS1 /* EEP CS1 (Negative-true logic) */
#define EEP_PCR_CS1 /* EEP CS1 (Negative-true logic) */
#endif /* #if (EEP_DEV_NUM > 1) */
```
### 2.4.6 mtl_com.h (Common Header File)

(1) Definition of header

Specify the header corresponding to the MCU to be used.
In the example below, the SH7206 is used.
If the header is not included, add the header and also create mtl_com.h.xxx for each MCU with reference to the provided program.

```
#include "mtl_com.h.7206"
```

### 2.4.7 mtl_com.h.xxx (File Prepared for Each Group)

Setting examples when using the SH7206 are shown below.

(1) Definition of OS header file

This software is an OS-independent program.
In the example below, the OS is not used.

```
/* Include an OS header file with a prototype declaration */ /** SET **/
/* because wai_sem/sig_sem/dly_tsk is used. */ /** SET **/
/* The define and include statements below should be */ /** SET **/
/* comments when the OS is not used. */ /** SET **/
#elif define MTL_OS_USE /* OS usage */ /** SET **/
#elif include <mr30.h> /* OS include file */ /** SET **/
```

(2) Definition of header file specifying common access area

Includes the header file in which the MCU registers are defined.
This file needs to be included because it is mainly used by the device driver for controlling the ports.
In the example below, the SH7206 header file is included. Include the header file in accordance with the MCU.

```
/* Include the header file containing the define statement */ /** SET **/
/* for the I/O periphery because the define value of the */ /** SET **/
/* SFR area of the MCU is used. */ /** SET **/
#include "7206.h" /* SH7206 include file */ /** SET **/
```
(3) Definition of endian type

This is a setting for another purpose. Do not modify the definition.

```c
/* Specify the endian type of the MCU used. */ /* *** SET ***/
/* When big endian is specified, little endian definition should be a comment. */ /* *** SET ***/
//#define MTL_MCU_LITTLE /* Little endian */ /* *** SET ***/
```

(4) Specification of standard library type used

This is a setting for another purpose. Do not modify the specification.

```c
/* Specify the standard library type used. */ /* *** SET ***/
/* When the processing below is used in the library provided with the */ /* *** SET ***/
/* compiler, the define statement below should be a comment. */ /* *** SET ***/
/* memcmp() / memcpy() / memset() / strcat() / strcmp() / strcpy() / strlen() */ /* *** SET ***/
//#define MTL_USER_LIB /* Optimized library usage */ /* *** SET ***/
```

(5) Definition of RAM area accessed by processing group used

Define MTL_MEM_NEAR for the SuperH family.

```c
/* Define the RAM area accessed by the processing group used. */ /* *** SET ***/
/* Standard function or processing efficient for some */ /* *** SET ***/
/* processing is applied. */ /* *** SET ***/
//#define MTL_MEM_FAR /* Supports even external RAM area */ /* *** SET ***/
#define MTL_MEM_NEAR /* Supports only internal RAM area */ /* *** SET ***/
```

Set only the above define statement and do not make any other modifications.
(6) Definition of software timer

Sets the internal software timer used.
Make this setting in accordance with the system.
The following reference values are obtained at 16.67-MHz operation without wait and when the instruction cache is disabled.
Note that the settings differ when the instruction cache is enabled and so that the setting should be made in accordance with the system.

/*-------------------------------------------------------------------------*/
/* Define the counter value of the timer. */
/* Note: Calculated at 16.67-MHz operation. */
#define MTL_T_1US 28  /* 1-us loop count */
#define MTL_T_2US 64  /* 2-us loop count */
#define MTL_T_4US 138 /* 4-us loop count */
#define MTL_T_5US 174 /* 5-us loop count */
#define MTL_T_10US 358 /* 10-us loop count */
#define MTL_T_20US 724 /* 20-us loop count */
#define MTL_T_30US 1090 /* 30-us loop count */
#define MTL_T_50US 1823 /* 50-us loop count */
#define MTL_T_100US 3655 /* 100-us loop count */
#define MTL_T_200US 7320 /* 200-us loop count */
#define MTL_T_300US 10990 /* 300-us loop count */
#define MTL_T_400US ( MTL_T_200US * 2 ) /* 400-us loop count */
#define MTL_T_1MS 36650 /* 1-ms loop count */
// #define MTL_T_2MS ( MTL_T_1MS * 2 ) /* 2-ms loop count */
// #define MTL_T_5MS ( MTL_T_1MS * 5 ) /* 5-ms loop count */
2.5 Usage Notes

The sample programs show description examples in which the clock synchronous SCI is used.

When using another resource, set the software in accordance with the hardware.

2.6 Notes at Embedment

To embed the sample programs, include eep.h.

2.7 Usage of Another SuperH Family MCU

Usage of another SuperH family MCU is supported easily.

The following files must be prepared.

(1) I/O module common definition equivalent of eep_io.h.xxx
   - Define the I/O pins to be used with reference to the SFR header of the MCU used.

(2) SFR common definition equivalent of eep_sfr.h.xxx
   - Define the UART to be used with reference to the SFR header of the MCU used.

(3) Header definition equivalent of mtl_com.h.xxx
   - Create and define a header for the MCU used.

Create the above files with reference to the provided programs.

In addition, specify the created header in eep_io.h, eep_sfr.h, and mtl_com.h.
### 2.8 File Configuration

<table>
<thead>
<tr>
<th>Directory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\com</td>
<td>Directory for common functions</td>
</tr>
<tr>
<td>mtl_com.c</td>
<td>Various definitions for common functions</td>
</tr>
<tr>
<td>mtl_com.h.1650</td>
<td>Common header file</td>
</tr>
<tr>
<td>mtl_com.h.1657</td>
<td>Common header file</td>
</tr>
<tr>
<td>mtl_com.h.2378</td>
<td>Common header file</td>
</tr>
<tr>
<td>mtl_com.h.3029</td>
<td>Common header file</td>
</tr>
<tr>
<td>mtl_com.h.36049</td>
<td>Common header file</td>
</tr>
<tr>
<td>mtl_com.h.38024</td>
<td>Common header file</td>
</tr>
<tr>
<td>mtl_com.h.38076</td>
<td>Common header file</td>
</tr>
<tr>
<td>mtl_com.h.7149</td>
<td>Common header file</td>
</tr>
<tr>
<td>mtl_com.h.7206</td>
<td>Common header file</td>
</tr>
<tr>
<td>\drv</td>
<td>Sample device driver directory</td>
</tr>
<tr>
<td>\seep_spi</td>
<td>Serial EEPROM directory</td>
</tr>
<tr>
<td>eep.h</td>
<td>Driver common definition</td>
</tr>
<tr>
<td>eep_usr.c</td>
<td>Driver user I/F module</td>
</tr>
<tr>
<td>eep_io.c</td>
<td>I/O module</td>
</tr>
<tr>
<td>eep_io.h.1650</td>
<td>I/O module common definition</td>
</tr>
<tr>
<td>eep_io.h.1657</td>
<td>I/O module common definition</td>
</tr>
<tr>
<td>eep_io.h.2378</td>
<td>I/O module common definition</td>
</tr>
<tr>
<td>eep_io.h.3029</td>
<td>I/O module common definition</td>
</tr>
<tr>
<td>eep_io.h.36049</td>
<td>I/O module common definition</td>
</tr>
<tr>
<td>eep_io.h.38024</td>
<td>I/O module common definition</td>
</tr>
<tr>
<td>eep_io.h.38076</td>
<td>I/O module common definition</td>
</tr>
<tr>
<td>eep_io.h.7149</td>
<td>I/O module common definition</td>
</tr>
<tr>
<td>eep_io.h.7206</td>
<td>I/O module common definition</td>
</tr>
<tr>
<td>eep_sfr.h</td>
<td>SFR common definition</td>
</tr>
<tr>
<td>eep_sfr.h.1650</td>
<td>SFR common definition</td>
</tr>
<tr>
<td>eep_sfr.h.1657</td>
<td>SFR common definition</td>
</tr>
<tr>
<td>eep_sfr.h.2378</td>
<td>SFR common definition</td>
</tr>
<tr>
<td>eep_sfr.h.3029</td>
<td>SFR common definition</td>
</tr>
<tr>
<td>eep_sfr.h.36049</td>
<td>SFR common definition</td>
</tr>
<tr>
<td>eep_sfr.h.38024</td>
<td>SFR common definition</td>
</tr>
<tr>
<td>eep_sfr.h.38076</td>
<td>SFR common definition</td>
</tr>
<tr>
<td>eep_sfr.h.7149</td>
<td>SFR common definition</td>
</tr>
<tr>
<td>eep_sfr.h.7206</td>
<td>SFR common definition</td>
</tr>
<tr>
<td>\sample</td>
<td>Sample program directory</td>
</tr>
<tr>
<td>usr_tst tsk.c</td>
<td>Sample program for operation verification</td>
</tr>
<tr>
<td>tsk.c</td>
<td>Various definitions for common functions</td>
</tr>
<tr>
<td>common.c</td>
<td>Various definitions for common functions</td>
</tr>
<tr>
<td>common.h</td>
<td>Various definitions for common functions</td>
</tr>
</tbody>
</table>
## Revision Record

<table>
<thead>
<tr>
<th>Rev.</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>Dec. 15.06</td>
<td>First edition issued</td>
</tr>
</tbody>
</table>

---

**Control Using Clock Synchronous SCI of Renesas SH**
Notes regarding these materials

1. This document is provided for reference purposes only so that Renesas customers may select the appropriate
Renesas products for their use. Renesas neither makes warranties or representations with respect to the
accuracy or completeness of the information contained in this document nor grants any license to any intellectual
property rights or any other rights of Renesas or any third party with respect to the information in this document.

2. Renesas shall have no liability for damages or infringement of any intellectual property or other rights arising out
of the use of any information in this document, including, but not limited to, product data, diagrams, charts,
programs, algorithms, and application circuit examples.

3. You should not use the products or the technology described in this document for the purpose of military
applications such as the development of weapons of mass destruction or for the purpose of any other military
use. When exporting the products or technology described herein, you should follow the applicable export
control laws and regulations, and procedures required by such laws and regulations.

4. All information included in this document such as product data, diagrams, charts, programs, algorithms, and
application circuit examples, is current as of the date this document is issued. Such information, however, is
subject to change without any prior notice. Before purchasing or using any Renesas products listed in this
document, please confirm the latest product information with a Renesas sales office. Also, please pay regular
and careful attention to additional and different information to be disclosed by Renesas such as that disclosed
through our website. (http://www.renesas.com)

5. Renesas has used reasonable care in compiling the information included in this document, but Renesas
assumes no liability whatsoever for any damages incurred as a result of errors or omissions in the information
included in this document.

6. When using or otherwise relying on the information in this document, you should evaluate the information in light
of the total system before deciding about the applicability of such information to the intended application.
Renesas makes no representations, warranties or guaranties regarding the suitability of its products for any
particular application and specifically disclaims any liability arising out of the application and use of the
information in this document or Renesas products.

7. With the exception of products specified by Renesas as suitable for automobile applications, Renesas products
are not designed, manufactured or tested for applications or otherwise in systems the failure or malfunction of
which may cause a direct threat to human life or create a risk of human injury or which require especially high
quality and reliability such as safety systems, or equipment or systems for transportation and traffic, healthcare,
combustion control, aerospace and aeronautics, nuclear power, or undersea communication transmission. If you
are considering the use of our products for such purposes, please contact a Renesas sales office beforehand.
Renesas shall have no liability for damages arising out of the uses set forth above.

8. Notwithstanding the preceding paragraph, you should not use Renesas products for the purposes listed below:

   (1) artificial life support devices or systems
   (2) surgical implantations
   (3) healthcare intervention (e.g., excision, administration of medication, etc.)
   (4) any other purposes that pose a direct threat to human life

Renesas shall have no liability for damages arising out of the uses set forth in the above and purchasers who
elect to use Renesas products in any of the foregoing applications shall indemnify and hold harmless Renesas
Technology Corp., its affiliated companies and their officers, directors, and employees against any and all
damages arising out of such applications.

9. You should use the products described herein within the range specified by Renesas, especially with respect to
the maximum rating, operating supply voltage range, movement power voltage range, heat radiation
characteristics, installation and other product characteristics. Renesas shall have no liability for malfunctions or
damages arising out of the use of Renesas products beyond such specified ranges.

10. Although Renesas endeavors to improve the quality and reliability of its products, IC products have specific
characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions.
Please be sure to implement safety measures to guard against the possibility of physical injury, and injury or
damage caused by fire in the event of the failure of a Renesas product, such as safety design for hardware and
software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment
for aging degradation or any other applicable measures. Among others, since the evaluation of microcomputer
software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.

11. In case Renesas products listed in this document are detached from the products to which the Renesas products
are attached or affixed, the risk of accident such as swallowing by infants and small children is very high. You
should implement safety measures so that Renesas products may not be easily detached from your products.
Renesas shall have no liability for damages arising out of such detachment.

12. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written
approval from Renesas.

13. Please contact a Renesas sales office if you have any questions regarding the information contained in this
document, Renesas semiconductor products, or if you have any other inquiries.

© 2006. Renesas Technology Corp., All rights reserved.