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](http://www.renesas.com)

April 1st, 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.
H8S/20103, H8S/20203, and H8S/20223 Groups
Reprogramming Data-Flash Memory in EW1 Mode

Introduction
Data-flash areas within products of the H8S/20103, H8S/20203, and H8S/20223 Groups can be reprogrammed in EW1 mode.

Target Devices
H8S/20103 (R4F20103)
H8S/20203 (R4F20203)
H8S/20223 (R4F20223)

Frequency Used in Confirming Operation
System clock $\phi = \phi_{osc} = 20$ MHz

Contents
1. Specifications ...................................................................................................................... 2
2. Description of Module Used .......................................................................................... 4
3. Principle of Operation ..................................................................................................... 24
4. Description of Software ................................................................................................. 25
5. Flowcharts ...................................................................................................................... 30
6. Program Listing .............................................................................................................. 39
1. Specifications

Specifications of this sample task are as given below. Figure 1 shows an overview of the reprogramming of data-flash memory. In this application note, the first 128 bytes in data-flash area A are reprogrammed.

(1) The 4 Kbytes in data-flash area A are backed up by being transferred to a RAM area.
(2) Data-flash area A is erased.
(3) Blank checking is executed for data-flash area A.
(4) Data in the first 128 bytes of the RAM area where the back-up was made are reprogrammed.
(5) After reprogramming, the 4-Kbytes that have been backed-up in the RAM area are written to data-flash area A.
(6) Steps 1 to 5 accomplish reprogramming of the first 128 bytes of data-flash area A.
### Reconfiguring Data Flash in EW1 Mode

#### Backup Data
- H'000000
- H'003FFF
- H'004000
- H'00BFFF
- H'00C000
- H'013FFF
- H'014000
- H'01BFFF
- H'01C000
- H'01FFFF
- H'F00000
- H'F00FFF
- H'F01000
- H'F01FFF

#### Flashblock 1
- Data flash A: H'0F00000
- Data flash B: H'0F00FFF
- Backup data: H'0F01000
- RAM: H'FFDF80

#### Flashblock 2
- Interrupt vectors and reprogramming-control program

#### Flashblock 3
- Erasure
- H'FF at all locations is confirmed.

#### Flashblock 4
- Backup data are transferred to RAM.
- Data flash A is erased.

#### Flashblock 5
- Blank checking of data flash A is executed.

### Figure 1 Overview of Reprogramming the Data Flash in EW1 Mode

- The reprogrammed data and backup data are written to data flash A.
- Some of the backup data are reprogrammed.
2. Description of Module Used

2.1 ROM

The features of the on-chip flash memory are described below.

- Programming/erasing method
  Four bytes are programmed simultaneously. A single block is erased at a time; only one block should be erased at a time even when the entire ROM area is to be erased.

- Programming/erasing time
  Program ROM programming time: 150 µs (typ.) for 4-byte simultaneous programming, i.e., 38 µs (typ.) per byte
  Data-flash programming time: 300 µs (typ.) for 4-byte simultaneous programming, i.e., 75 µs (typ.) per byte
  Erasing time: 200 ms (typ.) per block for the program ROM and data-flash areas.

- Reprogramming capability
  The program ROM area can be reprogrammed up to 1,000 times and the data-flash area can be reprogrammed up to 10,000 times.

- Two on-board programming modes
  Boot mode: The on-chip SCI can be used for programming/erasing the user ROM area. In this mode, the communications bit rate between the host and this LSI can be automatically adjusted.
  User mode: Any interface can be used for programming/erasing the user ROM area.

- Programmer mode
  A PROM programmer is used for programming/erasing.

- Protection function
  Flash memory can be protected against erroneous programming and erasure.
  Lock-bit protection function can be set through software.

- PROM-programmer protection/Boot-mode protection
  By writing specified data to a specified address range in user ROM, protection of the user-ROM area in boot mode and PROM-programmer mode can be established.

- Access cycle
  Program ROM: One state
  Data flash: Two states
### 2.1.1 Block Configuration

Figure 2 shows the blocks of the flash memory. The user ROM area contains the program ROM area for storing the microcomputer’s operating program and the data-flash area for storing data. In the figure, each thick-line frame indicates an erasure block (erasing unit); each thin-line frame indicates a programming unit. The values in the frames are addresses. Erasure is done in units of the erasure blocks shown in figure 2. Programming is done in 2-word (4-byte) units at addresses with the lower-order four-bit values H’0, H’4, H’8, or H’C.

![Block Configuration of Flash Memory](image)

**Figure 2 Block Configuration of Flash Memory**
2.1.2 CPU Reprogramming Mode

In CPU reprogramming mode, the user ROM area can be reprogrammed by executing the software commands by the CPU. The software commands should be issued to the specific area to be reprogrammed in the user ROM area.

If an interrupt is requested during erasure operation in CPU reprogramming mode, erasure can be suspended to process the interrupt. This is referred to as erase-suspend function. In erase-suspend mode, the user ROM area can be read through programming.

CPU reprogramming is performed in either of two modes, EW0 mode and EW1 mode. Table 1 shows differences between the two modes.

<table>
<thead>
<tr>
<th>Table 1 Differences between EW0 Mode and EW1 Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
</tr>
<tr>
<td>Area in which a reprogramming-control program can be located</td>
</tr>
<tr>
<td>Area in which a reprogramming-control program can be executed</td>
</tr>
<tr>
<td>Area which can be reprogrammed</td>
</tr>
<tr>
<td>Limitations on software commands</td>
</tr>
<tr>
<td>Mode after software command execution</td>
</tr>
<tr>
<td>CPU state during auto-programming and auto-erasure</td>
</tr>
<tr>
<td>Flash memory state detection</td>
</tr>
<tr>
<td>Conditions of transition to erase-suspend state</td>
</tr>
</tbody>
</table>
| Conditions of Interrupt generation              | • The flash memory returns from the busy state to the ready state\(^1\).  
|                                               | • The user ROM area is read in the busy state\(^1\). | Use of interrupts is prohibited. |
| Usage of DTC                                    | Usable\(^2\)                                   | Usable\(^2\)                                   |

Notes:
1. To avoid the generation of access to the user ROM area, set VOFR so that the variable vectors and interrupt processing routines are allocated to RAM.
2. Allocate DTC vectors and processing routines to RAM. Do not use the DTC for access to the user ROM area during E/W processing. If this is ignored, values read will be invalid.
3. Do not use the DTC if the reprogramming-control program is allocated to RAM.
2.1.3 EW0 Mode

EW0 mode can be selected by transferring the reprogramming-control program to the RAM, branching to the program in the RAM, setting the FMEWMOD bit in FLMCR1 to 0, and setting the FMCMDEN bit in FLMCR1 to 1 (to enable software commands), in this order.

Programming and erasure operations can be controlled through software commands. Completion of the software command and related information can be read out from the FLMSTR register.

To cause a transition to erase-suspend mode during erasure, set both the FMSPEN and FMSPREQ bits in FLMCR2 to 1 (to enable a transition to erase-suspend mode and to request a transition to erase-suspend mode, respectively). Then wait for the transition time to erase-suspend mode (approximately 50 μs), check that the FMRDY bit in FLMSTR is 1 (ready state), and access the user ROM area. Setting the FMSPREQ bit to 0 resumes erasure.

When the interrupt is used, set the interrupt vector offset register (VOFR) such that access to the user ROM area is not generated. That is, the vectors should be at addresses within the RAM and point to interrupt processing routines that are also in the RAM.

2.1.4 EW1 Mode

EW1 mode can be selected by setting the FMEWMOD bit in FLMCR1 to 1, and then setting the FMCMDEN bit in FLMCR1 to 1 (to enable software commands).

Programming and erasure operations can be controlled through software commands. Completion of the software command and related information can be read out from the FLMSTR register.

To cause a transition to erase-suspend mode during erasure, set the FMSPEN bit in FLMCR2 to 1 (to enable a transition to erase-suspend mode), and then execute the erasure command. Note that the interrupt for causing a transition to erase-suspend mode must be enabled beforehand. This allows the interrupt request to be accepted when the transition time to erase-suspend mode has elapsed after the erasure command is executed.

When an interrupt is requested, the FMSPREQ bit is automatically set to 1 (to request a transition to erase-suspend mode), thus suspending erasure. If erasure has not been completed at the end of interrupt processing (FMERCF = 1 in FLMSTR), resume erasure by setting the FMSPREQ bit to 0.
2.1.5 Programming/Erasing

In what is termed CPU reprogramming, the CPU executes software commands to program and erase flash memory on board.

2.1.6 Software Commands

Table 2 shows a list of the software commands with word-length instructions and table 3 shows a list of the software commands with byte-length instructions. Whether an instruction is to be byte-length or word-length is specified by the FMWUS bit in FLMCR1.

Table 2  Software Commands (Word Length: FMWUS = 1)

<table>
<thead>
<tr>
<th>Software Command</th>
<th>First Command Cycle</th>
<th>Second Command Cycle</th>
<th>Third Command Cycle</th>
<th>Command Use in Modes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mode</td>
<td>Addr.</td>
<td>Data</td>
<td>Mode</td>
</tr>
<tr>
<td>Erasure</td>
<td>Write</td>
<td>×</td>
<td>H'2020</td>
<td>Write</td>
</tr>
<tr>
<td>Programming</td>
<td>Write</td>
<td>WA</td>
<td>H'4141</td>
<td>Write</td>
</tr>
<tr>
<td>Blank checking</td>
<td>Write</td>
<td>×</td>
<td>H'2525</td>
<td>Write</td>
</tr>
<tr>
<td>Lock-bit programming</td>
<td>Write</td>
<td>×</td>
<td>H'7777</td>
<td>Write</td>
</tr>
<tr>
<td>Read-array</td>
<td>Write</td>
<td>×</td>
<td>H'FFFF</td>
<td></td>
</tr>
<tr>
<td>Clear-status</td>
<td>Write</td>
<td>×</td>
<td>H'5050</td>
<td></td>
</tr>
<tr>
<td>Lock-bit reading</td>
<td>Write</td>
<td>×</td>
<td>H'7171</td>
<td>Read</td>
</tr>
</tbody>
</table>

[Legend]

×: Arbitrary address in the user ROM area
xx: Eight-bit arbitrary data
BA: Arbitrary address in the target block
WA: Programming address. (The lower two bits of an address are ignored. WA should be the same in each command cycle.)
WDn: Programming data (16 bits)
### Table 3: Software Commands (Byte Length: FMWUS = 0)

<table>
<thead>
<tr>
<th>Software Command</th>
<th>First Command Cycle</th>
<th>Second Command Cycle</th>
<th>Third Command to Fifth Command Cycle</th>
<th>Command Use in Modes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mode</td>
<td>Addr.</td>
<td>Data</td>
<td>Mode</td>
</tr>
<tr>
<td>Erasure</td>
<td>Write</td>
<td>×</td>
<td>H'20</td>
<td>Write</td>
</tr>
<tr>
<td>Programming</td>
<td>Write</td>
<td>WA</td>
<td>H'41</td>
<td>Write</td>
</tr>
<tr>
<td>Blank checking</td>
<td>Write</td>
<td>×</td>
<td>H'25</td>
<td>Write</td>
</tr>
<tr>
<td>Lock-bit programming</td>
<td>Write</td>
<td>×</td>
<td>H'77</td>
<td>Write</td>
</tr>
<tr>
<td>Read-array</td>
<td>Write</td>
<td>×</td>
<td>H'FF</td>
<td></td>
</tr>
<tr>
<td>Clear-status</td>
<td>Write</td>
<td>×</td>
<td>H'50</td>
<td></td>
</tr>
<tr>
<td>Lock-bit reading</td>
<td>Write</td>
<td>×</td>
<td>H'71</td>
<td>Read</td>
</tr>
</tbody>
</table>

**[Legend]**
- ×: Arbitrary address in the user ROM area
- xx: Eight-bit arbitrary data
- BA: Arbitrary address at the target block
- WA: Programming address. (The lower two bits of an address are ignored. WA should be the same in each command cycle.)
- WDn: Programming data (8 bits)
(1) Initialization for CPU Reprogramming

Before software commands are issued, settings for CPU reprogramming mode must be made and issuing of software commands must be permitted.

Figures 3 and 4 show initialization for CPU reprogramming mode.

Flow of initialization for EW0 mode*1

Start

1. Transfer the reprogramming-control program to RAM.
2. Set the interrupt vector offset by VOFR and place the interrupt vectors in RAM.*1
3. Jump to the reprogramming-control program in RAM.

FMEWMOD = 0

FMCMDEN = 1

Start

1. Command issued for the program area?
   No
   FMLBD = 1
   Yes
   Command issued for data flash?
   No
   DFPR[x] = 0*2
   Yes

To processing for issuing commands

Notes: For any interrupts that are in use, allocate the interrupt vector entries and interrupt routines to RAM. If interrupts are not to be used, allocation to RAM is not necessary.

1. Within the flow, set the CPU reprogramming unit select bit (FMWUS) to select the size of instructions.
2. Set the DFPR according to the area of data-flash memory for which commands are to be issued.

Figure 3  Initialization for EW0 Mode
Flow of initialization for EW1 mode

Start

FMEWMOD = 1

FMCMDEN = 1

Command issued for the program area?

Yes

FMLBD = 1

Command issued for data flash?

Yes

DFPR[x] = 0

No

To processing for issuing commands

Notes: For any interrupts that are in use, allocate the interrupt vector entries and interrupt routines to RAM. If interrupts are not to be used, allocation to RAM is not necessary.
1. Within the flow, set the CPU reprogramming unit select bit (FMWUS) to select the unit of overwriting.
2. Set the DFPR according to the area of data-flash memory for which commands are to be issued.

Figure 4  Initialization for EW1 Mode
(2) Erasure

When H'20 is written in the first command cycle and H'D0 is written to any address in the block in the second command cycle, erase/erase-verify of the specified block is automatically started. Completion of erasure is indicated by the FMRDY bit in FLMSTR. The FMRDY bit is read as 0 during erasure, and read as 1 after erasure has been completed.

When erasure has been completed, the erasure result can be checked by reading the FMEBSF bit in FLMSTR. (See the description in (9) below, Full Status Checking.)

Note that if the lock bit is 0 (locked) in the specified block and the FMLBE bit is 0 (lock bit enabled), an erasure command is not accepted for the specified block. Figure 5 to 7 are flowcharts of erasure, with figure 5 depicting the case where the erase-suspend function is not in use and figures 6 and 7 depicting the cases where it is in use.

When the erase-suspend function is being employed and erasure is resumed immediately after a period in erase-suspend mode, instruction fetching with normal incrimination of the program counter will not be possible. To avoid this problem, add two NOP instructions immediately after the instruction that writes FMSPREQ = 0. Furthermore, do not use the DTC when erasure has been suspended in EW1 mode and the reprogramming control program has been allocated to RAM.

In EW1 mode, do not execute this command for the block in which the reprogramming-control program is located. The FMRDY bit in FLMSTR changes to 0 when erasure is started, and changes to 1 when completed.

Figure 5  Flowchart of Erasure When the Erase-Suspend Function is Not in Use
Write software command H’20.
Write H’D0 to any address in the specified block.

FMRDY = 1?
Yes
Erasure end

No
Full status checking

FMSREQ = 1

Yes
Access to flash memory

No
RTE

Notes:
1. In EW0 mode, set VOFR to allocate the interrupt vector table to RAM, and ensure that the interrupt routines for any interrupts that are in use are also in RAM.
2. When an interrupt request is generated, it takes the transition time to erase-suspend mode for the request to be accepted. To allow a transition to the erase-suspend state, enable the relevant interrupt beforehand.

Figure 6 Flowchart of Erasure in EW0 Mode When the Erase-Suspend Function is in Use
Write software command H'20.
Write H'D0 to any address in the specified block.

!(FMSPREQ = 0)

NOP
NOP

(FMRDY = 1?)
Yes
Full status checking
Erasure end

Note: 1. When an interrupt request is generated, it takes the transition time to erase-suspend mode for the request to be accepted. To allow a transition to the erase-suspend state, enable the relevant interrupt beforehand.

Figure 7 Flowchart of Erasure in EW1 Mode When the Erase-Suspend Function is in Use
(3) Programming

A program command is used to program data in flash memory in 4-byte units. Command or data size can be set depending on the FMWUS bit in FLMCR1. Setting the FMWUS bit to 0 enables using byte-length instructions. When H’41 is written in the first command cycle and data are written to the programming address in the second through fifth command cycles, programming and verifying are automatically started*.

Setting the FMWUS bit to 1 enables using word-length instructions. When H’4141 is written in the first command cycle and data are written to the programming address in the second and third command cycles, programming and verifying are started*.

Completion of programming is indicated by the FMRDY bit in FLMSTR. The FMRDY bit is read as 0 during programming, and as 1 once programming has been completed.

When programming has been completed, the results of programming can be checked by reading the FMPRSF bit in FLMSTR (see the description under (9), Full Status Checking below). Figure 8 shows the programming flowchart.

Do not attempt further programming at addresses that have already been programmed.

Note that if the lock bit is 0 (locked) in the specified block and the FMLBD bit is 0 (lock bit enabled), a programming command is not accepted for the specified block.

In EW1 mode, do not execute this command for the block in which the reprogramming-control program is located.

The FMRDY bit in FLMSTR changes to 0 when programming is started, and changes to 1 when completed.

Note: * The lower two bits of the programming addresses are ignored.

---

**Figure 8 Programming Flowchart**
(4) Blank Checking
When H'25 is written in the first command cycle and H'D0 is written to any address in the block in the second command cycle, blank checking of the specified block is started.
Completion of blank checking is indicated by the FMRDY bit in FLMSTR. The FMRDY bit is read as 0 during blank checking, and read as 1 after blank checking has been completed.
When blank checking has been completed, the results of blank checking can be checked by reading the FMEBSF bit in FLMSTR (see the description under (9), Full Status Checking below).
Figure 9 shows the blank checking flowchart.
The FMRDY bit in FLMSTR changes to 0 when blank checking is started, and changes to 1 when completed.

Figure 9   Blank Checking Flowchart
(5) Lock-Bit Programming

When H’77 is written in the first command cycle and H’D0 is written to any address in the block in the second command cycle, lock-bit programming of the specified block is started.

Completion of lock-bit programming is indicated by the FMRDY bit in FLMSTR. The FMRDY bit is read as 0 during lock-bit programming, and read as 1 after lock-bit programming has been completed.

When lock-bit programming has been completed, the lock-bit programming result can be checked by reading the FMPRSF bit in FLMSTR (see the description under (9), Full Status Checking below).

Figure 10 shows the lock-bit programming flowchart.

The FMRDY bit in FLMSTR changes to 0 when lock-bit programming is started, and changes to 1 when completed.

![Lock-Bit Programming Flowchart](image-url)
(6) Read-Array Command
A read-array command is to cause a transition to a mode in which data can be read from flash memory. When H'FF is written in the first command cycle, a transition to read array mode is caused. When the specified addresses are read out in the subsequent command cycles, data are read from the specified addresses. Since read-array mode is retained until any other command is written, multiple addresses can be read successively.

(7) Lock-Bit Reading Command
A lock-bit reading command is to cause a transition to a mode in which the lock bit in flash memory can be read. When H'71 is written in the first command cycle and H'D0 is written to any address in the block in the second command cycle, lock-bit reading of the specified block is started. After transition to lock-bit read mode, reading the specified block address BA returns the lock-bit value in the bit 14 value to be read. Do not execute a lock-bit read command in the ROM.

(8) Status Clearing Command
A clear-status command is used to clear the status flag to 0. When H'50 is written in the first command cycle, the FMPRSF and FMEBSF bits in FLMSTR are cleared to 0.

(9) Full Status Checking
When any command (other than the read-array command, lock-bit reading command and status clearing command) is issued, full-status checking is performed to confirm whether or not there was an error. When an error occurs, the FMPRSF and FMEBSF bits in FLMSTR are set to 1, indicating the occurrence of the relevant errors.
Table 4 shows the bit values in FLMSTR and the corresponding errors. Figure 11 shows the full status checking flowchart and figure 12 shows procedures for handling each error.
### Table 4 Bit Values in FLMSTR and Corresponding Errors

<table>
<thead>
<tr>
<th>Bit Values in FLMSTR</th>
<th>Error</th>
<th>Error Generation Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>FMEBSF 0  FMPRSF 0</td>
<td>None (normal end)</td>
<td>—</td>
</tr>
<tr>
<td>FMEBSF 0  FMPRSF 1</td>
<td>Programming error</td>
<td>The programming command is executed and results in unsuccessful programming.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lock-bit programming error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The lock-bit programming command is executed and results in unsuccessful programming.</td>
</tr>
<tr>
<td>FMEBSF 1  FMPRSF 0</td>
<td>Erasure error</td>
<td>The erasure command is executed and results in unsuccessful erasure.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Blank checking error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The blank checking command is executed and it is detected that the specified block is not blank.</td>
</tr>
<tr>
<td>FMEBSF 1  FMPRSF 1</td>
<td>Command sequence error</td>
<td>• A command is not written correctly.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A data value other than H'D0 and H'FF is written in the last cycle of the command that consists of two command cycles.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The erasure command is input in erase-suspend mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The programming command is input for the suspended block in erase-suspend mode.</td>
</tr>
</tbody>
</table>

---

**Figure 11 Full Status Checking Flowchart**
Command sequence error

- Execute the status clearing command. (Clear the status flag to 0).
- Check that the command is input correctly.
- The erasure command is re-executed.

Erasure error

- Execute the status clearing command. (Clear the status flag to 0).
- Has the erase command been re-executed three or less times?
- The target block for erasure is unavailable.
- The erasure command is re-executed.

Programming error

- Execute the status clearing command. (Clear the status flag to 0).
- Specify the different address from the address having caused an error as the programming address.
- The erasure command is re-executed.

Lock-bit programming error

- Execute the status clearing command. (Clear the status flag to 0).
- Has the lock-bit programming command been executed 1,000 or less times in total?
- The target block is unavailable.
- The lock-bit programming command is re-executed.

Figure 12 Procedures for Handling Errors
2.1.7 Protection
Three modes are available to protect the flash memory against reading, programming, and erasing: software protection, lock-bit protection, and protection to restrict access in programmer mode and boot mode.

2.1.8 Software Protection
Software commands can be disabled by clearing the FMCMDEN bit in the flash memory control register (FLMCR1) through software. In this state, software commands are not executed even if they are input.

Data-flash areas can be protected in block units by using the flash memory data-flash protect register (DFPR). Setting bits DFPR1 and DFPR0 in DFPR to 1 places all the data-flash areas in protect mode.

2.1.9 Lock-Bit Protection
The programming/erasure commands can be disabled by programming the lock bits using the lock-bit programming command. In this state, the erasure/programming commands are not executed even if input. This prevents flash memory from being erroneously erased or programmed due to CPU runaway.

The protection function can be temporarily disabled by setting the FMLBD bit in FLMCR1 to 1.

To clear the lock bit, erase the specified block. Note that lock bits are unavailable in data-flash areas.
2.1.10 Usage Notes

(1) Prohibited Instruction

In EW0 mode, the following instruction cannot be used because it refers to the data in the flash memory area.

- TRAP

(2) Interrupts

Table 5 shows interrupt handling in CPU reprogramming mode.

Table 5  Interrupt Handling in CPU Reprogramming Mode

| Mode    | Command being Executed                  | On Reception of an Interrupt Request                                                                 | On Generation of a Watchdog Timer Reset, LVD Reset, Software Reset, or Pin Reset |
|---------|-----------------------------------------|------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------
| EW0     | Erasure                                 | Interrupts can be handled if interrupt vectors are located in RAM. For details, see the section on the interrupt vector offset register (VOFR) of H8S/20103, H8S/20203, H8S/20223 Group Hardware Manual. | Immediately after a reset is generated, a software command is forcibly terminated, and flash memory and LSI are reset. Because of the forced termination, it might be impossible to read correct values from the block or address for which the software command has been executed; execute erasure again after restarting and confirm that erasure is completed successfully. The watchdog timer does not stop even during command execution; initialize the timer periodically. |
|         | Programming                             |                                                                                                       |                                                                                   |
|         | Lock-bit programming                    |                                                                                                       |                                                                                   |
|         | Blank checking                          |                                                                                                       |                                                                                   |
| EW1     | Erasure (with the erase-suspend function not in use) | Erasure is given priority, keeping the interrupt request waiting. When erasure is completed, execution of the interrupt processing is started. | Immediately after a reset is generated, a software command is forcibly terminated, and flash memory and LSI are reset. Because of the forced termination, it might be impossible to read correct values from the block or address for which the software command has been executed; execute erasure again after restarting and confirm that erasure is completed successfully. Since the watchdog timer does not stop even during command execution, set the overflow time of the watchdog timer longer than the erasure/programming execution time. |
|         | Erasure (with the erase-suspend function in use) | After the transition time to erase-suspend mode, erasure is suspended starting execution of the interrupt processing. When the interrupt processing is completed, setting the FMSPREQ bit in FLMCR2 to 0 resumes erasure. |                                                                                   |
|         | Programming                             | A software command is given priority, keeping the interrupt request waiting. When the software command is completed, execution of the interrupt processing is started. |                                                                                   |
|         | Lock-bit programming                    |                                                                                                       |                                                                                   |
|         | Blank checking                          |                                                                                                       |                                                                                   |
(3) Method of Access

When writing values to the protected bits indicated below, start by writing 0 to the bit and then write 1 to it or by writing 1 to the bit and then write 0 to it. Do not allow the generation of any interrupt or any access to other I/O registers between the two operations. For writing, always use the MOV instruction.

a. Bits that are set to 1 by writing 0 and then 1 consecutively
   - FLMCR1: FMLBD and FMCMDEN bits
   - FLMCR2: FMISPE and FMSPLN bits

b. Bits that are cleared to 0 by writing 1 and then 0 consecutively
   - DFPR: DFPR1 and DFPR0 bits

The example below is of code for use when the FMCMDEN and FMLBD bits in FLMCR1 are to be changed from 0 to 1.

```
MOV.B @FLMCR1,R0L :FLMCR1=H'04  R0L=H'04  R0H=H'xx
MOV.B @FLMCR1,R0H  :FLMCR1=H'04  R0L=H'04  R0H=H'04
BSET  #0,R0H       :FLMCR1=H'04  R0L=H'04  R0H=H'05
BSET  #3,R0H       :FLMCR1=H'04  R0L=H'04  R0H=H'0D
MOV.B R0L,@FLMCR1  :FLMCR1=H'0D  R0L=H'04  R0H=H'0D
MOV.B R0H,@FLMCR1  :FLMCR1=H'0D  R0L=H'04  R0H=H'0D
```

(4) Reprogramming User ROM Area

When it is necessary to reprogram the block containing the reprogramming-control program, use boot mode. This is because if the power supply voltage drops in EW0 mode while the block containing the reprogramming-control program is being reprogrammed, the reprogramming-control program cannot be correctly reprogrammed, and this might disable further reprogramming of the flash memory. Only proceed with overwriting of the programming-control program after securing ample stabilization time for the power supply.

(5) Programming

Do not program addresses that have already been programmed.

(6) LSI Mode Transition

During software command execution, do not cause a transition to standby mode or sleep mode.

(7) Reset during Execution of Software Command in Flash Memory

Do not apply a pin reset, LVD reset, watchdog timer reset or software reset during execution of the programming, lock-bit programming, blank-checking, and erasure commands. If applied, the currently executed command is forcibly terminated. In this case, execute the erasure command of the specified block again and confirm that erasure is completed successfully.
3. Principle of Operation

Figure 13 shows the principle of operation in this sample task. Data-flash memory is reprogrammed by means of hardware and software processing as shown in figure 13.

**Figure 13 Principle of Operation in This Sample Task**
4. Description of Software

4.1 Descriptions of Functions

The functions are listed and described in table 6.

Table 6 Descriptions of Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Main Routine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Declaration</td>
<td>void main(void)</td>
</tr>
<tr>
<td>Argument</td>
<td>Argument name Type Meaning</td>
</tr>
<tr>
<td>er_blk</td>
<td>unsigned char *</td>
</tr>
<tr>
<td>Returned value</td>
<td>Type Value Meaning</td>
</tr>
<tr>
<td>Returned value</td>
<td>none</td>
</tr>
<tr>
<td>Description</td>
<td>Calls most of the other functions.</td>
</tr>
</tbody>
</table>
Function Name | Blank Checking Routine
---|---
Declaration | unsigned char ew1_blank_check(unsigned char *blank_blk)
Argument | Argument name | Type | Meaning
| blank_blk | unsigned char * | Address setting to indicate the target block for blank checking
Returned value | Type | Value | Meaning
| unsigned char | H'00: Normal end | Result of executing a blank checking command
| H'10: Blank checking error |
| H'18: Command sequence error |
Description | Executes blank checking in data-flash area A in EW1 mode.

Function Name | Programming Routine
---|---
Declaration | unsigned char ew1_write(volatile unsigned char *wr_top, volatile unsigned char *wr_end, volatile unsigned char *wr_data)
Argument | Argument name | Type | Meaning
| wr_top | volatile unsigned char * | First address for programming
| wr_end | volatile unsigned char * | End address for programming + 1
| wr_data | volatile unsigned char * | Address of the data for programming
Returned value | Type | Value | Meaning
| unsigned char | H'00: Normal end |
| H'08: Programming error |
| H'18: Command sequence error |
Description | Writes to data-flash area A in EW1 mode.

Function Name | Full Status Checking Routine
---|---
Declaration | unsigned char full_status_check(unsigned char *addr)
Argument | Argument name | Type | Meaning
| addr | unsigned char * | Setting of the target address for full status checking
Returned value | Type | Value | Meaning
| unsigned char | H'00: Normal end |
| H'08: Programming error or lock-bit programming error |
| H'10: Erasure error or blank checking error |
| H'18: Command sequence error |
Description | Checks full status.
## 4.2 Description of Internal Registers

Table 7 gives descriptions of how internal registers are used in this sample task.

### Table 7 Description of Internal Registers

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Symbol</th>
<th>Function</th>
<th>Address</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMRJ</td>
<td>PMRJ[1:0]</td>
<td>The OSC1 and OSC2 functions are selected for pins PJ0/OSC1 and PJ1/OSC2.</td>
<td>H'FF000C</td>
<td>B'11</td>
</tr>
<tr>
<td>FLMCR1</td>
<td>FMWUS</td>
<td>Setting to select reprogramming by the CPU to be initiated by byte-length instructions.</td>
<td>H'FF0660</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>FMEWMOD</td>
<td>EW1 mode is selected.</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>FMCMDEN</td>
<td>Software commands for flash memory are enabled.</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>DFPR</td>
<td>DFPR1#6</td>
<td>E/W of data-flash B is disabled</td>
<td>HFF0662</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>DFPR0#6</td>
<td>E/W of data-flash A is enabled.</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FLMSTR</td>
<td>FMEBSF#8</td>
<td>Erasure/blank checking status flag</td>
<td>H'FF0663</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>FMPRSF#8</td>
<td>Programming status flag</td>
<td></td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>FMRDY</td>
<td>Flash memory ready/busy status flag</td>
<td></td>
<td>—</td>
</tr>
<tr>
<td>SYSCCR</td>
<td>PHIHSEL</td>
<td>φhigh clock source is set to φosc.</td>
<td>H'FF06D0</td>
<td>1</td>
</tr>
<tr>
<td>LPCR1</td>
<td>PSCSTP</td>
<td>PSC divider is operating.</td>
<td>H'FF06D1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>PHIBSEL</td>
<td>φbase clock source is set to φhigh.</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>LPCR2</td>
<td>PHI[2:0]</td>
<td>System clock φ is set to φbase.</td>
<td>H'FF06D2</td>
<td>B’000</td>
</tr>
<tr>
<td>LPCR3</td>
<td>PHIS[2:0]</td>
<td>Bus-master operating clock φs is set to φ.</td>
<td>H'FF06D3</td>
<td>B’000</td>
</tr>
<tr>
<td>OSCCSR</td>
<td>φosc oscillation settling time is set.</td>
<td>H'FF06D5</td>
<td>H’0E</td>
<td></td>
</tr>
</tbody>
</table>

[Setting conditions]  
- The erasure command is executed and results in unsuccessful erasure.  
- The blank-checking command is executed and it is detected that the specified block is not blank.  

[Clearing condition]  
- The status clearing command is issued.

[Setting conditions]  
- The programming command is executed and results in unsuccessful programming.  
- The lock-bit programming command is executed and results in unsuccessful programming.  

[Clearing condition]  
- The status clearing command is issued.
### Register

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Symbol</th>
<th>Function</th>
<th>Address</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMWD</td>
<td></td>
<td>Clock input to the WDT is prohibited.</td>
<td>H'FFFFF99</td>
<td>H'F7</td>
</tr>
<tr>
<td>TCSRWD</td>
<td></td>
<td>Writing to the TMWD register is enabled.</td>
<td>H'FFFFF9A</td>
<td>H'A3</td>
</tr>
<tr>
<td>MSTCR1</td>
<td>MSTWDT</td>
<td>WDT is released from the module standby mode.</td>
<td>H'FFFFFDC</td>
<td>0</td>
</tr>
</tbody>
</table>

**Notes:**
1. When setting the bit to 1, first clear the bit to 0 and then immediately set the bit to 1; do not allow any interrupt to be generated between these operations.
2. The bit is cleared to 0 when the FMRDY bit changes from 0 to 1.
3. Set the FMEWMOD bit and then set the FMCMDEN bit to 1.
4. When setting the FMCMDEN bit to 1 while the FMEWMOD bit is 0, be sure to execute the program in the RAM.
5. When setting the bit to 0, first set the bit to 1 and then immediately set the bit to 0; do not allow any interrupt to be generated between these operations.
6. The DFPR bits are set to 1 when the FMCMDEN bit changes from 0 to 1.
7. The bit cannot be set to 1 through software.
8. The bit is cleared to 0 when the status clearing command is executed.

### 4.3 RAM Usage

Table 8 gives a description of RAM usage in this sample task.

<table>
<thead>
<tr>
<th>Label Name</th>
<th>Functions</th>
<th>Memory</th>
<th>Module Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>Data-flash area A for back up (H'FFDF80 to H'FFE7F)</td>
<td>4 KB</td>
<td>—</td>
</tr>
</tbody>
</table>
4.4 Descriptions of Definitions in Use

Table 9 gives descriptions of the definitions used in this sample task.

<table>
<thead>
<tr>
<th>Label Name</th>
<th>Description</th>
<th>Constant</th>
</tr>
</thead>
<tbody>
<tr>
<td>FULL_STATUS</td>
<td>Mask value for use in full status checking of FLMSTR</td>
<td>H'28</td>
</tr>
<tr>
<td>COMMAND_SEQUENCE_ERR</td>
<td>Value corresponding to a command sequence error; for use in full status checking of FLMSTR</td>
<td>H'28</td>
</tr>
<tr>
<td>ERASE_BLANK_ERR</td>
<td>Value corresponding to an erasure error or blank checking error; for use in full status checking of FLMSTR</td>
<td>H'20</td>
</tr>
<tr>
<td>PRG_LOCKBIT_ERR</td>
<td>Value corresponding to a programming error or lock-bit programming error; for use in full status checking of FLMSTR</td>
<td>H'08</td>
</tr>
<tr>
<td>NO_ERR</td>
<td>Value corresponding to normal completion; for use in full status checking of FLMSTR</td>
<td>H'00</td>
</tr>
<tr>
<td>WRITE_SIZE</td>
<td>Size of the portion of data-flash A to be reprogrammed</td>
<td>H'80</td>
</tr>
<tr>
<td>BACK_UP_AREA</td>
<td>First address in the back-up RAM area of data-flash A</td>
<td>H'FFDF80</td>
</tr>
<tr>
<td>FLASH_BLK_A</td>
<td>First address of data-flash area (erasure block) A</td>
<td>H'F00000</td>
</tr>
<tr>
<td>FLASH_BLK_B</td>
<td>First address of data-flash area (erasure block) B</td>
<td>H'F01000</td>
</tr>
<tr>
<td>FLASH_BLK_B_END</td>
<td>End address of data-flash area (erasure block) B</td>
<td>H'F01FFF</td>
</tr>
</tbody>
</table>
5. Flowcharts

5.1 Main Routine

```
main

Set the I bit to 1 to disable interrupts.

h8s_sysinit()

Contents of the entire 4-KB data-flash area A are transferred to RAM as backup data.

ew1_erase

Argument: Address of data-flash area A

ew1_blank_check

Argument: Address of data-flash area A

Reprogramming of the first 128 bytes in the RAM area where data-flash A has been backed up.

ew1_write

1st argument: First address of data-flash area A
2nd argument: First address of data-flash area B
3rd argument: First address of the RAM area where data-flash A is backed up

----- 4 KB of data are transferred to the area of RAM from H'FFDF80 to H'FFEF7F.

----- Argument: Address of data-flash area A

----- Argument: Address of data-flash area A
```
5.2 System Initialization Routine

```
h8s_sysinit

Set the MSTWDT bit in MSTCR1 to 0.  

Set the B4WI and TCSRWE bits in TCSRWD to 0 and 1, respectively.  

Set the TMWLOCK and TMWI bits in TCSRWD to 0.  

Set CKS[3:0] in TMWD to B'0111.*  

Set CKS[3:0] in TMWD to B'1000.  

Set the B4WI and TCSRWE bits in TCSRWD to 0.  

Set the φosc oscillation settling time in the active bits of OSCCSR.  

Set PMRJ[1:0] to B'11.  

Set the WI and WE bits in SYSCCR to 0 and 1, respectively.  

Set the PHIHSEL bit in SYSCCR to 1.  

Set the WI and WE bits in SYSCCR to 0.  

1

The WDT is released from module standby mode.  

Writing to the TMWLOCK and TMWI bits in TCSRWD is enabled.  

Writing to TMWD is enabled.  

Value to disable clock input to the WDT module (halts the WDT).  

Writing the bit-wise inverse of the value set in * above makes the value set in * effective.  

Writing to the TMWLOCK and TMWI bits in TCSRWD is disabled.  

Set the bits so that the waiting time will be at least 6.5 ms given the frequency of the oscillator.  

The OSC1 and OSC2 functions are selected for pins PJ0/OSC1 and PJ1/OSC2.  

Writing to bits 5, 4, 2, and 1 in SYSCCR is enabled.  

φosc is selected as the φhigh clock source.  

Writing to bits 5, 4, 2, and 1 in SYSCCR is disabled.  
```
Set the WI and WE bits in LPCR1 to 0 and 1, respectively.

Set the PSCSTP and PHIBSEL bits in LPCR1 to 0 and 1, respectively.

Set PHI[2:0] in LPCR2 to B'000.

Set PHIS[2:0] in LPCR3 to B'000.

return

Writing to bits 5 to 2, and 0 in LPCR1 is enabled.

φhigh is selected as the clock source for φbase and for operation of the PSC divider.

Writing to bits 5 to bit 2, and 0 in LPCR1 is disabled.

Writing to bits 2 to 0 in LPCR2 is enabled.

φbase is selected as the system clock (φ).

Writing to bits 2 to 0 in LPCR2 is disabled.

Writing to bits 2 to 0 in LPCR3 is enabled.

φ is selected as the bus-master operating clock (φs).

Writing to bits 2 to 0 in LPCR3 is disabled.
5.3 Erasure Routine

```plaintext
ew1_erase

Set the FMEWMOD bit in FLMCR1 to 1.

Set the FMCMDEN bit in FLMCR1 to 0, then 1.

EW1 is selected as the mode of CPU reprogramming.

Software commands for flash memory are enabled.

Determination of target block for erasure

Data-flash area A?

Yes

Data-flash area B?

No

Set the DFPR1 bit in DFPR to 1, then 0.

E/W of data-flash A is enabled.

No

Set the DFPR0 bit in DFPR to 1, then 0.

E/W of data-flash B is enabled.

Write software command H'20 to any address in the target block for erasure.

Write H'D0 to the specified address in the target block for erasure.

FMRDY = 1?

Yes

Is erasure in progress?

No

full_status_check

Full status checking

return
```
5.4 Programming Routine

ew1_write

Set the FMWUS bit in FLMCR1 to 0.

Set the FMEWMOD bit in FLMCR1 to 1.

CPU reprogramming to continue?

Yes

No

3

Set the FMCMDEN bit in FLMCR1 to 0, then 1.

Setting for CPU reprogramming in byte units is made.

EW1 is set as the mode of CPU reprogramming.

Software commands in flash memory are enabled.

Determination of the target block for erasure

Data-flash area A?

Yes

No

Determination of the target block for erasure

Data-flash area B?

Yes

Set the DFPR0 bit in DFPR to 1, then 0.

E/W of data flash A is enabled.

No

Set the DFPR1 bit in DFPR to 1, then 0.

E/W of data flash B is enabled.

Write software command H'41 to the target address for programming.

Write programming data to the target address for programming.

4-byte units are written to the successive target addresses for reprogramming.
FMRDY = 1?

Is programming in progress?

full_status_check

Full status checking

No errors?

Result of full status checking is determined.

return

Result of full status checking is returned.
5.5 Blank Checking Routine

ew1_blank_check

Set the FMEWMOD bit in FLMCR1 to 1.

---- EW1 is set as the mode of CPU reprogramming.

Set the FMCMDEN bit in FLMCR1 to 0, then 1.

---- Software commands in flash memory are enabled.

Determination of the target block for blank checking

Data-flash area A?

Yes

No

Determination of the target block for blank checking

Data-flash area B?

Yes

Set the DFPR0 bit in DFPR to 1, then 0.

---- E/W of data flash A is enabled.

No

Set the DFPR1 bit in DFPR to 1, then 0.

---- E/W of data flash B is enabled.

Write software command H'25 to any address in the target block for blank checking.

Write H'D0 to any address of the target block for blank checking.

FMRDY = 1?

Yes

---- Is execution of blank checking in progress?

No

full_status_check

---- Full status checking

return
## 5.6 Full Status Checking Routine

- **full_status_check**
  
  Read the FMEBSF and FMPRSF bits in FLMSTR.

  - **Command sequence error?**
    - **Yes**
    - **No**

  - **Error in erasure or blank checking?**
    - **Yes**
    - **No**

  - **Error in programming or lock-bit programming?**
    - **Yes**
      - Set the FMCMDEN bit in FLMCR1 to 0, then 1.
      - **Determination of the target of full status checking**
        - **Data-flash area A?**
          - **Yes**
            - **E/W of data flash A is enabled.**
            - Set the DFPR0 bit in DFPR to 1, then 0.
          - **No**
            - **Determination of the target of full status checking**
              - **Data-flash area B?**
                - **Yes**
                  - **Set the DFPR1 bit in DFPR to 1, then 0.**
                - **No**
                  - **Set the DFPR1 bit in DFPR to 1, then 0.**

  - **No**
    - **Set the DFPR0 bit in DFPR to 1, then 0.**
Clear status flags by writing software command H’50 to a target address.

Result of full status checking is returned.
6. Program Listing

/***************************************************************
/* H8S/2000 Tiny Series -H8S/20203- */
/* Application Note */
/* */
/* data flash read and write */
/* */
/* Function: data flash read and write (EW1 mode) */
/* */
/* */
/* External Clock: 20MHz */
/* */
/* Internal Clock: 20MHz */
/***************************************************************
#include <machine.h>
#include "iodefine.h"
#define FULL_STATUS 0x28  /* mask FLMSTR of FMEBSF,FMPRSF */
#define COMMAND_SEQUENCE_ERR 0x28  /* FMEBSF=1, FMPRSF=1 */
#define ERASE_BLANK_ERR 0x20  /* FMEBSF=1, FMPRSF=0 */
#define PRG_LOCKBIT_ERR 0x08  /* FMEBSF=0, FMPRSF=1 */
#define NO_ERR 0x00  /* FMEBSF=0, FMPRSF=0 */
#define WRITE_SIZE 0x80  /* data size written to data flash A */
#define BACK_UP_AREA (volatile unsigned char *)0xFFDF80
/* Data Flash block area */
#define FLASH_BLK_A (volatile unsigned char *)0xF00000
#define FLASH_BLK_B (volatile unsigned char *)0xF01000
#define FLASH_BLK_B_END (volatile unsigned char *)0xF01FFF
/***************************************************************
Declaration of function prototype */
/***************************************************************
void main(void);
unsigned char ew1_write( volatile unsigned char *wr_top, 
    volatile unsigned char *wr_end, volatile unsigned char *wr_data );
unsigned char ew1_erase( unsigned char *er_blk );
unsigned char ew1_blank_check( unsigned char *blank_blk );
unsigned char full_status_check( unsigned char *adrs );
void h8s_sysinit(void);
/*******************************************************************************/
/* Name : main */
/* Parameters : None */
/* Returns : None */
/* Description : User main */
/*******************************************************************************/

void main(void)
{
    unsigned char ii, chk;
    volatile unsigned char *df_p, *ram_p;

    set_ccr(0x80); /* set CCR-Ibit */
    h8s_sysinit(); /* initialize system */

    /* back up data of data flash A(4KB) */
    for ( df_p=FLASH_BLK_A, ram_p=BACK_UP_AREA; df_p<FLASH_BLK_B; df_p++, ram_p++ ){
        (*ram_p) = (*df_p);
    }

    /* EW1 erase of data flash A */
    chk = ew1_erase( FLASH_BLK_A );

    /* blank check data flash A */
    chk = ew1_blank_check( FLASH_BLK_A );

    /* create write data of data flash */
    for ( ii=0, ram_p=BACK_UP_AREA; ram_p<(BACK_UP_AREA+WRITE_SIZE); ii++, ram_p++ ){
        (*ram_p) = ii;
    }

    /* EW1 write of data flash */
    chk = ew1_write( FLASH_BLK_A, FLASH_BLK_B, BACK_UP_AREA );

    while(1);
}
unsigned char ew1_write( volatile unsigned char *wr_top, 
        volatile unsigned char *wr_end, volatile unsigned char *wr_data )
{
    volatile unsigned char *ptr;
    unsigned char result;
    unsigned char ii;

    FLASH.FLMCR1.BIT.FMWUS = 0;       /* byte write */
    FLASH.FLMCR1.BIT.FMEWMOD = 1; /* select EW1 mode */

    for( ptr=wr_top; ptr<wr_end; ptr+=4 ){
        FLASH.FLMCR1.BIT.FMCMDEN = 0;
        FLASH.FLMCR1.BIT.FMCMDEN = 1; /* flash memory software command enable */

        /* Data Flash A ? */
        if ( (FLASH_BLK_A <= ptr) && (ptr < FLASH_BLK_B) ){
            FLASH.DFPR.BIT.DFPR0 = 1; /* E/W enable of Data Flash A */
            FLASH.DFPR.BIT.DFPR0 = 0; /* E/W enable of Data Flash A */
        }
        /* Data Flash B ? */
        else if ( (FLASH_BLK_B <= ptr) && (ptr <= FLASH_BLK_B_END) ){
            FLASH.DFPR.BIT.DFPR1 = 1; /* E/W enable of Data Flash B */
            FLASH.DFPR.BIT.DFPR1 = 0; /* E/W enable of Data Flash B */
        }

        (*ptr) = 0x41; /* software command 0x41 */
        /* 4byte write */
        for( ii=0; ii<4; ii++, wr_data++ ){
            (*ptr) = (*wr_data);
        }

        while( FLASH.FLMSTR.BIT.FMRDY != 1 ); /* write complete ? */
        result = full_status_check(ptr); /* full status check */

        if ( result != NO_ERR ){
            return;
        }
    }

    return (result);
/**************************************************************/
/* Name               : ewl_erase */
/* Parameters          : (er_blk)address of erase BLOCK */
/* Returns             : erase result */
/* Description        : data flash erase program */
/* of EW1 mode        */
/**************************************************************/

unsigned char ewl_erase( unsigned char *er_blk )
{
    unsigned char result;

    FLASH.FLMCR1.BIT.FMEWMOD = 1; /* select EW1 mode */

    FLASH.FLMCR1.BIT.FMCMDEN = 0;
    FLASH.FLMCR1.BIT.FMCMDEN = 1; /* flash memory software command enable */

    /* Data Flash A ? */
    if ( (FLASH_BLK_A <= er_blk) && (er_blk < FLASH_BLK_B) ){
        FLASH.DFPR.BIT.DFPR0 = 1; /* E/W enable of Data Flash A */
        FLASH.DFPR.BIT.DFPR0 = 0; /* E/W enable of Data Flash A */
    }

    /* Data Flash B ? */
    else if ( (FLASH_BLK_B <= er_blk) && (er_blk <= FLASH_BLK_B_END) ){
        FLASH.DFPR.BIT.DFPR1 = 1; /* E/W enable of Data Flash B */
        FLASH.DFPR.BIT.DFPR1 = 0; /* E/W enable of Data Flash B */
    }

    (*er_blk) = 0x20;        /* write software command H'20 */
    (*er_blk) = 0xD0;        /* write software command H'D0 */

    while( FLASH.FLMSTR.BIT.FMRDY != 1 );  /* erase complete ? */

    result = full_status_check(er_blk);   /* full status check */

    return (result);
}
unsigned char ew1_blank_check( unsigned char *blank_blk )
{
    unsigned char result;

    FLASH.FLMCR1.BIT.FMEWMOD = 1; /* select EW1 mode */
    FLASH.FLMCR1.BIT.FMCMDEN = 0;
    FLASH.FLMCR1.BIT.FMCMDEN = 1; /* flash memory software command enable */

    /* Data Flash A ? */
    if ( (FLASH_BLK_A <= blank_blk) && (blank_blk < FLASH_BLK_B) )
    {
        FLASH.DFPR.BIT.DFPR0 = 1; /* E/W enable of Data Flash A */
        FLASH.DFPR.BIT.DFPR0 = 0; /* E/W enable of Data Flash A */
    }
    /* Data Flash B ? */
    else if ( (FLASH_BLK_B <= blank_blk) && (blank_blk <= FLASH_BLK_B_END) )
    {
        FLASH.DFPR.BIT.DFPR1 = 1; /* E/W enable of Data Flash B */
        FLASH.DFPR.BIT.DFPR1 = 0; /* E/W enable of Data Flash B */
    }

    (*blank_blk) = 0x25; /* blank check software command H'25 */
    (*blank_blk) = 0xD0; /* blank check software command H'D0 */

    while( FLASH.FLMSTR.BIT.FMRDY != 1 ); /* blank check complete ? */

    result = full_status_check(blank_blk); /* full status check */

    return (result);
}
unsigned char full_status_check( unsigned char *addr )
{
    unsigned char tmp_flmstr;

    /* Full status check */
    tmp_flmstr = FLASH.FLMSTR.BYTE & FULL_STATUS; /* read FLMSTR */

    switch ( tmp_flmstr ){
        case COMMAND_SEQUENCE_ERR: /* command sequence error */
        case ERASE_BLANK_ERR: /* erase or blank check error */
        case PRG_LOCKBIT_ERR: /* program or lock bit program error */
            /* error processing */
            FLASH.FLMCR1.BIT.FMCMDEN = 0;
            FLASH.FLMCR1.BIT.FMCMDEN = 1; /* flash memory software command enable */
            break;
        default : /* No error */
            break;
    }

    (*addr) = 0x50; /* Clear status command */

    return (tmp_flmstr);
}
# H8S/20103, H8S/20203, H8S/20223 Group
Reprogramming Data Flash in EW1 Mode

---

### 6.1 Designation of Link Addresses

<table>
<thead>
<tr>
<th>Section Name</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>PResetPRG, PIntPRG</td>
<td>H'000400</td>
</tr>
<tr>
<td>P, C$DSEC, C$BSEC, D</td>
<td>H'000800</td>
</tr>
<tr>
<td>B, R</td>
<td>H'FFEF80</td>
</tr>
<tr>
<td>S</td>
<td>H'FFFD80</td>
</tr>
</tbody>
</table>
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>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>Oct.31.08</td>
<td>First edition issued</td>
</tr>
</tbody>
</table>

All trademarks and registered trademarks are the property of their respective owners.
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.