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

# 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

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.

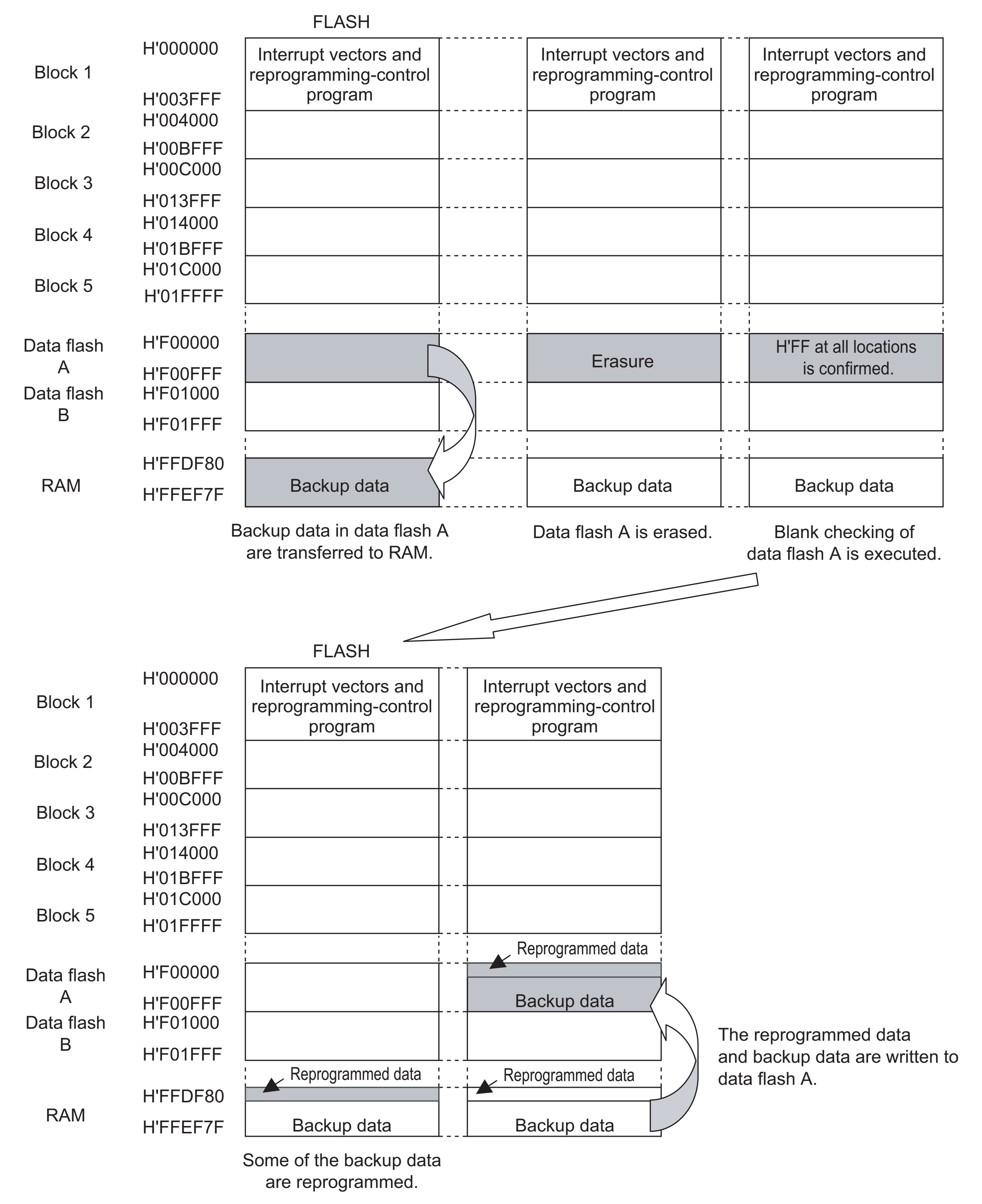



**Figure 1 Overview of Reprogramming the Data Flash in EW1 Mode**

# 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.

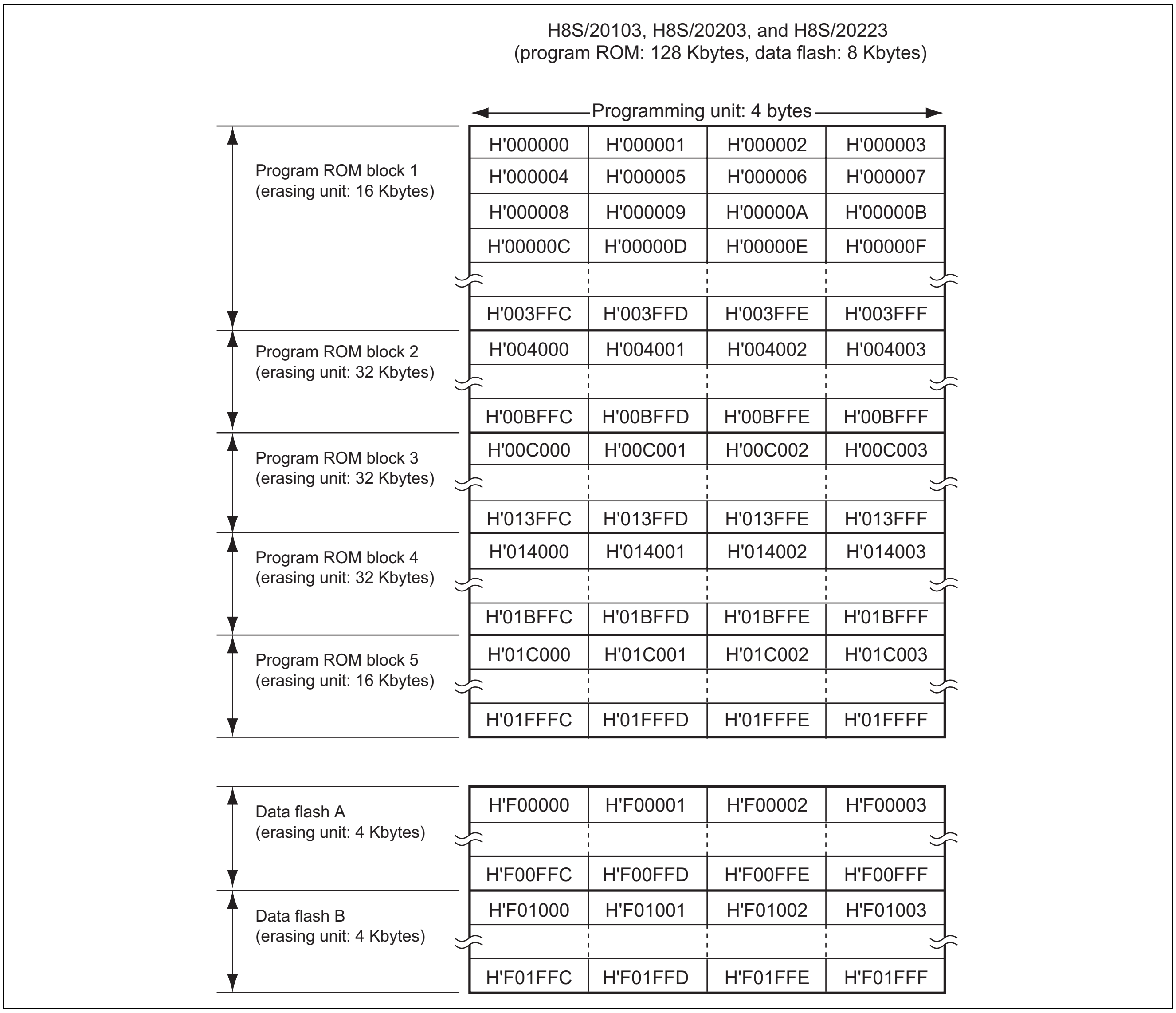



**Figure 2 Block Configuration of Flash Memory**

### 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 1 Differences between EW0 Mode and EW1 Mode**

<table>
<tr><th>Item</th><th>EW0 Mode</th><th>EW1 Mode</th></tr>
<tr><td>Area in which a reprogramming-control program can be located</td><td colspan="2">User ROM area</td></tr>
<tr><td>Area in which a reprogramming-control program can be executed</td><td>A reprogramming-control program must be transferred to RAM before execution.</td><td>A reprogramming-control program can be executed in the user ROM area.</td></tr>
<tr><td>Area which can be reprogrammed</td><td>User ROM area</td><td>User ROM area excluding the blocks in which a reprogramming-control program is located.</td></tr>
<tr><td>Limitations on software commands</td><td>None</td><td>The program and erasure commands must not be executed on any block in which a reprogramming-control program is located.</td></tr>
<tr><td>Mode after software command execution</td><td colspan="2">Read-array mode</td></tr>
<tr><td>CPU state during auto-programming and auto-erasure</td><td>Operating state</td><td>Hold state (I/O ports retain the states in which they have been before the command is executed.)</td></tr>
<tr><td>Flash memory state detection</td><td colspan="2">Read the FMPRSF, FMERSF, and FMEBSF bits in FLMSTR in a program.</td></tr>
<tr><td>Conditions of transition to erase-suspend state</td><td>Both the FMSPEN and FMSPREQ bits in FLMCR2 are set to 1.<br>Or, both the FMSPEN and FMISPE bits in FLMCR2 are set to 1 and an interrupt is requested.</td><td>The FMSPEN bit in FLMCR2 is set to 1 and an interrupt is requested.</td></tr>
<tr><td>Conditions of Interrupt generation</td><td>• The flash memory returns from the busy state to the ready state*[1].<br>• The user ROM area is read in the busy state*[1].</td><td>Use of interrupts is prohibited.</td></tr>
<tr><td>Usage of DTC</td><td>Usable*[2]</td><td>Usable*[2]*[3]</td></tr>
</table>

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)**

| Software Command | First Command Cycle | | | Second Command Cycle | | | Third Command Cycle | | | Command Use in Modes | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Mode | Addr. | Data | Mode | Addr. | Data | Mode | Addr. | Data | EW0 | EW1 |
| Erasure | Write | × | H'2020 | Write | BA | H'D0D0 | | | | Possible | Possible |
| Programming | Write | WA | H'4141 | Write | WA | WD1 | Write | WA | WD2 | Possible | Possible |
| Blank checking | Write | × | H'2525 | Write | BA | H'D0D0 | | | | Possible | Possible |
| Lock-bit programming | Write | × | H'7777 | Write | BA | H'D0D0 | | | | Possible | Possible |
| Read-array | Write | × | H'FFFF | | | | | | | Possible | — |
| Clear-status | Write | × | H'5050 | | | | | | | Possible | Possible |
| Lock-bit reading | Write | × | H'7171 | Read | BA | H'xxxx | | | | Possible | Impossible |

[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)**

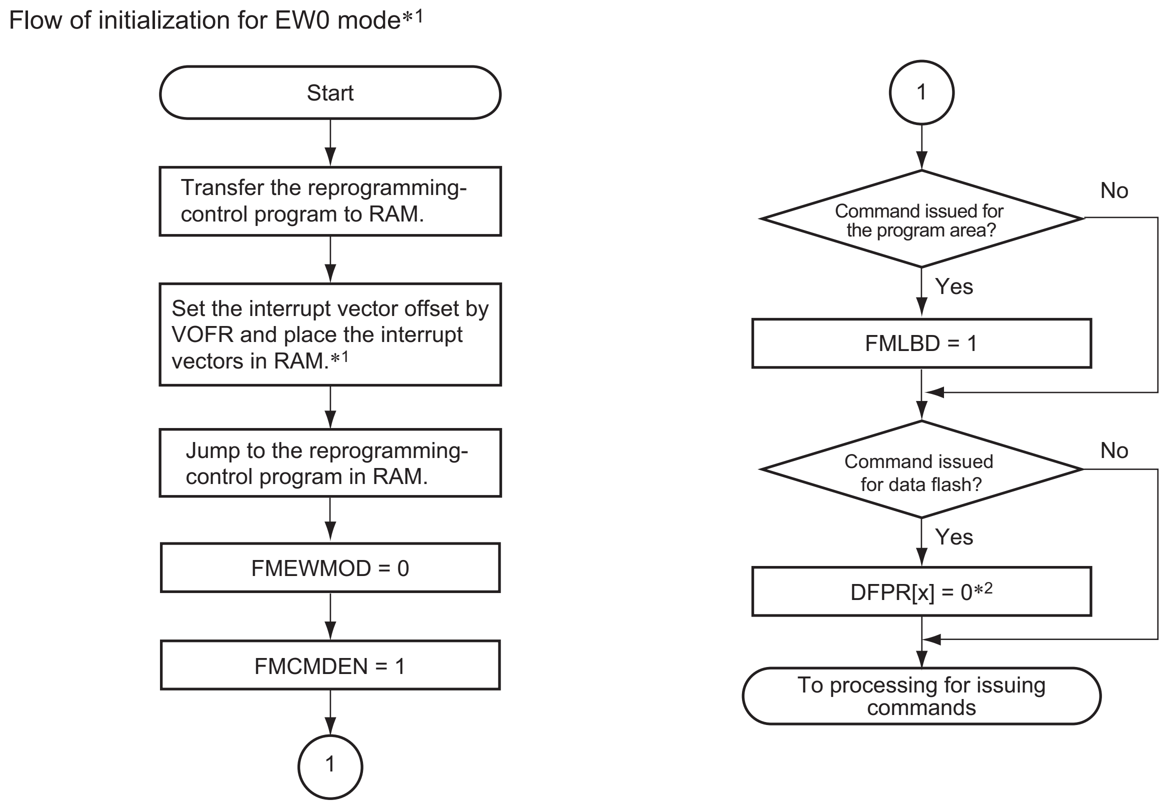
| Software Command | First Command Cycle | | | Second Command Cycle | | | Third Command to Fifth Command Cycle | | | Command Use in Modes | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Mode | Addr. | Data | Mode | Addr. | Data | Mode | Addr. | Data | EW0 | EW1 |
| Erasure | Write | × | H'20 | Write | BA | H'D0 | | | | Possible | Possible |
| Programming | Write | WA | H'41 | Write | WA | WD1 | Write | WA | WD2 to WD4 | Possible | Possible |
| Blank checking | Write | × | H'25 | Write | BA | H'D0 | | | | Possible | Possible |
| Lock-bit programming | Write | × | H'77 | Write | BA | H'D0 | | | | Possible | Possible |
| Read-array | Write | × | H'FF | | | | | | | Possible | — |
| Clear-status | Write | × | H'50 | | | | | | | Possible | Possible |
| Lock-bit reading | Write | × | H'71 | Read | BA | H'xx | | | | Possible | Impossible |

[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

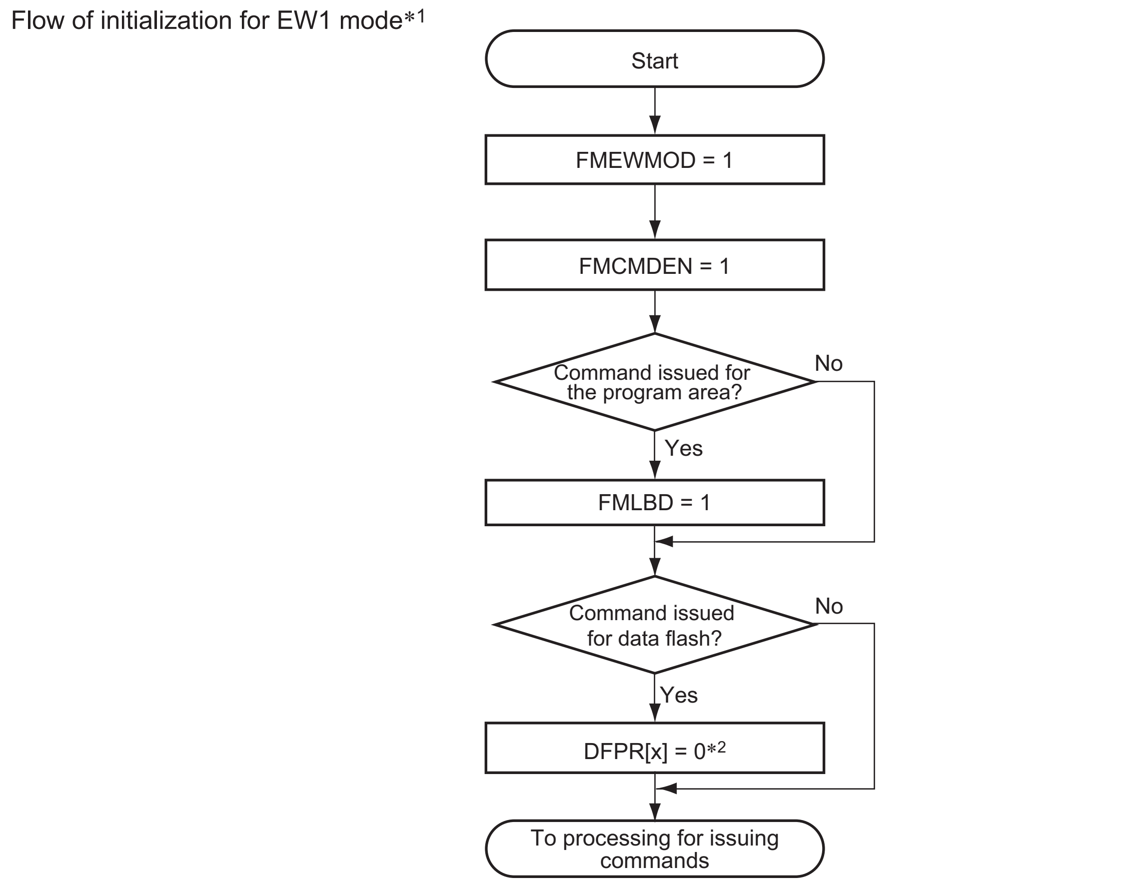
Before software commands are issued, settings for CPU reprogramming mode must be made and issuing of software commands must be permitted.

Figures 3 and 4 show initialization for CPU reprogramming mode.




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**




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.

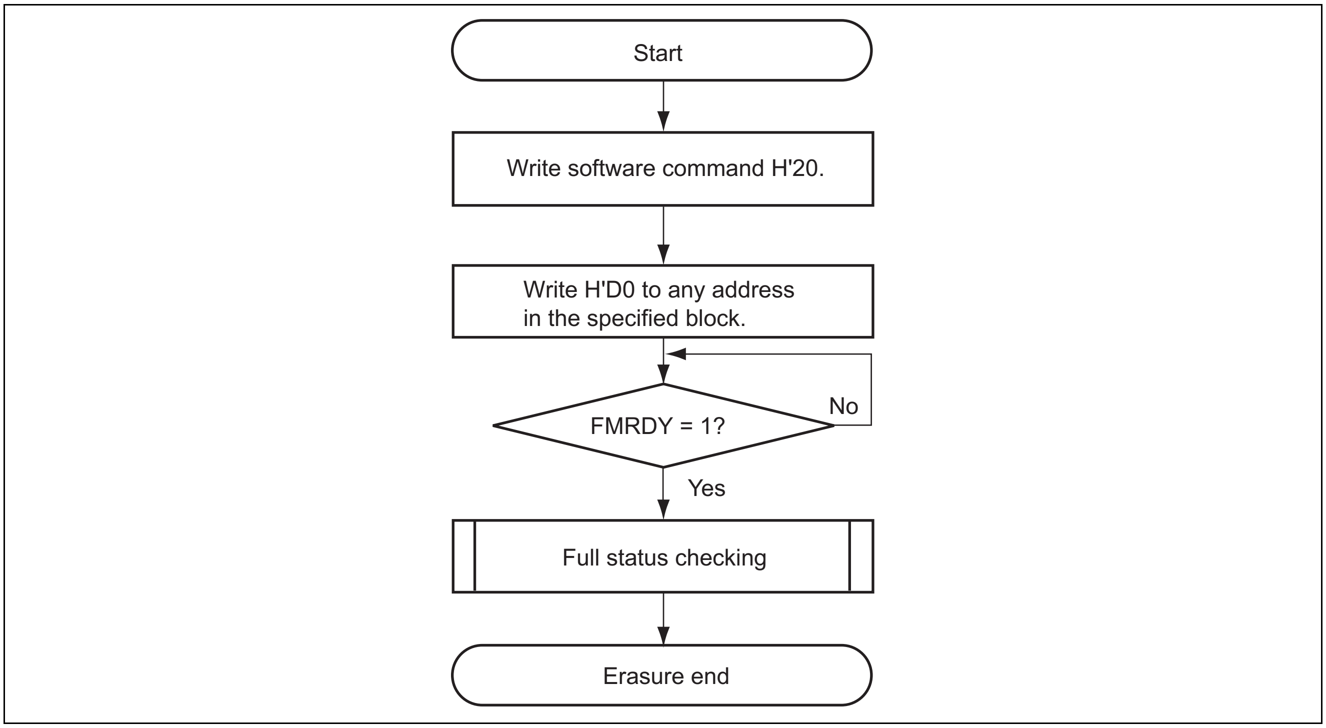
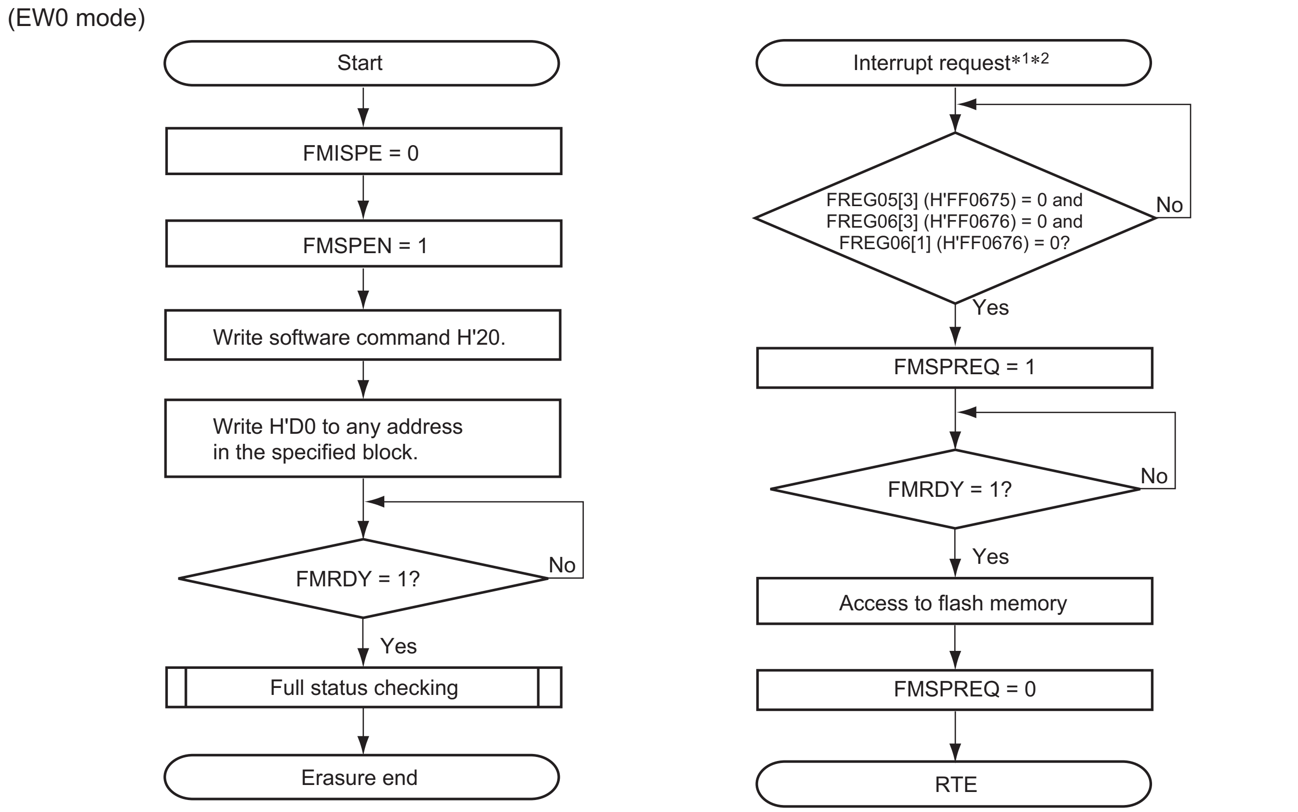
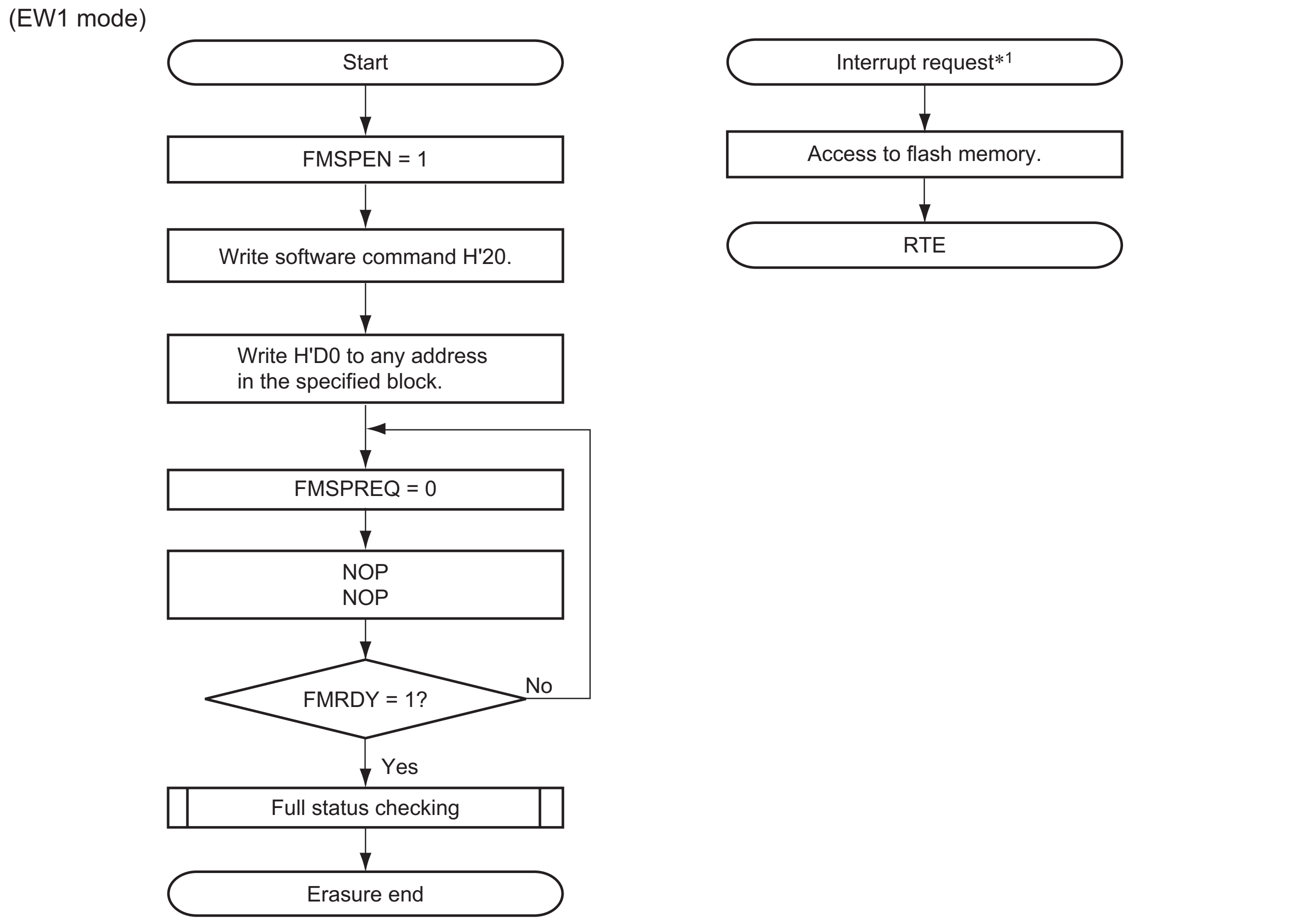



**Figure 5 Flowchart of Erasure When the Erase-Suspend Function is Not in Use**




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**




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.

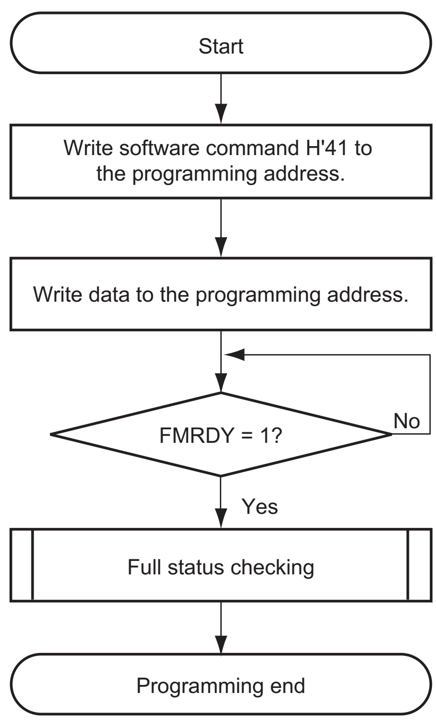



**Figure 8 Programming Flowchart**

(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.

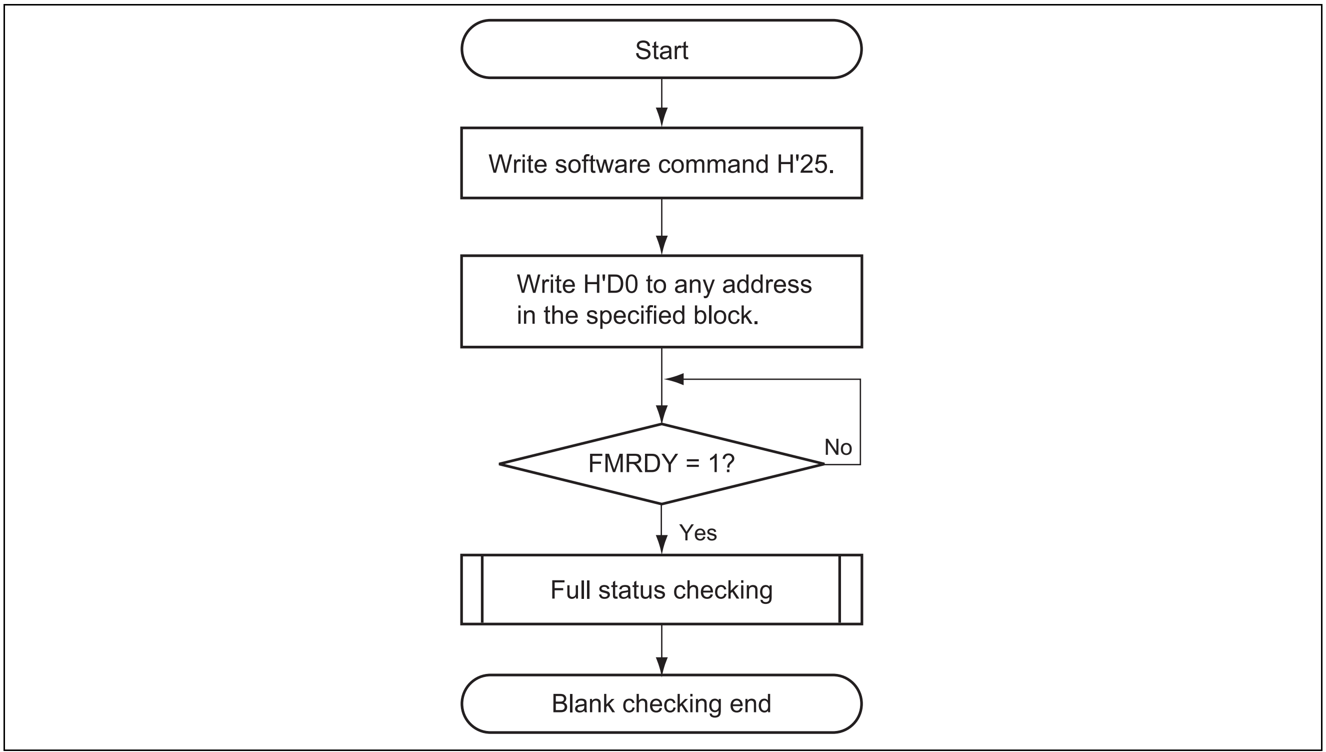



**Figure 9   Blank Checking Flowchart**

(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.

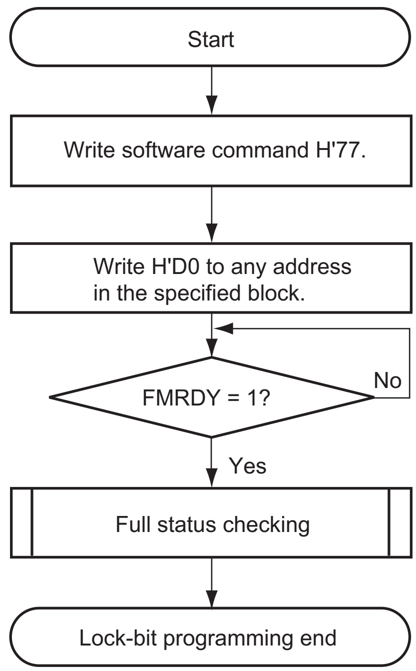



**Figure 10 Lock-Bit Programming Flowchart**

(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**

| Bit Values in FLMSTR | | | |
|---|---|---|---|
| **FMEBSF** | **FMPRSF** | **Error** | **Error Generation Conditions** |
| 0 | 0 | None (normal end) | — |
| 0 | 1 | Programming error | The programming command is executed and results in unsuccessful programming. |
| | | Lock-bit programming error | The lock-bit programming command is executed and results in unsuccessful programming. |
| 1 | 0 | Erasure error | The erasure command is executed and results in unsuccessful erasure. |
| | | Blank checking error | The blank checking command is executed and it is detected that the specified block is not blank. |
| 1 | 1 | Command sequence error | • A command is not written correctly.<br>• 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.<br>• The erasure command is input in erase-suspend mode.<br>• The programming command is input for the suspended block in erase-suspend mode. |

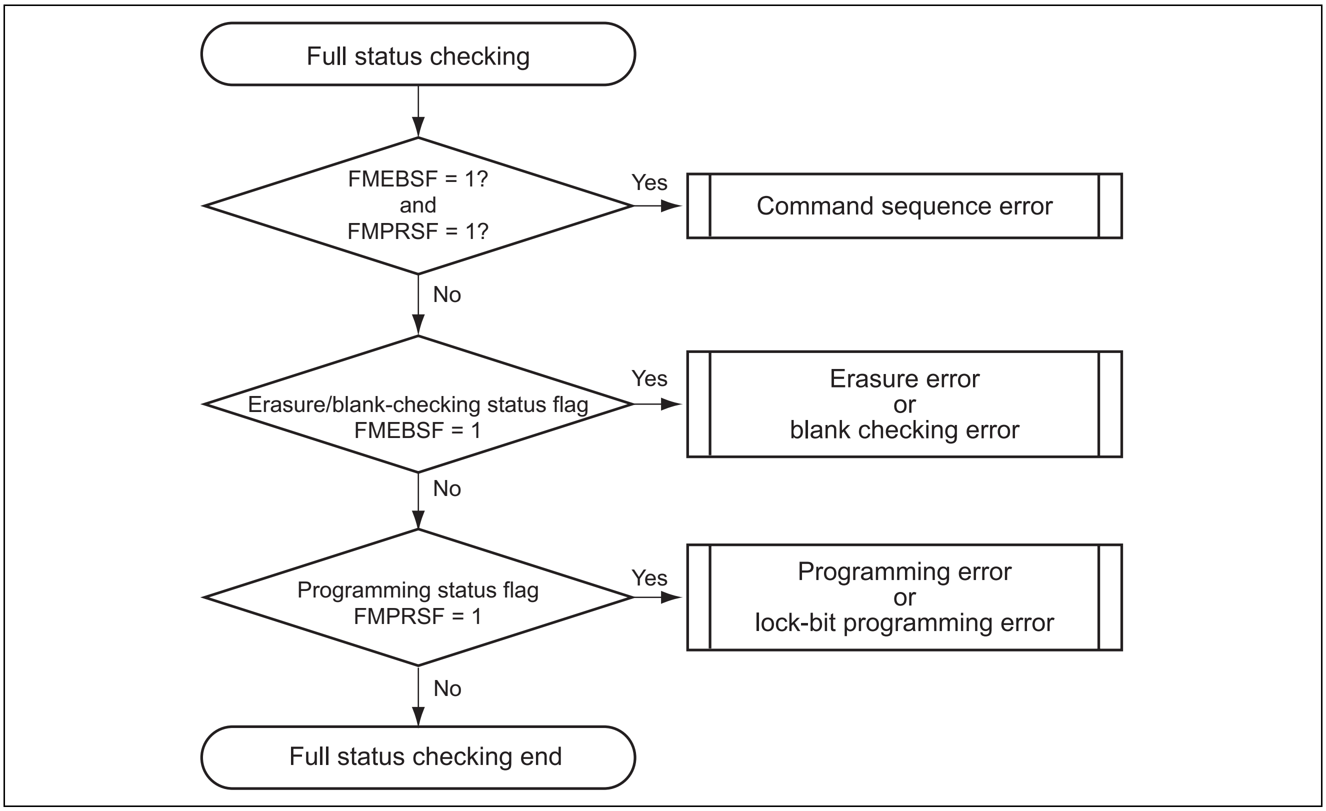



**Figure 11 Full Status Checking Flowchart**

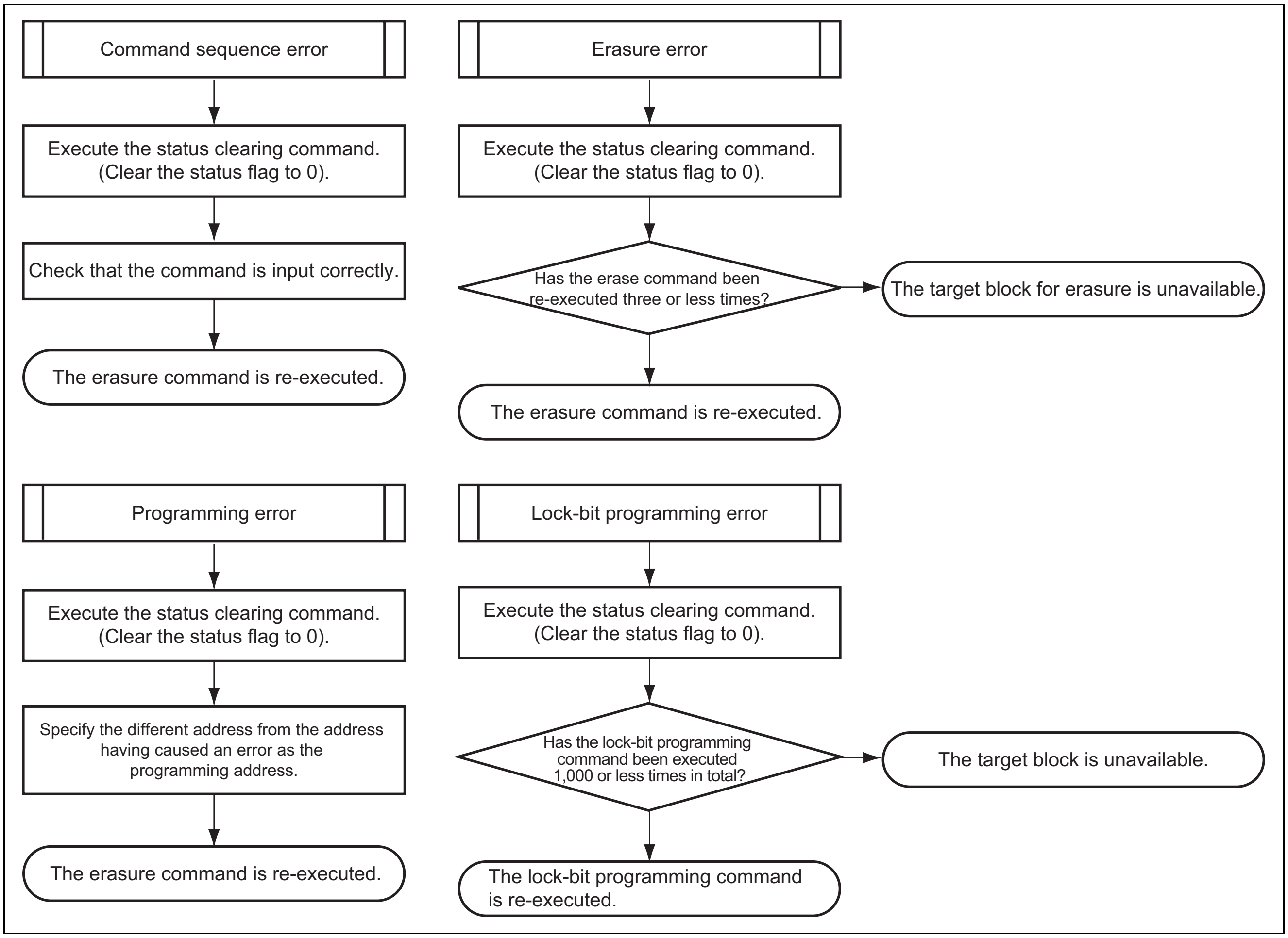



**Figure 12 Procedures for Handling Errors**

### 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<br>Programming<br>Lock-bit programming<br>Blank checking | 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.<br>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.<br>The watchdog timer does not stop even during command execution; initialize the timer periodically. |
| 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.<br>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.<br>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<br>Lock-bit programming<br>Blank checking | A software command is given priority, keeping the interrupt request waiting. When the software command is completed, execution of the interrupt processing is started. | |

(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 FMSPEN 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'04   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.

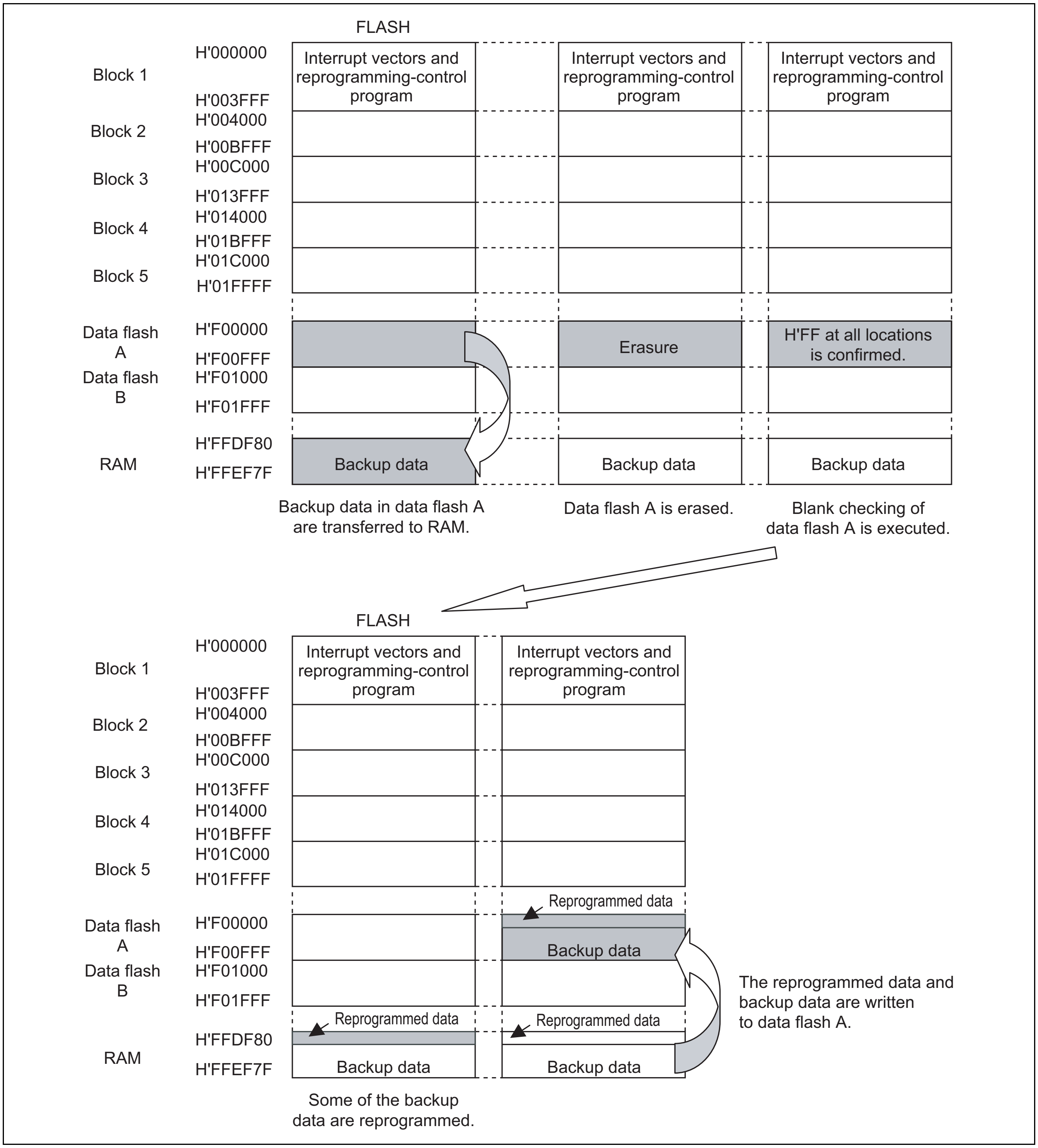



**Figure 13 Principle of Operation in This Sample Task**

# 4. Description of Software

## 4.1 Descriptions of Functions

The functions are listed and described in table 6.

**Table 6 Descriptions of Functions**

| **Function Name** | Main Routine | | |
|---|---|---|---|
| **Declaration** | void main(void) | | |
| **Argument** | **Argument name** | **Type** | **Meaning** |
| | None | None | None |
| **Returned value** | **Type** | **Value** | **Meaning** |
| | None | None | None |
| **Description** | Calls most of the other functions. | | |

| **Function Name** | System Initialization Routine | | |
|---|---|---|---|
| **Declaration** | void h8s_sysinit(void) | | |
| **Argument** | **Argument name** | **Type** | **Meaning** |
| | None | None | None |
| **Returned value** | **Type** | **Value** | **Meaning** |
| | None | None | None |
| **Description** | Makes settings for module standby, system clock and bus-master operating clock, and halts the WDT. | | |

| **Function Name** | Erasure Routine | | |
|---|---|---|---|
| **Declaration** | unsigned char ew1_erase (unsigned char *er_blk) | | |
| **Argument** | **Argument name** | **Type** | **Meaning** |
| | er_blk | unsigned char * | Address setting to indicate the target block for erasure |
| **Returned value** | **Type** | **Value** | **Meaning** |
| | unsigned char | H'00: Normal end<br>H'10: Erasure error<br>H'18: Command sequence error | Result of executing an erasure command |
| **Description** | Erases data-flash area A in EW1 mode. | | |

| **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<br>H'10: Blank checking error<br>H'18: Command sequence error | Result of executing a blank checking command |
| **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<br>H'08: Programming error<br>H'18: Command sequence error | Result of executing a programming command |
| **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<br>H'08: Programming error or lock-bit programming error<br>H'10: Erasure error or blank checking error<br>H'18: Command sequence error | Result of executing full status checking |
| **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**

| Register Name | Symbol | Function | Address | Setting |
|---|---|---|---|---|
| PMRJ | PMRJ[1:0] | The OSC1 and OSC2 functions are selected for pins PJ0/OSC1 and PJ1/OSC2. | H'FF000C | B'11 |
| FLMCR1 | FMWUS | Setting to select reprogramming by the CPU to be initiated by byte-length instructions. | H'FF0660 | 0 |
| | FMEWMOD | EW1 mode is selected. | | 1 |
| | FMCMDEN*1*2*3*4 | Software commands for flash memory are enabled. | | 1 |
| DFPR | DFPR1*5*6 | E/W of data-flash B is disabled | H'FF0662 | 1 |
| | DFPR0*5*6 | E/W of data-flash A is enabled. | | 0 |
| FLMSTR | FMEBSF*7*8 | Erasure/blank checking status flag<br>0: Normal end<br>1: End with an error<br>[Setting conditions]<br>• The erasure command is executed and results in unsuccessful erasure.<br>• The blank-checking command is executed and it is detected that the specified block is not blank.<br>[Clearing condition]<br>• The status clearing command is issued. | H'FF0663 | — |
| | FMPRSF*7*8 | Programming status flag<br>0: Normal end<br>1: End with an error<br>[Setting conditions]<br>• The programming command is executed and results in unsuccessful programming.<br>• The lock-bit programming command is executed and results in unsuccessful programming.<br>[Clearing condition]<br>• The status clearing command is issued. | | — |
| | FMRDY | Flash memory ready/busy status flag<br>0: Busy (programming or erasure in progress)<br>1: Ready<br>[Setting condition]<br>• Flash memory is not being programmed or erased.<br>[Clearing condition]<br>• Flash memory is being programmed or erased. | | — |
| SYSCCR | PHIHSEL | ϕhigh clock source is set to ϕosc. | H'FF06D0 | 1 |
| LPCR1 | PSCSTP | PSC divider is operating. | H'FF06D1 | 0 |
| | PHIBSEL | ϕbase clock source is set to ϕhigh. | | 1 |
| LPCR2 | PHI[2:0] | System clock ϕ is set to ϕbase. | H'FF06D2 | B'000 |
| LPCR3 | PHIS[2:0] | Bus-master operating clock ϕs is set to ϕ. | H'FF06D3 | B'000 |
| OSCCSR | | ϕosc oscillation settling time is set. | H'FF06D5 | H'0E |

| Register Name | Symbol | Function | Address | Setting |
|---|---|---|---|---|
| TMWD | | Clock input to the WDT is prohibited. | H'FFFF99 | H'F7 |
| TCSRWD | | Writing to the TMWD register is enabled. | H'FFFF9A | H'A3 |
| MSTCR1 | MSTWDT | WDT is released from the module standby mode. | H'FFFFDC | 0 |

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 8 RAM Usage**

| Label Name | Functions | Memory | Module Name |
|---|---|---|---|
| — | Data-flash area A for back up (H'FFDF80 to H'FFEF7F) | 4 KB | — |

## 4.4 Descriptions of Definitions in Use

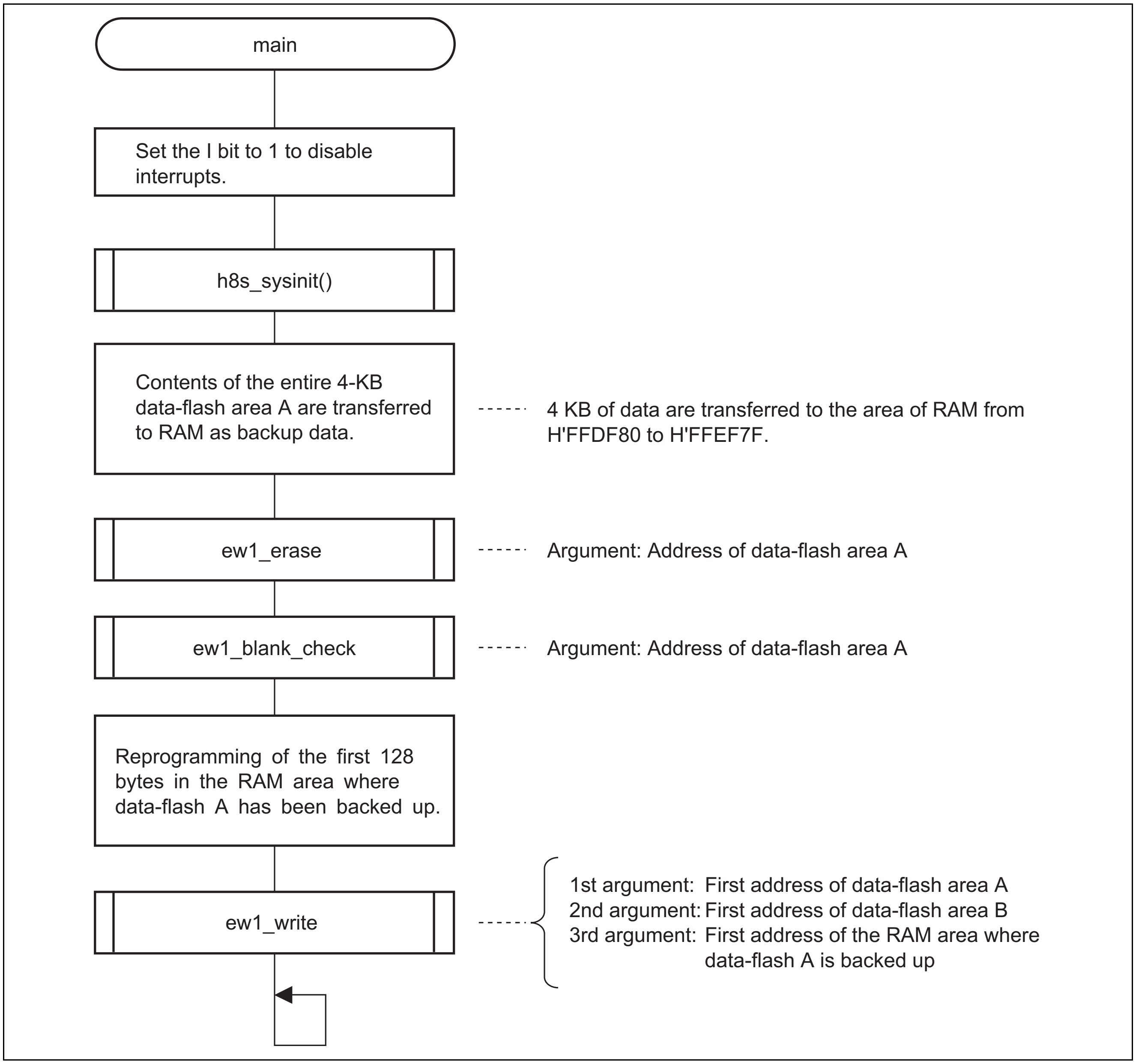
Table 9 gives descriptions of the definitions used in this sample task.

**Table 9 Descriptions of Definitions in Use**

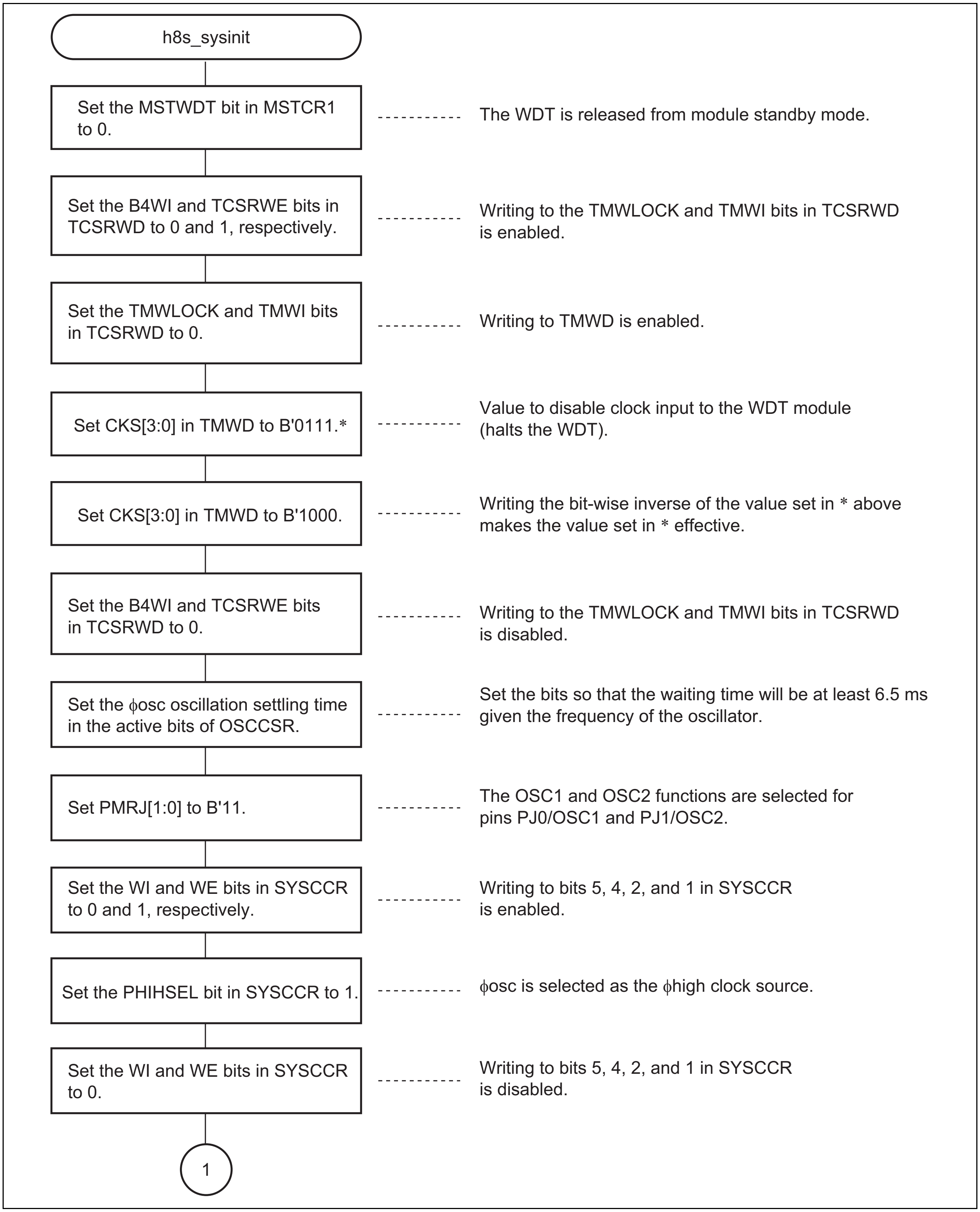
| Label Name | Description | Constant |
|---|---|---|
| FULL_STATUS | Mask value for use in full status checking of FLMSTR | H'28 |
| COMMAND_SEQUENCE_ERR | Value corresponding to a command sequence error; for use in full status checking of FLMSTR | H'28 |
| ERASE_BLANK_ERR | Value corresponding to an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

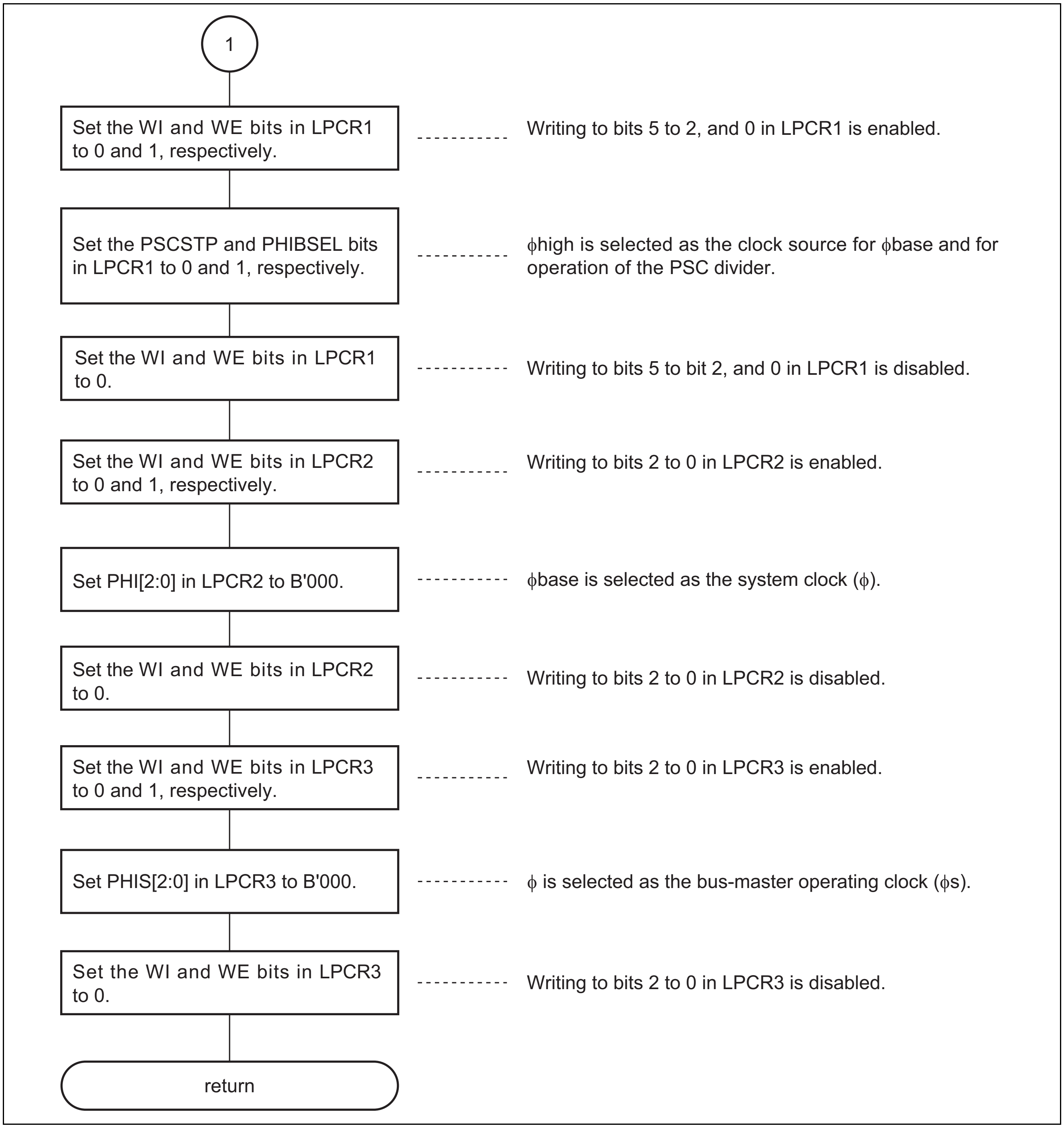
## 5. Flowcharts

### 5.1 Main Routine

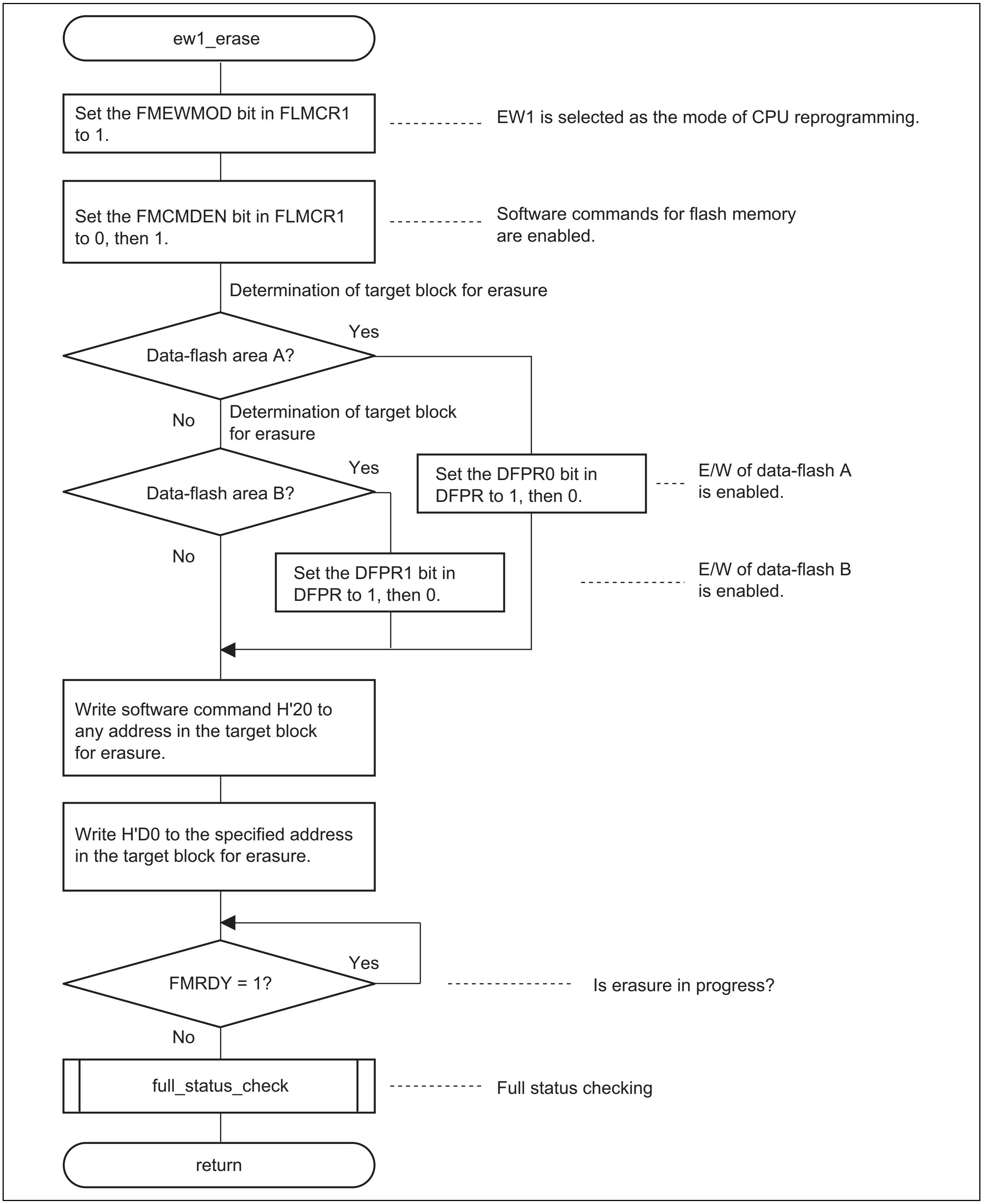


## 5.2 System Initialization Routine

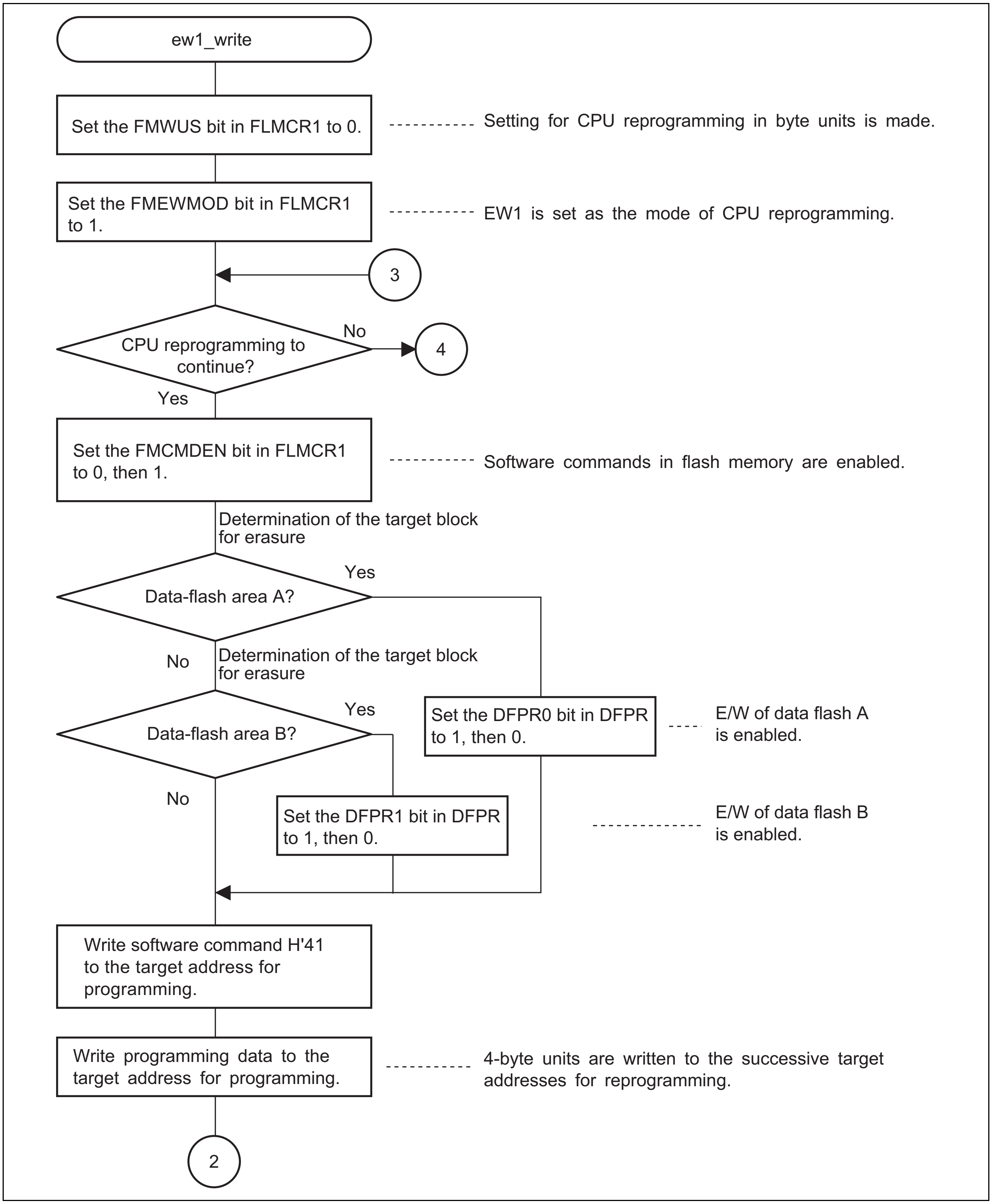


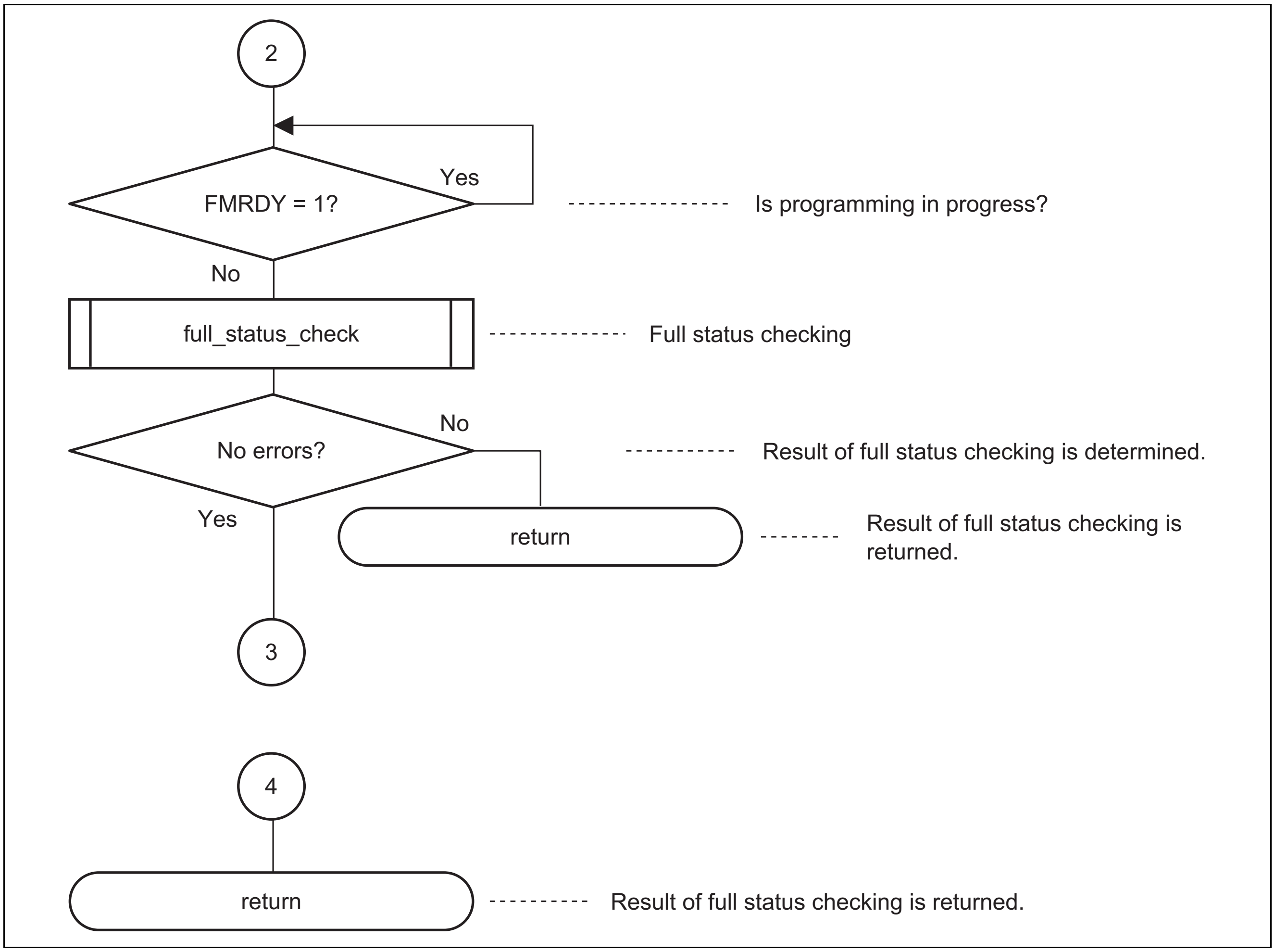

1
Set the WI and WE bits in LPCR1 to 0 and 1, respectively.
Writing to bits 5 to 2, and 0 in LPCR1 is enabled.
Set the PSCSTP and PHIBSEL bits in LPCR1 to 0 and 1, respectively.
ϕhigh is selected as the clock source for ϕbase and for operation of the PSC divider.
Set the WI and WE bits in LPCR1 to 0.
Writing to bits 5 to bit 2, and 0 in LPCR1 is disabled.
Set the WI and WE bits in LPCR2 to 0 and 1, respectively.
Writing to bits 2 to 0 in LPCR2 is enabled.
Set PHI[2:0] in LPCR2 to B'000.
ϕbase is selected as the system clock (ϕ).
Set the WI and WE bits in LPCR2 to 0.
Writing to bits 2 to 0 in LPCR2 is disabled.
Set the WI and WE bits in LPCR3 to 0 and 1, respectively.
Writing to bits 2 to 0 in LPCR3 is enabled.
Set PHIS[2:0] in LPCR3 to B'000.
ϕ is selected as the bus-master operating clock (ϕs).
Set the WI and WE bits in LPCR3 to 0.
Writing to bits 2 to 0 in LPCR3 is disabled.
return

## 5.3 Erasure Routine

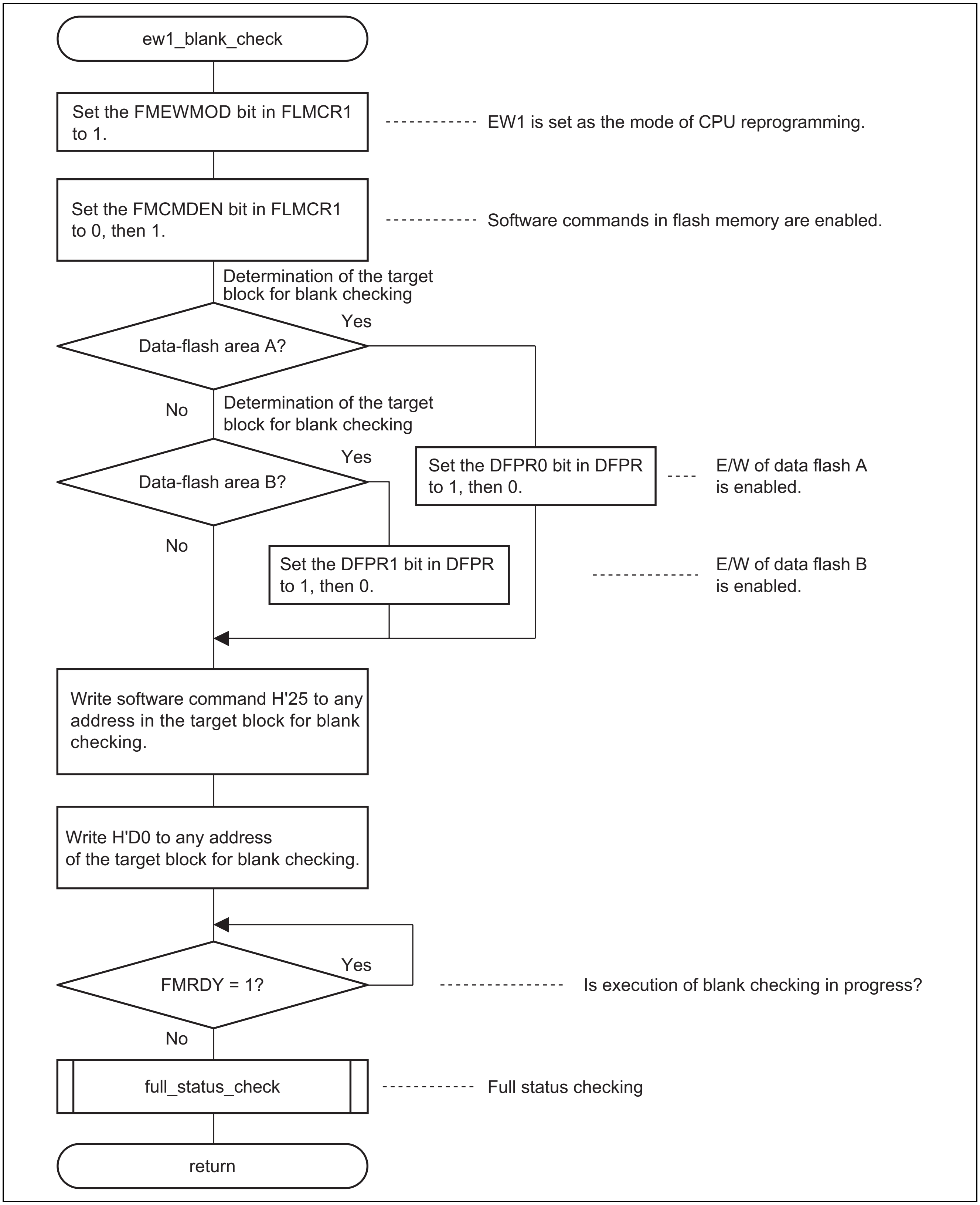


## 5.4 Programming Routine

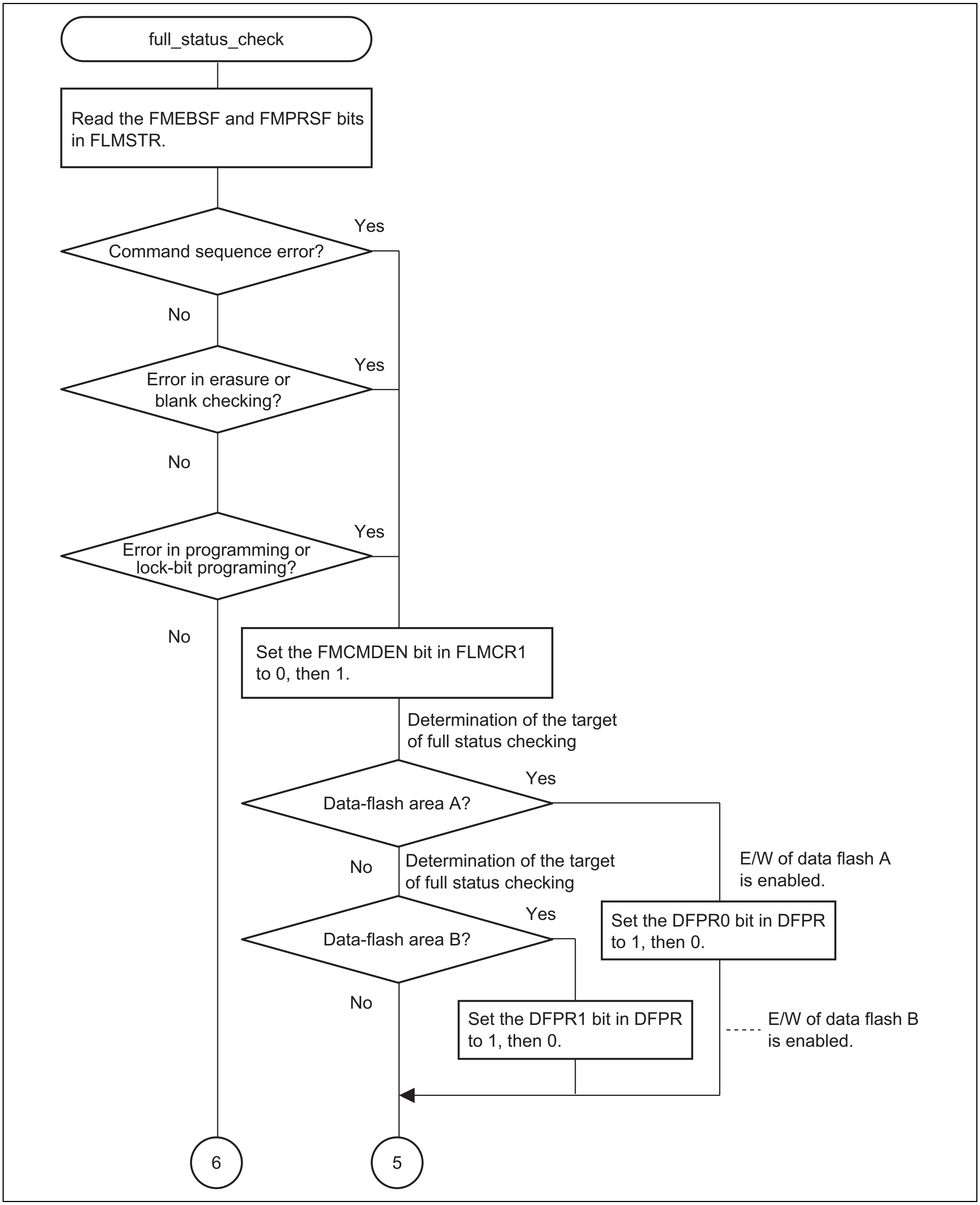


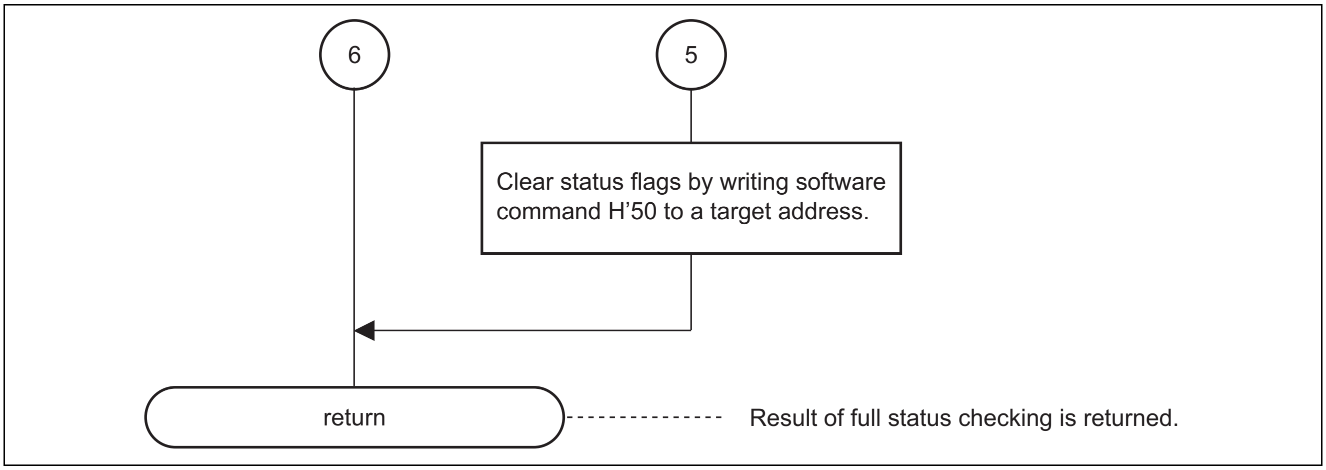

2
FMRDY = 1?
Yes
No
Is programming in progress?
full_status_check
Full status checking
No errors?
No
Yes
Result of full status checking is determined.
return
Result of full status checking is returned.
3
4
return
Result of full status checking is returned.

## 5.5 Blank Checking Routine



## 5.6 Full Status Checking Routine




6
5
Clear status flags by writing software command H'50 to a target address.
return
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);
}
```

```
/*****************************************************/
/* Name        : ew1_write                          */
/* Parameters  : (wr_top)address of write top       */
/*               (wr_end)address of write end       */
/* Returns     : write result                       */
/* Description : data flash write program           */
/*               of EW1 mode                        */
/*****************************************************/
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        : ew1_erase                            */
/* Parameters  : (er_blk)address of erase BLOCK       */
/* Returns     : erase result                         */
/* Description : data flash erase program             */
/*               of EW1 mode                          */
/******************************************************/
unsigned char ew1_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);
}
```

```
/*****************************************************/
/* Name        : ew1_blank_check                     */
/* Parameters  : (blk_top)address of blank check     */
/* Returns     : blank check result                  */
/* Description : blank check program                 */
/*****************************************************/
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);
}
```

```
/******************************************************/
/* Name        : full_status_check                    */
/* Parameters  : (addr)E/W block address              */
/* Returns     : full status check result             */
/* Description : full status check                    */
/******************************************************/
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 */

            /* Data Flash A ? */
            if ( (FLASH_BLK_A <= addr) && (addr < 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 <= addr) && (addr <= 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 */
            }

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

            break;

    default :                                        /* No error */

            break;
    }

    return (tmp_flmstr);
}
```

```
/*****************************************************/
/* Name        : h8s_sysinit                         */
/* Parameters  : None                                */
/* Returns     : None                                */
/* Description : initialize H8S/20203               */
/*****************************************************/
void h8s_sysinit(void)
{
    MSTCR1.BIT.MSTWDT = 0;                              /* WDT module standby off */

    /* stop WDT */
    WDT.TCSRWD.BYTE = 0x97;                             /* write enable TMWLOCK, TMWI */
    WDT.TCSRWD.BYTE = 0xA3;                             /* write enable TMWD */
    WDT.TMWD.BYTE = 0xF7;                               /* Not select clock source */
    WDT.TMWD.BYTE = 0xF8;                               /* write bit inversion */
    WDT.TCSRWD.BYTE = 0x87;                             /* write disable TMWLOCK, TMWI */

    CPG.OSCCSR.BYTE = 0x0E;                             /* wait over 6.5ms, Phi_osc=20MHz */
    PMRJ.BYTE = 0x03;                                   /* select OSC1,OSC2 */

    CPG.SYSCCR.BYTE = (CPG.SYSCCR.BYTE & 0x7F)| 0x40;   /* WI=0, WE=1 */
    CPG.SYSCCR.BYTE = 0x60;                             /* high=Phi_osc, Phi_low=Phi_loco */
    CPG.SYSCCR.BYTE = CPG.SYSCCR.BYTE & 0x3F;           /* WI=0, WE=0 */

    CPG.LPCR1.BYTE = (CPG.LPCR1.BYTE & 0x7F) | 0x40;    /* WI=0, WE=1 */
    CPG.LPCR1.BYTE = 0x41;                              /* PSC on, Phi_base=Phi_high */
    CPG.LPCR1.BYTE = CPG.LPCR1.BYTE & 0x3F;             /* WI=0, WE=0 */

    CPG.LPCR2.BYTE = (CPG.LPCR2.BYTE & 0x7F) | 0x40;    /* WI=0, WE=1 */
    CPG.LPCR2.BYTE = 0x40;                              /* select system clock */
    CPG.LPCR2.BYTE = CPG.LPCR2.BYTE & 0x3F;             /* WI=0, WE=0 */

    CPG.LPCR3.BYTE = (CPG.LPCR3.BYTE & 0x7F) | 0x40;    /* WI=0, WE=1 */
    CPG.LPCR3.BYTE = 0x40;                              /* select clock of bus master */
    CPG.LPCR3.BYTE = CPG.LPCR3.BYTE & 0x3F;             /* WI=0, WE=0 */
}
```

## 6.1 Designation of Link Addresses

| Section Name | Address |
|---|---|
| PResetPRG, PIntPRG | H'000400 |
| P, C$DSEC, C$BSEC, D | H'000800 |
| B, R | H'FFEF80 |
| S | H'FFFD80 |

## Website and Support

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

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

## Revision Record

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.00 | Oct.31.08 | — | First edition issued |

## 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.