To our customers,

# Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: http://www.renesas.com

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

# Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

# H8S Family

## Example of Reprogramming On-Chip Flash Memory in User Program Mode "SCI (Clock Synchronous Mode)"

### Introduction

This Application Note is a summary of sample reprogramming operations performed on the on-chip flash memory (user MAT) of the H8S/2556, 2552 or 2506 group MCU operated in the user program mode by clock synchronous communication over a serial communications interface (hereafter called SCI).

### Target Device

H8S/2500 Series H8S/2556 Group MCU

### Contents

1. Specifications ..... 2
2. Description of Software Operation ..... 4
3. Description of Registers ..... 5
4. Flowchart ..... 19
5. Application Memory Map ..... 29
6. Program Listings ..... 30
7. Appendix ..... 36

# 1. Specifications

In this sample task, the MCU is started up in the user program mode. During operation, an IRQ0 interrupt is generated by the sending side of programming data. After three blocks of the flash memory, from EB10 to EB12, have been erased, the received programming data is programmed to the same three blocks from EB10 through EB12. A serial communications interface (hereafter called SCI) is used in the the clock synchronous mode. To be more specific, the MCU uses the SCI2.

Figure 1 shows a block diagram of an on-board reprogramming setup using an SCI (clock synchronous mode).

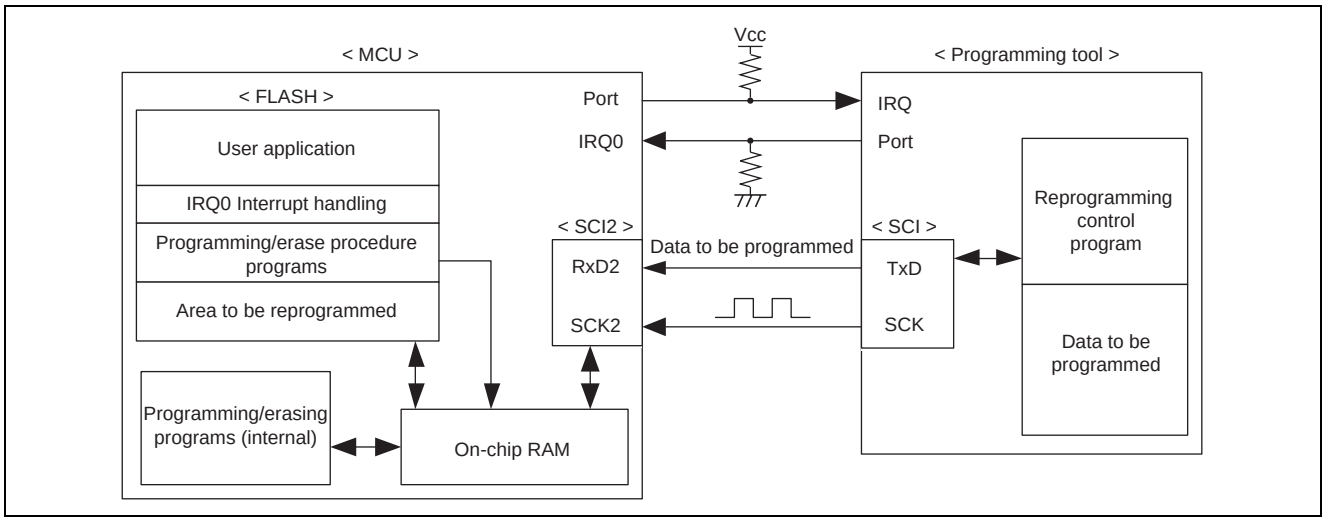



**Figure 1 Block diagram of on-board reprogramming setup using an SCI (clock synchronous mode)**

## 1.1 Operation Mode

User-program-mode pin settings are shown as follows.

**Table 1 Pin settings**

| Pin | Level |
|---|---|
| RES | 1 |
| MD0 | 0/1 |
| MD1 | 1 |
| MD2 | 1 |

## 1.2 Software Development Environment

For software development of this sample task, High-performance Embedded Workshop3 (HEW3) Ver. 3.01.06.001 is used. For the programming of software in the user program mode, Flash Development Toolkit (FDT) 2.0 is used.

## 1.3 Interface Specifications

The SCI interface specifications are shown below.

**Table 2 Interface specifications**

| Item | MCU | Programming Tool |
|---|---|---|
| Channel used | Channel 2 (SCI2) for reception only | For transmission only |
| Clock (SCK) | Input | Output |
| Communications mode | Clock synchronous | Clock synchronous |
| Baud rate | 1 Mbps | 1 Mbps |

## 1.4 Control Specifications

The control specifications are shown below.

**Table 3 Control Specifications**

| Control | MCU | Programming Tool |
|---|---|---|
| Flash reprogramming request signal | IRQ0 (rising edge) | Port (output) |
| Programming data request signal | Port PJ0 (pin 53: output) | IRQ (falling edge) |

## 1.5 Explanation of Control Signal Behavior

In this sample task, an IRQ0 interrupt is used for receiving the flash reprogramming request signal. (During the flash memory programming/erase operations, interrupts are disabled.)

Figure 2 shows a series of control operations that occur from the receipt of the flash reprogramming request signal up until the completion of the flash memory reprogramming while the user application program is being run.

Note that, in this sample task, no error handling procedure is defined. The programming tool used in this sample task is designed to assumes that reprogramming operation has successfully completed in the absence of a programming data request signal from the MCU within a predefined duration after a reprogramming start signal has been transmitted.

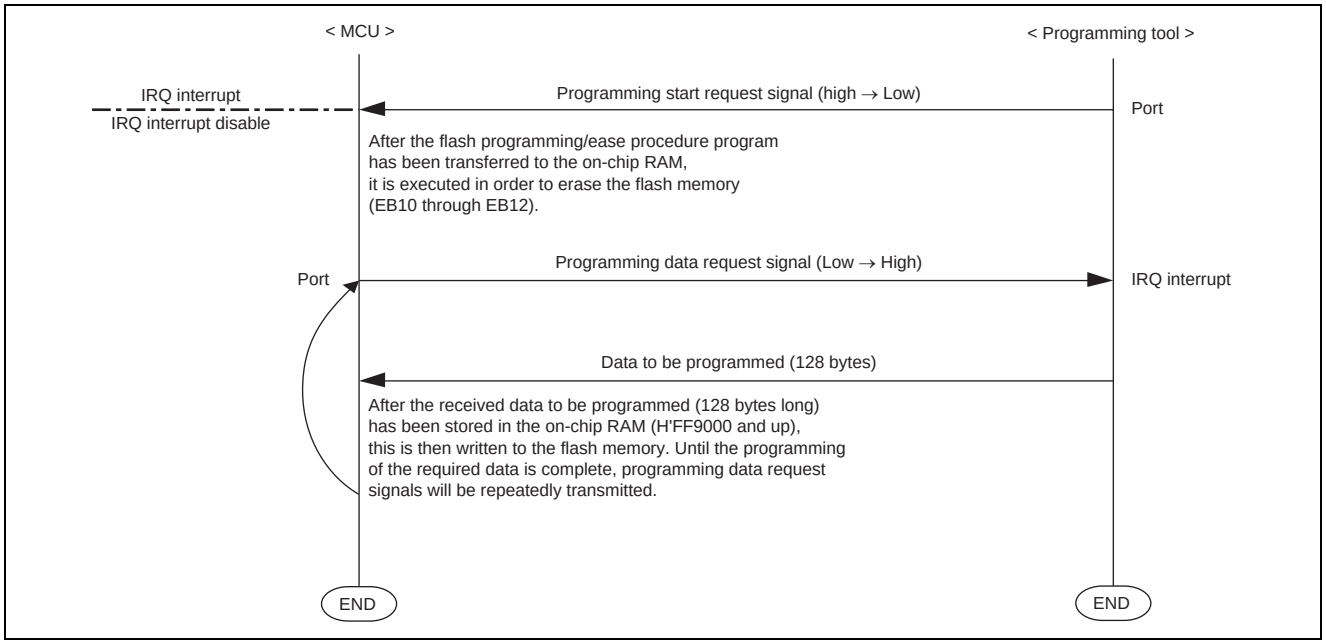



**Figure 2 Control Operation**

## 2. Description of Software Operation

An overview of the flash reprogramming operation for this sample task is shown below.

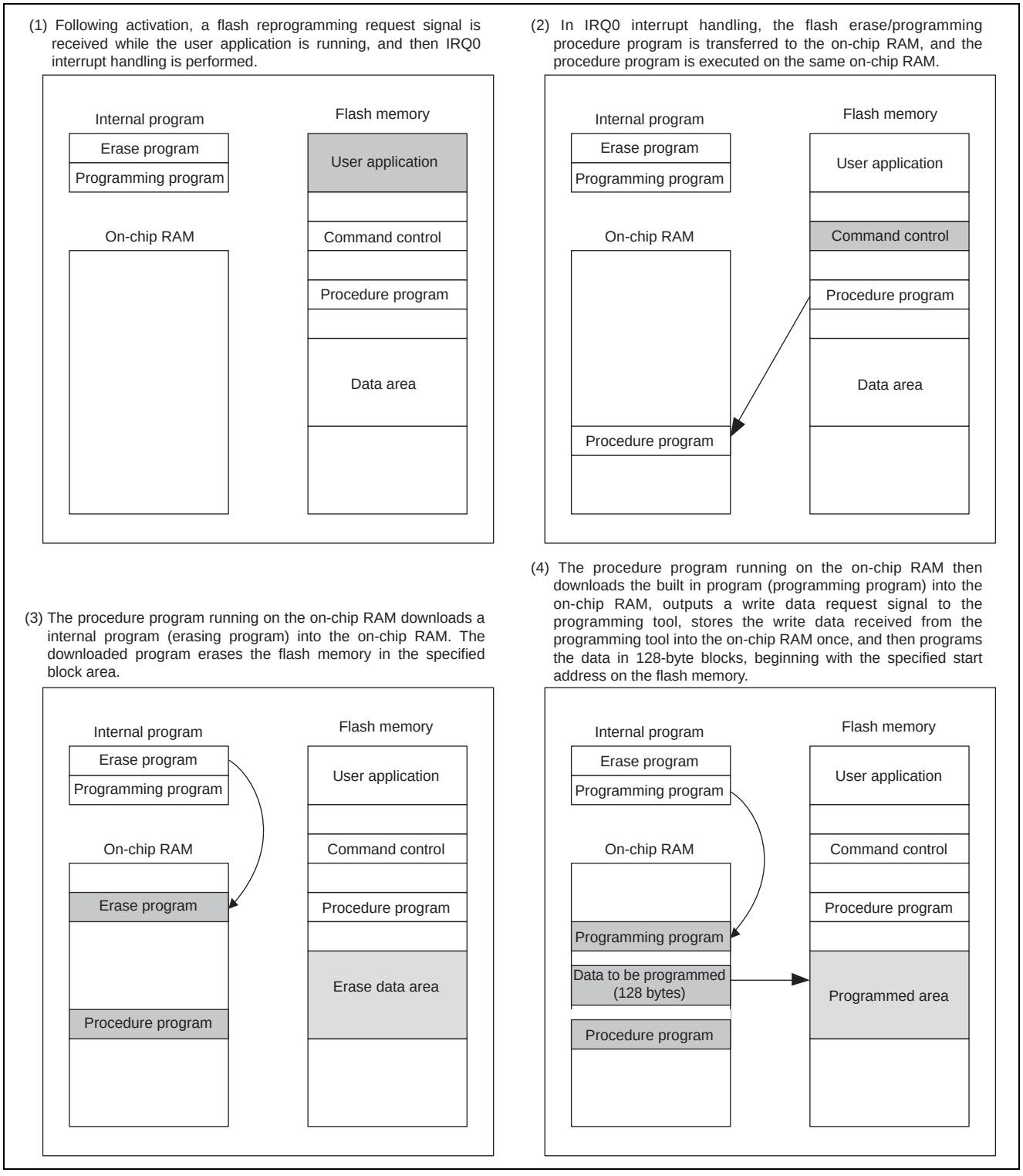



**Figure 3 Explanation of Software Operation**

# 3. Description of Registers

The registers and parameters for controlling flash memory units are described below.

In order to enable access to a register controlling flash memory other than RAMER (RAM emulation register), the SYSCR2's FLSHE bit must be set to 1 in a mode where the on-chip flash memory is operational. However, when FLSHE = 1, certain TPU control registers (H'FFFE80 to H'FFFEB1) become inaccessible. Always clear the FLSHE bit to 0 before accessing TPU registers.

## 3.1 Programming/Erase Interface Registers

This section describes the programming/erase interface registers. All are 8-bit registers allowing byte access only. These registers, save the FLER bit of the FCCS register, are initialized upon power-on reset, as well as upon entering hardware standby mode, software standby mode, or watch modes, respectively. The FLER bit, however, is not initialized upon entering software standby or watch modes.

### 3.1.1 Flash-Code-Control Status Register (FCCS) Initial Value: H'80

The FCCS consists of a monitor bit for checking for errors during the programming/erasing of flash memory, and a bit requesting download of an internal program.

**Table 4 Flash-Code-Control Status Register (FCCS) Initial Value**

| Bit | Bit Name | R/W | Description |
|---|---|---|---|
| 7 | — | R | Reserved bit<br>This bit is always read as 1. Always set this bit to 1 when writing as well. |
| 6, 5 | — | R | Reserved bits<br>These bits are always read as 0. Always set these bits to 0 when writing as well. |
| 4 | FLER | R | Flash memory error<br>This is a bit for indicating that an error has occurred during flash memory programming/erase processing. If FLER = 1 is set, the flash memory enters an error protection state. It is initialized at power-on reset or transition into the hardware standby mode. When FLER becomes 1, a high voltage will be applied inside the flash memory. Therefore, in order to prevent possible damage to the flash memory, release a reset after a reset input period of 100 ms, which is longer than usual.<br>0: Flash memory is operating normally.<br>Programming/erase protection (error protection) on the flash memory is disabled.<br>[Clearing condition] Cleared at power-on reset or transition into the hardware standby mode.<br>1: Indicates that an error has occurred during flash memory programming or erase operation. Programming/erase protection (error protection) on the flash memory is enabled. |
| 3 to 1 | — | R | Reserved bits<br>These bits are always read as 0. Always set these bits to 0 when writing as well. |

**Table 4 Flash-Code-Control Status Register (FCCS) Initial Value (cont)**

| Bit | Bit Name | R/W | Description |
|---|---|---|---|
| 0 | SCO | (R)/W | Source program copy operation<br>This is a request bit to download the internal programming/erase program into the on-chip RAM. If 1 is written to this bit, the internal program selected by the FPCS or FECS register will be automatically downloaded into the on-chip RAM area specified by the FTDAR register. In order to write 1 to this bit, it is necessary to clear the RAM emulation state, write H'A5 to the FKEY register, and execute the downloaded program on the on-chip RAM.<br>Immediately after writing 1 to this bit, always execute four NOP instructions.<br>Note that, when downloading is completed, this bit is cleared to 0 and thus it is impossible to read a 1 state from this bit.<br>0: Downloads of the internal programming/erase program to the on-chip RAM<br>[Clearing condition] Cleared when a download completes.<br>1: Generates a request to download the internal programming/erase program into the on-chip RAM.<br>[Setting condition] The bit is set when 1 is written with all of the following conditions being satisfied.<br>(1) H'A5 has been written in the FKEY register.<br>(2) The code is being executed in the on-chip RAM.<br>(3) Not in the RAM emulation mode. (That is, the RAMS bit of RAMER is 0.) |

### 3.1.2 Flash Program Code Select Register (FPCS) Initial Value: H'00

FPCS is a register to select/unselect the internal programming program to be downloaded.

**Table 5 Flash Program Code Select Register (FPCS) Initial Value**

| Bit | Bit Name | R/W | Description |
|---|---|---|---|
| 7 to 1 | — | R | Reserved bits<br>These bits are always read as 0. Always set these bits to 0 when writing as well. |
| 0 | PPVS | R/W | Program pulse verify<br>Selects the programming program.<br>0: Does not select the internal programming program.<br>[Clearing condition] Cleared when a transfer completes.<br>1: Selects the internal programming program. |

### 3.1.3 Flash Erase Code Select Register (FECS) Initial Value: H'00

FECS is a register to select/deselect the internal erase program to be downloaded.

**Table 6 Flash Erase Code Select Register (FECS) Initial Value**

| Bit | Bit Name | R/W | Description |
|---|---|---|---|
| 7 to 1 | — | R | Reserved bits<br>These bits are always read as 0. Always set these bits to 0 when writing as well. |
| 0 | EPVB | R/W | Erase pulse verify block<br>Selects the erase program.<br>0: Does not select the internal erase program.<br>[Clearing condition] Cleared when a transfer completes.<br>1: Selects the internal erase program. |

### 3.1.4 Flash Key Code Register (FKEY) Initial Value: H'00

FKEY is a register for software protection purposes that permits the download of an internal program and the programming/erasing of the flash memory. Unless a key code is entered before writing 1 to the SCO bit for downloading an internal program or before executing the downloaded programming/erase program, such processing cannot be performed.

**Table 7 Flash Key Code Register (FKEY) Initial value**

| Bit | Bit Name | R/W | Description |
|---|---|---|---|
| 7 | K7 | R/W | Key code |
| 6 | K6 | R/W | Writing to the SCO bit is only enabled when H'A5 has been written into this register. If any value other than H'A5 is written into the FKEY register, writing 1 to the SCO bit is not allowed and, therefore, a program cannot be downloaded to the on-chip RAM. Only after H'5A has been written, the programming/erasing of the flash memory becomes possible. Even if an internal programming/erase program is executed with any value other than H'A5 written in the FKEY register, the flash memory cannot be programmed or erased.<br>H'A5: Permits writing to the SCO bit. (Any value other than H'A5 would not allow setting of the SCO bit.)<br>H'5A: Permits flash memory programming/erasing. (Any value other than H'5A would cause the software protection state to be retained.)<br>H'00: Initial value |
| 5 | K5 | R/W | |
| 4 | K4 | R/W | |
| 3 | K3 | R/W | |
| 2 | K2 | R/W | |
| 1 | K1 | R/W | |
| 0 | K0 | R/W | |

### 3.1.5 Flash MAT Select Register (FMATS) Initial Value: H'00*

FMATS is a register that specifies the selection of either the user or the user boot MAT respectively.

**Table 8 Flash MAT Select Register (FMATS) Initial Value**

| Bit | Bit Name | R/W | Description |
|---|---|---|---|
| 7 | MS7 | R/W | MAT select<br>Any value other than H'AA specifies the selection of the user MAT, while H'AA written in this register specifies the selection of the user boot MAT. By writing a value to FMATS, MAT switching is effected. In the user program mode, the user boot MAT cannot be rewritten even if selected in FMAT. To rewrite the user boot MAT, perform reprogramming either in the boot or programmer mode.<br>H'AA: Selects the user boot MAT.<br>(A value other than H'AA specifies the user MAT.)<br>This is the initial value when startup is done in the user boot mode.<br>H'00: This is the initial value when startup is done in a mode other than the user boot mode.<br>(The user MAT is selected.)<br>[Programming enable condition] The code is being executed within the on-chip RAM. |
| 6 | MS6 | R/W | |
| 5 | MS5 | R/W | |
| 4 | MS4 | R/W | |
| 3 | MS3 | R/W | |
| 2 | MS2 | R/W | |
| 1 | MS1 | R/W | |
| 0 | MS0 | R/W | |

Note *The initial value is H'AA when in the user boot mode; H'00 otherwise.

### 3.1.6 Flash Transfer Destination Address Register (FTDAR) Initial Value: H'00

FTDAR is a register used to specify the destination address on the on-chip RAM to which the internal program is to be downloaded. Make the setting of this register before writing 1 to the SCO bit in the FCCS register.

**Table 9 Flash Transfer Destination Address Register (FTDAR) Initial Value**

| Bit | Bit Name | R/W | Description |
|---|---|---|---|
| 7 | TDER | R/W | Transfer destination address setting error<br>When an error has occurred in the download start address specification using the bits from TDA6 to TDA0, 1 is set to this bit. Concerning evaluation of a potential error in the address specification, a check is made of the value in bits TDA6 to TDA0 to determine whether it falls within the range between H'00 to H'07 when a program is downloaded with the FCCS register's SCO bit set to 1. Before setting the SCO bit to 1, set the FTDAR value to between H'00 to H'07 in addition to setting this bit to 0.<br>0: The value set to bits TDA6 to TDA0 is normal.<br>1: The value set to bits TDA6 to TDA0 is outside the range of H'00 to H'07, meaning that download has been aborted. |
| 6 | TDA6 | R/W | Transfer destination address<br>Specifies the download start address. Using a setting value within the permissible range of H'00 to H'07, the download start address on the on-chip RAM can be specified in increments of 4 kilobytes.<br>H'00: Sets the download start address to H'FF9000.<br>H'01: Sets the download start address to H'FFA000.<br>H'02: Sets the download start address to H'FFB000.<br>H'03: Sets the download start address to H'FFC000.<br>H'04: Sets the download start address to H'FFD000.<br>H'05: Sets the download start address to H'FFE000.<br>H'06: Sets the download start address to H'FF8000.<br>H'07: Sets the download start address to H'FF7000.<br>H'08 - H'FF: Must not be set. If any of these values are set, the TDER bit becomes 1 during the download operation and the download operation of the internal program is aborted. |
| 5 | TDA5 | R/W | |
| 4 | TDA4 | R/W | |
| 3 | TDA3 | R/W | |
| 2 | TDA2 | R/W | |
| 1 | TDA1 | R/W | |
| 0 | TDA0 | R/W | |

### 3.1.7 System Control Register (SYSCR2) Initial Value: H'00*

The SYSCR2 register controls register access.

**Table 10 System Control Register (SYSCR2) Initial Value**

| Bit | Bit Name | R/W | Description |
|---|---|---|---|
| 7 to 4 | — | — | Reserved bits<br>Write 0s. |
| 3 | FLSHE | R/W | Flash memory control register enable<br>Writes 0s to control the CPU access made by the flash memory control register. Setting the FLSHE bit to 1 enables read/programming of the flash memory control register. Clearing the FLSHE bit to 0 deselects the flash memory control register. At this time, the contents of the flash memory control register are retained.<br>0: Flash control logic unit which controls H'FFFFA4 to H'FFFFAF is disabled.<br>1: Flash control logic unit which controls H'FFFFA4 to H'FFFFAF is disabled. |
| 2 | — | — | Reserved bit<br>Write 0. |
| 1,0 | — | R/W | Reserved bits<br>Write 0s. |

Note * The initial values of bits 7 to 4 and 2 are undefined. The initial value of other bits is 0.

## 3.2 Programming/Erase Interface Parameters

Programming/erase interface parameters are used for specifying operating frequency, user branch destination address, programming data storing address, blocks to be erased, and so on of an internal program that has been downloaded as well as exchanging processing and operation results. For these parameters, the CPU's general registers (ER0, ER1) and on-chip RAM areas are used. Initial values at the time of power-on reset and hardware standby remain undefined.

During download, initialization, or the execution of an internal program, the values of the CPU registers other than that of R0L are saved. In R0L, the value returned as the result of processing is written. Since the stack area is used for saving the values of registers other than that of the R0L, always ensure this is allocated prior to any processing. (The maximum usable size of the stack area is 128 bytes.)

Programming/erase interface parameters are used in the following four types of processing:

1. Download control
2. Pre-programming/erase operation initialization
3. Programming operation
4. Erase operation

Different parameters are used for different types of processing. The following table shows which parameters are used for which types of operation.

Note that the FPFR parameter values are returned as the result of the initialization, programming and erase operations. However, the meanings of individual bits vary depending on the type of processing performed.

**Table 11 Parameters and Modes in Which They are Used**

| Parameter Name | Abbr. | Down-load | Initializa-tion | Pro-gramming | Erase | R/W | Initial Value | Assigned to |
|---|---|---|---|---|---|---|---|---|
| Download pass/fail result | DPFR | Used | | | | R/W | Undefined | On-chip RAM* |
| Flash pass/fail result | FPFR | | Used | Used | Used | R/W | Undefined | CPU's R0L |
| Flash program erase frequency control | FPEFEQ | | Used | | | R/W | Undefined | CPU's ER0 |
| Flash user branch address set parameter | FUBRA | | Used | | | R/W | Undefined | CPU's ER1 |
| Flash multi-purpose address area | FMPAR | | | Used | | R/W | Undefined | CPU's ER1 |
| Flash multi-purpose data destination area | FMPDR | | | Used | | R/W | Undefined | CPU's ER0 |
| Flash erase block select | FEBS | | | | Used | R/W | Undefined | CPU's ER0 |

Note *A single byte in the download destination start address specified by the FTDAR register.

### 3.2.1 Download Control

The internal program is automatically downloaded by setting the SCO bit to 1. The area on the on-chip RAM into which the program is downloaded is a 2-kilobyte- area from the start address specified by the FTDAR register. Download control is set by means of the programming/erase interface registers, and a return value is passed as the DPFR parameter.

(1) Download Pass/Fail Parameter (DPFR: 1 byte of start address on the on-chip RAM specified by the FTDAR register)

This is a value returned as the result of a download. The success or otherwise of the download can be assessed by the value of this parameter. Since it is difficult to verify whether the setting of the SCO bit to 1 was successful, set the single-byte start address on the on-chip RAM specified in the FTDAR register prior to download (that is, before setting the SCO bit to 1) to a specification other than the download return value (for example, to H'FF), ensure positive verification.

**Table 12 Download Pass/Fail Result Parameter**

| Bit | Bit Name | R/W | Description |
|---|---|---|---|
| 7 to 3 | — | — | Reserved bits<br>Value 0 is returned. |
| 2 | SS | R/W | Source select error detection bit<br>As a downloadable program, only one type of internal program can be specified. If none or multiple types are selected, or a program is selected without mapping, an error occurs.<br>0: Program to be downloaded is correctly selected.<br>1: Download error. (Multiple programs selected or program selected without mapping.) |
| 1 | FK | R/W | Flash key register error detection bit<br>This is a bit returning the result following checking of whether or not the FKEY register value is H'A5.<br>0: FKEY register setting is correct. (FKEY = H'A5)<br>1: FKEY register setting value error. (The FKEY value is not H'A5.) |
| 0 | SF | R/W | Success/fail bit<br>This is a bit that specifies whether downloading has successfully completed. By reading back the program that has been downloaded to the on-chip RAM, a check is made to determine whether it has been successfully transferred to the on-chip RAM, and the result of this check is returned.<br>0: Download of an internal program has been completed successfully (without error).<br>1: Download of an internal program has failed. (An error has occurred.) |

### 3.2.2 Programming/Erase Initialization

An internal program to be downloaded includes an initialization module.

The programming/erase operation requires the application of pulses of a given time width, and the required pulse width is created by means of constructing a wait loop using the CPU instructions. Accordingly, it is necessary to set the operating frequency of the CPU.

It is the initialization program that makes settings such as the programming/erase program parameters of the downloaded program.

(1) Flash Programming/Erasing Frequency Parameter (FPEFEQ: CPU's general register ER0)

This is a parameter used to set the operating frequency of the CPU.

**Table 13 Flash Programming/Erasing Frequency Parameter**

| Bit | Bit Name | R/W | Description |
|---|---|---|---|
| 31 to 16 | F31 to F16 | R/W | Reserved bits<br>These bits should be cleared to 0. |
| 15 to 0 | F15 to F0 | R/W | Frequency setting bits<br>Set a CPU operating frequency. Calculate the settable value as follows:<br>• Round off the operating frequency in MHz to two decimal places.<br>• Multiply the value by 100, convert the obtained value to binary form, and write it to the FPEFEQ parameter (general register ER0).<br>As a concrete example, when the CPU's operating frequency is 25.000 MHz, calculations are as follows:<br>Round off 25.000 at the third decimal place to obtain 25.00. Convert 25.00 × 100 = 2500 into binary form to obtain B'0000, 1001, 1100, 0100 (H'09C4), and set it to ER0. |

(2) Flash User Branch Address Setting Parameter (FUBRA: CPU's general register ER1)

This parameter sets the user branch destination address. The specified user program can be executed in units of a predetermined amount of processing during programming/erase operations.

**Table 14 Flash User Branch Address Setting Parameter**

| Bit | Bit Name | R/W | Description |
|---|---|---|---|
| 31 to 0 | UA31 to UA16 | R/W | User branch destination address<br>When no user branching is required, set address 0 (H'00000000).<br>A user branch destination must be within the RAM space other than the area occupied by the internal program transferred or the external bus space. Proceed with caution to avoid branching to an area without execution code, which would cause a runaway, and avoid corrupting the internal program area or a stack area. In the event of a program runaway, flash memory values are not guaranteed.<br>During user-branched processing, do not download, initialize, or invoke programming/erase program routines of the internal program. Programming/erasure subsequent to the user branch routine cannot be otherwise guaranteed. In addition, do not modify pre-prepared data to be written. Likewise, do not rewrite the programming/erase interface register or make a transition to the RAM emulation mode during user branched processing. After completing the user-branch processing, return to the programming/erase program by the RTS instructions. |

(3) Flash Pass/Fail Parameter (FPFR: CPU's general register R0L)

This section describes the FPFR as a return value, indicating the result of initialization.

**Table 15 Flash Pass/Fail Parameter**

| Bit | Bit Name | R/W | Description |
|---|---|---|---|
| 7 to 3 | — | — | Reserved bits<br>Value 0 is returned. |
| 2 | BR | R/W | User branch error detection bit<br>Returns the result of a check performed to determine whether or not the specified user branch destination address is outside the storage area of a programming/erase-related programs that have been downloaded.<br>0: User branch address setting is correct.<br>1: User branch address setting is incorrect. |
| 1 | FQ | R/W | Frequency error detection bit<br>Returns the result of a check determining whether or not the specified CPU operating frequency is within the supported range.<br>0: Operating frequency setting is correct.<br>1: Operating frequency setting is incorrect. |
| 0 | SF | R/W | Success/fail bit<br>This is a bit that is returned to indicate whether or not initialization has been successfully terminated.<br>0: Initialization terminated successfully (without error).<br>1: Initialization terminated abnormally (and resulted in error). |

### 3.2.3 Performing Programming Operation

To program to the flash memory, it is necessary to pass the programming destination address on the user MAT to the downloaded programming program alongside the data to be programmed.

1. Set the start address of the programming destination on the user MAT to the general register ER1. This parameter is called FMPAR (flash multi-purpose address area parameter). Since programming data is always in 128-byte blocks, the programming start address boundary on the user MAT should always have lower 8 bits (A7 to A0) of either H'00 or H'80.
2. Ensure that programming data to be programmed to the user MAT is ready in a continuous area. The programming data must be in a continuous space accessible by the CPU's MOV.B instruction and outside the on-chip flash memory space. Even if the data you wish to program is less than 128 bytes long, prepare programming data a full 128 bytes by padding with dummy code (H'FF). Set the start address of the area storing the prepared programming data to general register ER0. This parameter is called FMPDR (flash multi-purpose data destination area parameter).

(1) Flash Multi-Purpose Address Area Parameter (FMPAR: CPU's general register ER1)

Sets the programming destination start address on the user MAT.

When the address of any area outside the flash memory space is set, an error will occur.

In addition, the programming destination start address must be within a 128-byte boundary. Any address outside this boundary also will result in an error, which will be reflected in the WA bit of the FPFR parameter.

**Table 16 Flash Multi-Purpose Address Area Parameter**

| Bit | Bit Name | R/W | Description |
|---|---|---|---|
| 31 to 0 | MOA31 to MOA0 | R/W | Stores the programming destination start address on the user MAT. From the start address on the user MAT specified in this register, 128 bytes of continuous data will be written.<br>Thus, the programming destination start address should be on a 128-byte boundary, meaning that bits MOA6 to MOA0 are always 0. |

(2) Flash Multi-Purpose Data Destination Parameter (FMPDR: CPU's general register ER0)

Sets the start address of the area storing data to be written to the user MAT. If the area storing programming data is in the flash memory, an error will occur. This error will be reflected in the WD bit of the FPFR parameter.

**Table 17 Flash Multi-Purpose Data Destination Parameter**

| Bit | Bit Name | R/W | Description |
|---|---|---|---|
| 31 to 0 | MOA31 to MOA0 | R/W | Stores the start address of the area that is storing data to be programmed to the user MAT. A continuous 128 bytes of data from the start address specified here onwards will be programmed to the user MAT. |

(3) Flash Pass/Fail Parameter (FPFR: CPU's general register R0L)

This is a value returned as the result of programming processing.

**Table 18 Flash Pass/Fail Parameter**

| Bit | Bit Name | R/W | Description |
|---|---|---|---|
| 7 | — | — | Reserved bit<br>Value 0 is returned. |
| 6 | MD | R/W | Programming-mode-related setting error detection bit<br>Returns the result of a check run to determine whether error protection is enabled.<br>0: FLER state is normal (FLER = 0).<br>1: FLER = 1 and programming cannot be performed. |
| 5 | EE | R/W | Programming operation error detection bit<br>If the specified data cannot be programmed due to failure to erase the user MAT or if a part of a flash-memory-related registers is rewritten when control returns from user-branch processing, 1 is set to this bit. In addition, when the FMATS register value is H'AA and if writing is attempted with the user boot MAT selected, a programming operation error will result. In this case, neither the user MAT nor the user boot MAT has been programmed. In order to write to the user boot MAT, program in the boot or programmer mode.<br>0: Programming operation completed successfully.<br>1: Programming operation terminated abnormally. The result of the programming is not guaranteed. |
| 4 | FK | R/W | Flash key register error detection bit<br>Returns the result of checking the FKEY register value prior to a programming operation.<br>0: FKEY register is set correctly (FKEY = H'5A)<br>1: FKEY register setting indicates an error (the FKEY value is other than H'5A.) |
| 3 | — | — | Reserved bit<br>Value 0 is returned. |
| 2 | WD | R/W | Programming data address detection bit<br>If the specified start address of the data to be written is in either of the following areas, an error will occur.<br>• Address in an area on the on-chip RAM where a downloaded programming/erase program resides.<br>• Address in the flash memory area<br>0: Programming data address is set to a normal value.<br>1: Programming data address is set to an abnormal value. |
| 1 | WA | R/W | Programming address error detection bit<br>If an address that meets either of the following conditions is specified as the programming destination start address, an error will occur.<br>• The destination address is outside the flash memory area.<br>• The specified address is not on a 128-byte boundary. (Not all of A6 to A0 are 0.)<br>0: Setting of the programming destination address is a normal value.<br>1: Setting of the programming destination address is an abnormal value. |
| 0 | SF | R/W | Success/fail bit<br>This bit indicates whether the programming process has completed successfully.<br>0: Programming completed successfully (without error).<br>1: Programming terminated abnormally (an error has occurred). |

### 3.2.4 Erase Operation

During flash memory erase operations, the number of the block to be erased on the user MAT must be transmitted to the downloaded erase program. Set this information to the FEBS parameter (general register ER0).

Specify one block from block numbers 0 to 15.

(1) Flash Erase Block Select Parameter (FEBS: CPU's general register ER0)

Specifies the number of the block to erase. You cannot specify two or more block numbers.

**Table 19 Flash Erase Block Select Parameter**

| Bit | Bit Name | R/W | Description |
|---|---|---|---|
| 31 to 8 | — | — | Reserve bits<br>Set value 0s. |
| 7 to 0 | EB7 to EB0 | R/W | Erase block<br>Sets the block number of a block to be erased within the range of 0 to 15.<br>0 represents block EB0, while 15 represents block EB15. Any setting other than 0 to 15 results in an error. |

(2) Flash Pass/Fail Parameter (FPFR: CPU's register R0L)

This is a value returned as the result of erase processing.

**Table 20 Flash Erase Block Select Parameter**

| Bit | Bit Name | R/W | Description |
|---|---|---|---|
| 7 | — | — | Reserved bit<br>Value 0 is returned. |
| 6 | MD | R/W | Erase mode-related setting error detection bit<br>Returns the result of a check performed to ensure error protection is not enabled.<br>0: FLER is in normal state. (FLER = 0)<br>1: FLER = 1 and erasure cannot be performed. |
| 5 | EE | R/W | Erase operation time error detection bit<br>If erasure of the user MAT failed or if part of the flash-related register values have been changed when control returned from user branch processing, 1 is returned to this bit. Also, when erase operation is performed with the FMATS register value set to H'AA and with the user boot MAT selected, an erase operation time error occurs. In this case, neither the user MAT nor the user boot MAT has been erased. To erase the user boot MAT, erase in the boot or programmer mode.<br>0: Erase operation completed successfully.<br>1: Erase operation terminated abnormally, and the erase result is not guaranteed. |
| 4 | FK | R/W | Flash key register error detection bit<br>Returns the result of checking the FKEY register value before the start of the erase operation.<br>0: FKEY register setting is normal. (FKEY = H'5A)<br>1: Error in FKEY register setting. (FKEY value other than H'5A) |

**Table 20 Flash Erase Block Select Parameter (cont)**

| Bit | Bit Name | R/W | Description |
|---|---|---|---|
| 3 | EB | R/W | Erase block select error detection bit<br>This is the result after checking whether the specified erase block number is within the range of user MAT block numbers.<br>0: Erase block number setting is normal.<br>1: Erase block number setting is abnormal. |
| 2 to 1 | — | — | Reserved bits<br>Value 0s are returned. |
| 0 | SF | R/W | Success/fail bit<br>This bit indicates whether the erase operation has completed successfully.<br>0: Erasure completed successfully. (Without error)<br>1: Erasure terminated abnormally. (An error occurred.) |

# 4. Flowchart

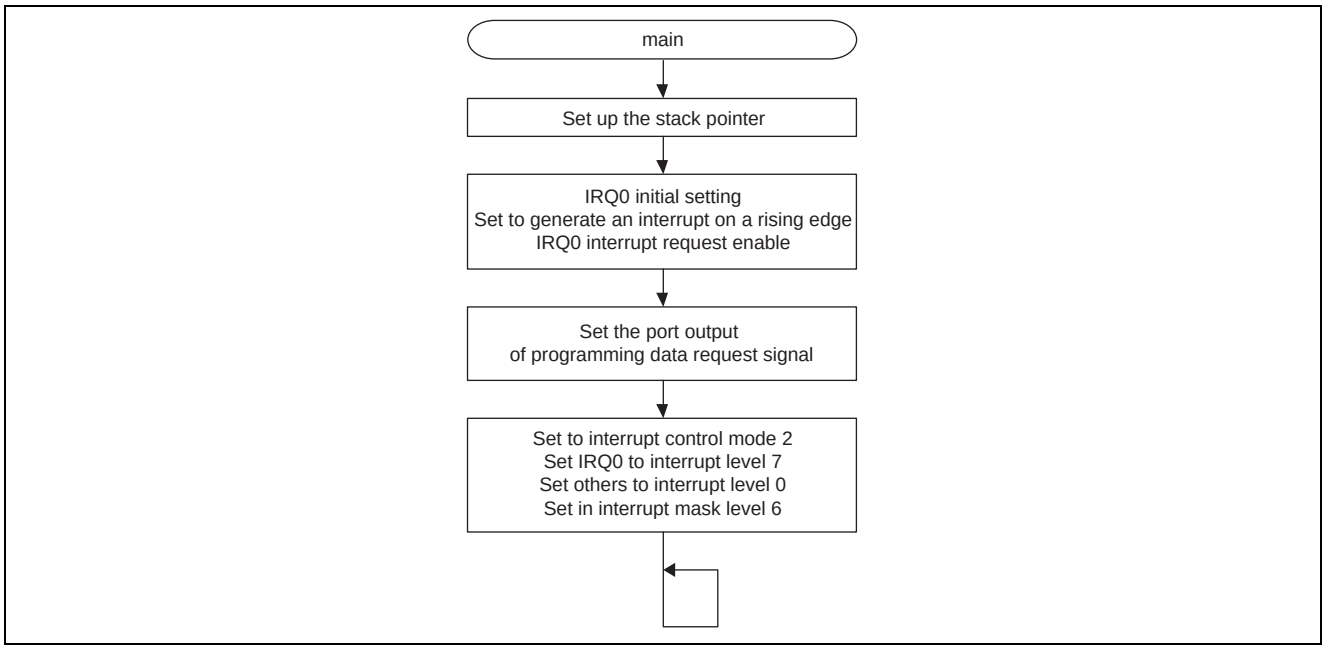
## 4.1 Main Processing (User Application)

Following activation, the user application sets up the stack pointer, initializes the IRQ0 interrupt and programming request signal output port, sets interrupt-control mode and interrupt-level settings, and then make transition to the IRQ0 interrupt wait state.

The processing described earlier is implemented by the user application.

**Table 21 Main Processing**

| | |
|---|---|
| Specification | void main(void) |
| Returned value | None |
| Argument | None |
| Function call | None |
| Section | MAIN |

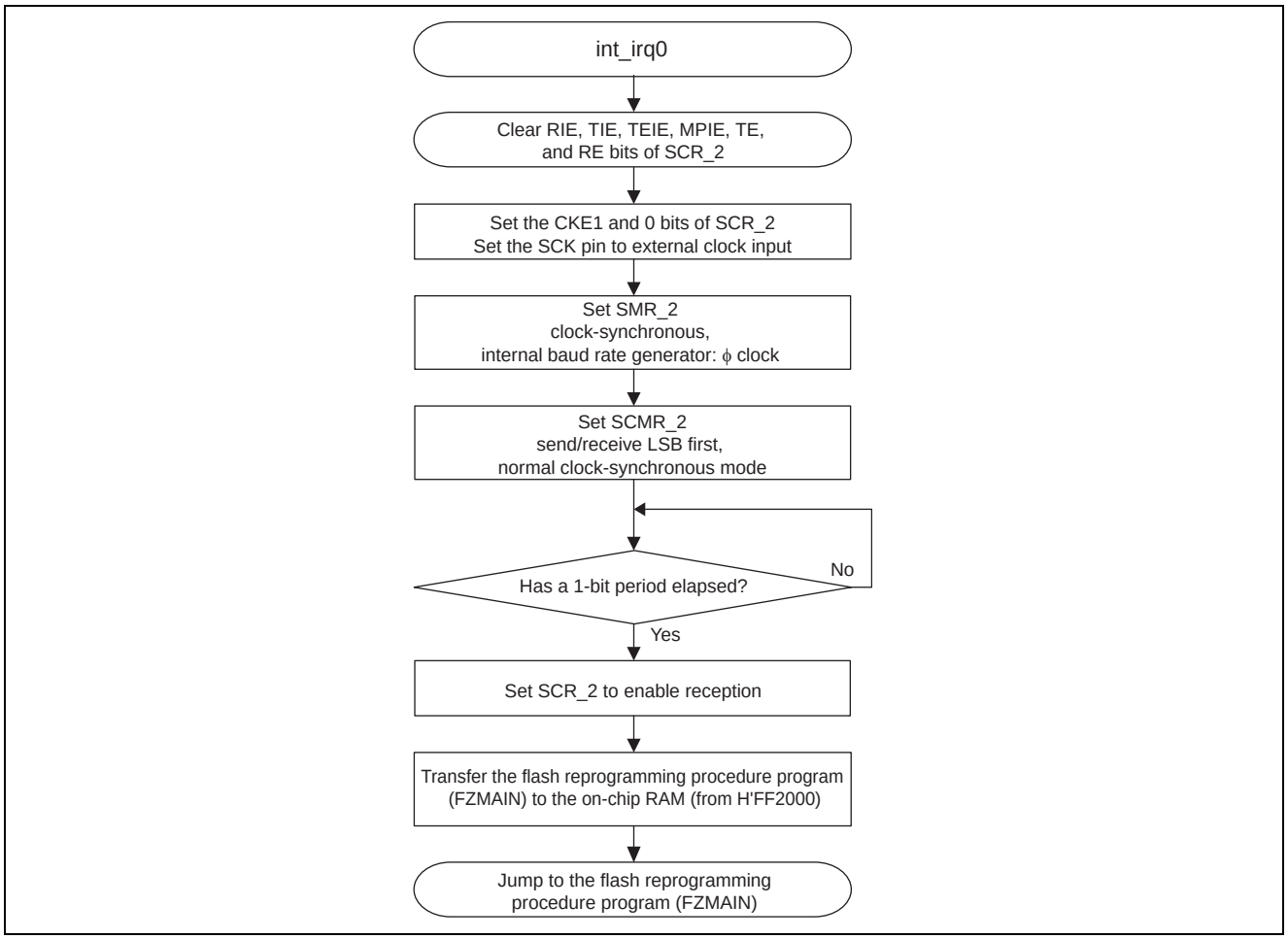


## 4.2 IRQ0 Interrupt Handling

Upon receipt of a programming start signal from the programming tool, the SCI2 initial setting is performed. After the flash programming/erase procedure program is transferred to the on-chip RAM, control jumps to the flash programming/erase procedure program.

**Table 22 IRQ0 Interrupt Handling**

| | |
|---|---|
| Function name | void int_irq0 (void) |
| Return value | None |
| Argument | None |
| Function calls | None |
| Section | INT_IRQ0 |

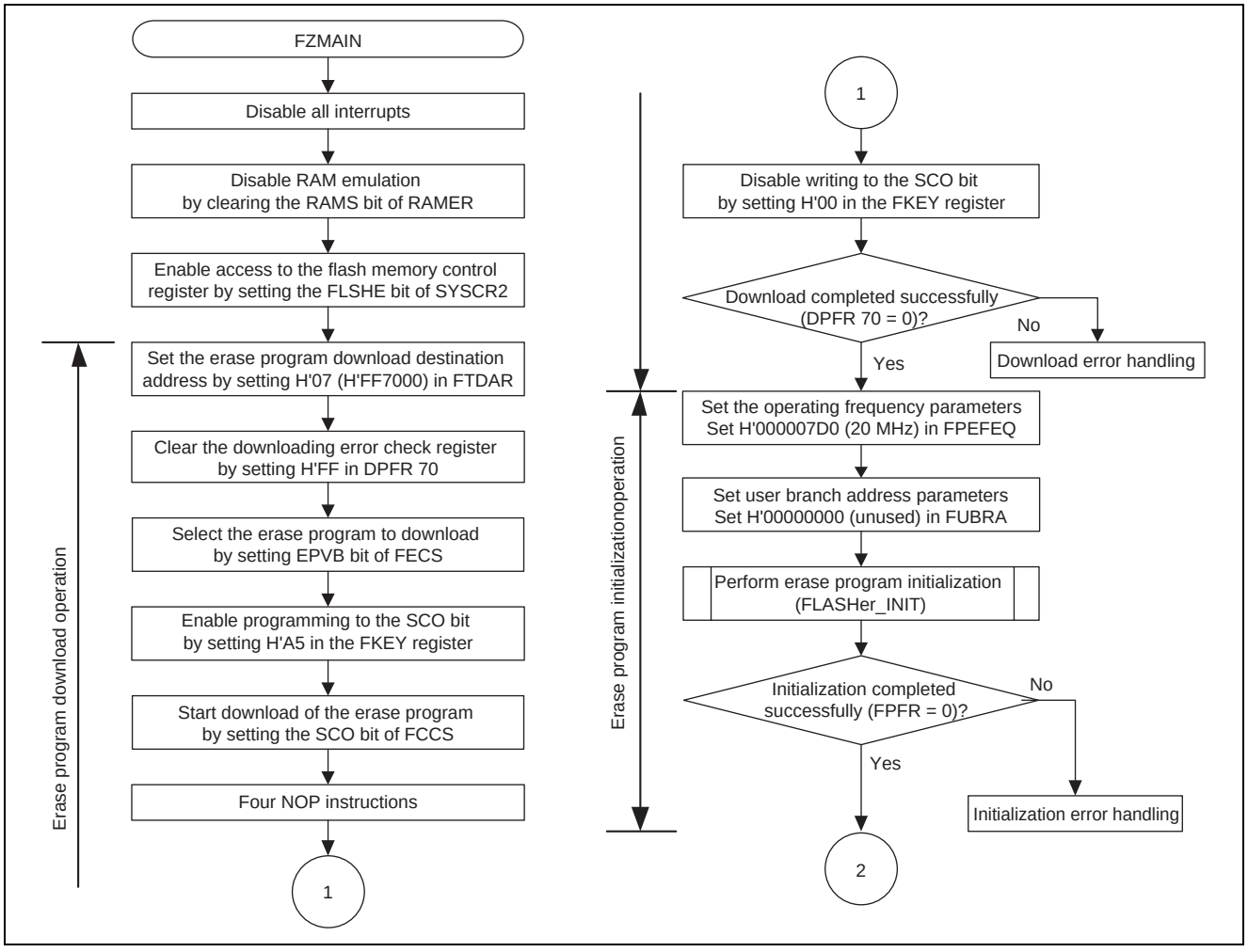


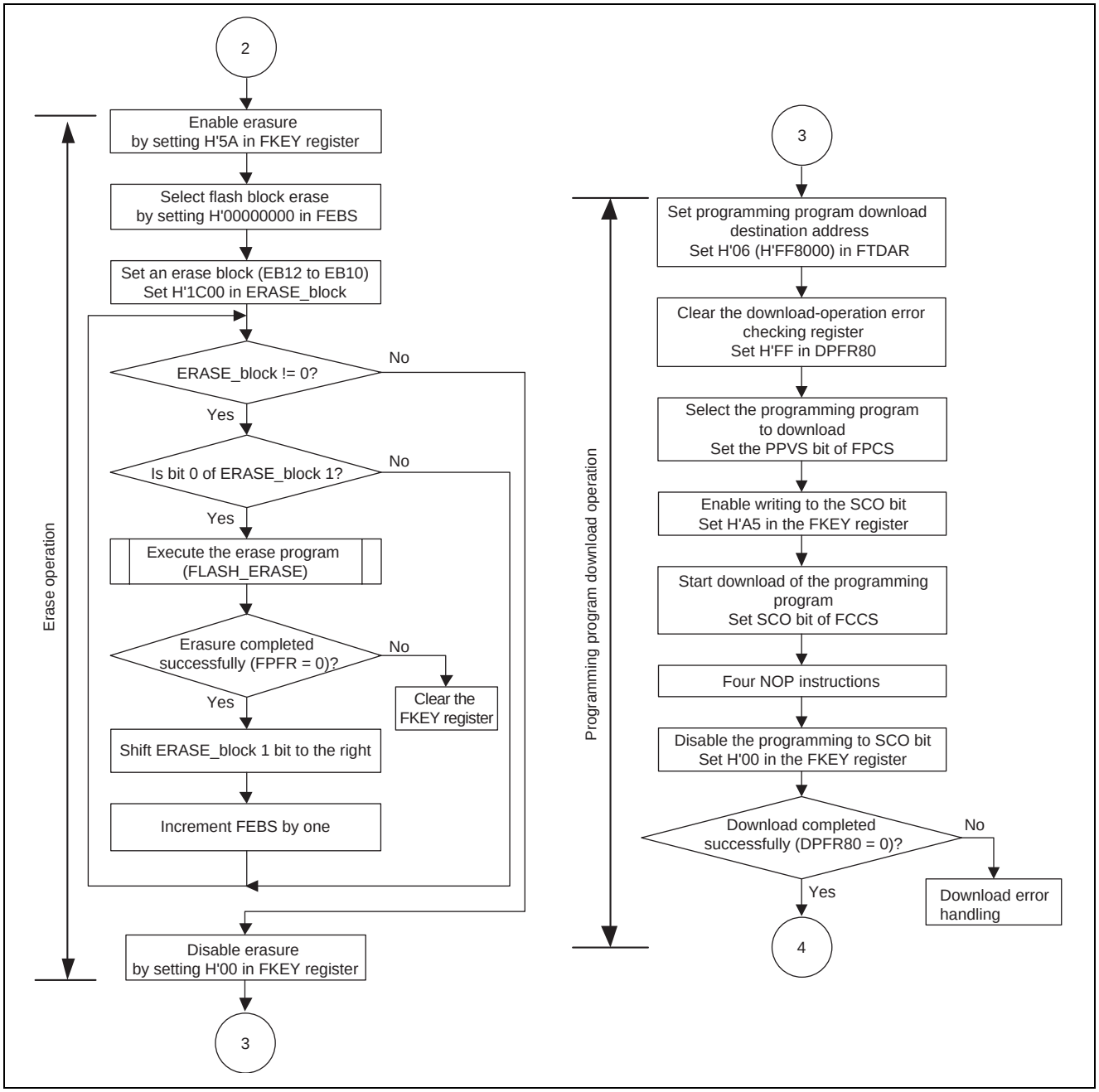
## 4.3 Flash Programming/Erase Procedure Program

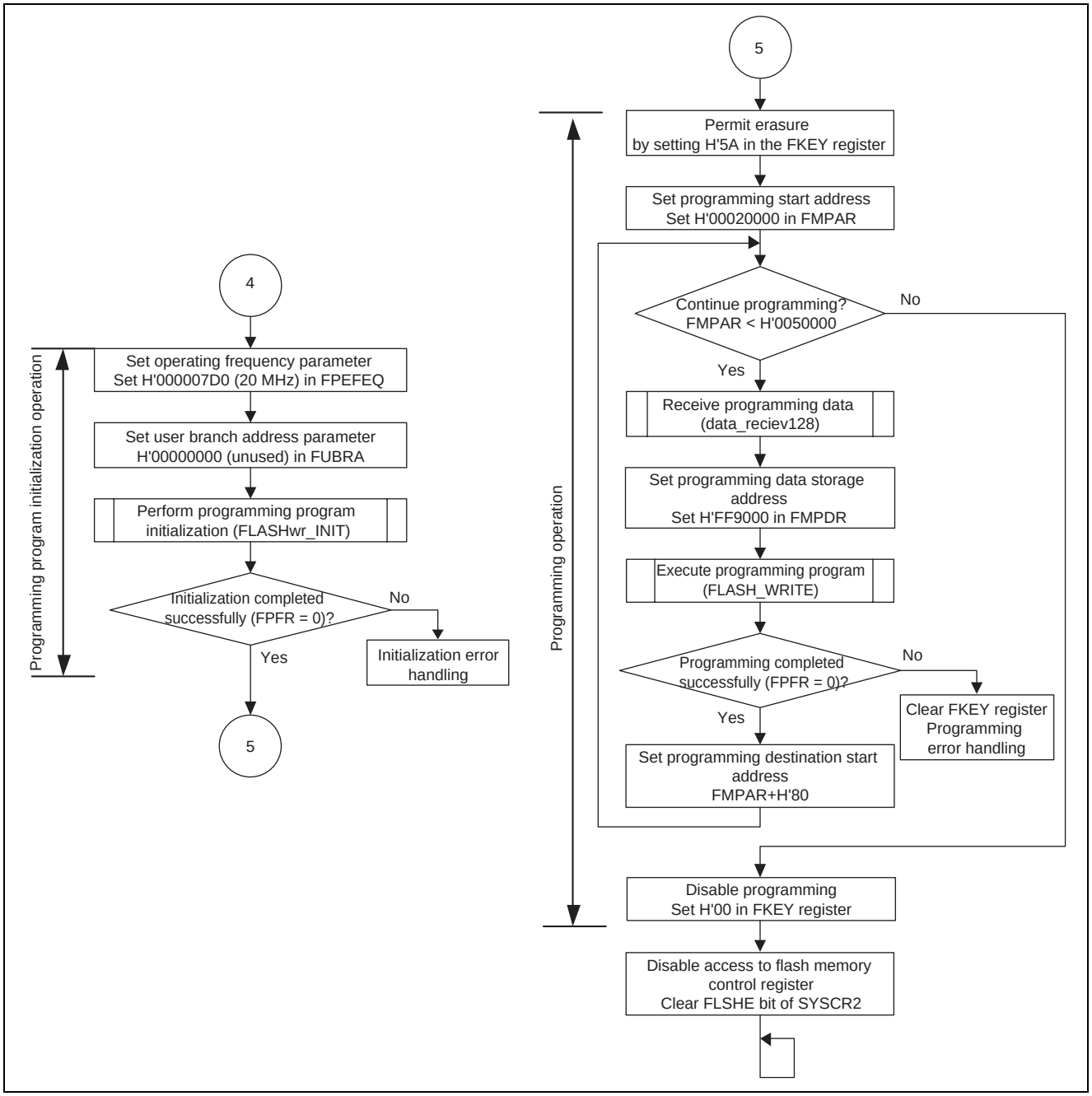
The flash programming/erase procedure program executed on the on-chip RAM, requests a download of the programming/erase program, performs a programming/erase procedure, makes a pass/fail judgment on the result, sends a command, and receives the data to be programmed.

**Table 23 Flash Programming/Erase Procedure Program**

| | |
|---|---|
| Function name | void FZMAIN(void) |
| Returned value | None |
| Argument | None |
| Function call | unsigned char FLASHer_INIT(unsigned long FPEFEQ, unsigned long FUBRA)<br>unsigned char FLASH_ERASE(unsigned long FEBS)<br>unsigned char FLASHwr_INIT(unsigned long FPEFEQ, unsigned long FUBRA)<br>unsigned char FLASH_WRITE(unsigned long FMPDR, unsigned long FMPAR)<br>void data_reciev128(void) |
| Section | Flash memory storage area: FWRMAIN<br>On-chip RAM execution area: RWRMAIN (H'FF7800 and up) |




2
Enable erasure
by setting H'5A in FKEY register
Select flash block erase
by setting H'00000000 in FEBS
Set an erase block (EB12 to EB10)
Set H'1C00 in ERASE_block
ERASE_block != 0?
No
Yes
Is bit 0 of ERASE_block 1?
No
Yes
Execute the erase program
(FLASH_ERASE)
Erasure completed
successfully (FPFR = 0)?
No
Yes
Clear the
FKEY register
Shift ERASE_block 1 bit to the right
Increment FEBS by one
Disable erasure
by setting H'00 in FKEY register
3
Erase operation
3
Set programming program download
destination address
Set H'06 (H'FF8000) in FTDAR
Clear the download-operation error
checking register
Set H'FF in DPFR80
Select the programming program
to download
Set the PPVS bit of FPCS
Enable writing to the SCO bit
Set H'A5 in the FKEY register
Start download of the programming
program
Set SCO bit of FCCS
Four NOP instructions
Disable the programming to SCO bit
Set H'00 in the FKEY register
Download completed
successfully (DPFR80 = 0)?
No
Yes
Download error
handling
4
Programming program download operation


4
Set operating frequency parameter
Set H'000007D0 (20 MHz) in FPEFEQ
Set user branch address parameter
H'00000000 (unused) in FUBRA
Perform programming program
initialization (FLASHwr_INIT)
Initialization completed
successfully (FPFR = 0)?
No
Yes
Initialization error
handling
5
Programming program initialization operation
5
Permit erasure
by setting H'5A in the FKEY register
Set programming start address
Set H'00020000 in FMPAR
Continue programming?
FMPAR < H'0050000
No
Yes
Receive programming data
(data_reciev128)
Set programming data storage
address
Set H'FF9000 in FMPDR
Execute programming program
(FLASH_WRITE)
Programming completed
successfully (FPFR = 0)?
No
Yes
Clear FKEY register
Programming
error handling
Set programming destination start
address
FMPAR+H'80
Disable programming
Set H'00 in FKEY register
Disable access to flash memory
control register
Clear FLSHE bit of SYSCR2
Programming operation

## 4.4 Flash Memory Programming Initialization Operation (Dummy)

During the flash memory programming initialization operation, an internal program is downloaded to the on-chip RAM via the flash programming/erase procedure program, and then executed within the on-chip RAM.

In this sample task, the function name alone is declared as a dummy beforehand and its execution address is specified so that the flash memory programming initialization process function can be called using its function name (FLASHwr_INIT) when the program is downloaded to the specified address using the procedure program.

Because of a dummy declaration, only a 2-byte RTS instruction is stored on the flash memory. Note that no transfer is performed from the flash memory to the on-chip RAM.

**Table 24 Flash Memory Programming Initialization Operation (Dummy)**

| | |
|---|---|
| Specification | unsigned char FLASHwr_INIT(unsigned long FPEFEQ, unsigned long FUBRA) |
| Returned value | Flash pass/fail parameter: FPFR(R0L)<br>This is a value returned as the result of the programming initialization. In the case of a successful completion, value H'00 is returned. |
| Argument | Flash program erase frequency control: FPEFEQ(ER0)<br>Sets the CPU's operating frequency. The operating frequency should be rounded off in MHz at the third decimal place to two decimal places. Multiply the obtained value by 100, and then convert the result into hexadecimal notation. Then, set the value thus obtained.<br>In this application example, the operating frequency is 20 MHz. So set 20000=H'07D0.<br>Flash user branch address set parameter: FUBRA(ER1)<br>This is a parameter for setting the user branch destination address.<br>Since this parameter value is not necessary in this application example, set address 0 (H'00000000). |
| Function call | — |
| Section | Flash storage area: FWRINI<br>On-chip RAM execution area: RWRINI (from H'FF7020) |

## 4.5 Flash Memory Programming Operation (Dummy)

During flash memory programming operation, an internal program is downloaded to the on-chip RAM by the flash programming/erase procedure program, and then executed on the on-chip RAM.

In this sample task, the function name alone is declared as a dummy beforehand and its execution address is specified so that calling of the flash memory write process function by its function name (FLASH_WRITE) is enabled when the internal program is downloaded to the specified address by the procedure program

Because of a dummy declaration, only a 2-byte RTS instruction is stored on the flash memory (boot MAT). Note that no transfer is performed from the flash memory to the on-chip RAM.

**Table 25 Flash Memory Programming Operation (Dummy)**

| | |
|---|---|
| Specification | unsigned char FLASH_WRITE(unsigned long FMPDR, unsigned long FMPAR) |
| Returned value | Flash pass/fail parameter: FPFR(R0L)<br>This is a value returned as the result of the programming operation. When the programming operation is successfully completed, the value H'00 is returned. |
| Argument | Flash multi-purpose data destination parameter: FMPDR(ER0)<br>Sets the start address of the area storing data to be programmed to the flash memory. In this sample task, since the area is the 128 bytes of space starting from H'FF9000 in the on-chip RAM, H'FF9000 is set.<br>Flash multi-purpose address area parameter: FMPAR (ER1)<br>Sets the start address of the programming destination on the flash memory. The set address must be within a 128-byte boundary. |
| Function call | — |
| Section | Flash storage area: FWRIT<br>On-chip RAM execution area: RWRIT (from H'FF7010) |

## 4.6 Flash Memory Erase Initialization Operation (Dummy)

For the flash memory erase initialization operation, an internal program is downloaded to the on-chip RAM by the flash programming/erase procedure program, and then executed on the on-chip RAM.

In this sample task, the function name alone is declared as a dummy beforehand and its execution address is specified and calling of the flash memory write process function by its function name (FLASHer_INIT) is enabled when the program is downloaded to the specified address by the procedure program

Because of a dummy declaration, only a 2-byte RTS instruction is stored on the flash memory. Note that no transfer is performed from the flash memory to the on-chip RAM.

**Table 26 Flash Memory Erase Initialization Operation (Dummy)**

| | |
|---|---|
| Specification | unsigned char FLASHer_INIT(unsigned long FPEFEQ, unsigned long FUBRA) |
| Returned value | Flash pass/fail parameter: FPFR(R0L)<br>This is a value returned as the result of erase initialization. When erase initialization is successfully completed, the value H'00 is returned. |
| Argument | Flash program erase frequency control: FPEFEQ(ER0)<br>Sets the CPU's operating frequency. Round off the operating frequency value in MHz at the third decimal place to two decimal places. Multiply the obtained value by 100, and then convert the result into hexadecimal notation. The thus obtained value is set.<br>In this application example, the operating frequency is 20 MHz. So set 20000=H'07D0.<br>Flash user branch address set parameter: FUBRA(ER1)<br>This is a parameter for setting the user branch destination address.<br>Since this parameter value is not required in this sample task, set address 0 (H'00000000). |
| Function call | — |
| Section | Flash storage area: FERINI<br>On-chip RAM execution area: RERINI (from H'FF7020) |

## 4.7 Flash Memory Erase Operation (Dummy)

During the flash memory erase operation, an internal program is downloaded to the on-chip RAM by the flash programming/erase procedure program, and then executed on the on-chip RAM.

In this sample task, the function name alone is declared as a dummy beforehand and its execution address is specified so that calling of the flash memory programming process function by its function name (FLASH_ERASE) is enabled when the program is downloaded to the specified address by the procedure program.

Because of a dummy declaration, only a 2-byte RTS instruction is stored on the flash memory. Note that no transfer is performed from the flash memory to the on-chip RAM.

**Table 27 Flash Memory Erase Operation (Dummy)**

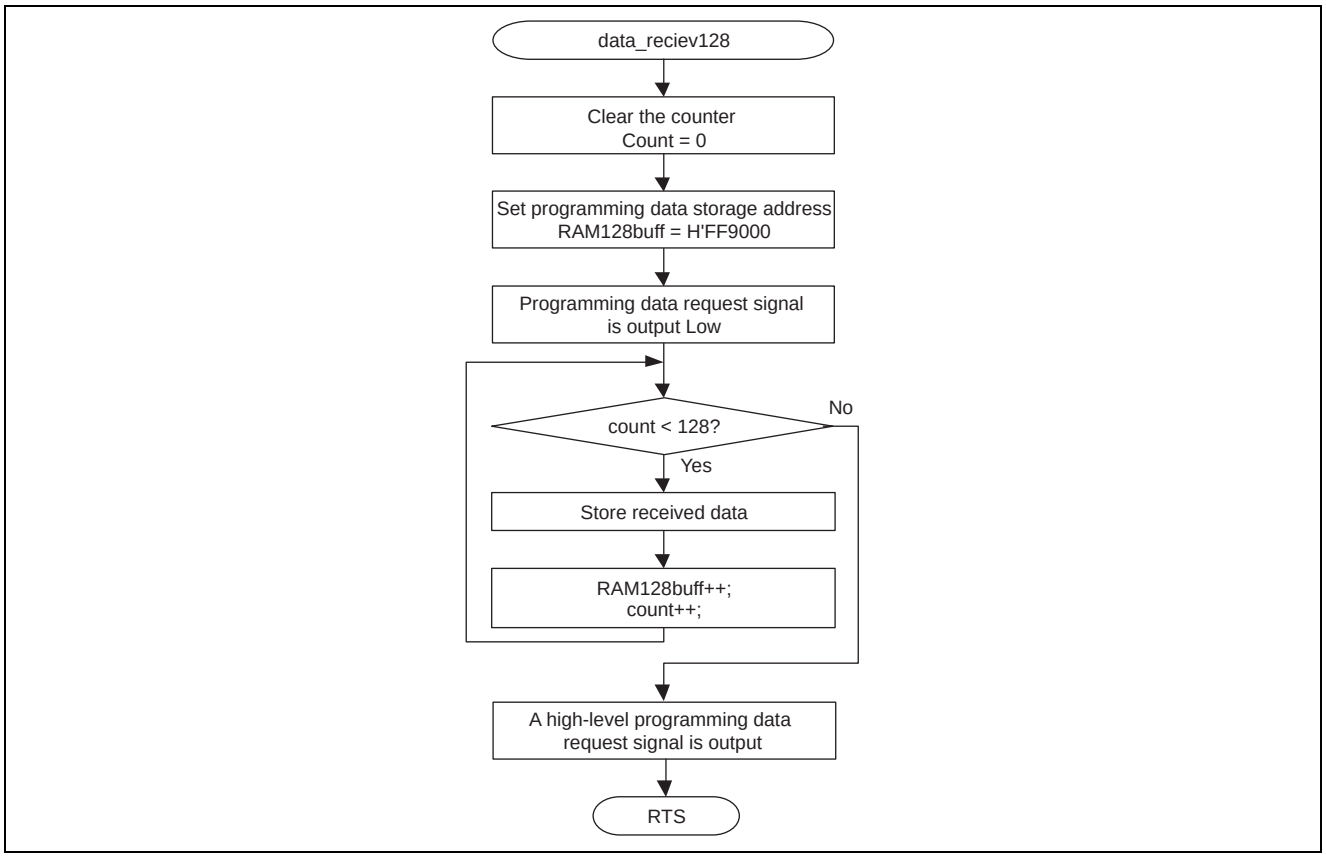
| | |
|---|---|
| Function name | unsigned char FLASH_ERASE(unsigned long FEBS) |
| Returned value | Flash pass/fail parameter: FPFR(R0L)<br>This is a value returned as the result of the erase operation. When the erase is successfully completed, the value H'00 is returned. |
| Argument | Flash erase block select parameter: FEBS(ER0)<br>Sets the block number of the block to be erased. It is not possible to specify two or more block numbers.<br>In this sample task, EB12 to EB10 are erased. Set H'0000000C for EB12, H'0000000B for EB11, and H'0000000A for EB10 respectively. |
| Function call | None |
| Section | Flash storage area: FERAS<br>On-chip RAM execution area: RERAS (H'FF7010 and up) |

## 4.8 Programming Data Receive Operation

Before the flash memory programming operation, the programming data request command is sent to the programming tool, and then 128 bytes of data to be written is received and stored on the fly into the on-chip RAM.

**Table 28 Programming Data Receive Operation**

| | |
|---|---|
| Function name | void data_reciev128(void) |
| Returned value | None |
| Argument | None |
| Function call | None |
| Section | Placed in the same section (FWRMAIN) as the flash programming/erase procedure program and after the FZMAIN. Also the same with the execution area. |



## 5. Application Memory Map

The table below shows a memory map of this sample task.

**Table 29 Application Memory Map**

- Flash Memory (User MAT)

| | Section | Description |
|---|---|---|
| From H'000000 | VECT | Vector table |
| From H'000000 | MAIN | See section 4.1 |
| From H'004000 | INT_IRQ0 | See section 4.2 |
| H'020000 to H'04FFFF | DATA | Data area (EB10 to EB12) |
| From H'05FFFF | FWRMAIN | See sections 4.3 and 4.8 |
| | FWRINI | See section 4.4 |
| | FWRIT | See section 4.5 |
| | FERINI | See section 4.6 |
| | FERAS | See section 4.7 |

- On-chip RAM

| | Section | Description |
|---|---|---|
| H'FF7000 to H'FF77FF | — | Erase/programming program download destination |
| H'FF7010 | RERAS / RWRIT | See sections 4.7 and 4.5 |
| H'FF7020 | RERINI /RWRINI | See sections 4.6 and 4.4 |
| From H'FF7800 | RWRMAIN | See section 4.3 |

# 6. Program Listings

## 6.1 Main Processing (User Application) Source Program

```
/*******************************/
/* main                        */
/*******************************/
#pragma entry main(sp=0xFFEFB0) /* Stack pointer setting                        */
void main(void){
    *MSTPCRA = 0x3f;
    *MSTPCRB = 0x5f;            /* SCI2 module stop-bit clear                   */
    *MSTPCRC = 0xff;
    *ISCRL = 0x02;              /* Interrupt occurs on the rising edge of IRQ0  */
    *IER = 0x01;                /* IRQ0 enable                                  */

    *PJDDR = 0xFF;              /* Initial setting of write request signal port */
    *PJDR = 0x01;               /* Write request signal is output High          */
    *SYSCR = 0x21;              /* Interrupt control mode 2                     */
    *IPRA = 0x70;               /* IRQ0 interrupt level 7                       */
    set_exr(0x06);              /* Interrupt mask level 6                       */
    while(1);
}
```

## 6.2 IRQ0 Interrupt Handling Source Program

The IRQ0 interrupt handling source is shown below along with the source to jump to the on-chip RAM used in the IRQ0 interrupt handling as well as the section labels.

```
/*******************************/
/*  IRQ0 Interrupt              */
/*******************************/
#pragma interrupt(int_irq0)
void int_irq0 (void){
    unsigned char *dst,*src;
    unsigned long  i=0;
    *SCR2  = 0x00;             /* Clearing of RIE, TIE, TEIE, MPIE, TE, and RE bits    */
    *SCR2  = 0x03;             /* External clock select                                */
    *SMR2  = 0x80;             /* Clock synchronous                                    */
    *SCMR2 = 0xF2;             /* Send/receive LSB first. Normal synchronous mode      */
    for( i=0 ; i<2100 ; i++ ); /* wait                                                 */
    *SCR2  = 0x13;             /* RE set                                               */
    for(src=_FWRMAIN_BGN, dst=_RWRMAIN_RAM; src<=_FWRMAIN_END; src++, dst++) {
        *dst = *src;
    }
    *ISR &= 0x00;
    jmp_RAM();
}
/*******************************/
/*  Transfer to on-chip RAM     */
/*******************************/
#pragma inline_asm(jmp_RAM)
void    jmp_RAM(void){
    MOV.L   #H'FFA000,ER0
    JMP @ER0
    NOP
}
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;   Section Label Table
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    .SECTION    FWRMAIN,    CODE,ALIGN=2
    .SECTION    RWRMAIN,    CODE,ALIGN=2
    .SECTION    C,DATA,ALIGN=2
    .EXPORT __FWRMAIN_BGN
    .EXPORT __FWRMAIN_END
    .EXPORT __RWRMAIN_RAM
__FWRMAIN_BGN  .data.l (STARTOF FWRMAIN)
__FWRMAIN_END  .data.l (STARTOF FWRMAIN) + (SIZEOF FWRMAIN)
__RWRMAIN_RAM  .data.l (STARTOF RWRMAIN)
    .END
```

## 6.3 Flash Programming/Erase Procedure Program Source Program

The program source is shown below along with the programming data receive process used during the flash programming/erase procedure.

```
#define DPFR70 (*(unsigned char  *)0xff7000) /* Download process error check register (DPFR)  */
#pragma inline_asm(NOP4)
static void NOP4(void){                      /* Four NOP instructions                         */
    NOP
    NOP
    NOP
    NOP
}
void FZMAIN(void){          /* for 0xFF2000                                        */
    unsigned long FPEFEQ;   /* ER0                                                 */
    unsigned long FUBRA;    /* ER1                                                 */
    unsigned char FPFR;     /* R0L                                                 */
    unsigned long FEBS;     /* ER0                                                 */
    unsigned long FMPAR;    /* ER1                                                 */
    unsigned long FMPDR;    /* ER0                                                 */
    *IER = 0x00;
    *IPRA = 0x00;          /* IRQ0 interrupt inhibit                               */
    set_exr(0x07);         /* Interrupt inhibit (interrupt mask level 7)           */
    *SCR2  = 0x30;         /* TE, RE set (RIE clear)                               */
    *RAMER=0;              /* Emulation deselect                                   */
    *SYSCR2=0x08;          /* Flash memory control register access enabled         */
/* Erase program download process   */
    *FTDAR = 0x07;         /* Download destination address setting (H'FF7000)      */
    DPFR70 = 0xFF;         /* Download process error check register (DPFR) clear   */
    *FECS = 0x01;          /* Erase program select (EPVB set)                      */
    *FKEY = 0xA5;          /* SCO bit write enabled                                */
    *FCCS |= 0x01;         /* Erase program download request (SCO set)             */
    NOP4();                /* NOP×4                                                */
    *FKEY = 0x00;          /* FKEY clear                                           */
    if(DPFR70 != 0x00){    /* Download error?                                      */
        goto end_p;
    }
/* Erase initialization process     */
    FPEFEQ = 0x000007D0;            /* Operating frequency parameter setting  20 MHz  H'07D0 =>
                                       D'2000                                                */
    FUBRA = 0x00000000;             /* User branch address parameter setting                 */
    FPFR = FLASH_INIT(FPEFEQ,FUBRA); /* FPFR--> R0L,FPEFEQ-->ER0,FUBRA-->ER1                 */
    if(FPFR != 0x00){               /* Initial setting error?                                */
        goto end_p;
    }
/* Erase process   */
    *FKEY = 0x5A;                    /* FKEY set                                          */
    FEBS=0x00000000;
    ERASE_block = 0x1C00;            /* EB12 to EB10 erase                                */
    while(ERASE_block != 0x0000){    /* Block erasing routine                             */
```

```
        if((ERASE_block & 0x0001)==0x0001){
            FPFR = FLASH_ERASE(FEBS);     /* FPFR-->R0L, FEBS-->ER0                          */
            if(FPFR != 0x00){             /* Erase error?                                    */
                goto end_p;
            }
        }
        ERASE_block = (ERASE_block>>1);   /* Erase block specification register shift        */
        FEBS++;                           /* Flash erase select parameter                    */
    }
    *FKEY = 0x00;                         /* FKEY clear                                      */
/* Write program download process   */
    *FTDAR = 0x07;          /* Download destination address setting (H'FF7000)     */
    DPFR70 = 0xFF;          /* Download process error check register (DPFR) clear  */
    *FPCS = 0x01;           /* Write program select (PPVS set)                     */
    *FKEY = 0xA5;           /* Write program download enable                       */
    *FCCS |= 0x01;          /* Download request (SCO set)                          */
    NOP4();                 /* NOP×4                                               */
    *FKEY = 0x00;           /* FKEY clear                                          */
    if(DPFR70 != 0x00){     /* Download error?                                     */
        goto end_p;
    }
/* Write initialization process                    */
    FPEFEQ = 0x000007D0;              /* Operating frequency parameter setting  20 MHz  H'07D0 =>
                                         D'2000                                          */
    FUBRA = 0x00000000;               /* User branch address parameter setting           */
    FPFR = FLASHwr_INIT(FPEFEQ,FUBRA);/* FPEFEQ-->ER0,FUBRA-->R5,ERR_CHECK-->R0          */
    if(FPFR != 0x00){                 /* Initial setting error?                          */
        goto end_p;
    }
/* Write process   */
    *FKEY = 0x5A;                         /* FKEY set                                    */
    FMPAR = 0x00020000;                   /* Write start address setting                 */
    while(FMPAR < 0x00050000){            /* Writing complete?                           */
        data_reciev128();                 /* Write data receive routine                  */
        FMPDR=0xFFFF9000;
        FPFR = FLASH_WRITE(FMPDR,FMPAR);  /* FMPDR-->R4,FMPAR-->R5,ERR_CHECK-->R0        */
        if(FPFR != 0x00){                 /* Write error?                                */
            goto end_p;
        }
        FMPAR+=0x80;
    }
    *FKEY = 0x00;                         /* FKEY clear                                  */
end_p:                                    /* Destination of jump at error occurrence     */
    *SYSCR2=0x00;
    while(1);
}

void data_reciev128(void){
    unsigned char count=0;                /* Receive data counter                        */
    unsigned char *RAM128buff;            /* Write data storage address in on-chip RAM   */
*PJDR &= 0xFE;                            /* Write data request signal output Low        */
RAM128buff=(unsigned char *)0xFFFF9000;
    while(count < 128){
```

```
    while((*SSR2 & 0x40)!=0x40);
        *SSR2 &= 0xbf;
        *RAM128buff = *RDR2;            /* Received data storage                        */
        RAM128buff++;
        count++;
    }
*PJDR |= 0x01;                          /* Write data request signal output High        */
}
```

## 6.4 Flash Memory Erase Initialization Process Dummy Source Program

See section 4.6.

```
unsigned char FLASHer_INIT(unsigned long FPEFEQ, unsigned long FUBRA){}
```

## 6.5 Flash Memory Erase Process Dummy Source Program

See section 4.7.

```
unsigned char FLASH_ERASE(unsigned long FEBS){}
```

## 6.6 Flash Memory Programming Initialization Process Dummy Source Program

See section 4.4.

```
unsigned char FLASHwr_INIT(unsigned long FPEFEQ,unsigned long FUBRA){}
```

## 6.7 Flash Memory Programming Process Dummy Source Program

See section 4.5.

```
unsigned char FLASH_WRITE(unsigned long FMPDR, unsigned long FMPAR){}
```

## 7. Appendix

I/O definition header file (ioaddrs.h)

```
/************************/
/*     System           */
/************************/
#define SYSCR2  (volatile char*)(0xfffde2)
#define SBYCR  (volatile char*)(0xfffde4)
#define SYSCR  (volatile char*)(0xfffde5)
#define SCKCR  (volatile char*)(0xfffde6)
#define MDCR   (volatile char*)(0xfffde7)
#define MSTPCRA  (volatile char*)(0xfffde8)
#define MSTPCRB  (volatile char*)(0xfffde9)
#define MSTPCRC  (volatile char*)(0xfffdea)
#define LPWRCR  (volatile char*)(0xfffdec)

/********************/
/*       INTC       */
/********************/
#define   ISCRH  (volatile char*)(0xfffe12)
#define   ISCRL  (volatile char*)(0xfffe13)
#define   IER(volatile char*)(0xfffe14)
#define   ISR(volatile char*)(0xfffe15)
#define   IPRA   (volatile char*)(0xfffec0)
#define   IPRB   (volatile char*)(0xfffec1)
#define   IPRC   (volatile char*)(0xfffec2)
#define   IPRD   (volatile char*)(0xfffec3)
#define   IPRE   (volatile char*)(0xfffec4)
#define   IPRF   (volatile char*)(0xfffec5)
#define   IPRG   (volatile char*)(0xfffec6)
#define   IPRH   (volatile char*)(0xfffec7)
#define   IPRI   (volatile char*)(0xfffec8)
#define   IPRJ   (volatile char*)(0xfffec9)
#define   IPRK   (volatile char*)(0xfffeca)
#define   IPRL   (volatile char*)(0xfffecb)
#define   IPRM   (volatile char*)(0xfffecc)
#define   IPRO   (volatile char*)(0xfffece)

/****************/
/* SCI2   */
/****************/
#define SMR2  (volatile char*)(0xffff88)
#define BRR2  (volatile char*)(0xffff89)
#define SCR2  (volatile char*)(0xffff8a)
#define TDR2  (volatile char*)(0xffff8b)
#define SSR2  (volatile char*)(0xffff8c)
#define RDR2  (volatile char*)(0xffff8d)
#define SCMR2 (volatile char*)(0xffff8e)
```

```
/******************/
/*    I/O Port    */
/******************/
#define P1DDR (volatile char*)(0xfffe30)
#define P2DDR (volatile char*)(0xfffe31)
#define P3DDR (volatile char*)(0xfffe32)
#define P5DDR (volatile char*)(0xfffe34)
#define P7DDR (volatile char*)(0xfffe36)

#define P3ODR (volatile char*)(0xfffe46)

#define P1DR  (volatile char*)(0xffff00)
#define P2DR  (volatile char*)(0xffff01)
#define P3DR  (volatile char*)(0xffff02)
#define P5DR  (volatile char*)(0xffff04)
#define P7DR  (volatile char*)(0xffff06)

#define PORT1 (volatile char*)(0xffffb0)
#define PORT2 (volatile char*)(0xffffb1)
#define PORT3 (volatile char*)(0xffffb2)
#define PORT4 (volatile char*)(0xffffb3)
#define PORT5 (volatile char*)(0xffffb4)
#define PORT7 (volatile char*)(0xffffb6)
#define PORT9 (volatile char*)(0xffffb8)

#define PADDR (volatile char*)(0xfffe39)
#define PADR  (volatile char*)(0xffff09)
#define PORTA (volatile char*)(0xffffb9)
#define PAPCR (volatile char*)(0xfffe40)
#define PAODR (volatile char*)(0xfffe47)

#define PBDDR (volatile char*)(0xfffe3a)
#define PBDR  (volatile char*)(0xffff0a)
#define PORTB (volatile char*)(0xffffba)
#define PBPCR (volatile char*)(0xfffe41)

#define PCDDR (volatile char*)(0xfffe3b)
#define PCDR  (volatile char*)(0xffff0b)
#define PORTC (volatile char*)(0xffffbb)
#define PCPCR (volatile char*)(0xfffe42)

#define PDDDR (volatile char*)(0xfffe3c)
#define PDDR  (volatile char*)(0xffff0c)
#define PORTD (volatile char*)(0xffffbc)
#define PDPCR (volatile char*)(0xfffe43)

#define PEDDR (volatile char*)(0xfffe3d)
#define PEDR  (volatile char*)(0xffff0d)
#define PORTE (volatile char*)(0xffffbd)
#define PEPCR (volatile char*)(0xfffe44)

#define PFDDR (volatile char*)(0xfffe3e)
#define PFDR  (volatile char*)(0xffff0e)
```

```
#define PORTF (volatile char*)(0xffffbe)

#define PGDDR (volatile char*)(0xfffe3f)
#define PGDR  (volatile char*)(0xffff0f)
#define PORTG (volatile char*)(0xffffbf)

#define PHDDR (volatile char*)(0xfffa10)
#define PJDDR (volatile char*)(0xfffa11)
#define PHDR (volatile char*)(0xfffa12)
#define PJDR (volatile char*)(0xfffa13)
#define PORTH (volatile char*)(0xfffa14)
#define PORTJ (volatile char*)(0xfffa15)

/******************/
/*    ROM         */
/******************/
#define RAMER   (volatile char*)(0xfffedb)
#define FCCS  (volatile char*)(0xffffa4)
#define FPCS  (volatile char*)(0xffffa5)
#define FECS  (volatile char*)(0xffffa6)
#define FKEY  (volatile char*)(0xffffa8)
#define FMATS  (volatile char*)(0xffffa9)
#define FTDAR    (volatile char*)(0xffffaa)
#define FVACR    (volatile char*)(0xffffab)
#define FVADRR  (volatile char*)(0xffffac)
#define FVADRE  (volatile char*)(0xffffad)
#define FVADRH  (volatile char*)(0xffffae)
#define FVADRL  (volatile char*)(0xffffaf)
```

## Revision Record

| | | Description | |
|---|---|---|---|
| **Rev.** | **Date** | **Page** | **Summary** |
| 1.00 | Mar.09.05 | — | First edition issued |
| | | | |
| | | | |
| | | | |
| | | | |

## Keep safety first in your circuit designs!

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

## Notes regarding these materials

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