H8S Family

Example of Reprogramming On- Chip Flash Memory in the User Boot Mode "SCI (Asynchronous 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 when operated in the user boot mode through asynchronous communication using a serial communications interface (hereafter called SCI).

Target Device

H8S/2500 Series H8S/2556 Group MCU

Contents

1. Specifications

In this sample task, the MCU is started up in the user boot mode. After three blocks from EB10 to EB12 of the flash memory have been erased, a signal requesting the transmission of data to be programmed is sent to a programming tool, and then data received in response is programmed to the three blocks from EB10 through EB12. As for interfacing, a serial communications interface (hereafter called SCI) and the asynchronous mode are used. To be more specific, the MCUuses the SCI2.

Figure 1 shows a block diagram of an on-board programming using an SCI (asynchronous mode).

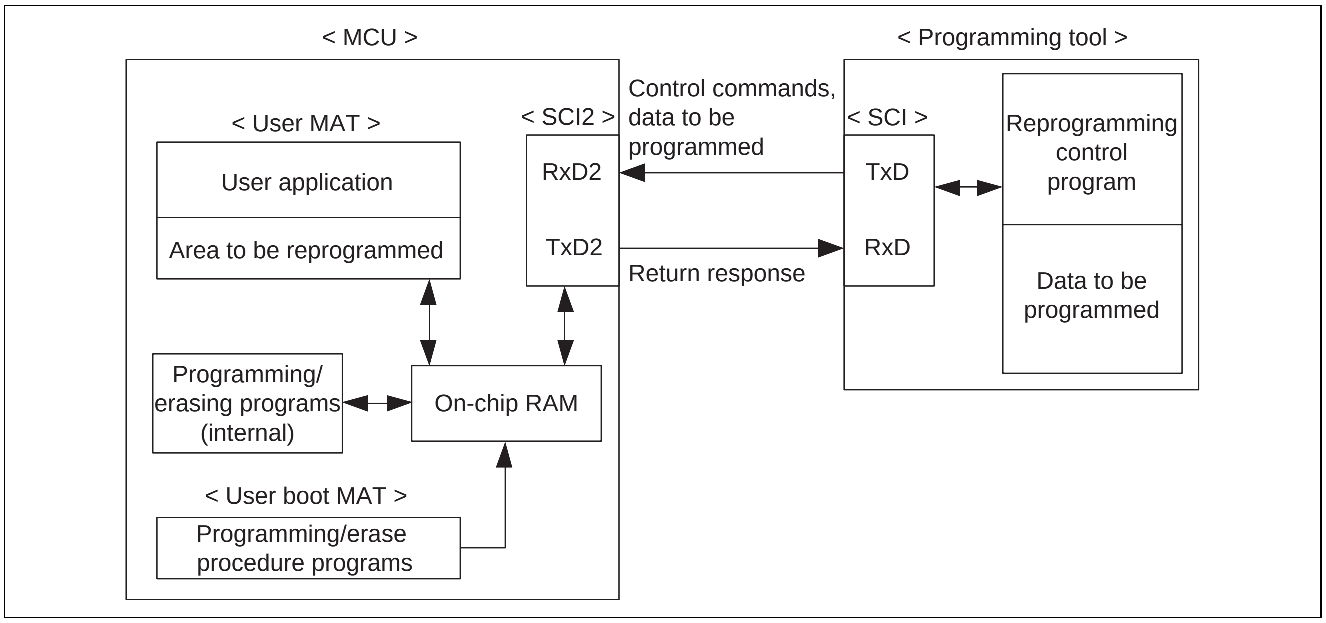


Figure 1 Block Diagram of On-Board Reprogramming using an SCI (asynchronous mode)

1.1 Operation Mode

User-boot-mode pin settings are shown as follows:

Table 1 Pin Settings

Pin	Level
RES	1
MD0	1
MD0	0
MD2	0

1.2 Software Development Environment

For software development of this sample task, High-performance Embedded Workshop 3 (HEW3) Ver. 3.01.06.001 is used. For the programming the software in the user boot mode, flash development tool kit (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)	—
Communication mode	Asynchronous	Asynchronous
Data length	8 bits	8 bits
Parity	None	None
Stop bit	1 stop bit	1 stop bit
Baud rate	38400 bps	38400 bps

1.4 Control Specifications

Specifications of control commands are as follows:

Table 3 Control Specifications

Command	Function
H'66	Flash erase complete command
H'77	Programming data request command
H'AA	Programming successfully completed command

1.5 Description of Control Command Behaviors

In this sample task, SCI2 is used for transmitting/receiving flash reprogramming control commands. (Interrupts are disabled during the flash-memory programming/erase operations.)

Figure 2 shows a series of control command behaviors that take place up until the reprogramming of the flash memory is complete.

Note that, in the case of this sample task, no error command is defined. The programming tool used in this sample task assumes that an error has occurred if no response is returned from the MCU following a predefined duration.

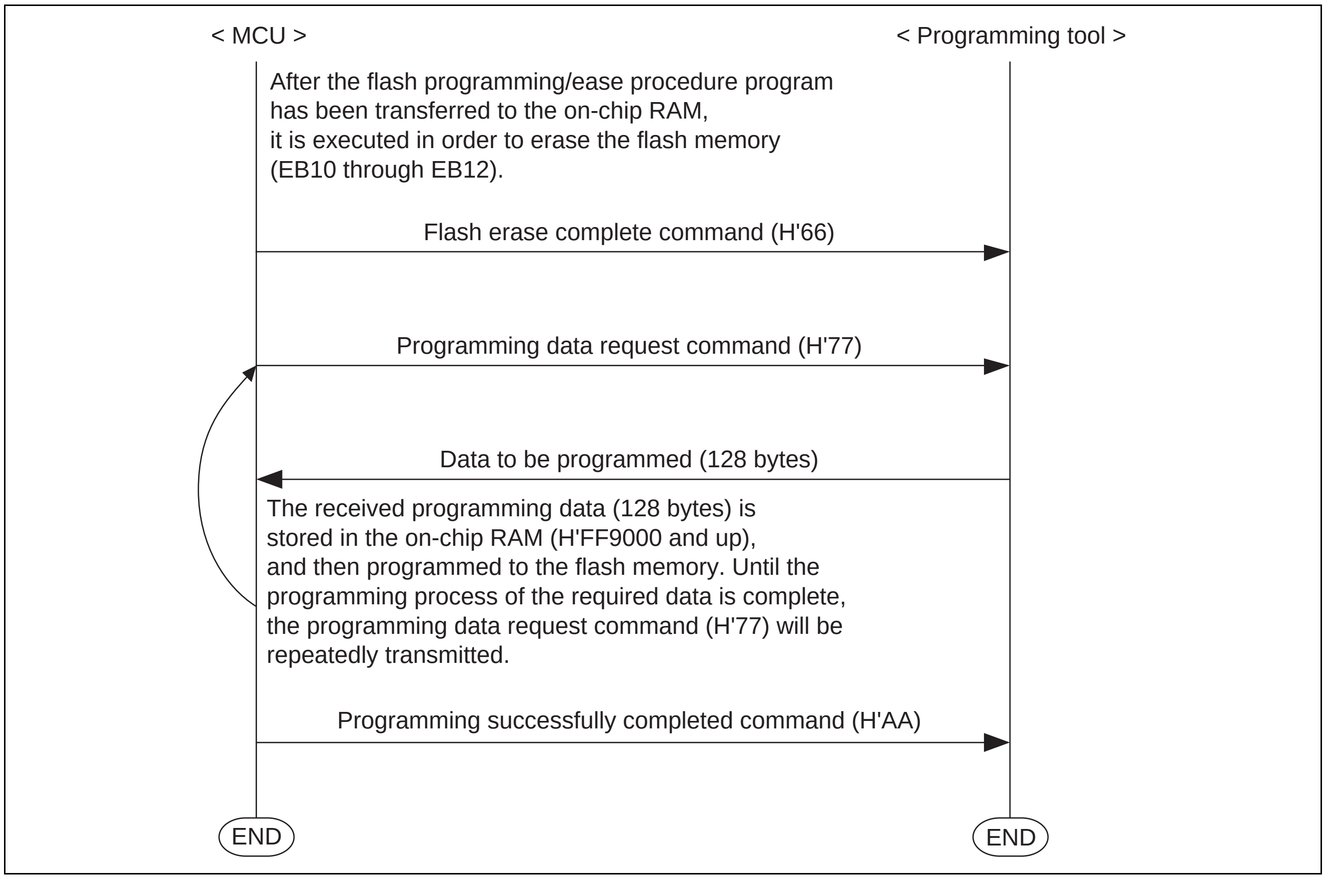
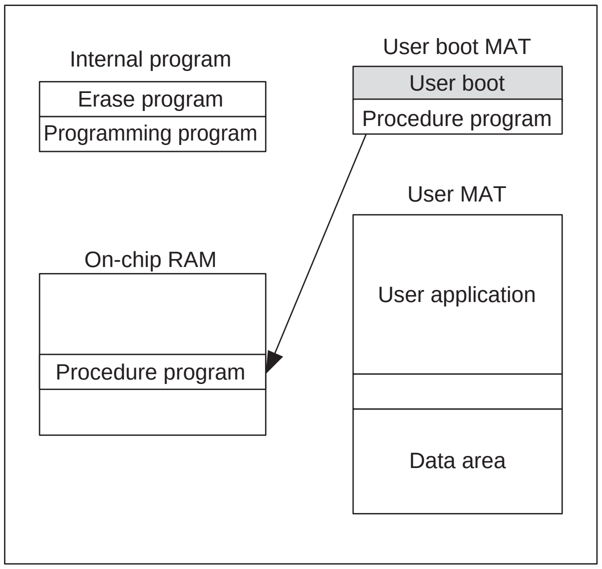

< MCU >
< Programming tool >
After the flash programming/ease procedure program has been transferred to the on-chip RAM, it is executed in order to erase the flash memory (EB10 through EB12).
Flash erase complete command (H'66)
Programming data request command (H'77)
Data to be programmed (128 bytes)
The received programming data (128 bytes) is stored in the on-chip RAM (H'FF9000 and up), and then programmed to the flash memory. Until the programming process of the required data is complete, the programming data request command (H'77) will be repeatedly transmitted.
Programming successfully completed command (H'AA)
END
END

Figure 2 Control Command Behaviors

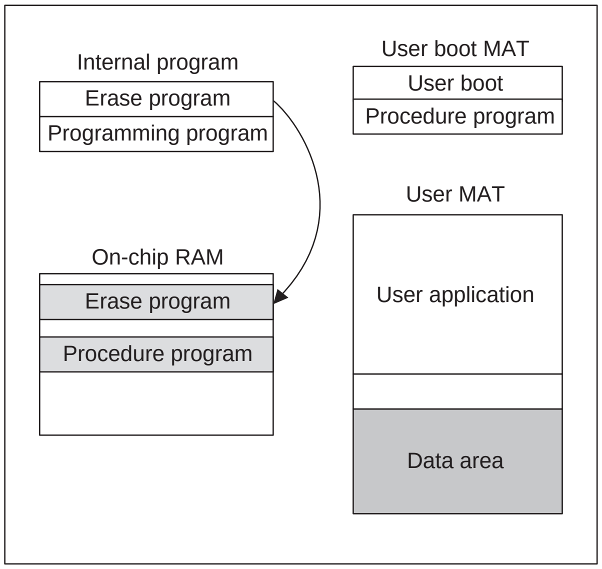
2. Description of Software Operation

The overview of the flash reprogramming operation, as performed in this sample task, is shown below:

(1) Following system startup, the user boot program transfers the flash memory erase and write procedure programs into the on-chip RAM, from where the procedure programs are then executed.



(2) The procedure program in the on-chip RAM then downloads the internal program (for erasing) into the on-chip RAM, which subsequently erases the flash memory in the specified block area.



(3) The procedure program in the on-chip RAM downloads the internal program (programming program) into the on-chip RAM, stores the programming data received from the programming tool into the on-chip RAM on a single occasion, and writes the data in 128-byte blocks from the specified start address.

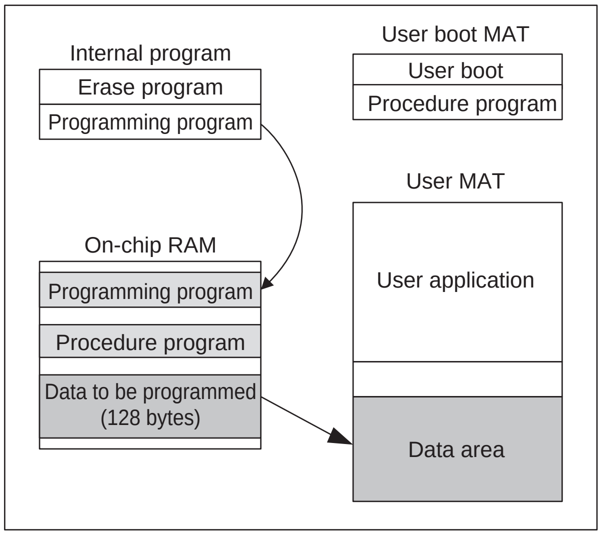


Figure 3 Description of Software Operation

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 This bit is always read as 1. Always set this bit to 1 when writing as well.
6, 5	—	R	Reserved bits These bits are always read as 0. Always set these bits to 0 when writing as well.
4	FLER	R	Flash memory error 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 to 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 µs, which is longer than usual. 0: Flash memory is operating normally. Programming /erase protection (error protection) on the flash memory is disabled. [Clearing condition] Cleared at power-on reset or transition into the hardware standby mode. 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 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 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. Immediately after writing 1 to this bit, always execute four NOP instructions. Note that, when downloading is completed, this bit is cleared to 0 and thus it is impossible to read 1 from this bit. 0: Downloading of the internal programming/erase program to the on-chip RAM is not performed. [Clearing condition] Cleared when a download completes. 1: Generates a request to download the internal programming /erase program into the on-chip RAM. [Setting condition] The bit is set when 1 is written with all of the following conditions being satisfied. (1) H'A5 has been written in the FKEY register. (2) The code is being executed in the on-chip RAM. (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 These bits are always read as 0. Always set these bits to 0 when writing as well.
0	PPVS	R/W	Program pulse verify Selects the write program. 0: Does not select the internal programming program. [Clearing condition] Cleared when a transfer completes. 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 These bits are always read as 0. Always set these bits to 0 when writing as well.
0	EPVS	R/W	Erase pulse verify block Selects the erase program. 0: Does not select the internal erase program. [Clear condition] Cleared when a transfer completes. 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 write/erase program, such processing cannot be performed.

Table 7 Flash Key Code Register (FKEY) Initial Value

Bit	Bit Name	R/W	Description
7 6 5 4 3 2 1 0	K7 K6 K5 K4 K3 K2 K1 K0	R/W R/W R/W R/W R/W R/W R/W R/W	Key code 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 write/erase program is executed with any value other than H'A5 written in the FKEY register, the flash memory cannot be programmed or erased. H'A5: Permits writing to the SCO bit. (Any value other than H'A5 would not allow setting of the SCO bit.) H'5A: Permits flash memory programming/erasing. (Any value other than H'5A would cause the software protection state to be retained.) H'00: Initial value

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

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 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. H'AA: Selects the user boot MAT. (A value other than H'AA specifies the user MAT.) This is the initial value when startup is done in the user boot mode. H'00: This is the initial value when startup is done in a mode other than user boot mode. (The user MAT is selected.) [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 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. 0: The value set to bits TDA6 to TDA0 is normal. 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 5 4 3 2 1 0	TDA6 TDA5 TDA4 TDA3 TDA2 TDA1 TDA0	R/W R/W R/W R/W R/W R/W R/W	Transfer destination address 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. H'00: Sets the download start address to H'FF9000. H'01: Sets the download start address to H'FFA000. H'02: Sets the download start address to H'FFB000. H'03: Sets the download start address to H'FFC000. H'04: Sets the download start address to H'FFD000. H'05: Sets the download start address to H'FFE000. H'06: Sets the download start address to H'FF8000. H'07: Sets the download start address to H'FF7000. 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.

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 Write 0s.
3	FLSHE	R/W	Flash memory control register enable 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. 0: Flash control logic unit which controls H'FFFFA4 to H'FFFFAF is disabled. 1: Flash control logic unit which controls H'FFFFA4 to H'FFFFAF is enabled.
2	—	—	Reserved bit Write 0.
1, 0	—	R/W	Reserved bits Write 0s.

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

3.2 Programming/Erasing Interface Parameters

Write/erase interface parameters are used for specifying operating frequency, user branch destination address, write 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.)

Write/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 EFFR 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), in order to ensure positive verification is possible.

Table 12 Download Pass/Fail Parameter

Bit	Bit Name	R/W	Description
7 to 3	—	—	Reserved bits Return 0.
2	SS	R/W	Source select error detection bit 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. 0: Program to be downloaded is correctly selected. 1: Download error. (Multiple programs selected or program selected without mapping.)
1	FK	R/W	Flash key register error detection bit This is a bit returning the result following checking of whether or not the FKEY register value is H'A5. 0: FKEY register setting is correct. (FKEY = H'A5) 1: FKEY register setting value error. (The FKEY value is not H'A5.)
0	SF	R/W	Success/fail bit 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. 0: Download of an internal program has been completed successfully (without error). 1: Download of an internal program has failed. (An error has occurred.)

3.2.2 Programming /Erase Initialization

An internal program to be downloaded also 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 These bits should be cleared to 0.
15 to 0	F15 to F0	R/W	Frequency setting bits Set a CPU operating frequency. Calculate the settable value as follows: • Round off the operating frequency in MHz to two decimal places. • Multiply the value by 100, convert the obtained value to binary form, and write it to the FPEFEQ parameter (general register ER0). As a concrete example, when the CPU's operating frequency is 25.000 MHz, calculations are as follows: Round off 25.000 at the third decimal place to obtain 25.00. Convert 25.00 $\times$ 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	R/W	User branch destination address When no user branching is required, set address 0 (H'00000000). 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. During user-branched processing, do not download, initialize, or invoke programming/erase program routines of the internal program. Programming/erasing subsequent to the user branch routine cannot be otherwise guaranteed. In addition, do not modify pre-prepared data to be written. Likewise, do not reprogramming 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 Value 0 is returned.
2	BR	R/W	User branch error detection bit 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. 0: User branch address setting is correct. 1: User branch address setting is incorrect.
1	FQ	R/W	Frequency error detection bit Returns the result of a check determining whether or not the specified CPU operating frequency is within the supported range. 0: Operating frequency setting is normal. 1: Operating frequency setting is abnormal.
0	SF	R/W	Success/fail bit This is a bit that is returned to indicate whether or not initialization has been successfully terminated. 0: Initialization terminated successfully (without error). 1: Initialization terminated abnormally (and resulted in error).

3.2.3 Programming Operation

To program 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 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 programming 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. 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	MOD31 to MOD0	R/W	Stores the start address of the area that is storing data to be written 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 Value 0 is returned.
6	MD	R/W	Programming-mode-related setting error detection bit Returns the result of a check run to determine whether error protection is enabled. 0: FLER state is normal (FLER = 0). 1: FLER = 1 and programming cannot be performed.
5	EE	R/W	Programming operation error detection bit If the specified data cannot be programmed due to failure to erase the user MAT or if a part of 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 programming is attempted with the user boot MAT selected, a writing operation error will result. In this case, neither the user MAT nor the user boot mat has been reprogrammed. In order to programming to the user boot mat, program in the boot or programmer mode. 0: Programming operation completed successfully. 1: Programming operation terminated abnormally. The result of the programming is not guaranteed.
4	FK	R/W	Flash key register error detection bit Returns the result of checking the FKEY register value prior to a programming operation. 0: FKEY register is set normally (FKEY=H'5A) 1: FKEY register setting indicates an error (the FKEY value is other than H'5A.)
3	—	—	Reserved bit Value 0 is returned.
2	WD	R/W	Write data address detection bit If the specified start address of the data to be written is in either of the following areas, an error will occur. • Address in an area on the on-chip RAM where a downloaded programming/erase program resides. • Address in the flash memory area 0: Programming data address is set to a normal value. 1: Programming data address is set to an abnormal value.

Table 18 Flash Pass/Fail Parameter (cont)

Bit	Bit Name	R/W	Description
1	WA	R/W	Write address error detection bit If an address that meets either of the following conditions is specified as the programming destination start address, an error will occur. • The destination address is outside the flash memory area. • The specified address is not on a 128-byte boundary. (Not all of A6 to A0 are 0.) 0: Setting of the programming destination address is a normal value. 1: Setting of the programming destination address is an abnormal value.
0	SF	R/W	Success/fail bit This bit indicates whether the programming process has completed successfully. 0: Programming completed successfully (without error). 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	—	—	Reserved bits Set value 0s.
7 to 0	EB7 to EB0	R/W	Erase block Sets the block number of a block to be erased within the range of 0 to 15. 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 Pass/Fail Parameter

Bit	Bit Name	R/W	Description
7	—	—	Reserved bit Value 0 is returned.
6	MD	R/W	Erase mode-related setting error detection bit Returns the result of a check performed to ensure error protection is not enabled. 0: FLER is in normal state. (FLER = 0) 1: FLER = 1 and erasure cannot be performed.
5	EE	R/W	Erase operation time error detection bit If erasure of the user MAT failed or if a 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. 0: Erase operation completed successfully. 1: Erase operation terminated abnormally, and the erase result is not guaranteed.
4	FK	R/W	Flash key register error detection bit Returns the result of checking the FKEY register value before the start of the erase operation. 0: FKEY register setting is normal. (FKEY = H'5A) 1: Error in FKEY register setting. (FKEY value other than H'5A)
3	EB	R/W	Erase block select error detection bit This is the result after checking whether the specified erase block number is within the range of user MAT block numbers. 0: Erase block number setting is normal. 1: Erase block number setting is abnormal.
2, 1	—	—	Reserved bits Value 0s are returned.
0	SF	R/W	Success/fail bit This bit indicates whether the erase operation has completed successfully. 0: Erasure completed successfully. (Without error) 1: Erasure terminated abnormally. (An error occurred.)

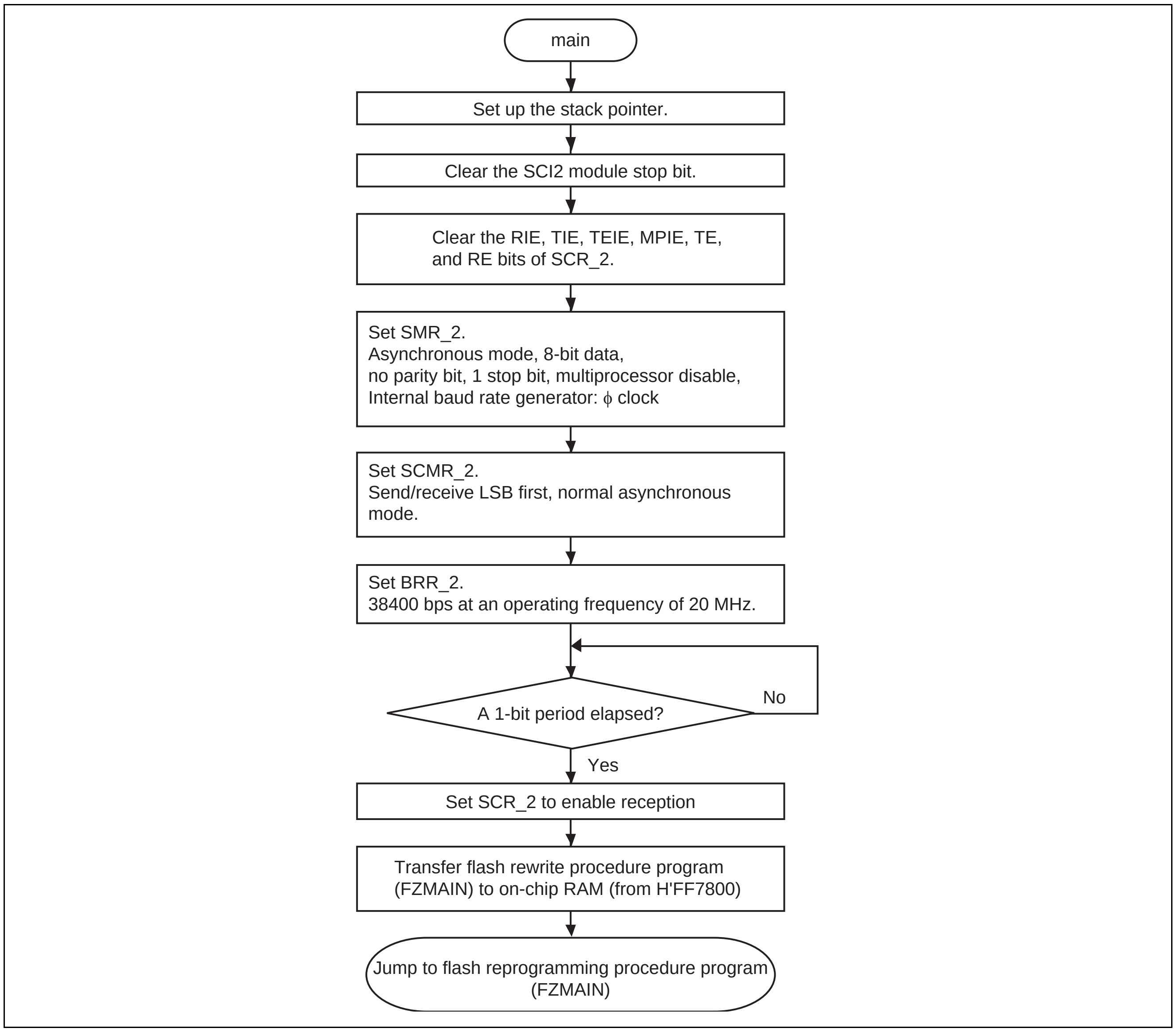
4. Flowchart

4.1 Main Processing (User Boot Program)

After activation, the user boot program sets up the stack pointer, clears the SCI2 module stop bit, sets the interrupt-control mode and interrupt-level settings, transfers the flash programming/erase procedure program into the on-chip RAM, and then jumps to the flash programming/erase procedure program.

Table 21 Main Processing (User Boot Program)

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

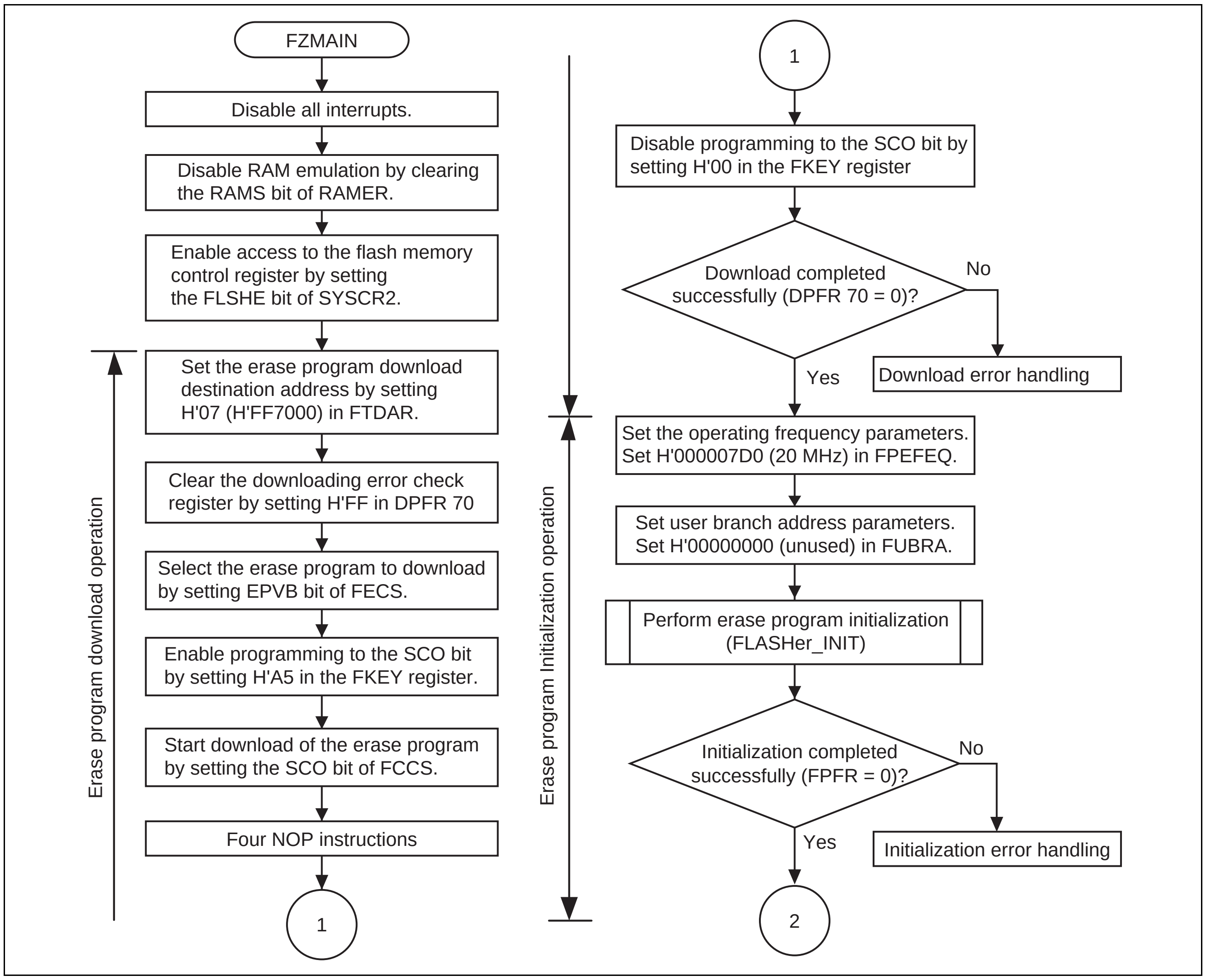

main
Set up the stack pointer.
Clear the SCI2 module stop bit.
Clear the RIE, TIE, TEIE, MPIE, TE, and RE bits of SCR_2.
Set SMR_2. Asynchronous mode, 8-bit data, no parity bit, 1 stop bit, multiprocessor disable, Internal baud rate generator: ϕ clock
Set SCMR_2. Send/receive LSB first, normal asynchronous mode.
Set BRR_2. 38400 bps at an operating frequency of 20 MHz.
A 1-bit period elapsed?
No
Yes
Set SCR_2 to enable reception
Transfer flash rewrite procedure program (FZMAIN) to on-chip RAM (from H'FF7800)
Jump to flash reprogramming procedure program (FZMAIN)

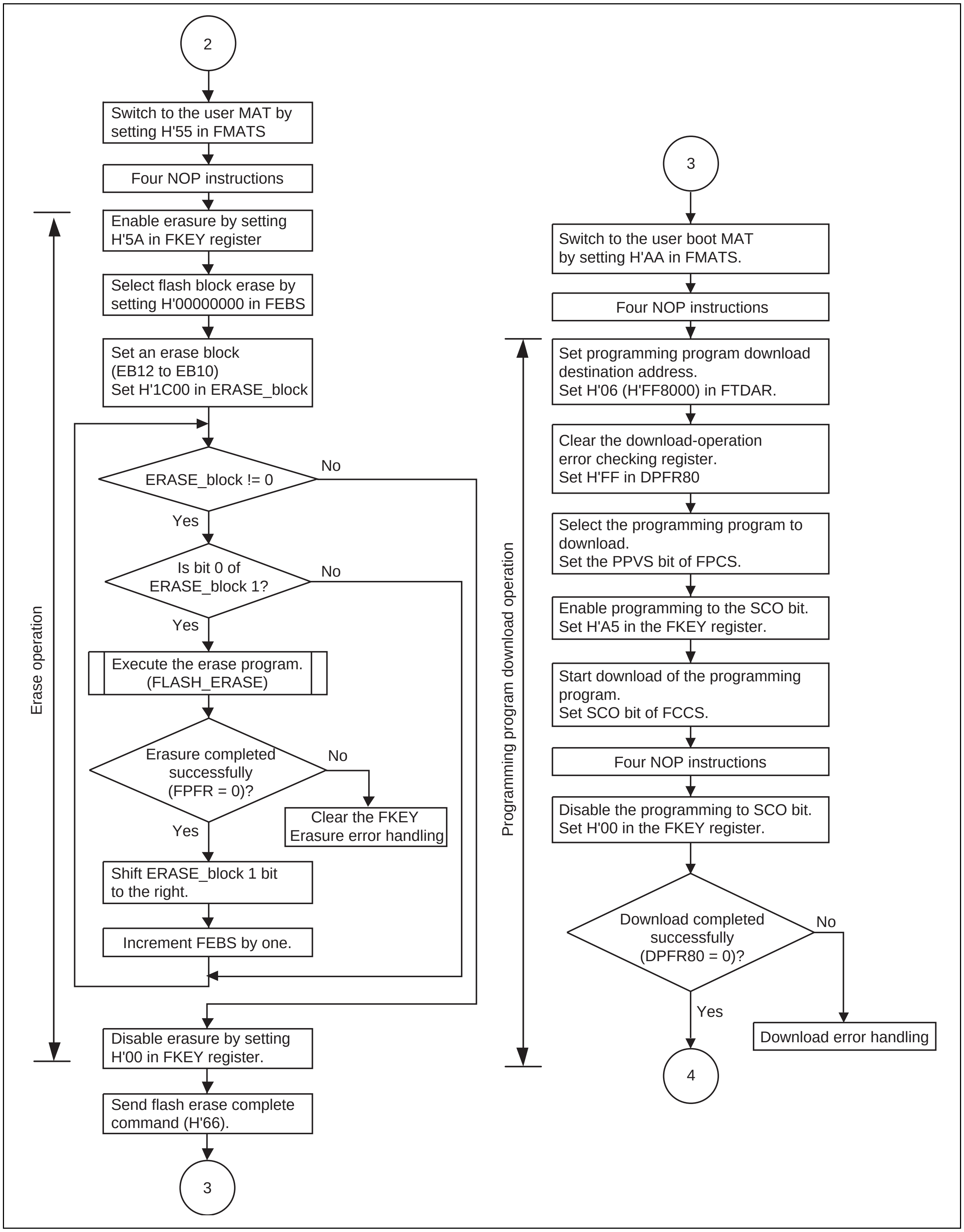
4.2 Flash Programming/Erase Procedure Program

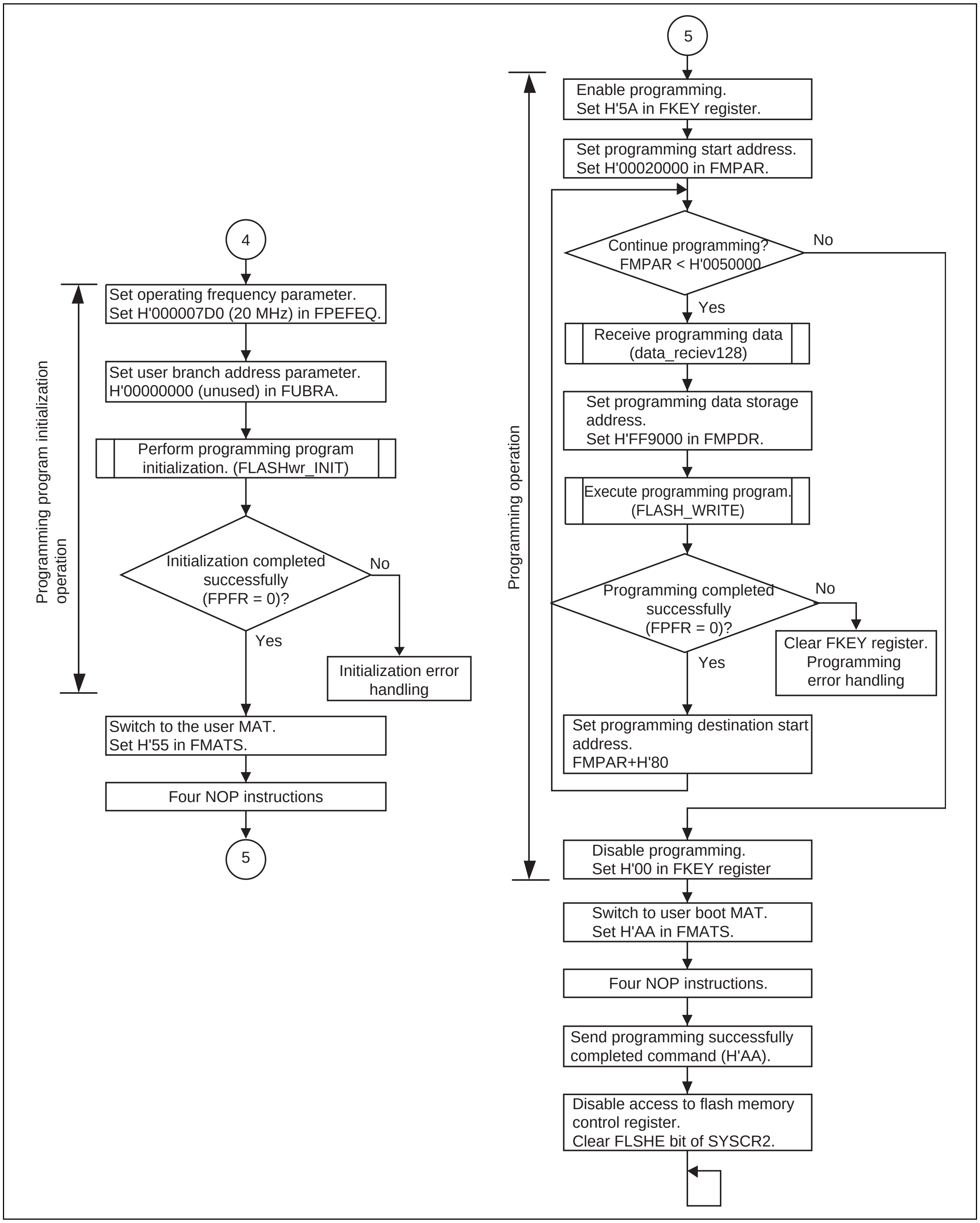
The procedure program, being executed on the on-chip RAM, requests download of the programming/erase program, performs a programming/erase procedure, makes a pass/fail judgment on the result and sends a command, and receives data to be programmed.

Table 22 Flash Programming/Erase Procedure Program

Function name	void FZMAIN (void)
Return value	None
Argument	None
Function calls	unsigned char FLASHer_INIT (unsigned long FPEFEQ, unsigned long FUBRA) unsigned char FLASH_ERASE (unsigned long FEBS) unsigned char FLASHwr_INIT (unsigned long FPEFEQ, unsigned long FUBRA) unsigned char FLASH_WRITE (unsigned long FMPDR, unsigned long FMPAR) void data_reciev128 (void)
Section	Flash memory storage area: FWRMAIN On-chip RAM execution area: RWRMAIN (from H'FF7800)


FZMAIN
Disable all interrupts.
Disable RAM emulation by clearing the RAMS bit of RAMER.
Enable access to the flash memory control register by setting the FLSHE bit of SYSCR2.
Set the erase program download destination address by setting H'07 (H'FF7000) in FTDAR.
Clear the downloading error check register by setting H'FF in DPFR 70
Select the erase program to download by setting EPVB bit of FECS.
Enable programming to the SCO bit by setting H'A5 in the FKEY register.
Start download of the erase program by setting the SCO bit of FCCS.
Four NOP instructions
1
Erase program download operation
Erase program Initialization operation
1
Disable programming to the SCO bit by setting H'00 in the FKEY register
Download completed successfully (DPFR 70 = 0)?
No
Download error handling
Yes
Set the operating frequency parameters. Set H'000007D0 (20 MHz) in FPEFEQ.
Set user branch address parameters. Set H'00000000 (unused) in FUBRA.
Perform erase program initialization (FLASHer_INIT)
Initialization completed successfully (FPFR = 0)?
No
Initialization error handling
Yes
2


2
Switch to the user MAT by setting H'55 in FMATS
Four NOP instructions
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 Erasure error handling
Shift ERASE_block 1 bit to the right.
Increment FEBS by one.
Erase operation
Disable erasure by setting H'00 in FKEY register.
Send flash erase complete command (H'66).
3
3
Switch to the user boot MAT by setting H'AA in FMATS.
Four NOP instructions
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 programming 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.
Programming program download operation
Download completed successfully (DPFR80 = 0)?
No
Yes
Download error handling
4


4
Programming program initialization operation
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
Switch to the user MAT. Set H'55 in FMATS.
Four NOP instructions
5
5
Programming operation
Enable programming. Set H'5A in 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
Switch to user boot MAT. Set H'AA in FMATS.
Four NOP instructions.
Send programming successfully completed command (H'AA).
Disable access to flash memory control register. Clear FLSHE bit of SYSCR2.

4.3 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 the dummy declaration, only a 2 byte RTS instruction is stored on the flash memory (user boot MAT). Note that transfer from flash memory to on-chip RAM is not performed.

Table 23 Flash Memory Programming Initialization Operation (Dummy)

Function name	unsigned char FLASHwr_INIT (unsigned long FPEFEQ, unsigned long FUBRA)
Returned value	Flash pass/fail parameter: FPFR (R0L) 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) 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. In this sample task, the operating frequency is 20 MHz. So set 20000=H'07D0. Flash user branch address set parameter: FUBRA (ER1) This is a parameter for setting the user branch destination address. Since this parameter value is not necessary in this sample task, set address 0 (H'00000000).
Function call	—
Section	Flash storage area: FWRINI On-chip RAM execution area: RWRINI (from H'FF7020)

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

Table 24 Flash Memory Programming Operation (Dummy)

Specification	unsigned char FLASH_WRITE (unsigned long FMPDR, unsigned long FMPAR)
Returned value	Flash pass/fail parameter: FPFR (R0L) 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) 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. Flash multi-purpose address area parameter: FMPAR (ER1) 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 On-chip RAM execution area: RWRIT (from H'FF7010)

4.5 Flash Memory Erase Initialization Operation (Dummy)

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

Table 25 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) 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) 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. In this sample task, the operating frequency is 20 MHz. So set 20000 = H'07D0. Flash user branch address set parameter: FUBRA (ER1) This is a parameter for setting the user branch destination address. Since this parameter value is not required in this sample task, set address 0 (H'00000000).
Function call	—
Section	Flash storage area: FERINI On-chip RAM execution area: RERINI (from H'FF7020)

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

Table 26 Flash Memory Erase Operation (Dummy)

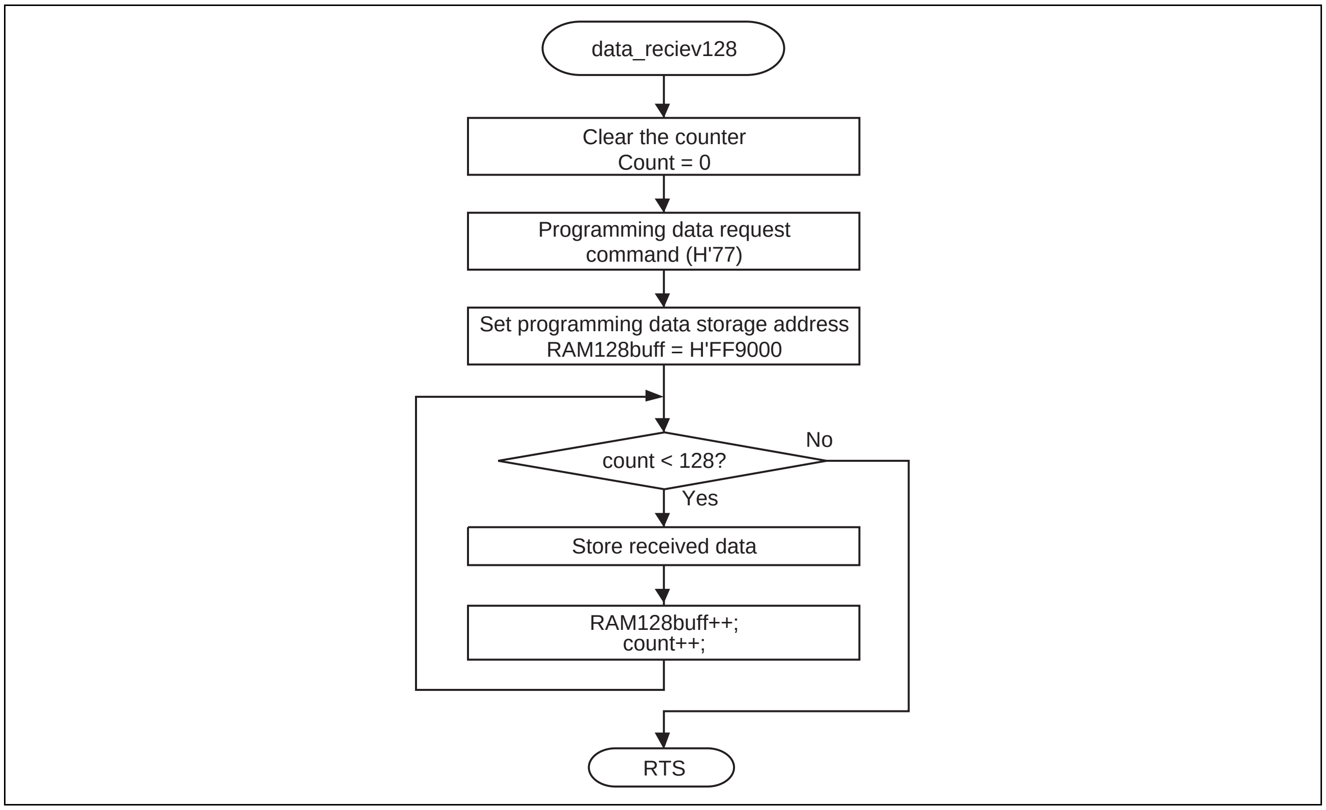
Specification	unsigned char FLASH_ERASE (unsigned long FEBS)
Returned value	Flash pass/fail parameter: FPFR (R0L) This is a value returned as the result of the erase operation. When the erase is successfully completed, the value H'00 is returned.
Argument	Flash erase block select parameter: FEBS (ER0) Sets the block number of the block to be erased. It is not possible to specify two or more block numbers. 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	—
Section	Flash storage area: FERAS On-chip RAM execution area: RERAS (from H'FF7010)

4.7 Programming Data Receive Operation

Before the flash memory programming operation, the programming data request command is sent to the programming tool via the SCI2 (in asynchronous mode), following which 128 bytes of data to be written is received and stored on the fly into the on-chip RAM.

Table 27 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. Memory Map

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

Table 28 Memory Map

- Flash Memory (User boot MAT)

	Section	Description
From H'000000	VECT	Vector table
From H'000400	MAIN	See section 4.1.
From H'000800	FWRMAIN	See sections 4.2 and 4.7.
	FWRINI	See section 4.3.
	FWRIT	See section 4.4.
	FERINI	See section 4.5.
	FERAS	See section 4.6.

- Flash Memory (User MAT)

	Section	Description
H'020000 to H'04FFFF	DATA	Data area (EB10 to EB12)

- On-chip RAM

		Section	Description
H'FF7000 to H'FF77FF		—	Erase/programming program download destination
	H'FF7010	See sections 4.6 and 4.4.	See sections 4.6 and 4.4.
	H'FF7020	See sections 4.5 and 4.3.	See sections 4.5 and 4.3.
From H'FF7800		R/WRMAIN	See section 4.2.

6. Program Listings

6.1 Main Processing (User Application) Source Program

The main processing source program along with the source program to jump to the on-chip RAM and section labels are shown below.

```
/*****************************/
/*  main                     */
/*****************************/
#pragma entry main(sp=0xFFEFB0)             /* Stack pointer setting                */
void main(void){
   *MSTPCRA = 0x3f;
   *MSTPCRB = 0x5f;                         /* SCI2 module stop-bit clear           */
   *MSTPCRC = 0xff;
   Sci2Init();                              /* SCI2 Initial setting                 */
   for(src=_FWRMAIN_BGN, dst=_R/WRMAIN_RAM; src<=_FWRMAIN_END; src++, dst++) {
      *dst = *src;                          /* Transfer to on-chip RAM              */
   }
   jmp_RAM();                               /* Store received data                  */
   while(1);
}
/*****************************/
/*  Jump 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       R/WRMAIN,           CODE,ALIGN=2
          .SECTION       C,DATA,ALIGN=2
          .EXPORT __FWRMAIN_BGN
          .EXPORT __FWRMAIN_END
          .EXPORT __R/WRMAIN_RAM
__FWRMAIN_BGN .data.l   (STARTOF FWRMAIN)
__FWRMAIN_END .data.l   (STARTOF FWRMAIN) + (SIZEOF FWRMAIN)
__R/WRMAIN_RAM   .data.l (STARTOF R/WRMAIN)
          .END
```

6.2 Source Program of SCI2 Initial Setting Operation

```
/*****************************/
/*  SCI2 Initial Setting     */
/*****************************/
void init_sci2(void){
   unsigned long i=0;
   *SCR2  = 0x00;              /* Clearing of RIE, TIE, TEIE, MPIE, TE, and RE bits */
   *SMR2  = 0x00;              /* Asynchronous, 8-bit data, no parity bit,          */
                               /* 1 stop bit, multi-processor disabled,             */
                               /* Internal baud rate generator: ϕ clock.            */
   *SCMR2 = 0xF2;              /* Send/receive LSB first, normal asynchronous mode. */
   *BRR2  = 0x0F;              /* 38400bps BRR = (Pϕ = 20MHz)                       */
   for( i=0 ; i<170 ; i++ );   /* 26 µs wait                                        */
   *SCR2  = 0x30;              /* Set TE and RE bits.                               */
}
```

6.3 Flash Programming/Erase Procedure Program Source Program

The flash programming/erase procedure program source code is shown below along with the programming data receive and command send processes that are used during the flash programming/erase procedure operation.

```
#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){              /* 0xFF7800 -                                */
    unsigned long FPEFEQ;       /* ER0                                       */
    unsigned long FUBRA;        /* ER1                                       */
    unsigned char FPFR;         /* R0L                                       */
    unsigned long FEBS;         /* ER0                                       */
    unsigned long FMPAR;        /* ER1                                       */
    unsigned long FMPDR;        /* ER0                                       */
    *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 = FLASHer_INIT(FPEFEQ,FUBRA); /* FPFR--> R0L,FPEFEQ-->ER0,FUBRA-->ER1          */
    if(FPFR != 0x00){                   /* Initial setting error?                        */
        goto end_p;
    }

/* Mat switching   */

    *FMATS = 0x55;              /* User mat select                                       */
    NOP4();                     /* NOP×4                                                 */
```

```
/* 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                                         */

/* Mat switching   */

    *FMATS = 0xAA;           /* User boot mat select                        */
    NOP4();                  /* NOP×4                                       */
    TXD_ex(0x66);            /* Send erase complete command                 */

/* Programming program download process    */

    *FTDAR = 0x07;           /* Download destination address setting (H'FF7000)          */
    DPFR70 = 0xFF;           /* Download process error check register (DPFR) clear       */
    *FPCS = 0x01;            /* Programming program select (PPVS set)                    */
    *FKEY = 0xA5;            /* Programming program download enable                      */
    *FCCS |= 0x01;           /* Download request (SCO set)                               */
    NOP4();                  /* NOP×4                                                    */
    *FKEY = 0x00;            /* FKEY clear                                               */
    if(DPFR70 != 0x00){      /* Download error?                                          */
        goto end_p;
    }

/* Programming 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;
    }

/* Mat switching   */

    *FMATS = 0x55;                  /* User mat select                                  */
    NOP4();                         /* NOP×4                                            */
```

```
/* Programming process */

    *FKEY = 0x5A;                       /* FKEY set                                         */
    FMPAR = 0x00020000;                 /* Programming start address setting                */
    while(FMPAR < 0x00050000){          /* Writing complete?                                */
        data_reciev128();               /* Programming 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                                       */

/* Mat switching   */

    *FMATS = 0xAA;                  /* User boot mat select                             */
    NOP4();                         /* NOP×4                                            */
    TXD_ex(0xAA);                   /* Send writing successfully completed command      */
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         */
    TXD_ex(0x77);                  /* Send programming data request command             */
    RAM128buff=(unsigned char *)0xFFFF9000;
    while(count < 128){
    while((*SSR2 & 0x40)!=0x40);
        *SSR2 &= 0xbf;
        *RAM128buff = *RDR2;       /* Received data storage                             */
        RAM128buff++;
        count++;
    }
}
void TXD_ex(unsigned char t_dt){   /*Send command                                       */
    while((*SSR2 & 0x80) != 0x80);
    *TDR2 = t_dt;
    *SSR2 &= 0x7f;
}
```

6.4 Flash Memory Erase Initialization Process Dummy Source Program

See section 4.5.

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

6.5 Flash Memory Erase Process Dummy Source

See section 4.6.

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

6.6 Flash Memory Programming Initialization Process Dummy Source Program

See section 4.3.

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

6.7 Flash Memory Programming Process Dummy Source Program

See section 4.4.

```
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

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