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 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 datasheets 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.
H8SX Family

0.18-µm Flash Memory Reprogramming in the User Boot Mode

Introduction

This application note describes the reprogramming of flash memory (the user MAT) via a clock-synchronous communications interface in the user boot mode of the H8SX/1582F, and mainly concerns the slave (receive) side. That is, unless otherwise specified, the descriptions are related to the slave side.

Target Device

H8SX/1582F

Contents

1. Specifications .................................................................................................................. 2
2. Applicable Conditions .................................................................................................... 6
3. Description of Modules Used ......................................................................................... 7
4. Description of Operation .............................................................................................. 10
5. Description of Software for Initialization Program on the Receive Side (Slave) .......... 18
6. Description of Software for Programming/Erasing Procedure Program on the Slave Side........ 24
7. Description of Software for the Clock Synchronous Serial Communications Program on the Slave Side ................................................................................................. 45
8. Documents for Reference (Note) .................................................................................. 57
1. Specifications

1.1 Specification in Outline

The user starts by downloading the erasing or programming module to the RAM and then calls subroutines to erase/program the flash memory.

In this sample task, data for reprogramming are placed in the flash memory on the master side and sent to the slave side by clock synchronous communications. The following procedure is then applied to reprogram the flash memory on the slave side. A sample configuration for on-board reprogramming is shown in figure 1, and the connections for clock synchronous communications between the master and slave are shown in figure 2.

- Power-on resets are applied, and the slave boots up in the user boot mode and the master boots up in the user mode.
- When master side switch 0 (SW0) is turned on, the master side sends the erase command to the slave side, and the slave side erases its own flash memory.
- The master side sends the data to be reprogrammed from its own flash memory to the slave side. The data are transferred in clock synchronous mode via serial communications interface (SCI) 3. The data are transmitted from the master side and received on the slave side.
- The slave programs the received data for reprogramming to its own flash memory.
- On both the master and slave sides, PD7 is low and PD6 is high while the flash memory is being reprogrammed, and PD7 is high and PD6 is low on completion of reprogramming.

![Figure 1 Sample Configuration for On-Board Reprogramming](image1)

![Figure 2 Wiring for Clock Synchronous Communications](image2)
1.2 User Boot Mode

Set the mode pins for the user boot mode (MD1 = 0, MD0 = 1), and perform a start from reset.

The RAM transfer program, clock synchronous communications program, and programming/erasing program must be written to the user boot MAT in advance.

1.3 Specifications for Communications

The specifications for communications between the master and slave in this sample task are listed below.

1.3.1 Type of Communications

Table 1 Type of Communications

<table>
<thead>
<tr>
<th>Item</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transfer rate</td>
<td>2.5 Mbps</td>
</tr>
<tr>
<td>Type</td>
<td>Clock synchronous communications</td>
</tr>
<tr>
<td>Data bits</td>
<td>8 bits (1 byte)</td>
</tr>
</tbody>
</table>

1.3.2 Communications Commands

Table 2 Communications Commands

<table>
<thead>
<tr>
<th>Command Name</th>
<th>Constant Name</th>
<th>Command Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Erase</td>
<td>ERASE</td>
<td>H'11</td>
</tr>
<tr>
<td>Write</td>
<td>WRITE</td>
<td>H'12</td>
</tr>
<tr>
<td>Read status</td>
<td>STATUSREAD</td>
<td>H'13</td>
</tr>
<tr>
<td>128-byte transmission request</td>
<td>TRS128</td>
<td>H'14</td>
</tr>
</tbody>
</table>

1.3.3 State Indicators

Table 3 State Indicators

<table>
<thead>
<tr>
<th>Status Name</th>
<th>Constant Name</th>
<th>Status Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>OK</td>
<td>H'00</td>
</tr>
<tr>
<td>Erase command error</td>
<td>ER_ECMD</td>
<td>H'C1</td>
</tr>
<tr>
<td>Erase download error</td>
<td>ER_EDWNLD</td>
<td>H'C2</td>
</tr>
<tr>
<td>Erase initialization error</td>
<td>ER_EINIT</td>
<td>H'C3</td>
</tr>
<tr>
<td>Erase error</td>
<td>ER_ERASE</td>
<td>H'C4</td>
</tr>
<tr>
<td>Program command error</td>
<td>ER_WCMD</td>
<td>H'A1</td>
</tr>
<tr>
<td>Program download error</td>
<td>ER_WDWNLD</td>
<td>H'A2</td>
</tr>
<tr>
<td>Program initialization error</td>
<td>ER_WINIT</td>
<td>H'A3</td>
</tr>
<tr>
<td>Program error</td>
<td>ER_WRITE</td>
<td>H'A4</td>
</tr>
</tbody>
</table>
1.3.4 Specifying Blocks to be Erased

The erase command "ERASE" is sent from the master to the slave and is immediately followed by an indicator of the block numbers of blocks to be erased. The communications format for block erasure is given in table 4. Block settings are made in a 4-byte (32-bit) unit where bits 11 to 0 correspond to blocks 11 to 0. Since bits 31 to 12 are not used, always set them to 0. Set bits corresponding to blocks to be erased to 1 and bits corresponding to blocks that are not to be erased to 0. An example of the data transmitted to erase block 11 is given in table 5.

### Table 4 Correspondence of Blocks to be Erased

<table>
<thead>
<tr>
<th>Bit</th>
<th>Erased Block</th>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 to 12</td>
<td>Not used</td>
<td>0 fixed</td>
<td>Not used. Set 0.</td>
</tr>
<tr>
<td>11 to 0</td>
<td>EB11 to EB0</td>
<td>0/1</td>
<td>0: The corresponding block is not erased. 1: The corresponding block is erased.</td>
</tr>
</tbody>
</table>

### Table 5 Example of Data for Transmission to Erase Block 11

<table>
<thead>
<tr>
<th>Item</th>
<th>1st Byte</th>
<th>2nd Byte</th>
<th>3rd Byte</th>
<th>4th Byte</th>
<th>5th Byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte</td>
<td>H'11</td>
<td>H'00</td>
<td>H'00</td>
<td>H'08</td>
<td>H'00</td>
</tr>
<tr>
<td>Bit</td>
<td>00010001</td>
<td>00000000</td>
<td>00000000</td>
<td>00001000</td>
<td>00000000</td>
</tr>
</tbody>
</table>

Note: The data is transmitted in byte units, with the LSB first.
1.4 Memory Map

The memory map for this sample task is given as figure 3.

![Memory Map Diagram]

**Figure 3 Memory Map**
2. Applicable Conditions

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating frequency</td>
<td>Input clock: 5 MHz&lt;br&gt;System clock (Iφ): 40 MHz&lt;br&gt;Peripheral module clock (Pφ): 20 MHz&lt;br&gt;External bus clock (Bφ): 20 MHz</td>
</tr>
<tr>
<td>Operating mode</td>
<td>Mode 1 (MD1 = 0, MD0 = 1)</td>
</tr>
<tr>
<td>On-board programming mode</td>
<td>User boot mode</td>
</tr>
<tr>
<td>Development tool</td>
<td>High-performance Embedded Workshop Ver. 4.00.02</td>
</tr>
<tr>
<td>C/C++ compiler</td>
<td>Renesas Technology Corp. H8S, H8/300 Series C/C++ Compiler Ver. 6.01.00</td>
</tr>
<tr>
<td>Compiler option</td>
<td>-cpu = h8sxa:24:md, -code = machinecode, -optimize = 1, -regparam = 3 -speed = (register, shift, struct, expression)</td>
</tr>
<tr>
<td>Optimizing linkage editor option</td>
<td>-rom = PFL_Code = RAM</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 7 Section Settings</th>
<th>Address</th>
<th>Section Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>H'001000</td>
<td>P</td>
<td>Program area</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PFL_Code</td>
<td>Area for storing programming/erasing procedure program</td>
</tr>
<tr>
<td></td>
<td>H'FF9000</td>
<td>RAM</td>
<td>Area for transferring programming/erasing procedure program</td>
</tr>
</tbody>
</table>
3. Description of Modules Used

3.1 User Boot Mode

3.1.1 User MAT and User Boot MAT

The on-chip flash memory consists of the two memory units (memory MATs) listed in table 8. Both are allocated to the same address. The user boot mode is an arbitrary boot program that suits the user system, and the user MAT can be programmed/erased.

When the CPU’s operating mode setting is for the user boot mode and it is started from a reset, processing starts from the execution start address contained in the reset vector. At this time the target memory MAT is the user boot MAT (FMATS = H'AA). If the user MAT is actually to be erased/programmed, the user boot MAT must be replaced by the user MAT in the memory map when the erasing/programming is to be done. This switching should be executed by code in RAM.

<table>
<thead>
<tr>
<th>Memory MAT</th>
<th>Activation</th>
<th>Amount of Memory Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>User MAT</td>
<td>Activates when a power-on reset is performed</td>
<td>256 Kbytes</td>
</tr>
<tr>
<td></td>
<td>in the user mode</td>
<td></td>
</tr>
<tr>
<td>User boot MAT</td>
<td>Activates when a power-on reset is performed</td>
<td>10 Kbytes</td>
</tr>
<tr>
<td></td>
<td>in the user boot mode</td>
<td></td>
</tr>
</tbody>
</table>
### 3.1.2 Downloading the On-Chip Program

Erasing/programming of the flash memory on this LSI is done by first downloading the on-chip module for either erasing/programming to the on-chip RAM and then running the individual programs.

The destination address for downloading is determined by the setting of the download destination specification register (FTDAR). For the RAM address map after downloading, refer to figure 4. As the figure shows, the on-chip RAM area is the destination for downloading of the erasing and programming programs. The corresponding program must be downloaded to the RAM area indicated by FTDAR prior to the required processing.

During erasing/programming, take care to ensure that the download area does not overlap with an area in use by the user.

![Figure 4: Normal Operation](image)

<table>
<thead>
<tr>
<th>[When erasing]</th>
<th>[When programming]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area to be used by the user</td>
<td>Area to be used by the user</td>
</tr>
<tr>
<td>RAM start address H'FF9000</td>
<td>Address specified by FTDAR</td>
</tr>
<tr>
<td>DPFR (return value: 1 byte)</td>
<td>DPFR (return value: 1 byte)</td>
</tr>
<tr>
<td>System area (15 bytes)</td>
<td>System area (15 bytes)</td>
</tr>
<tr>
<td>Erasing program entry</td>
<td>Programming program entry</td>
</tr>
<tr>
<td>Initialization program entry</td>
<td>Initialization program entry</td>
</tr>
<tr>
<td>Initialization and erasing programs</td>
<td>Initialization and programming programs</td>
</tr>
<tr>
<td>Area that can be used by the user</td>
<td>Area that can be used by the user</td>
</tr>
<tr>
<td>RAM end address H'FFBFFF</td>
<td></td>
</tr>
</tbody>
</table>
3.2 Block Configuration

Erase blocks of the user MAT are listed in table 9.

Table 9 User MAT Erase Blocks

<table>
<thead>
<tr>
<th>Block</th>
<th>Unit of Erasure</th>
<th>Addresses</th>
</tr>
</thead>
<tbody>
<tr>
<td>EB0</td>
<td>4 Kbytes</td>
<td>H'000000 to H'000FFF</td>
</tr>
<tr>
<td>EB1</td>
<td>4 Kbytes</td>
<td>H'001000 to H'001FFF</td>
</tr>
<tr>
<td>EB2</td>
<td>4 Kbytes</td>
<td>H'002000 to H'002FFF</td>
</tr>
<tr>
<td>EB3</td>
<td>4 Kbytes</td>
<td>H'003000 to H'003FFF</td>
</tr>
<tr>
<td>EB4</td>
<td>4 Kbytes</td>
<td>H'004000 to H'004FFF</td>
</tr>
<tr>
<td>EB5</td>
<td>4 Kbytes</td>
<td>H'005000 to H'005FFF</td>
</tr>
<tr>
<td>EB6</td>
<td>4 Kbytes</td>
<td>H'006000 to H'006FFF</td>
</tr>
<tr>
<td>EB7</td>
<td>4 Kbytes</td>
<td>H'007000 to H'007FFF</td>
</tr>
<tr>
<td>EB8</td>
<td>32 Kbytes</td>
<td>H'008000 to H'00FFFF</td>
</tr>
<tr>
<td>EB9</td>
<td>64 Kbytes</td>
<td>H'010000 to H'01FFFF</td>
</tr>
<tr>
<td>EB10</td>
<td>64 Kbytes</td>
<td>H'020000 to H'02FFFF</td>
</tr>
<tr>
<td>EB11</td>
<td>64 Kbytes</td>
<td>H'030000 to H'03FFFF</td>
</tr>
</tbody>
</table>

3.3 Serial Communications Interface

The SCI operates in the clock synchronous mode. It is used for command-related communications between the master and slave to transfer the data for reprogramming.
4. Description of Operation

4.1 User MAT Reprogramming Procedure

The procedure for reprogramming the user MAT in the user boot mode is shown in figure 5. In the user boot mode, the user boot MAT is accessible in the flash memory area, but the user MAT is hidden behind it. Therefore, user MAT must be chosen on the occasion of a reprogramming to user MAT. During programming of the user MAT, the user boot MAT is hidden, and the user MAT is being programmed, so the procedure program must be executed from an area other than the flash-memory area. After programming, the memory MATs are switched back to their initial mapping.

![Flowchart of User MAT Reprogramming Procedure](image-url)

**Figure 5** User MAT Reprogramming Procedure
4.2 Operation Overview

4.2.1 Start of On-Board Reprogramming

(1) A power-on reset is applied to the slave side with the mode pins set for the user boot mode, the RAM transfer program in the user boot MAT on the slave side is activated, and the programming/erasing procedure program is transferred to the on-chip RAM.

(2) Here, PD7 is low and PD6 is high on the slave side.

![Diagram showing the start of on-board reprogramming](image)

**Figure 6 Start of On-Board Reprogramming**
4.2.2 Activating the Programming/Erasing Procedure Program

(1) After the transfer programs have been loaded to RAM, control branches to the programming/erasing procedure program in RAM.

![Diagram of activation of Programming/Erasing Procedure Program]

**Figure 7 Activation of Programming/Erasing Procedure Program**
### 4.2.3 Erasing User MAT

1. When a low level input on the \( \text{IRQ0} \) pin as a trigger, the master sends the erasing command, "ERASE".
2. At this time, PD7 is low and PD6 is high on the master side.
3. Registers for controlling the flash memory are set (both the EPVB bit of the FECS register and the SCO bit of the FCCS register to 1), and the initialization program and erasure program are downloaded.
4. The initialization program is executed.
5. The erasing program is executed, and the target block for erasure on the user MAT is erased.

#### Figure 8 Erasing User MAT
4.2.4 Programming User MAT

(1) The master sends the programming command, "WRITE".

(2) Registers for controlling the flash memory are set (both the PPVS bit of the FPCS register and the SCO bit of the FCCS register to 1), and the initialization program and the programming program are downloaded.

(3) The initialization program is executed.

(4) The following a. to b. are repeated until all new data on the master side are programmed on the slave side.

a. The receive side (slave) receives 128 bytes of new data from the transfer source (master).

b. The receive side (slave) executes the programming program, and writes 128 bytes of data to the user MAT.

(5) On completion of programming, PD7 is high and PD6 is low on both the master and slave sides.

Figure 9 Programming User MAT
4.3 Sequence Diagram

4.3.1 Erasure Processing

![Sequence Diagram for Erasure Processing]

Figure 10 Erasure Processing
4.3.2 Processing to Receive the Position and Range for Programming

```
Master

Transmit programming command "WRITE"

Transmit programming start address (H'00002000)

Transmit programming size (H'00003000)

Wait for processing on the slave side

Transmit status-read command "STATUSREAD"

Slave

Receive 1 byte

Receive programming start address

Receive programming size

Receive data = "STATUSREAD"?

No

Yes

ER_WCMD → rtn

Wait for processing on the slave side

Receive 1 byte

Receive programming start address

Receive programming size

Receive data = "STATUSREAD"?

No

Yes

Transmit "OK"

PPVS of FPCS = 1

Download programming program

Download programming program

Programming program downloaded successfully?

Error

Successful

ER_WDWNLD → rtn

Initialization of programming processing

Initialize programming processing

Error

Successful

ER_WINIT → rtn

Programming processing

Programming processing

Figure 11 Processing to Receive the Position and Range for Programming
4.3.3 Programming Processing

**Figure 12** Programming Processing
5. Description of Software for Initialization Program on the Receive Side (Slave)

5.1 List of Functions

The initialization program (main.c) on the receive side is used to transfer the programming/erasing procedure program to the on-chip RAM. A list of functions used in the initialization program on the receive side is given in table 10, and the hierarchical structure of calls is given in figure 13.

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>init</td>
<td>Initialization routine. Releases from the module stop mode, makes clock settings, and calls the main function.</td>
</tr>
<tr>
<td>main</td>
<td>Main routine. Transfers the programming/erasing procedure program from the user MAT to on-chip RAM.</td>
</tr>
<tr>
<td>copyfzram</td>
<td>Transfers the programming/erasing procedure program from the user MAT to the on-chip RAM.</td>
</tr>
<tr>
<td>flew_main</td>
<td>Programming/erasing procedure program in the user MAT.</td>
</tr>
</tbody>
</table>

![Figure 13 Initialization Program on the Receive Side](image)

Table 10 Functions in the Initialization Program on the Receive Side
5.2 Description of Functions

5.2.1 init Function

1. **Overview**
   Initialization routine, releases the module stop mode, sets the clock, and calls the main function.

2. **Arguments**
   None

3. **Return value**
   None

4. **Internal registers used**
   The internal registers used in this sample task are shown below. The setting values are the values used in this sample task, and not the initial values.

   - **System Clock Control Register (SCKCR)** Address: H'FFFDC4
     ![System Clock Control Register Table]

     | Bit | Bit Name | Setting | R/W | Description |
     |-----|---------|---------|-----|-------------|
     | 10  | ICK2    | 0       | R/W | System clock (Iφ) select |
     | 9   | ICK1    | 0       | R/W | Selects the frequency of the CPU, DMAC, DTC module and system clock. |
     | 8   | ICK0    | 0       | R/W | 000: Input clock x 8 |
     | 6   | PCK2    | 0       | R/W | Peripheral module clock (Pφ) select |
     | 5   | PCK1    | 0       | R/W | Selects the frequency of peripheral module clock. |
     | 4   | PCK0    | 1       | R/W | 001: Input clock x 4 |
     | 2   | BCK2    | 0       | R/W | External bus clock (Bφ) select |
     | 1   | BCK1    | 0       | R/W | Selects the frequency of external bus clock. |
     | 0   | BCK0    | 1       | R/W | 001: Input clock x 4 |

   - **Module Stop Control Register A (MSTPCR)** Address: H'FFFDC8
     ![Module Stop Control Register Table]

     | Bit | Bit Name | Setting | R/W | Description |
     |-----|---------|---------|-----|-------------|
     | 15  | ACSE    | 0       | R/W | All-Module-Clock-Stop Mode Enable |
     | 13  | MSTPA13 | 1       | R/W | Enables/disables all-module-clock-stop mode for reducing current drawn by stopping the bus controller and I/O port operations when the CPU executes the SLEEP instruction after the module stop mode has been set for all the on-chip peripheral modules controlled by MSTPCR. |
     | 12  | MSTPA12 | 1       | R/W | 0: All-module-clock-stop mode disabled |
     | 11  | MSTPA11 | 0       | R/W | 1: All-module-clock-stop mode enabled |
     | 10  | MSTPA10 | 1       | R/W | DMA controller (DMAC) |
     | 9   | MSTPA9  | 1       | R/W | Data transfer controller (DTC) |
     | 8   | MSTPA8  | 1       | R/W | A/D converter unit 1 |
     | 7   | MSTPA7  | 1       | R/W | A/D converter unit 0 |
     | 6   | MSTPA6  | 1       | R/W | 16-bit timer pulse unit (TPU channels 11 to 6) |
     | 5   | MSTPA5  | 1       | R/W | 16-bit timer pulse unit (TPU channels 5 to 0) |
     | 4   | MSTPA4  | 1       | R/W | 16-bit timer pulse unit (TPU channels 11 to 6) |
     | 3   | MSTPA3  | 1       | R/W | 16-bit timer pulse unit (TPU channels 5 to 0) |
     | 2   | MSTPA2  | 1       | R/W | 16-bit timer pulse unit (TPU channels 11 to 6) |
     | 1   | MSTPA1  | 1       | R/W | 16-bit timer pulse unit (TPU channels 5 to 0) |
     | 0   | MSTPA0  | 1       | R/W | 16-bit timer pulse unit (TPU channels 5 to 0) |
• Module Stop Control Register B (MSTPCRB)  Address: H'FFFDCA

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Setting</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>MSTPB15</td>
<td>1</td>
<td>R/W</td>
<td>Programmable Pulse Generator (PPG)</td>
</tr>
<tr>
<td>12</td>
<td>MSTPB12</td>
<td>0</td>
<td>R/W</td>
<td>Serial Communication Interface_4 (SCI_4)</td>
</tr>
<tr>
<td>11</td>
<td>MSTPB11</td>
<td>0</td>
<td>R/W</td>
<td>Serial Communication Interface_3 (SCI_3)</td>
</tr>
</tbody>
</table>

• Module Stop Control Register C (MSTPCRC)  Address: H'FFFDCC

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Setting</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>MSTPC10</td>
<td>1</td>
<td>R/W</td>
<td>Synchronous serial communications unit 2 (SSU_2)</td>
</tr>
<tr>
<td>9</td>
<td>MSTPC9</td>
<td>1</td>
<td>R/W</td>
<td>Synchronous serial communications unit 1 (SSU_1)</td>
</tr>
<tr>
<td>8</td>
<td>MSTPC8</td>
<td>1</td>
<td>R/W</td>
<td>Synchronous serial communications unit 0 (SSU_0)</td>
</tr>
<tr>
<td>1</td>
<td>MSTPC1</td>
<td>0</td>
<td>R/W</td>
<td>On-chip RAM_1 (H'FF9000 to H'FFBFFF)</td>
</tr>
<tr>
<td>0</td>
<td>MSTPC0</td>
<td>0</td>
<td>R/W</td>
<td>Write a value so that MSTPC1 is always the same value as MSTPC0.</td>
</tr>
</tbody>
</table>

5. Flowchart

```
init
CCR = H'80
Initialize CCR by masking interrupts
SCKCR = H'0011
Clock (x8, x4, x4)
MSTPCRA = H'3FFF
MSTPCRB = H'E7FF
MSTPCRC = H'FF00
Clear the module stop mode
main()
Call the main routine
```
5.2.2 main Function

1. Overview
   Branches to programming/erasing procedure program.

2. Arguments
   None

3. Return value
   None

4. Internal registers used
   The internal registers used in this sample task are shown below. The setting values are the values used in this sample task, and not the initial values.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Setting</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>PD7DDR</td>
<td>1</td>
<td>R/W</td>
<td>0: Set the PD7 pin to input</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Set the PD7 pin to output</td>
</tr>
<tr>
<td>6</td>
<td>PD6DDR</td>
<td>1</td>
<td>R/W</td>
<td>0: Set the PD6 pin to input</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Set the PD6 pin to output</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Setting</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>PD7DR</td>
<td>Undefined</td>
<td>R/W</td>
<td>0: PD7 pin is driven low</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: PD7 pin is driven high</td>
</tr>
<tr>
<td>6</td>
<td>PD6DR</td>
<td>Undefined</td>
<td>R/W</td>
<td>0: PD6 pin is driven low</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: PD6 pin is driven high</td>
</tr>
</tbody>
</table>
5. Flowchart

main

- copyfzram() RAM transfer program
- sci3_init() SCI3 initialization
- PDDDR = H’C0 Set PD7 and PD6 pins as output
- Output flash reprogramming state from PD7 and PD6 pins
  - (During reprogramming)
  - PD7DR = 0
  - PD6DR = 1
- tmp = flew_main() Branch to programming/erasing control program in on-chip RAM
- tmp ≠ OK?
  - No
  - Output flash reprogramming state from PD7 and PD6 pins
    - (No error)
    - PD7DR = 1
    - PD6DR = 0
  - Yes
  - Output flash reprogramming state from PD7 and PD6 pins
    - (With error)
    - PD7DR = 0
    - PD6DR = 0
- SCI3.SCR = 0
  - TE, RE = 0
- End
5.2.3 copyfzram Function

1. Overview
   Transfers the programming/erasing procedure program to the on-chip RAM.

2. Arguments
   None

3. Return value
   None

4. Internal registers used
   None

5. Flowchart

5.2.4 flew_main Function

1. Overview
   Calls the main routine of the programming/erasing procedure program.
6. Description of Software for Programming/Erasing Procedure Program on the Slave Side

6.1 List of Functions

Programming/erasing procedure program (fwrite.c) performs erasing in erase block units, receives flash memory programming data, and performs programming to flash memory. A list of functions for the routines used in the programming/erasing procedure program is given in table 11. The hierarchical structure is shown in figure 14.

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>flew_main</td>
<td>Main processing of flash memory erasing/programming</td>
</tr>
<tr>
<td>erase_process</td>
<td>Erases flash memory</td>
</tr>
<tr>
<td>write_process</td>
<td>Programs flash memory</td>
</tr>
<tr>
<td>download</td>
<td>Downloads on-chip modules</td>
</tr>
<tr>
<td>fw_init</td>
<td>Initialization before flash memory erasing and programming</td>
</tr>
</tbody>
</table>

*Called from clock synchronous serial communications program.

Figure 14 Programming/Erasing Procedure Program
6.2 Description of Modules

6.2.1 flew_main Function

1. Overview
   Main processing of flash memory erasing/programming.

2. Arguments
   None

3. Return value
   Type  Description
   unsigned char  Error status

4. Internal registers used
   The internal registers used in this sample task are shown below. The setting values are the values used in this
   sample task, and not the initial values.

   • System Control Register (SYSCR)  Address: H'FFFD2C
     Bit  Bit Name  Setting  R/W  Description
     7    FLSHE    1       R/W  Flash Memory Control Register Enable
           Controls accesses by the CPU to the flash memory control
           registers. Setting this bit to 1 enables read from/write to the flash
           memory control registers. Clearing this bit to 0 disables the flash
           memory control registers. At this time, the contents of the flash
           memory control registers are retained.
           0: Disables the flash memory control registers
           1: Enables the flash memory control registers

   • Flash Program Code Select Register (FPCS)  Address: H'FFFD29
     Bit  Bit Name  Setting  R/W  Description
     0    PPVS     1       R/W  Program Pulse Verify
           Selects the programming program to be downloaded.
           0: Programming program is not selected
           [Clearing condition]
           • When transfer is completed
           1: Programming program is selected.

   • Flash Erase Code Select Register (FECS)  Address: H'FFFD2A
     Bit  Bit Name  Setting  R/W  Description
     0    EPVB     1       R/W  Erase Pulse Verify Block
           Selects the erasing program to be downloaded.
           0: Erasing program is not selected
           [Clearing condition]
           • When transfer is completed
           1: Erasing program is selected.
5. Flowchart

```
flow_main

sci3_rcvbyte()
Receive 5 bytes → rcvbuf

tmp = rcvbuf[0]
rcvdt. byte[0] to [3]
Read data

Yes

No

rtn = ER_ECMD
Error setting

end_main

FOHHE of SYSCR = 1
Enable access to flash memory
control registers
```
1
EPVB of FECS = 1
Select erasing program

fw_err = download()
Download erasing program

fw_err
= 0?
Yes
Error occurred?

No
rtn = ER_EDWNLD
Error setting

fw_err = fw_init()
Execute initialization
processing for erasure

fw_err
= 0?
Yes
Error occurred?

No
rtn = ER_EINIT
Error setting

fw_err
= erase_process(rcvdt)
Erase processing
rcvdt: Erase-block data

fw_err
= 0?
Yes
Error occurred?

No
rtn = ER_ERASE
Error setting

tmp = sci3_rcv1byte()
Receive 1 byte

tmp ≠ STATUSREAD
Received data
≠ STATUSREAD command?

Yes
sci3_trs1byte(OK)
Transmit "OK"

No
sci3_rcvbyte()
Receive 9 bytes → rcvbuf

tmp = rcvbuf[0]
wsize = rcvbuf[5] to [8]
Read data

tmp ≠ WRITE?
Received data ≠ ERASE command?

rtn = ER_WCMD
Error setting

sci3_trs1byte(OK)
Transmit "OK"

sci3_rcv1byte()
Receive 1 byte

tmp ≠ STATUSREAD
Received data ≠ STATUSREAD command?

end_main
PPVS of FPCS = 1
Select programming program

fw_err = download()
Download programming program

fw_err ≠ 0?
Error occurred?

Yes
No

rtn = ER_WDNLD
Error setting

fw_err = fw_init()
Execute initialization processing for programming

fw_err ≠ 0?
Error occurred?

Yes
No

rtn = ER_WINIT
Error setting

fw_err = write_process()
Programming processing

fw_err ≠ 0?
Error occurred?

Yes
No

rtn = ER_WRITE
Error setting

rtn = OK
Set OK

end_main

end_main

end_main

end_main
End

**Yes**

**Yes**

**No**

```c
sci3_trs1byte(rtn)
Transmit rtn
```

Return rtn.

**End**

**FLSHE of SYSCR = 0**

Disable access to flash memory control registers

```c
tmp = sci3_rcv1byte()
Receive 1 byte
```

**tmp ≠ STATUSREAD**

Received data ≠ STATUSREAD command?

**Scenarios**:

- **Yes**
  - **Yes**
    - **No**
      - **Return rtn.**

- **No**
  - **end_main**
6.2.2 erase_process Function

1. Overview
   Erases flash memory.

2. Arguments
<table>
<thead>
<tr>
<th>Type</th>
<th>Variable Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned char</td>
<td>ERASEBLK</td>
<td>Erases block</td>
</tr>
</tbody>
</table>

3. Return value
<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned char</td>
<td>Flash pass and fail parameter (FPFR). Return value of the erase result.</td>
</tr>
</tbody>
</table>

4. Internal registers used
   The internal registers used in this sample task are shown below. The setting values are the values used in this sample task, and not the initial values.

   • Flash Key Code Register (FKEY)  Address: H'FFFDEC
     | Bit | Bit Name | Setting | R/W | Description |
     |-----|----------|---------|-----|-------------|
     | 7   | K7       | 0       | R/W | Key Code    |
     | 6   | K6       | 1       | R/W | When H'A5 is written to FKEY, writing to the SCO bit in FCCS is enabled. When a value other than H'A5 is written, the SCO bit cannot be set to 1. Therefore, the on-chip program cannot be downloaded to the on-chip RAM. Only when H'5A is written can programming/erasing of the flash memory be executed. When a value other than H'5A is written, even if the programming/erasing program is executed, programming/erasing cannot be performed.
     | 5   | K5       | 0       | R/W | H'5A: Writing to the SCO bit is enabled. (The SCO bit cannot be set to 1 when FKEY is a value other than H'A5.)
     | 4   | K4       | 1       | R/W | H'5A: Programming/erasing of the flash memory is enabled. (When FKEY is a value other than H'5A, the software protection state is entered.)
     | 3   | K3       | 1       | R/W | H00: Initial value |
     | 2   | K2       | 0       | R/W | MAT Selection |
     | 1   | K1       | 1       | R/W | The memory MATs can be switched by writing a value to FMATS. When H'AA is written to FMATS, the user boot MAT is selected. When a value other than H'AA is written, the user MAT is selected. In the user boot mode: |
     | 0   | K0       | 0       | R/W | H'AA: The user boot MAT is selected. A value other than H'AA: The user MAT is selected. |

   • Flash MAT Select Register (FMATS)  Address: H'FFFEAD
     | Bit | Bit Name | Setting | R/W | Description |
     |-----|----------|---------|-----|-------------|
     | 7   | MS7      | 0       | R/W | MAT Selection |
     | 6   | MS6      | 0       | R/W | The memory MATs can be switched by writing a value to FMATS. When H'AA is written to FMATS, the user boot MAT is selected. |
     | 5   | MS5      | 0       | R/W | When a value other than H'AA is written, the user MAT is selected. In the user boot mode: |
     | 4   | MS4      | 0       | R/W | H'AA: The user boot MAT is selected. |
     | 3   | MS3      | 0       | R/W | A value other than H'AA: The user MAT is selected. |
     | 2   | MS2      | 0       | R/W | MAT Selection |
     | 1   | MS1      | 0       | R/W | The memory MATs can be switched by writing a value to FMATS. When H'AA is written to FMATS, the user boot MAT is selected. |
     | 0   | MS0      | 0       | R/W | H'AA: The user boot MAT is selected. A value other than H'AA: The user MAT is selected. |
Flash Pass and Fail Parameter (FPFR)  
Return value of the erase results

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Setting</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>MD</td>
<td>Undefined</td>
<td>R/W</td>
<td>Erasing Mode Related Setting Error Detect</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Detects the error protection state and returns the result. When the error protection state is entered, this bit is set to 1. Whether the error protection state is entered or not can be confirmed with the FLER bit in FCCS.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Normal operation (FLER = 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Error protection state, and programming cannot be performed (FLER = 1)</td>
</tr>
<tr>
<td>5</td>
<td>EE</td>
<td>Undefined</td>
<td>R/W</td>
<td>Erasing Execution Error Detect</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Writes 1 to this bit when the specified data could not be written because the user MAT was not erased. If this bit is set to 1, there is a high possibility that the user MAT has been written partially.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>In this case, after removing the error factor, erase the user MAT.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Erasure has ended normally</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Erasure has ended abnormally</td>
</tr>
<tr>
<td>4</td>
<td>FK</td>
<td>Undefined</td>
<td>R/W</td>
<td>Flash Key Register Error Detect</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Checks the FKEY value (H’A5) before programming starts, and returns the results.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: FKEY setting is normal (H’5A)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: FKEY setting is abnormal (a value other than H’5A)</td>
</tr>
<tr>
<td>3</td>
<td>EB</td>
<td>Undefined</td>
<td>R/W</td>
<td>Erase Block Selection Error Detect</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Checks whether the specified erase block number is in the block range of user MAT and returns the result.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Setting of erase block number is normal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Setting of erase block number is abnormal</td>
</tr>
<tr>
<td>0</td>
<td>SF</td>
<td>Undefined</td>
<td>R/W</td>
<td>Success/Fail</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Indicates the erasure results.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Erasure ended normally (no error)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Erasure ended abnormally (error occurred)</td>
</tr>
</tbody>
</table>

Flash Erase Block Select Parameter (FEBS)  
CPU General Register ER0
Sets the erase block number in the range from 0 to 11. Number 0 corresponds to block EB0 and number 11 corresponds to block EB11. An error occurs when a number other than 0 to 11 is set.
Setting: blk_no
5. Flowchart

```
erase_process

FMATS = H'00
Select user MAT

rtn = 0

FKEY = H'5A
Enable programming/erasing data in the flash memory

blk_no = 0
Clear the erase block no. to 0

e_pnt = *ERASE_ENT
Set the erase program address

wk_blk = ERASEBLK
Set the erase blocks in wk_blk

Erase blocks all erased?
wk_blk = 0

Yes

Erase target block?
wk_blk&0x01 == 1?

No

Yes

rtn = e_pnt(blk_no)
Execute the erase program

rtn = 0?

No

Yes

blk_no ++
Increment the erase block no.

wk_blk = (wk_blk >> 1)
Shift the erase block 1 bit to the right

FKEY = H'00
Clear the key code

FMATS = H'AA
Select user boot MAT

Return rtn

End
```
### 6.2.3 write_process Function

1. **Overview**
   Programs flash memory.

2. **Arguments**

<table>
<thead>
<tr>
<th>Type</th>
<th>Variable Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned long</td>
<td>fladr</td>
<td>Reprogramming start address</td>
</tr>
<tr>
<td>unsigned long</td>
<td>flsize</td>
<td>Reprogramming size</td>
</tr>
</tbody>
</table>

3. **Return value**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned char</td>
<td>Flash pass and fail parameter (FPFR). Return value of the programming result.</td>
</tr>
</tbody>
</table>

4. **Internal registers used**

   The internal registers used in this sample task are shown below. The setting values are the values used in this sample task, and not the initial values.

   - **Flash Key Code Register (FKEY)**  
     **Address:** H'FFFDEC

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Setting</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>K7</td>
<td>0</td>
<td>R/W</td>
<td>Key Code</td>
</tr>
<tr>
<td>6</td>
<td>K6</td>
<td>1</td>
<td>R/W</td>
<td>When H'A5 is written to FKEY, writing to the SCO bit in FCCS is enabled. When a value other than H'A5 is written, the SCO bit cannot be set to 1. Therefore, the on-chip program cannot be downloaded to the on-chip RAM. Only when H'5A is written can programming/erasing of the flash memory be executed. When a value other than H'5A is written, even if the programming/erasing program is executed, programming/erasing cannot be performed. H'A5: Writing to the SCO bit is enabled. (The SCO bit cannot be set to 1 when FKEY is a value other than H'A5.) H'5A: Programming/erasing of the flash memory is enabled. (When FKEY is a value other than H'5A, the software protection state is entered.)</td>
</tr>
<tr>
<td>5</td>
<td>K5</td>
<td>0</td>
<td>R/W</td>
<td>H'00: Initial value</td>
</tr>
<tr>
<td>4</td>
<td>K4</td>
<td>1</td>
<td>R/W</td>
<td>When H'AA is written to FMATS, the user boot MAT is selected. When a value other than H'AA is written, the user MAT is selected. In the user boot mode: H'AA: The user boot MAT is selected. A value other than H'AA: The user MAT is selected.</td>
</tr>
<tr>
<td>3</td>
<td>K3</td>
<td>1</td>
<td>R/W</td>
<td>When a value other than H'AA is written, the user MAT is selected.</td>
</tr>
<tr>
<td>2</td>
<td>K2</td>
<td>0</td>
<td>R/W</td>
<td>When H'A5 is written to FKEY, writing to the SCO bit in FCCS is enabled. When a value other than H'A5 is written, the SCO bit cannot be set to 1. Therefore, the on-chip program cannot be downloaded to the on-chip RAM. Only when H'5A is written can programming/erasing of the flash memory be executed. When a value other than H'5A is written, even if the programming/erasing program is executed, programming/erasing cannot be performed. H'A5: Writing to the SCO bit is enabled. (The SCO bit cannot be set to 1 when FKEY is a value other than H'A5.) H'5A: Programming/erasing of the flash memory is enabled. (When FKEY is a value other than H'5A, the software protection state is entered.)</td>
</tr>
<tr>
<td>1</td>
<td>K1</td>
<td>1</td>
<td>R/W</td>
<td>MAT Selection</td>
</tr>
<tr>
<td>0</td>
<td>K0</td>
<td>0</td>
<td>R/W</td>
<td>The memory MATs can be switched by writing a value to FMATS.</td>
</tr>
</tbody>
</table>

   - **Flash MAT Select Register (FMATS)**  
     **Address:** H'FFFEAD

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Setting</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>MS7</td>
<td>0</td>
<td>R/W</td>
<td>MAT Selection</td>
</tr>
<tr>
<td>6</td>
<td>MS6</td>
<td>0</td>
<td>R/W</td>
<td>The memory MATs can be switched by writing a value to FMATS.</td>
</tr>
<tr>
<td>5</td>
<td>MS5</td>
<td>0</td>
<td>R/W</td>
<td>When H'AA is written to FMATS, the user boot MAT is selected. When a value other than H'AA is written, the user MAT is selected.</td>
</tr>
<tr>
<td>4</td>
<td>MS4</td>
<td>0</td>
<td>R/W</td>
<td>H'AA: The user boot MAT is selected.</td>
</tr>
<tr>
<td>3</td>
<td>MS3</td>
<td>0</td>
<td>R/W</td>
<td>A value other than H'AA: The user MAT is selected.</td>
</tr>
<tr>
<td>2</td>
<td>MS2</td>
<td>0</td>
<td>R/W</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>MS1</td>
<td>0</td>
<td>R/W</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>MS0</td>
<td>0</td>
<td>R/W</td>
<td></td>
</tr>
</tbody>
</table>
Flash Pass and Fail Parameter (FPFR)  

Return value of the program results

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Setting</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
</table>
| 6   | MD       | Undefined | R/W | Programming Mode Related Setting Error Detect  
                  Dects the error protection state and returns the result. When  
                  the error protection state is entered, this bit is set to 1. Whether  
                  the error protection state is entered or not can be confirmed with  
                  the FLER bit in FCCS.  
                  0: Normal operation (FLER = 0)  
                  1: Error protection state, and programming cannot be performed  
                  (FLER = 1) |
| 5   | EE       | Undefined | R/W | Programming Execution Error Detect  
                  Writes 1 to this bit when the specified data could not be written  
                  because the user MAT was not erased. If this bit is set to 1, there  
                  is a high possibility that the user MAT has been written partially.  
                  In this case, after removing the error factor, erase the user MAT.  
                  0: Programming has ended normally  
                  1: Programming has ended abnormally |
| 4   | FK       | Undefined | R/W | Flash Key Register Error Detect  
                  Checks the FKEY value (H'A5) before programming starts, and  
                  returns the results.  
                  0: FKEY setting is normal (H'5A)  
                  1: FKEY setting is abnormal (a value other than H'5A) |
| 2   | WD       | Undefined | R/W | Write Data Address Detect  
                  When an address not in the flash memory area is specified as the  
                  start address of the storage destination for the program data, an  
                  error occurs.  
                  0: Setting of the start address of the storage destination for the  
                  program data is normal  
                  1: Setting of the start address of the storage destination for the  
                  program data is abnormal |
| 1   | WA       | Undefined | R/W | Write Address Error Detect  
                  When the following items are specified as the start address of the  
                  programming destination, an error occurs.  
                  • An area other than flash memory  
                  • The specified address is not aligned with the 128-byte  
                    boundary  
                    (lower eight bits of the address are other than H'00 and H'80)  
                  0: Setting of the start address of the programming destination is  
                  normal  
                  1: Setting of the start address of the programming destination is  
                  abnormal |
| 0   | SF       | Undefined | R/W | Success/Fail  
                  Indicates the programming results.  
                  0: Erasure ended normally (no error)  
                  1: Erasure ended abnormally (error occurred) |
- **Flash Multipurpose Address Area Parameter (FMPAR) CPU General Register ER1**
  FMPAR sets the start address of the programming destination on the user MAT.
  When an address in an area other than the flash memory is set, or the start address of the programming destination is not aligned with the 128-byte boundary, an error occurs. The error occurrence is indicated by the WA bit of the FPFR register.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Setting</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 to 0</td>
<td>MOA31 to MOA0</td>
<td>Fladr (local variable)</td>
<td>R/W</td>
<td>Sets the start address of the programming destination on the user MAT. Consecutive 128-byte programming is executed starting from the specified start address of the user MAT. Therefore, the specified start address of the programming destination becomes a 128-byte boundary, and MOA6 to MOA0 are always cleared to 0.</td>
</tr>
</tbody>
</table>

- **Flash Multipurpose Data Destination Parameter (FMPDR) CPU General Register ER0**
  FMPDR sets the start address in the area which stores the data to be programmed in the user MAT.
  When the storage destination for the program data is in flash memory, an error occurs. The error occurrence is indicated by the WD bit in FPFR.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Setting</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 to 0</td>
<td>MOD31 to MOD0</td>
<td>Fladr (local variable)</td>
<td>R/W</td>
<td>Sets the start address of the area which stores the program data for the user MAT. Consecutive 128-byte data is programmed to the user MAT starting from the specified start address.</td>
</tr>
</tbody>
</table>
5. Flowchart

write_process

FMATS = H'00
Select user MAT

FKEY = H'5A
Enable programming/erasing data in the flash memory

rtn = 0

w_pnt = *WRITE_ENT
Set the programming program address

i = 0

i < fsize?

Yes

No

rtn = E_WRTRUN

rtn == 0?

Yes

No

rtn

FKEY = H'00
Clear the key code

FMATS = H'AA
Select user boot MAT

Return rtn

End

rtn = w_pnt(w_adr,fladr)
Execute the programming program

fadr = fadr + WRT_NUM
Add WRT_NUM(128) bytes to the reprogramming address

rtn = w_pnt(w_adr,fladr)
Execute the programming program

rtn = E_WRTRUN

rtn = w_pnt(w_adr,fladr)
Execute the programming program

i = i + WRT_NUM

rtn

tmp = sci3_rcv1byte()
Execute the programming program

tmp ≠ STATUSREAD?

Yes

No

sci3_trs1byte(TRS128)
Transmit the 128-byte transmit request command

tmp = sci3_rcv1byte()
Receive WRT_NUM(128) bytes → store w_buf [128] storage

rtn = w_pnt(w_adr,fladr)
Execute the programming program

i < fsize?

Yes

No

rtn

rtn = 0

rtn ≠ 0?

Yes

No

rtn = E_WRTRUN

rtn = w_pnt(w_adr,fladr)
Execute the programming program
6.2.4 download Function

1. Overview
   Downloads the on-chip modules.

2. Arguments
   None

3. Return value
   Type             Description
   unsigned char    Download pass and fail result parameter (DPFR). Return value of the download result.

4. Internal registers used
   The internal registers used in this sample task are shown below. The setting values are the values used in this
   sample task, and not the initial values.

- **Flash Key Code Register (FKEY)**  Address: H'FFFDEC

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Setting</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>K7</td>
<td>1</td>
<td>R/W</td>
<td>Key Code</td>
</tr>
<tr>
<td>6</td>
<td>K6</td>
<td>0</td>
<td>R/W</td>
<td>When H'A5 is written to FKEY, writing to the SCO bit in FCCS is enabled. When a value other than H'A5 is written, the SCO bit cannot be set to 1. Therefore, the on-chip program cannot be downloaded to the on-chip RAM. Only when H'5A is written can programming/erasing of the flash memory be executed. When a value other than H'5A is written, even if the programming/erasing program is executed, programming/erasing cannot be performed. H'A5: Writing to the SCO bit is enabled. (The SCO bit cannot be set to 1 when FKEY is a value other than H'A5.) H'5A: Programming/erasing of the flash memory is enabled. (When FKEY is a value other than H'5A, the software protection state is entered.)</td>
</tr>
<tr>
<td>5</td>
<td>K5</td>
<td>1</td>
<td>R/W</td>
<td>H'00: Initial value</td>
</tr>
<tr>
<td>4</td>
<td>K4</td>
<td>0</td>
<td>R/W</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>K3</td>
<td>0</td>
<td>R/W</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>K2</td>
<td>1</td>
<td>R/W</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>K1</td>
<td>0</td>
<td>R/W</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>K0</td>
<td>1</td>
<td>R/W</td>
<td></td>
</tr>
</tbody>
</table>
## Flash Transfer Destination Address Register (FTDAR)  Address: H’FFFDEE

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Setting</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>TDER</td>
<td>0</td>
<td>R/W</td>
<td>Transfer Destination Address Setting Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This bit is set to 1 when an error has occurred in setting the start address specified by bits TDA6 to TDA0. A start address error is determined by whether the value set in bits TDA6 to TDA0 is within the range of H'00 to H'02 when download is executed by setting the SCO bit in FCCS to 1. Make sure that this bit is cleared to 0 before setting the SCO bit to 1 and the value specified by FTDAR should be within the range of H'00 to H'02.</td>
</tr>
<tr>
<td>6</td>
<td>TDA6</td>
<td>0</td>
<td>R/W</td>
<td>Transfer Destination Address</td>
</tr>
<tr>
<td>5</td>
<td>TDA5</td>
<td>0</td>
<td>R/W</td>
<td>Specifies the on-chip RAM start address of the download</td>
</tr>
<tr>
<td>4</td>
<td>TDA4</td>
<td>0</td>
<td>R/W</td>
<td>destination. A value between H'00 to H'02, and up to 4 Kbytes can be specified as the start address of the on-chip RAM.</td>
</tr>
<tr>
<td>3</td>
<td>TDA3</td>
<td>0</td>
<td>R/W</td>
<td>H'00: H'FF9000 is specified as the start address.</td>
</tr>
<tr>
<td>2</td>
<td>TDA2</td>
<td>0</td>
<td>R/W</td>
<td>H'01: H'FFA000 is specified as the start address.</td>
</tr>
<tr>
<td>1</td>
<td>TDA1</td>
<td>1</td>
<td>R/W</td>
<td>H'02: H'FFB000 is specified as the start address.</td>
</tr>
<tr>
<td>0</td>
<td>TDA0</td>
<td>0</td>
<td>R/W</td>
<td>H'03 to H'7F: Setting prohibited. (Specifying a value from H'03 to H'7F sets the TDER bit to 1 and stops download of the on-chip program.)</td>
</tr>
</tbody>
</table>
### Flash Code Control Status Register (FCCS)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Setting</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
</table>
| 4   | FLER     | 0       | R   | Flash Memory Error  
Indicates that an error has occurred during programming/erasing the flash memory. When this bit is set to 1, the flash memory enters the error protection state. When this bit is set to 1, high voltage is applied to the internal flash memory. To reduce the damage to the flash memory, the reset must be released after the reset input period (period of RES = 0) of at least 100 µs.  
0: Flash memory operates normally (Error protection is invalid)  
[Clearing condition]  
• At a power-on reset  
1: An error occurs during programming/erasing flash memory (Error protection is valid)  
[Setting conditions]  
• When an interrupt, such as NMI, occurs during programming/erasing.  
• When the flash memory is read during programming/erasing (including a vector read and an instruction fetch).  
• When the SLEEP instruction is executed during programming/erasing (including software standby mode).  
• When a bus master other than the CPU, such as the DTC and DMAC, obtains bus mastership during programming/erasing. |
| 0   | SCO*     | 0       | (R)/W | Source Program Copy Operation  
Requests the on-chip programming/erasing program to be downloaded to the on-chip RAM. When this bit is set to 1, the on-chip program which is selected by FPCS or FECS is automatically downloaded in the on-chip RAM area specified by FTDAR.  
In order to set this bit to 1, the RAM emulation mode must be canceled, H'A5 must be written to FKEY, and the setting of SCO bit must be executed in the on-chip RAM. Dummy read of FCCS must be executed twice immediately after setting this bit to 1. All interrupts must be disabled during download. This bit is cleared to 0 when download is completed.  
During program download initiated with this bit, particular processing which accompanies bank switching of the program storage area is executed.  
Before a download request, initialize the VBR contents to H'00000000. After download is completed, the VBR contents can be changed.  
0: Download of the programming/erasing program is not requested.  
[Clearing condition]  
• When download is completed  
1: Download of the programming/erasing program is requested.  
[Setting conditions] (When all of the following conditions are satisfied)  
• Not in RAM emulation mode (the RAMS bit of RAMER is cleared to 0)  
• H'A5 is written to FKEY  
• Setting of SCO bit in FCCS is executed in the on-chip RAM. |

Note: * SCO is a write-only bit. This bit is always read as 0.
Download Pass and Fail Result Parameter (DPFR)  Single Byte of Start Address in On-Chip RAM Specified by FTDAR

DPFR indicates the return value of the download result. The DPFR value is used to determine the download result.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Setting</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
</table>
| 2   | SS       | 1       | R/W | Source Select Error Detect  
|     |          |         |     | Only one type can be specified for the on-chip program which can be downloaded. When the program to be downloaded is not selected, more than two types of programs are selected, or a program which is not mapped is selected, an error occurs.  
|     |          |         |     | 0: Download program selection is normal  
|     |          |         |     | 1: Download program selection is abnormal |
| 1   | FK       | 1       | R/W | Flash Key Register Error Detect  
|     |          |         |     | Checks the FKEY value (H'A5) and returns the result.  
|     |          |         |     | 0: FKEY setting is normal (H'A5)  
|     |          |         |     | 1: FKEY setting is abnormal (value other than H'A5) |
| 0   | SF       | 1       | R/W | Success/Fail  
|     |          |         |     | Returns the download result. Reads back the program downloaded to the on-chip RAM and determines whether it has been transferred to the on-chip RAM.  
|     |          |         |     | 0: Download of the program has ended normally  
|     |          |         |     | 1: Download of the program has ended abnormally (error occurred) |
5. Flowchart

```
Flowchart

download

DPFR = H'FF
Clear DPFR

FTDAR = WKAREA
Specify the start address in the
on-chip RAM where the on-chip
program is downloaded

FKEY = H'A5
Enable writing to the SCO bit

VBR = H'00000000

SCO of FCCS = 1
tmp = FCCS (dummy read)
Request downloading of the
programming/erasing program

FKEY = H'00
Clear the key code

rtn = DPFR (downloaded result)

Return rtn (downloaded result)

End
```
6.2.5 fw_init Function

1. Overview
   Initializes flash memory before programming.

2. Arguments
   None

3. Return value
<table>
<thead>
<tr>
<th>Type</th>
<th>Variable Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned char</td>
<td>rtn</td>
<td>Flash pass and fail parameter (FPFR). Return value of the initialization result.</td>
</tr>
</tbody>
</table>

4. Internal registers used
   The internal registers used in this sample task are shown below. The setting values are the values used in this sample task, and not the initial values.

   - **Flash Program/Erase Frequency Parameter (FPEFEQ) CPU General Register ER0**
     FPEFEQ sets the operating frequency of the CPU. The CPU operating frequency available in this LSI ranges from 8 MHz to 48 MHz.

     | Bit  | Bit Name | Setting | R/W | Description |
     |------|----------|---------|-----|-------------|
     | 15 to 0 | F15 to F0 | CLOCK   | R/W | Frequency Set |

     Sets the operating frequency of the CPU. When the PLL multiplication function is used, set the multiplied frequency. The setting value must be calculated as follows:
     
     - The operating frequency shown in MHz units must be rounded in a number of three decimal places and be shown in a number of two decimal places.
     
     - The value multiplied by 100 is converted to the binary digit and is written to FPEFEQ (general register ER0). For example, when the operating frequency of the CPU is 35.000 MHz, the value is as follows:

     1. The number of three decimal places of 35.000 is rounded.
     2. The formula of 35.00 × 100 = 3500 is converted to the binary digit and B'0000 1101 1010 1100 (H'0CE4) is set to ER0.

   - **Flash Pass and Fail Parameter (FPFR) CPU General Register R0L**
     Return value of the initialization results

     | Bit | Bit Name | Setting | R/W | Description |
     |-----|----------|---------|-----|-------------|
     | 1   | FQ       | Undefined | R/W | Frequency Error Detect |

     Compares the specified CPU operating frequency with the operating frequencies supported by this LSI, and returns the result.

     0: Setting of operating frequency is normal
     1: Setting of operating frequency is abnormal

     | 0   | SF       | Undefined | R/W | Success/Fail |

     Returns the initialization result.

     0: Initialization has ended normally (no error)
     1: Initialization has ended abnormally (error occurred)
5. Flowchart

```
fw_init

rtn = 0

clk = CLOCK
Set the input clock

i_pnt = *INIT_ENT
Set the programming program address

rtn = w_pnt(w_adr,fladr)
Execute the programming program

Return rtn

End
```
7. Description of Software for the Clock Synchronous Serial Communications Program on the Slave Side

7.1 List of Functions

The clock synchronous serial communications program (sci3.c) performs communications processing to the master side. Table 12 is a list of functions in the clock synchronous serial communications program, and figure 15 shows the hierarchical structure.

Table 12 Functions in the Clock Synchronous Serial Communications Program

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sci3_init</td>
<td>Initializes clock synchronous serial communications</td>
</tr>
<tr>
<td>sci3_rcv1byte</td>
<td>Receives one byte of data</td>
</tr>
<tr>
<td>sci3_rcvnb1ye</td>
<td>Receives n bytes of data</td>
</tr>
<tr>
<td>sci3_trs1byte</td>
<td>Transmits one byte of data</td>
</tr>
<tr>
<td>sci3_trsnbyte</td>
<td>Transmits n bytes of data</td>
</tr>
</tbody>
</table>

Figure 15 Clock Synchronous Serial Communications Program
7.2 Description of Modules

7.2.1 sci3_init Function

1. Overview
   Initializes clock synchronous serial communications.

2. Arguments
   None

3. Return value
   None

4. Internal registers used
   The internal registers used in this sample task are shown below. The setting values are the values used in this sample task, and not the initial values.

- **Serial Mode Register_3 (SMR_3)**  Address: H'FFFE88

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Setting</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>C/A</td>
<td>0</td>
<td>R/W</td>
<td>Communications Mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Asynchronous mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Clocked synchronous mode</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Setting</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>CHR</td>
<td>0</td>
<td>R/W</td>
<td>Character Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Selects 8 bits as the data length.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Selects 7 bits as the data length.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Setting</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CKS1</td>
<td>0</td>
<td>R/W</td>
<td>Clock Selection 1, 0</td>
</tr>
<tr>
<td>0</td>
<td>CKS0</td>
<td>0</td>
<td>R/W</td>
<td>B'00: Clock source of the on-chip baud rate generator is set to Pphasis clock.</td>
</tr>
</tbody>
</table>

- **Serial Control Register_3 (SCR_3)**  Address: H'FFFE8A

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Setting</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>TE</td>
<td>0</td>
<td>R/W</td>
<td>Transmit Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disables transmission</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enables transmission</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Setting</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>RE</td>
<td>0</td>
<td>R/W</td>
<td>Receive Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disables reception</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enables reception</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Setting</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CKE1</td>
<td>0</td>
<td>R/W</td>
<td>Clock Selection 1, 0</td>
</tr>
<tr>
<td>0</td>
<td>CKE0</td>
<td>0</td>
<td>R/W</td>
<td>In the clock synchronous mode:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B'00: Internal clock is used for clock source, and SCK3 pin is set to clock output pin</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B'1X: External clock is used for clock source, and SCK3 pin is set to clock input pin</td>
</tr>
</tbody>
</table>

Legend
X: Don't care
### Serial Status Register_3 (SSR_3)  Address: H'FFFE8C

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Setting</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
</table>
| 7   | TDRE       | Undefined | R/(W)* | Transmit Data Register Empty
0: Transmit data written to TDR is not transferred to TSR
1: Transmit data is not written to TDR, or transmit data written to TDR is not transferred to TSR |
| 6   | RDRF       | Undefined | R/(W)* | Receive Data Register Full
0: No receive data is stored in RDR.
1: The receive data is stored in RDR. |
| 5   | ORER       | 0        | R/(W)* | Overrun Error
0: No overrun error
1: Overrun error occurred during receive operation |
| 2   | TEND       | Undefined | R | Transmit End
0: In transmission
1: Transmission completed |

Note: * Only 0 can be written here, to clear the flags for TDRE, RDRF, and ORER.

### Port 1 Input Buffer Control Register (P1ICR)  Address: H'FFFB90

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Setting</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
</table>
| 6   | P16ICR   | 1       | R/W | 0: Input buffer for P16 (SCK3) pin is disabled
1: Input buffer for P16 (SCK3) pin is enabled |
| 5   | P15ICR   | 1       | R/W | 0: Input buffer for P15 (RxD3) pin is disabled
1: Input buffer for P15 (RxD3) pin is enabled |

### Smart Card Mode Register_3 (SCMR_3)  Address: H'FFFE8E

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Setting</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
</table>
| 0   | SMIF     | 0       | R/W | Smart Card Interface Mode Select
0: Operates in normal asynchronous or clock synchronous mode
1: Operates in smart card interface mode |
5. Flowchart

```
sci3_init

SCR_3 & = H'CF
Disable transmission/reception

P16ICR = 1
P15ICR = 1

SCR3 = H'02
Set SCK as the clock input pin

SMR = H'80
SCMR = H'F2
Clock synchronous mode
8-bit data length
On-chip baud rate generator = φ clock

Clear ORER, FER, PER of SSR to 0

End
```
### 7.2.2 sci3_rcv1byte Function

1. **Overview**
   Receives one byte of clock synchronous serial data.

2. **Arguments**
   None

3. **Return value**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned char</td>
<td>Receives one byte of data</td>
</tr>
</tbody>
</table>

4. **Internal registers used**
   The internal registers used in this sample task are shown below. The setting values are the values used in this sample task, and not the initial values.

- **Serial Control Register_3 (SCR_3)** Address: H'FFFE8A

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Setting</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>RE</td>
<td>0</td>
<td>R/W</td>
<td>Receive Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disables reception</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enables reception</td>
</tr>
</tbody>
</table>

- **Serial Status Register_3 (SSR_3)** Address: H'FFFE8C

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Setting</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>RDRF</td>
<td>Undefined</td>
<td>R/(W)</td>
<td>Receive Data Register Full</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No receive data is stored in RDR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: The receive data is stored in RDR</td>
</tr>
<tr>
<td>5</td>
<td>ORER</td>
<td>Undefined</td>
<td>R/(W)</td>
<td>Overrun Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No overrun error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Overrun error occurred during reception</td>
</tr>
</tbody>
</table>

Note: Only 0 can be written here, to clear the flags for RDRF and ORER.

- **Receive Data Register_3 (RDR_3)** Address: H'FFFE8D

  Function: An 8-bit register for storing receive data.

  Setting: Undefined
5. Flowchart

```
sci3_rcv1byte

RE = 1

Yes

ORER of SSR3 == 1?

No

tmp ← RDR of SSR3

tmp = 0 ?

tmp ← RDRF of SSR3

tmp = 0?

No

Yes

End

sci3_rcv1byte

RE = 1

Yes

ORER = 1

No

tmp ← RDR of SSR3

RDRF = 0

RE = 0

Return tmp (received data)
```

7.2.3 sci3_rcvnbyte Function

1. Overview
   Receives n byte of clock synchronous serial data.

2. Arguments

<table>
<thead>
<tr>
<th>Type</th>
<th>Variable Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned char</td>
<td>dtno</td>
<td>No. of receive bytes</td>
</tr>
<tr>
<td>unsigned char</td>
<td>*ram</td>
<td>The start address of RAM in which the receive data is stored</td>
</tr>
</tbody>
</table>

3. Return value
   None

4. Internal registers used
   The internal registers used in this sample task are shown below. The setting values are the values used in this sample task, and not the initial values.

- **Serial Control Register_3 (SCR_3)** Address: H'FFFE8A

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Setting</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>RE</td>
<td>0</td>
<td>R/W</td>
<td>Receive Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disables reception</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enables reception</td>
</tr>
</tbody>
</table>

- **Serial Status Register_3 (SSR_3)** Address: H'FFFE8C

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Setting</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>RDRF</td>
<td>Undefined</td>
<td>R/(W)</td>
<td>Receive Data Register Full</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No receive data is stored in RDR.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: The receive data is stored in RDR.</td>
</tr>
<tr>
<td>5</td>
<td>ORER</td>
<td>Undefined</td>
<td>R/(W)</td>
<td>Overrun Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No overrun error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Overrun error occurred during receive operation</td>
</tr>
</tbody>
</table>

Note: Only 0 can be written here, to clear the flags for RDRF and ORER.

- **Receive Data Register_3 (RDR_3)** Address: H'FFFE8D

  Function: An 8-bit register for storing receive data.
  Setting: Undefined
5. Flowchart

```
sci3.rcvbyte
RE = 1

dtmo > 0? or
specified bytes completed?

RE = 1

dtmo > 0

ORER of SSR3 == 1?
No

tmp ← RDRF of SSR3

tmp = 0
Yes

tmp == 0?
No

tmp ← RDR

RDRF = 0

*ram = tmp

*ram ++
dtmo --

Yes

ORER = 1

dtmo = 0

RE = 0

End
```
7.2.4 sci3_trs1byte Function

1. Overview
   Receives one byte of clock synchronous serial data.

2. Arguments

<table>
<thead>
<tr>
<th>Type</th>
<th>Variable Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned char</td>
<td>tdt</td>
<td>Transmits one byte of data</td>
</tr>
</tbody>
</table>

3. Return value
   None

4. Internal registers used
   The internal registers used in this sample task are shown below. The setting values are the values used in this sample task, and not the initial values.

- **Serial Control Register_3 (SCR_3)** Address: H'FFFE8A
  
<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Setting</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>TE</td>
<td>0</td>
<td>R/W</td>
<td>Transmit Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disables transmission</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enables transmission</td>
</tr>
</tbody>
</table>

- **Transmit data Register_3 (TDR_3)** Address: H'FFFE8B
  Function: An 8-bit register for storing transmit data.
  Setting: Undefined

- **Serial Status Register_3 (SSR_3)** Address: H'FFFE8C
  
<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Setting</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>TDRE</td>
<td>Undefined</td>
<td>R/(W)</td>
<td>Transmit Data Register Empty</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Indicates that the transmit data written to TDR is not transferred to TSR.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Indicates that transmit data is not written to TDR or the transmit data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>written to TDR is sent to TSR.</td>
</tr>
<tr>
<td>2</td>
<td>TEND</td>
<td>Undefined</td>
<td>R</td>
<td>Transmit End</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: In transmission.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Transmission is completed.</td>
</tr>
</tbody>
</table>

Note: * Only 0 can be written here, to clear the flag for TDRE.
5. Flowchart

- sci3_trs1byte
- TE = 1
- TDRE == 0?
  - Yes: Data in TDR
  - No: No data in TDR
- TDR = tdt
- TDRE = 0
- TEND == 0?
  - Yes: In transmission
  - No: Transmission completed
- TE = 0
- End
7.2.5 sci3_trnbyte Function

1. Overview
   Transmits n bytes of clock synchronous serial data.

2. Arguments

<table>
<thead>
<tr>
<th>Type</th>
<th>Variable Name</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned short</td>
<td>dtno</td>
<td>Transmit size</td>
</tr>
<tr>
<td>unsigned char</td>
<td>*tdt</td>
<td>Start address of the transmit data</td>
</tr>
</tbody>
</table>

3. Return value
   None

4. Internal registers used
   The internal registers used in this sample task are shown below. The setting values are the values used in this sample task, and not the initial values.

• **Serial Control Register_3 (SCR_3)** Address: H'FFFE8A

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Setting</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>TE</td>
<td>0</td>
<td>R/W</td>
<td>Transmit Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disables transmission</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enables transmission</td>
</tr>
</tbody>
</table>

• **Transmit data Register_3 (TDR_3)** Address: H'FFFE8B
   Function: An 8-bit register for storing transmit data.
   Setting: Undefined

• **Serial Status Register_3 (SSR_3)** Address: H'FFFE8C

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Setting</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>TDRE</td>
<td>Undefined</td>
<td>R/(W)*</td>
<td>Transmit Data Register Full</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Indicates that the transmit data written to TDR is not transferred to TSR.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Indicates that transmit data is not written to TDR or the transmit data written to TDR is sent to TSR.</td>
</tr>
<tr>
<td>2</td>
<td>TEND</td>
<td>Undefined</td>
<td>R</td>
<td>Transmit End</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: In transmission.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Transmission is completed.</td>
</tr>
</tbody>
</table>

Note: * Only 0 can be written here, to clear the flag for TDRE.
5. Flowchart

- sci3_trnbyte
- TE = 1
- if dtno > 0? or specified bytes completed?
  - Yes: Data in TDR
    - TDR = *tdt
    - TDRE = 0
    - *tdt ++
    - dtno --
  - No: No data in TDR
    - TDRE == 0?
      - Yes: TDR = *tdt
      - No: TDRE = 0
      - *tdt ++
      - dtno --
- if TEND == 0?
  - Yes: In transmission
    - TE = 1
  - No: Transmission completed
    - TE = 0
    - End
8. Documents for Reference (Note)

- Hardware Manual
  H8SX/1582 Group Hardware Manual
  The most up-to-date version of this document is available on the Renesas Technology Website.

- Technical News/Technical Update
  The most up-to-date information is available on the Renesas Technology Website.
Website and Support

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

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

Revision Record

<table>
<thead>
<tr>
<th>Rev.</th>
<th>Date</th>
<th>Page</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>Mar.10.06</td>
<td>—</td>
<td>First edition issued</td>
</tr>
<tr>
<td>1.01</td>
<td>Sep.25.07</td>
<td>6</td>
<td>Page 6: An additional item, optimizing linkage editor option, in Section 2, Applicable Conditions</td>
</tr>
</tbody>
</table>
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 microcomputersoftware 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.