SH7280 Group

Using User Program Mode

R01AN0387EJ0100
Rev. 1.00
Nov. 30, 2010

Summary

This application note describes an example to run the flash memory reprogramming program in SH7286 microcomputers (MCUs) user program mode. An external device which is connected to the SH7286 stores the data to write to the flash memory, and communicates with the flash memory using the Serial Communication Interface with FIFO.

The flash memory reprogramming program described in this application note is stored on the SH7286 user MAT. The simple flash API for SH2 and SH2A (Standard API) provided by the Renesas Electronics is used to reprogram the flash memory.

Target Device

SH7286 MCU

Contents

1. Introduction

1.1 Specifications

This application programs, erases, and reads the flash memory using user program mode. User program mode handles programming, erasing, and reading with a desired interface. This application uses the serial communication between the host computer and the SH7286 to handle these processing.

When the SH7286 receives the flash reprogramming/erasing command (user control command) from the host computer while executing the user application, the SH7286 programs or erases the flash memory. When it receives the flash memory reading command from the host computer, it reads the flash memory.

Figure 1 shows the system configuration of this application.

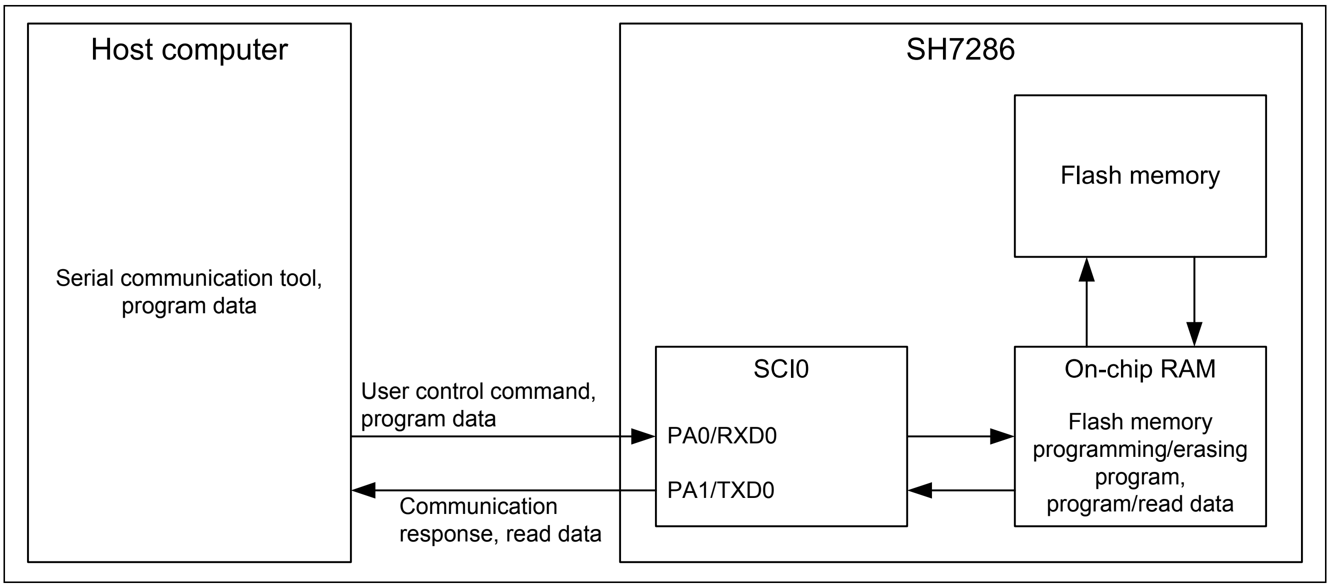

Host computer
Serial communication tool, program data
User control command, program data
Communication response, read data
SH7286
Flash memory
SCI0
PA0/RXD0
PA1/TXD0
On-chip RAM
Flash memory programming/erasing program, program/read data

Figure 1 System Configuration

1.2 Modules Used

- Serial Communication Interface (SCI)
- Flash Memory (ROM)

1.3 Applicable Conditions

MCU	SH7286 (1-MB flash memory version)
Operating Frequency [(1)]	Internal clock: 25 MHz Bus clock: 25 MHz Peripheral clock: 25 MHz
Integrated Development Environment [(2)]	Renesas Electronics Corporation High-performance Embedded Workshop Ver.4.04.01
C Compiler	Renesas Electronics SuperH RISC engine Family C/C++ compiler package Ver.9.01 Release 01
Compiler Options	Default setting in the High-performance Embedded Workshop (-cpu=sh2a -debug -gbr=auto -global_volatile=0 -opt_range=all -infinite_loop=0 -del_vacant_loop=0 -struct_alloc=1)

Notes: 1. When downloading the on-chip program to program or erase the flash memory, set the ratio of internal clock (Iφ), bus clock (Bφ), and peripheral clock (Pφ) by the Frequency control register (FRQCR) as 1/4:1/4:1/4 to the input clock frequency.

2. As the E10A-USB emulator does not support boot mode, user boot mode, and user program mode, the flash memory reprogramming program cannot be debugged by the E10A-USB emulator.

1.4 Related Application Note

For more information, refer to the following application note:

- SH Family Simple Flash API for SH2 and SH2A

2. Overview

This application uses the Serial Communication Interface (SCI) to connect the SH7286 with the external device.

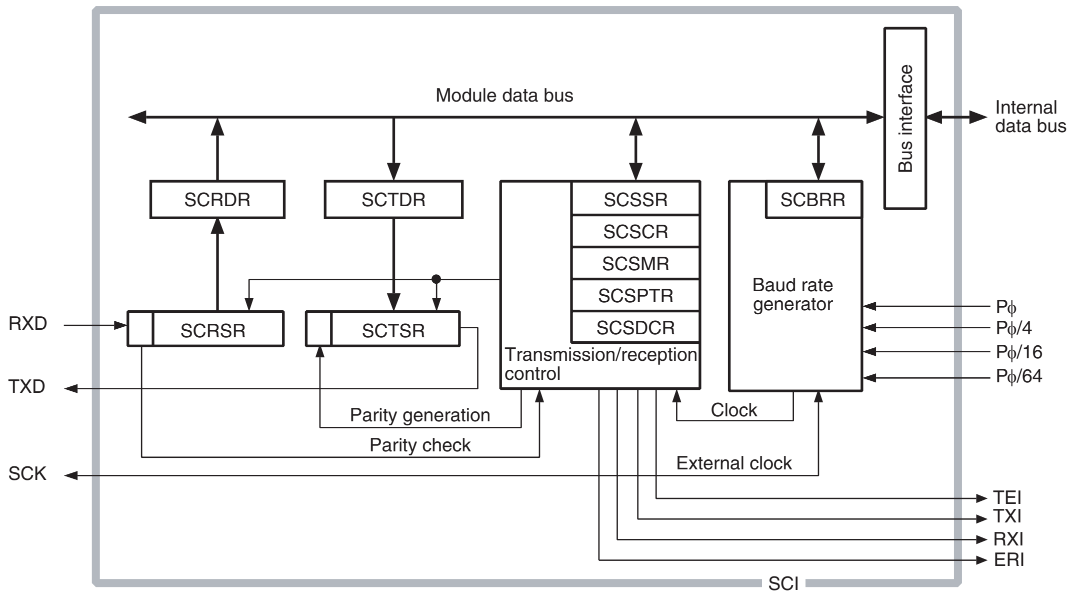
2.1 Overview of Modules

2.1.1 Serial Communication Interface (SCI)

SCI supports both asynchronous and clocked synchronous serial communication. It also supports full-duplex communication and allows double-buffering both at transmitter and receiver to transmit/receive the serial data continuously at high speed.

This application uses the SCI for the handshake between the SH7286 and an external device, and to transmit/receive the flash memory reprogram data.

Figure 2 shows the SCI block diagram.



[Legend]

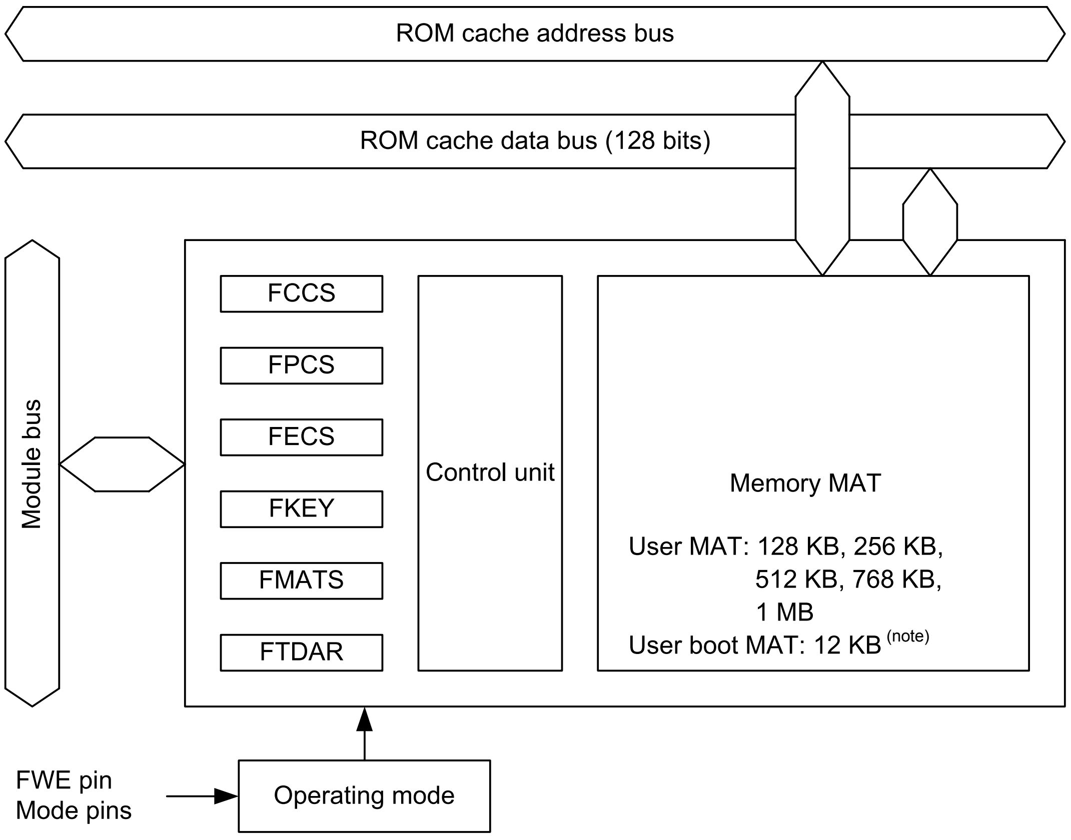
SCRSR:	Receive shift register
SCRDR:	Receive data register
SCTSR:	Transmit shift register
SCTDR:	Transmit data register
SCSMR:	Serial mode register
SCSCR:	Serial control register
SCSSR:	Serial status register
SCBRR:	Bit rate register
SCSPTR:	Serial port register
SCSDCR:	Serial direction control register

Figure 2 SCI Block Diagram

2.1.2 Flash Memory (ROM)

The SH7280 group MCU programs or erases the flash memory using its on-chip program.

Figure 3 shows the flash memory block diagram.



[Legend]

FCCS: Flash code control and status register

FPCS: Flash program code select register

FECS: Flash erase code select register

FKEY: Flash key code register

FMATS: Flash MAT select register

FTDAR: Flash transfer destination address register

Note: User boot MAT cannot be used with the SH7243.

Figure 3 Flash Memory Block Diagram

2.2 Programming/Erasing the Flash Memory

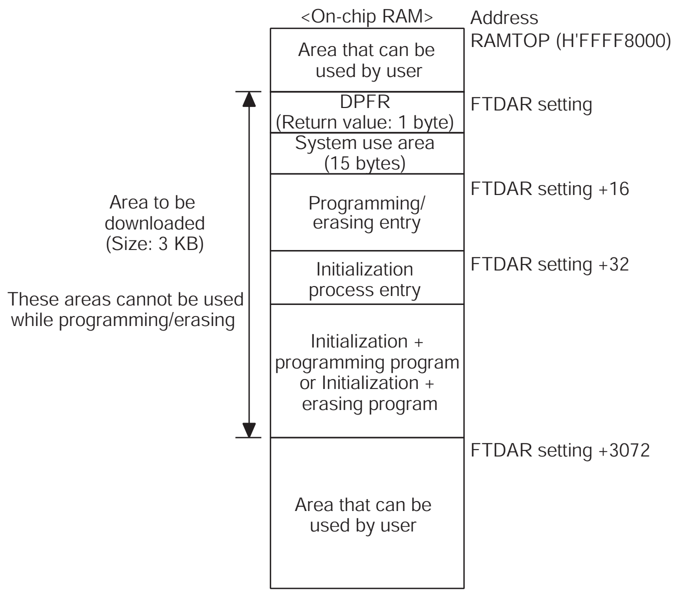
The SH7280 group MCU uses its on-chip program to program and erase the flash memory. This section describes how to reprogram the flash memory. For more information, refer to the SH7280 Group Hardware Manual. This application uses the Standard API. For more information about the API, refer to the related application note.

2.2.1 Preparing to Program/Erase the Flash Memory

To use the MCU on-chip program, the user must download the program to the on-chip RAM. After downloading is completed, specify the program address or data, erase block to the Programming/erasing interface registers/parameters and the downloaded program programs/erases the flash memory.

User must prepare programs to request to download, program and erase the flash memory, and detect the outcome, however, the SCO bit in the FCCS register must be set in on-chip RAM. As all downloaded on-chip programs are in on-chip RAM, make sure not to use the same area in on-chip RAM.

Figure 4 shows the downloaded program area memory map.



<On-chip RAM>
Address
RAMTOP (H'FFFF8000)
Area that can be used by user
DPFR (Return value: 1 byte)
FTDAR setting
System use area (15 bytes)
Programming/ erasing entry
FTDAR setting +16
Area to be downloaded (Size: 3 KB)
These areas cannot be used while programming/erasing
Initialization process entry
FTDAR setting +32
Initialization + programming program or Initialization + erasing program
FTDAR setting +3072
Area that can be used by user

Figure 4 Memory Map After Downloading the Program

2.2.2 Erasing the Flash Memory

Change the download destination on-chip RAM address in the FTDAR register to download the erasing program and programming program in other on-chip RAM areas separately.

Figure 5 shows the flow chart for erasing the flash memory.

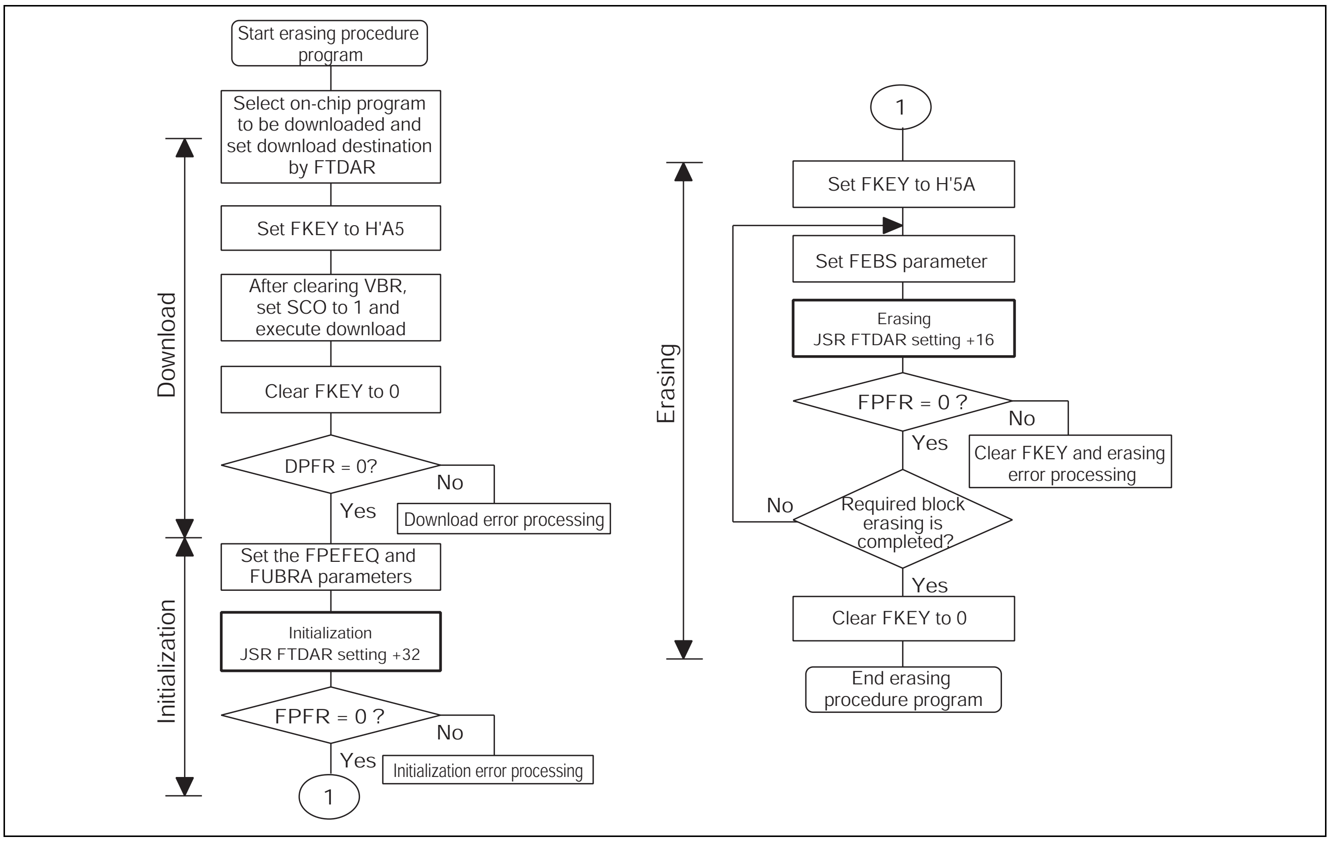

Start erasing procedure program
Select on-chip program to be downloaded and set download destination by FTDAR
Set FKEY to H'A5
After clearing VBR, set SCO to 1 and execute download
Clear FKEY to 0
DPFR = 0?
No
Download error processing
Yes
Set the FPEFEQ and FUBRA parameters
Initialization JSR FTDAR setting +32
FPFR = 0 ?
No
Initialization error processing
Yes
1
Download
Initialization
1
Set FKEY to H'5A
Set FEBS parameter
Erasing JSR FTDAR setting +16
FPFR = 0 ?
No
Clear FKEY and erasing error processing
Yes
Required block erasing is completed?
No
Yes
Clear FKEY to 0
End erasing procedure program
Erasing

Figure 5 Erasing the Flash Memory

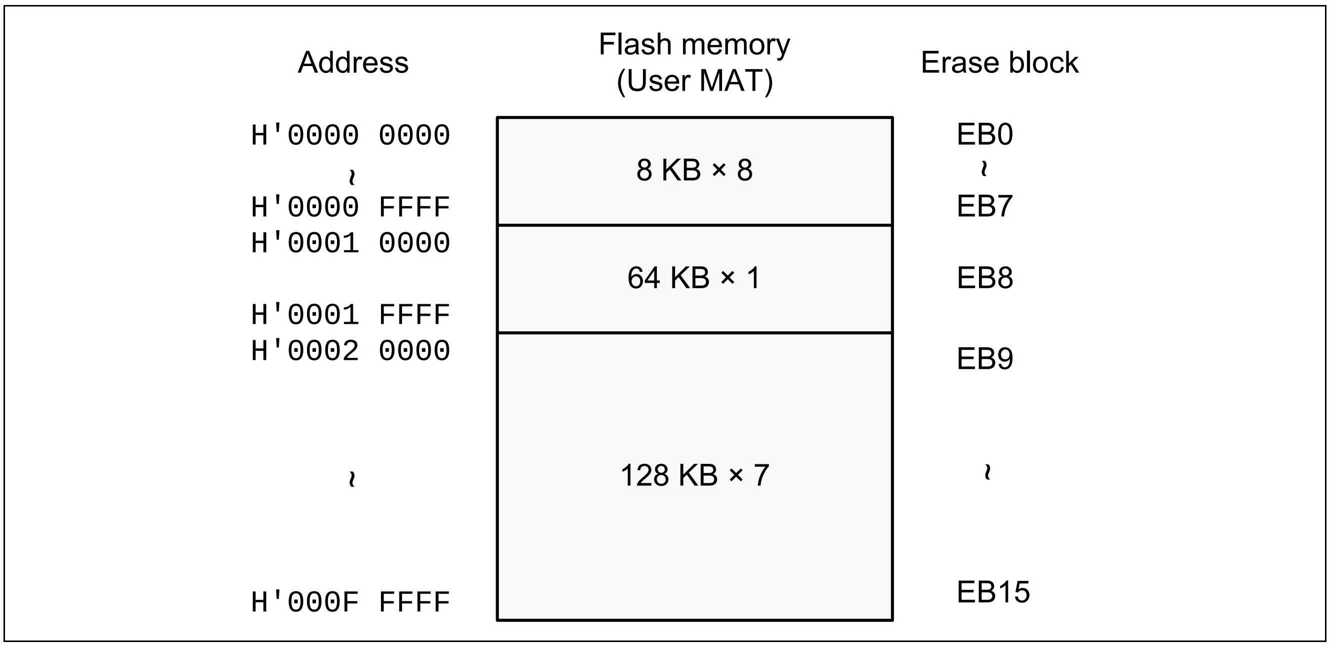


Figure 6 Dividing the Flash Memory Erase Block

Table 1 Erase Block and Address

Erase Block	Address	Capacity
EB0	H'0000_0000 to H'0000_1FFF	8 KB
EB1	H'0000_2000 to H'0000_3FFF	
EB2	H'0000_4000 to H'0000_5FFF	
EB3	H'0000_6000 to H'0000_7FFF	
EB4	H'0000_8000 to H'0000_9FFF	
EB5	H'0000_A000 to H'0000_BFFF	
EB6	H'0000_C000 to H'0000_DFFF	
EB7	H'0000_E000 to H'0000_FFFF	
EB8	H'0001_0000 to H'0001_FFFF	64 KB
EB9	H'0002_0000 to H'0003_FFFF	128 KB
EB10	H'0004_0000 to H'0005_FFFF	
EB11	H'0006_0000 to H'0007_FFFF	
EB12	H'0008_0000 to H'0009_FFFF	
EB13	H'000A_0000 to H'000B_FFFF	
EB14	H'000C_0000 to H'000D_FFFF	
EB15	H'000E_0000 to H'000F_FFFF	

2.2.3 Programming the Flash Memory

Change the download destination on-chip RAM address in the FTDAR register to download the erasing program and programming program in other on-chip RAM areas separately.

Figure 7 shows the flow chart for programming the flash memory.

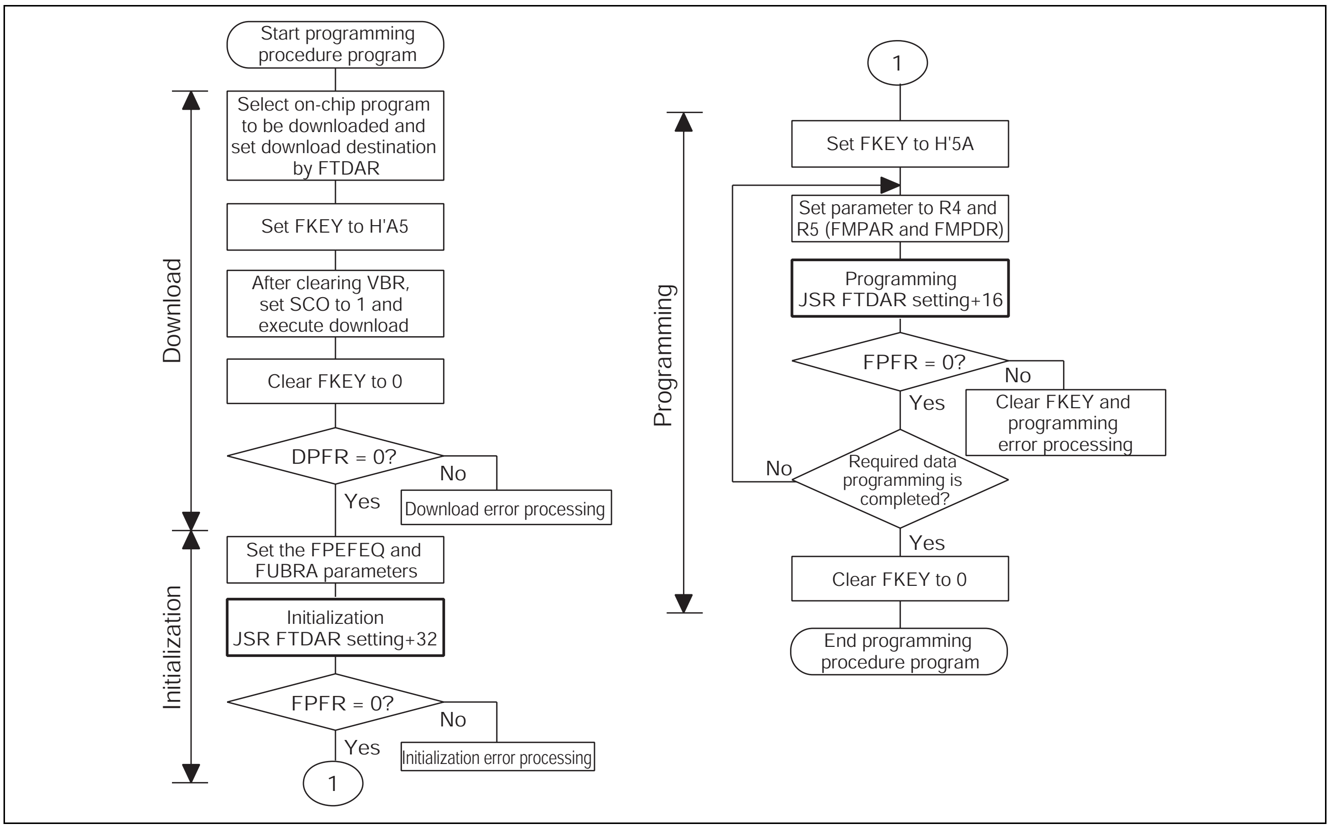

Start programming procedure program
Select on-chip program to be downloaded and set download destination by FTDAR
Set FKEY to H'A5
After clearing VBR, set SCO to 1 and execute download
Clear FKEY to 0
DPFR = 0?
No
Yes
Download error processing
Set the FPEFEQ and FUBRA parameters
Initialization JSR FTDAR setting+32
FPFR = 0?
No
Yes
Initialization error processing
1
Download
Initialization
1
Set FKEY to H'5A
Set parameter to R4 and R5 (FMPAR and FMPDR)
Programming JSR FTDAR setting+16
FPFR = 0?
No
Yes
Clear FKEY and programming error processing
No
Required data programming is completed?
Yes
Clear FKEY to 0
End programming procedure program
Programming

Figure 7 Programming the Flash Memory

2.3 Flash Program Data Buffer

This application has the buffer area to hold the program data in the SH7286 on-chip RAM. The capacity of the buffer area is 256 bytes, which is equivalent to a flash programming.

Figure 8 shows the operation image of the buffer. Table 2 lists the data buffer area address (note).

Note: Data buffer area is divided into sections. Change the section allocation address to set the desired buffer area address. Make sure not to use the same area as the on-chip program in on-chip RAM.

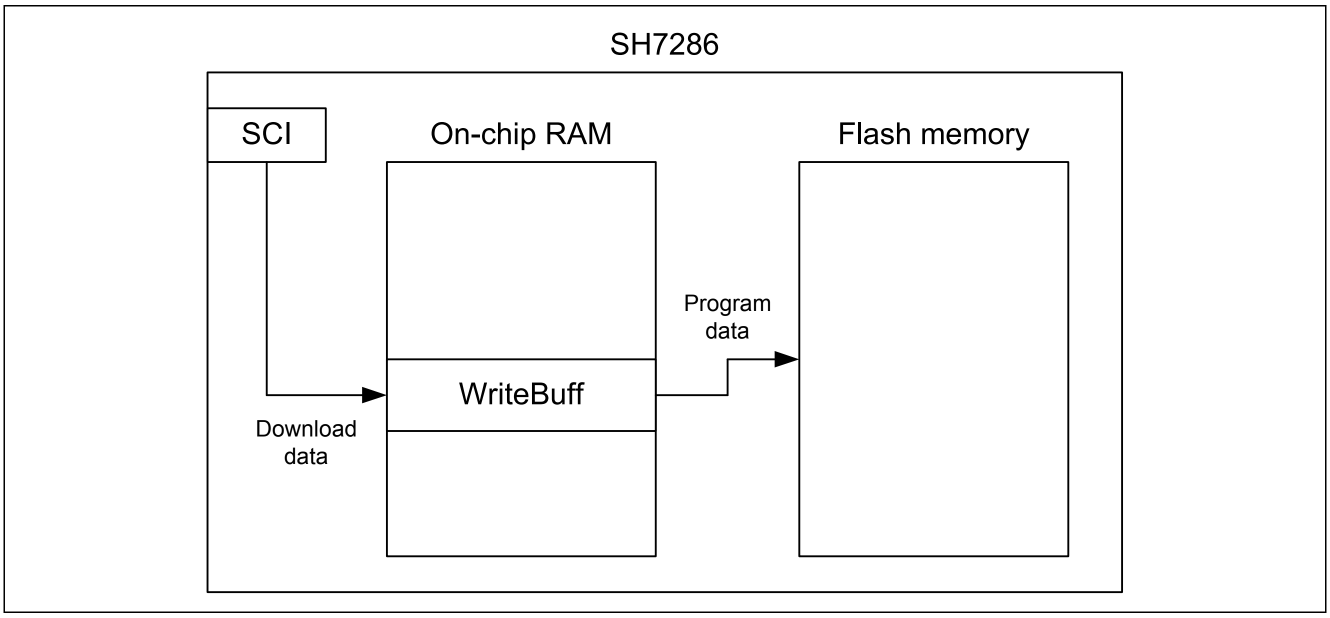


Figure 8 Buffer Operating Image

Table 2 Data Buffer Area Address

Buffer Name	Address	Capacity
WriteBuff	H'FFF8_5000 to H'FFF8_50FF	256 bytes

3. Sample Program External Specifications

This application allocates the flash memory reprogramming sample program including main function (sample program) in EB0 block in the user MAT (address: H'0000 0000 to H'0000 1FFF). Sample program consists of the user application (main function), serial communication program, flash memory reprogramming program, and Standard API.

The target area to program or erase in the flash memory is the user MAT (EB1 to EB15 block address: H'0000 2000 to H'000F FFFF) other than EB0 block where the sample program is allocated.

Figure 9 shows the image of programming and erasing the flash memory by the sample program.

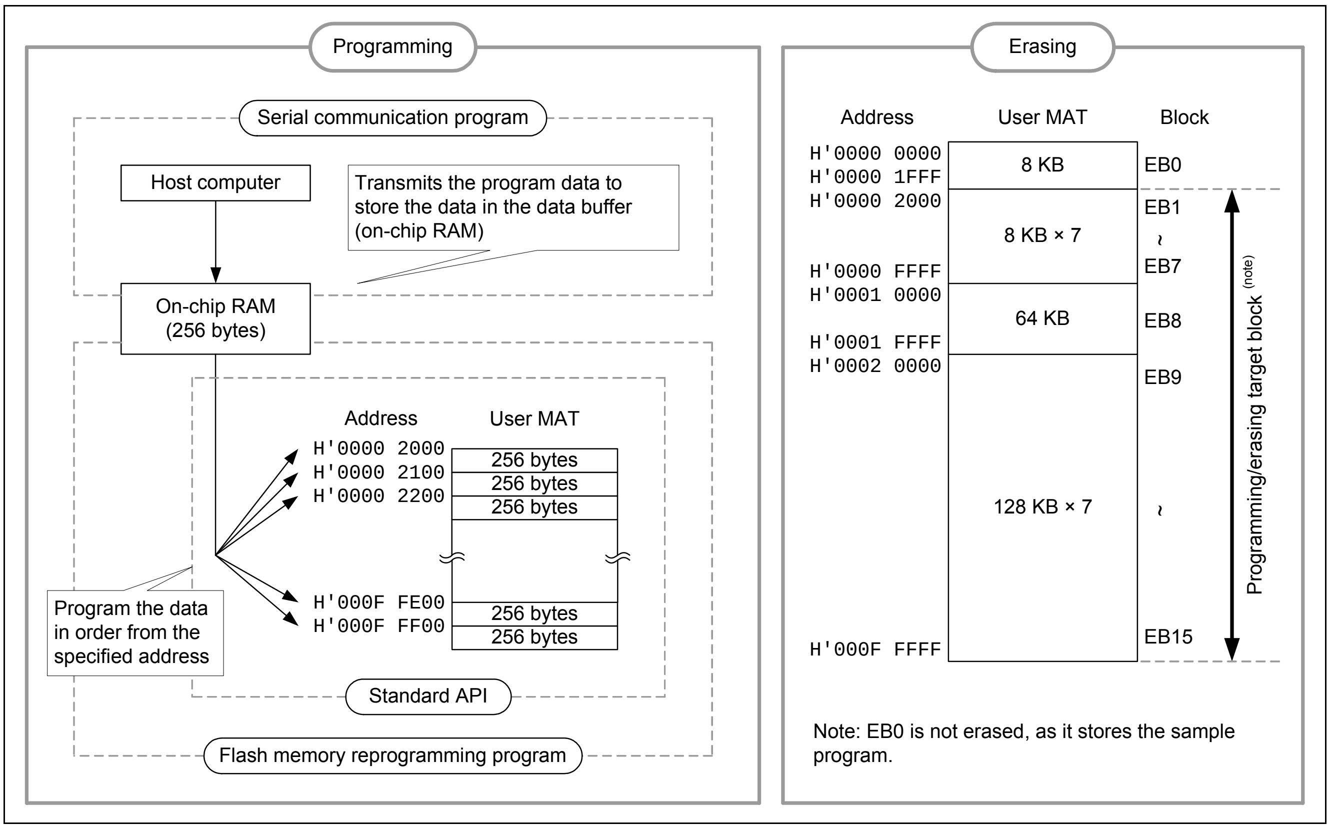


Figure 9 Programming and Erasing the Flash Memory

3.1 Sample Program Operation

This application executes the serial communication with the host computer and transmits/receives the user control commands for communication and data to program, erase and read the flash memory. It uses SCI channel 0 (SCI0) for the serial communication. The sample program these processing to control the flash memory in on-chip RAM.

The sample program checks whether the flash memory is program-/erase-enabled or not. When the flash memory is program-/erase-enabled, the sample program requests the host computer to issue the user control command for communication; otherwise, the sample program polls the FWE bit until the flash memory is program-/erase-enabled.

Figure 10 shows the main processing flow chart.

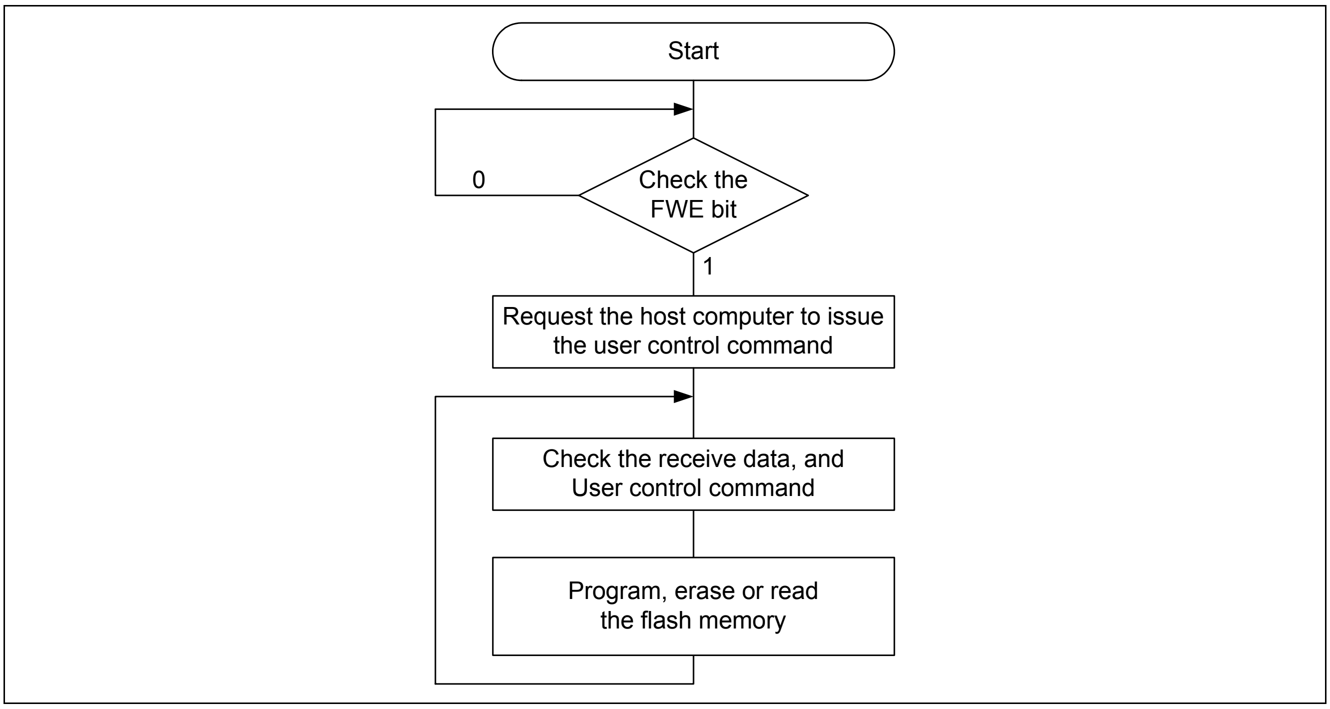

Start
Check the FWE bit
0
1
Request the host computer to issue the user control command
Check the receive data, and User control command
Program, erase or read the flash memory

Figure 10 Main Processing Flow Chart

Table 3 lists the user control commands for communication from the host computer. Table 4 lists the notification from the SH7286.

When an error occurs while programming or erasing the flash memory, the sample program notifies the error end (RET_NG) to the host computer and enters an infinite loop. Add the error processing as appropriate.

Table 3 User Control Commands from the Host Computer to SH7286

Command Name	Value	Description
CMD_GO	H'55	Starts programming/erasing the flash memory
CMD_READ	H'AA	Reads the flash memory
CMD_ERASE	H'77	Erases the flash memory
CMD_WRITE	H'88	Programs the flash memory
CMD_WEND	H'99	Ends programming/erasing the flash memory

Table 4 Notifications from the SH7286 to the Host Computer

Notification Name	Value	Description
Normal end (RET_OK)	H'00	Notifies the host computer that the command handling ends successfully
Error end (RET_NG)	H'01	Notifies the host computer that the command handling ends in error
Transmit request (RET_REQ)	H'11	Notifies the host computer that the sample program is requesting to transmit the user control command or the program data

3.1.1 Programming or Erasing the Flash Memory

When the host computer transmits the flash memory programming/erasing start command (CMD_GO), the sample program transitions to the flash memory programming/erasing state, and notifies the transmission request (RET_REQ) to the host computer.

Next, the host computer transmits the flash memory erasing command (CMD_ERASE), and specifies the program/erase destination block number (other than EB0) in units of 2 bytes. This 2-byte data must be set to 1 to the bit that indicates the specified block number (i.e. Set the data to H'0002 for programming EB1, to H'0800 for programming EB11.)

When the specified block does not exist (H'0000 is specified), or bit 0 corresponding to EB0 is set to 1, the sample program notifies the error end (RET_NG) to the host computer, and enters an infinite loop. When erasing the flash memory in the specified block is completed, the sample program notifies the normal end (RET_OK) to the host computer.

Then, the host computer transmits the flash memory programming command (CMD_WRITE), and the destination start address and program data size (4-byte data). Make sure to specify the address (H'0000 2000 to H'000F FFFF) within the specified block (EB1 to EB15) when erasing the flash memory at 256-byte boundary. Otherwise, the operation is not guaranteed.

When the host computer transmits the destination start address and program data size, the sample program notifies the host computer to request transmitting the program data (RET_REQ), and the host computer transmits the program data size data. As the program data in the user MAT must be in units of 256 bytes, the sample program programs the flash memory at every 256-byte data is received. (When the specified program data size is less than 256 bytes, the sample program sets the remaining data to H'FF.)

When the total number of programming the flash memory does not reach the program data size, the sample program notifies the host computer to request transmitting (RET_REQ) the program data. The host computer must repeat transmitting data until the size reaches the program data size. When the total number of programming the flash memory reaches the program data size, the sample program notifies the normal end (RET_OK) to the host computer.

Finally, the host computer transmits the flash memory programming/erasing end command (CMD_WEND), and the sample program ends programming or erasing the flash memory.

Figure 11 shows the communication command sequence when programming or erasing the flash memory by the sample program.

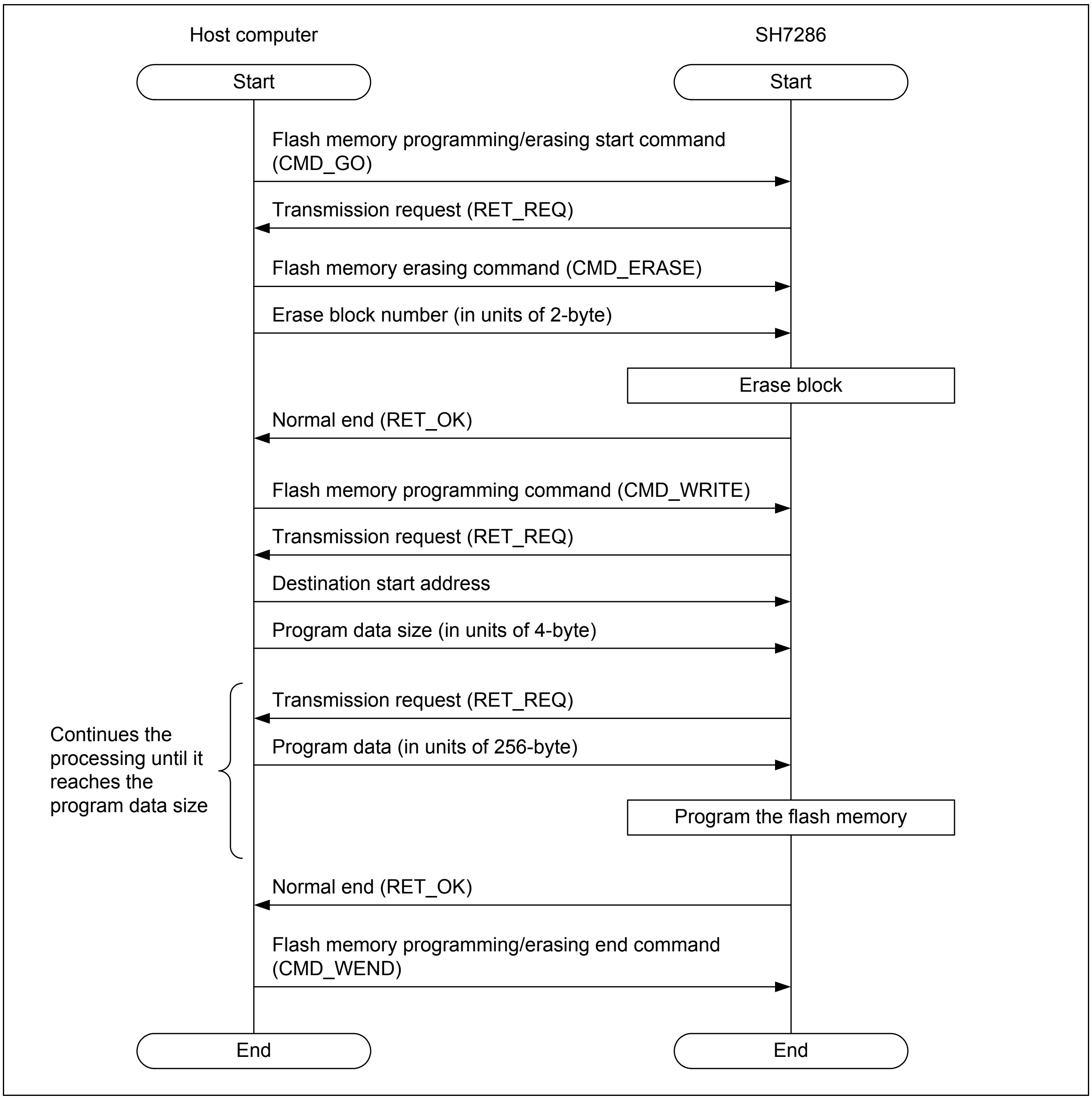

Host computer
SH7286
Start
Start
Flash memory programming/erasing start command (CMD_GO)
Transmission request (RET_REQ)
Flash memory erasing command (CMD_ERASE)
Erase block number (in units of 2-byte)
Erase block
Normal end (RET_OK)
Flash memory programming command (CMD_WRITE)
Transmission request (RET_REQ)
Destination start address
Program data size (in units of 4-byte)
Continues the processing until it reaches the program data size
Transmission request (RET_REQ)
Program data (in units of 256-byte)
Program the flash memory
Normal end (RET_OK)
Flash memory programming/erasing end command (CMD_WEND)
End
End

Figure 11 Communication Command Sequence When Programming/Erasing the Flash Memory

3.1.2 Reading the Flash Memory

The sample program reads the specified size of data from the destination start address and transmits the data to the host computer by the flash memory reading command (CMD_READ).

When the sample program receives the flash memory reading command (CMD_READ), it notifies the transmission request (RET_REQ) to the host computer. When the sample program receives the destination start address (in units of 4-byte) and read data size (in units of 4-byte) from the host computer (8 bytes in total), it reads the specified size of data from the destination address, and transmits the data to the host computer.

Specify the address (H'0000 0000 to H'000F FFFF) within blocks EB0 to EB15 (User MAT) as the read destination start address. Otherwise, the sample program does not read the flash memory, notifies the error end (RET_NG) to the host computer to enter an infinite loop. As the sample program does not include the error check when the specified address is not on the user MAT, do not specify the address that is out of bounds.

Figure 12 shows the communication command sequence when reading the flash memory.

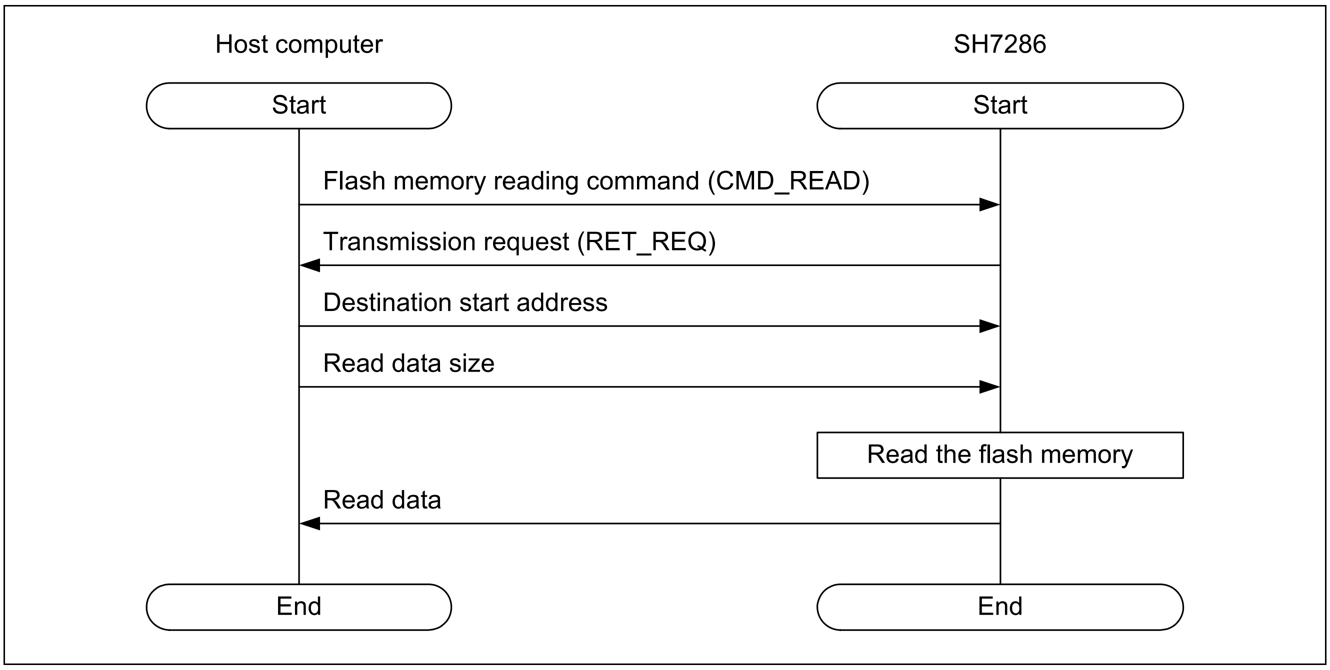

Host computer
SH7286
Start
Start
Flash memory reading command (CMD_READ)
Transmission request (RET_REQ)
Destination start address
Read data size
Read the flash memory
Read data
End
End

Figure 12 Communication Command Sequence When Reading the Flash Memory

4. Sample Program Internal Specifications

4.1 Modules

Table 5 lists the specifications of sample program modules.

Table 5 Sample Program Modules

Type	Module Name	Function Name	Description	Flow Chart
User application	Main processing	main	Executes the user application	See Figure 13
Flash memory reprogramming	Flash memory programming/ erasing	ocf_write	Programs or erasing the flash memory	See Figure 14 and Figure 15
	Flash memory reading	ocf_read	Reads the flash memory	See Figure 16
	Flash memory program-/erase-enabled check	ocf_pe_chk	Checks that the flash memory is program-/erase-enabled	See Figure 17
Serial communication control	SCI configuration	io_sci_init	Configures the SCI (channel 0)	–
	SCI receive data existence check	io_sci_chk_rcv	Checks if the receive data is stored in the SCRDR register	–
	SCI transmit	io_sci_snd	Transmits one-byte data	–
	SCI receive	io_sci_rcv	Receives the specified bytes of data	–
	SCI module stop	io_sci_stop	Stop supplying the clock to the SCI (channel 0)	–
Standard API	Block erase	R_FlashErase	Erases the data in the specified block	–
	Flash memory programming	R_FlashWrite	Programs the data in the specified address	–

4.2 Variable Used

Table 6 lists a variable used in the sample program.

Table 6 Variable

Variable Label Name	Description	Module to Use
unsigned char WriteBuff[256]	Stores the program data	ocf_write

4.3 Register Settings

Table 7 lists the register settings for the peripherals.

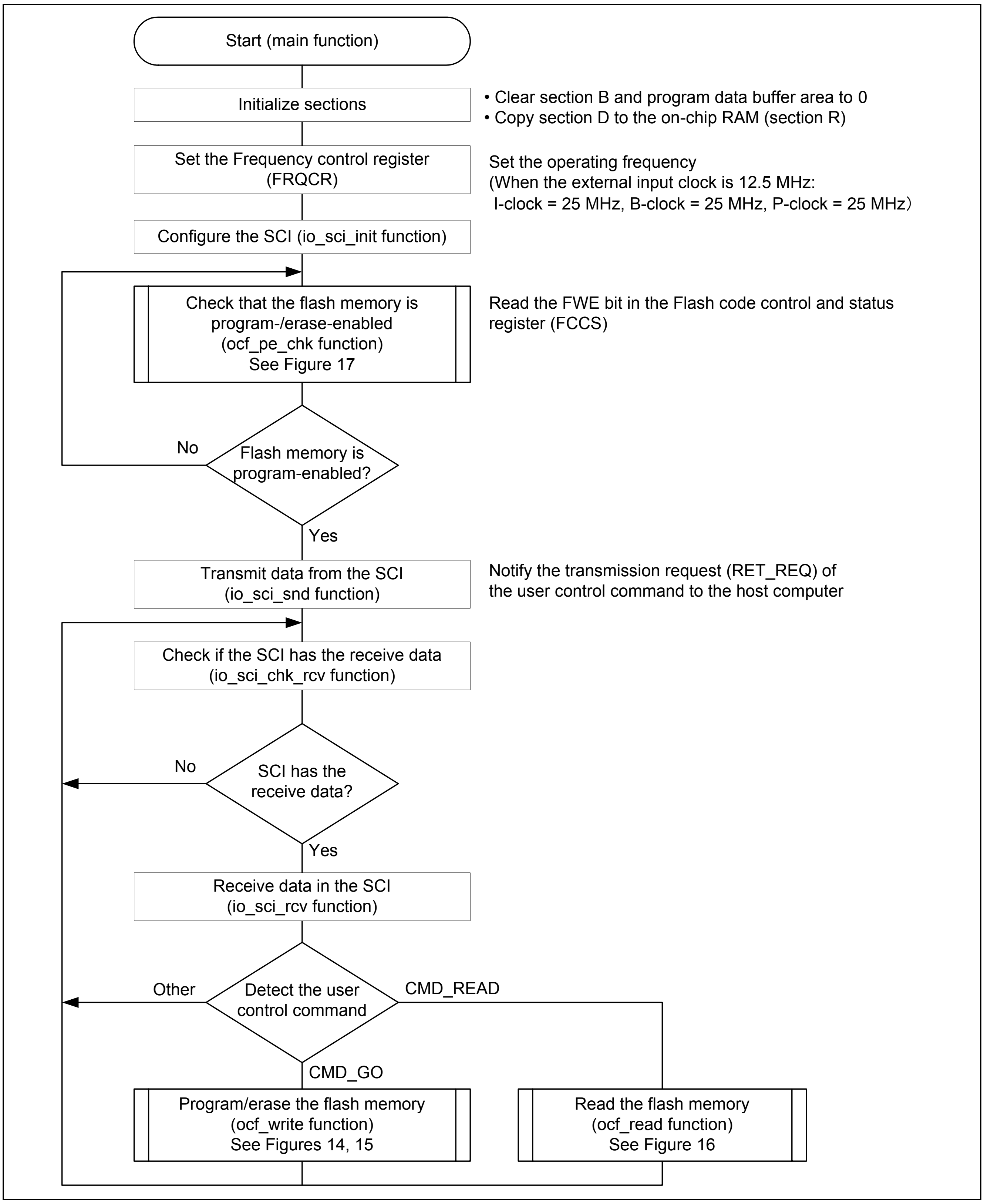
Table 7 Register Settings in the Sample Program

Register Name	Address	Setting	Description
Frequency control register (FRQCR)	H'FFFE 0010	H'0333	• STC [2:0] = "B'011": Bus clock division ratio = 1/4 • IFC [2:0] = "B'011": Internal clock division ratio = 1/4 • PFC [2:0] = "B'011": Peripheral clock division ratio = 1/4
Standby control register 5 (STBCR5)	H'FFFE 0418	H'7F	MSTP57 = "0": SCI0 is operating
Serial mode register_0 (SCSMR_0)	H'FFFF 8000	H'00	• C/A# = "0": Asynchronous mode • CHR = "0": 8-bit data • PE = "0": Disables to add and check the parity bit • STOP = "0" 1 stop bit • MP = "0": Disables the multiprocessor mode • CKS [1:0] = "B'00": Peripheral clock
Bit rate register_0 (SCBRR_0)	H'FFFF 8002	D'80	Bit rate = 9600 bps (Peripheral clock = 25 MHz)
Serial control register_0 (SCSCR_0)	H'FFFF 8004	H'30	• TE = "1": Enables the transmitter • RE = "1": Enables the receiver
Port A control register L1 (PACRL1)	H'FFFE 3816	H'0055	• PA1MD [2:0] = "B'101": Outputs TXD0 (SCI0) • PA0MD [2:0] = "B'101": Inputs RXD0 (SCI0)

4.4 Flow Charts

This section describes the flow charts of the sample program.

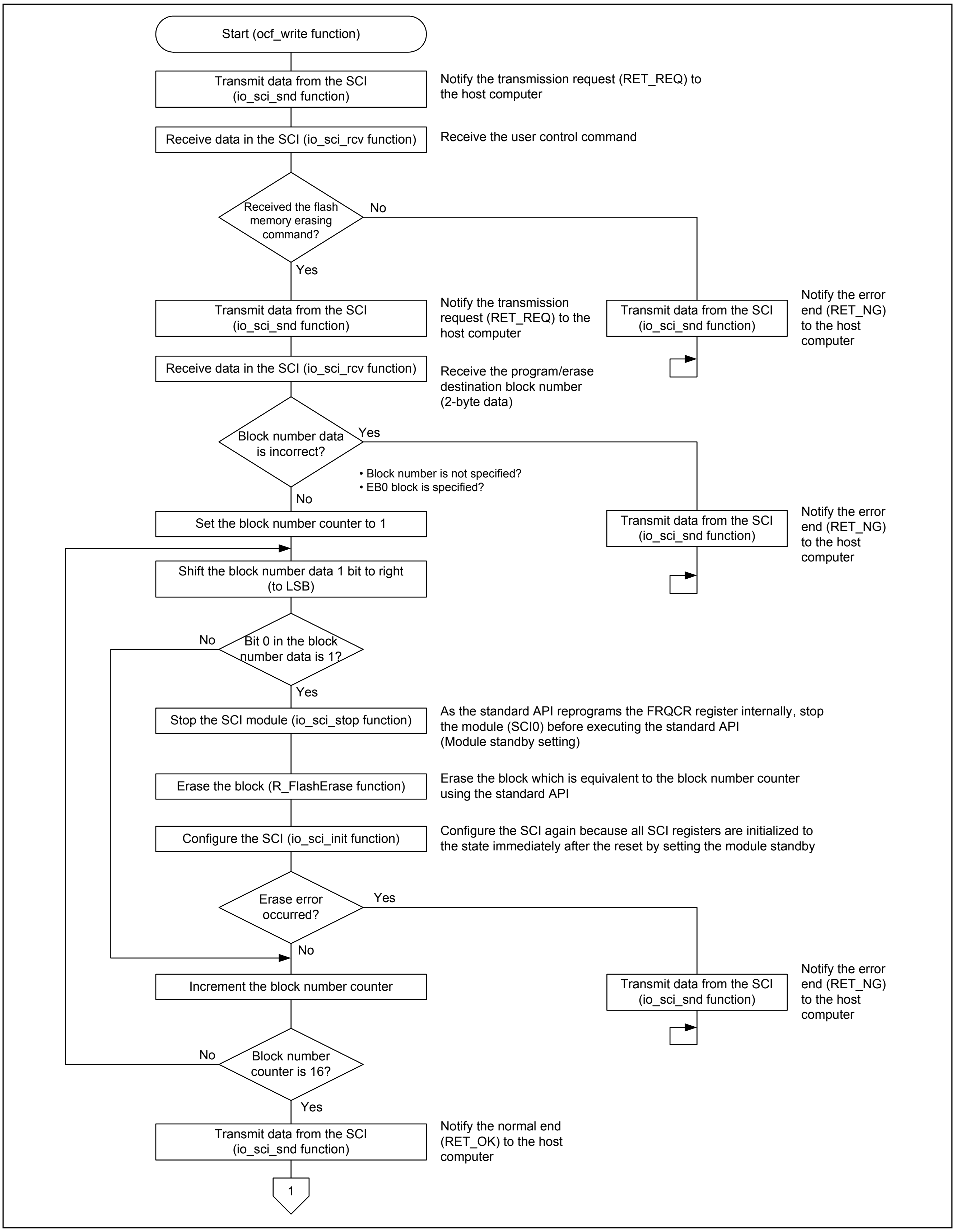
4.4.1 Main Flow Chart



Start (main function)
Initialize sections
• Clear section B and program data buffer area to 0
• Copy section D to the on-chip RAM (section R)
Set the Frequency control register (FRQCR)
Set the operating frequency (When the external input clock is 12.5 MHz: I-clock = 25 MHz, B-clock = 25 MHz, P-clock = 25 MHz)
Configure the SCI (io_sci_init function)
Check that the flash memory is program-/erase-enabled (ocf_pe_chk function) See Figure 17
Read the FWE bit in the Flash code control and status register (FCCS)
Flash memory is program-enabled?
No
Yes
Transmit data from the SCI (io_sci_snd function)
Notify the transmission request (RET_REQ) of the user control command to the host computer
Check if the SCI has the receive data (io_sci_chk_rcv function)
SCI has the receive data?
No
Yes
Receive data in the SCI (io_sci_rcv function)
Detect the user control command
Other
CMD_READ
CMD_GO
Program/erase the flash memory (ocf_write function) See Figures 14, 15
Read the flash memory (ocf_read function) See Figure 16

Figure 13 Main Processing Flow Chart

4.4.2 Programming/Erasing the Flash Memory



Start (ocf_write function)
Transmit data from the SCI (io_sci_snd function)
Notify the transmission request (RET_REQ) to the host computer
Receive data in the SCI (io_sci_rcv function)
Receive the user control command
Received the flash memory erasing command?
No
Yes
Transmit data from the SCI (io_sci_snd function)
Notify the transmission request (RET_REQ) to the host computer
Transmit data from the SCI (io_sci_snd function)
Notify the error end (RET_NG) to the host computer
Receive data in the SCI (io_sci_rcv function)
Receive the program/erase destination block number (2-byte data)
Block number data is incorrect?
Yes
• Block number is not specified?
• EB0 block is specified?
No
Set the block number counter to 1
Transmit data from the SCI (io_sci_snd function)
Notify the error end (RET_NG) to the host computer
Shift the block number data 1 bit to right (to LSB)
No
Bit 0 in the block number data is 1?
Yes
Stop the SCI module (io_sci_stop function)
As the standard API reprograms the FRQCR register internally, stop the module (SCI0) before executing the standard API (Module standby setting)
Erase the block (R_FlashErase function)
Erase the block which is equivalent to the block number counter using the standard API
Configure the SCI (io_sci_init function)
Configure the SCI again because all SCI registers are initialized to the state immediately after the reset by setting the module standby
Erase error occurred?
Yes
No
Increment the block number counter
Transmit data from the SCI (io_sci_snd function)
Notify the error end (RET_NG) to the host computer
No
Block number counter is 16?
Yes
Transmit data from the SCI (io_sci_snd function)
Notify the normal end (RET_OK) to the host computer
1

Figure 14 Programming/Erasing the Flash Memory (1/2)

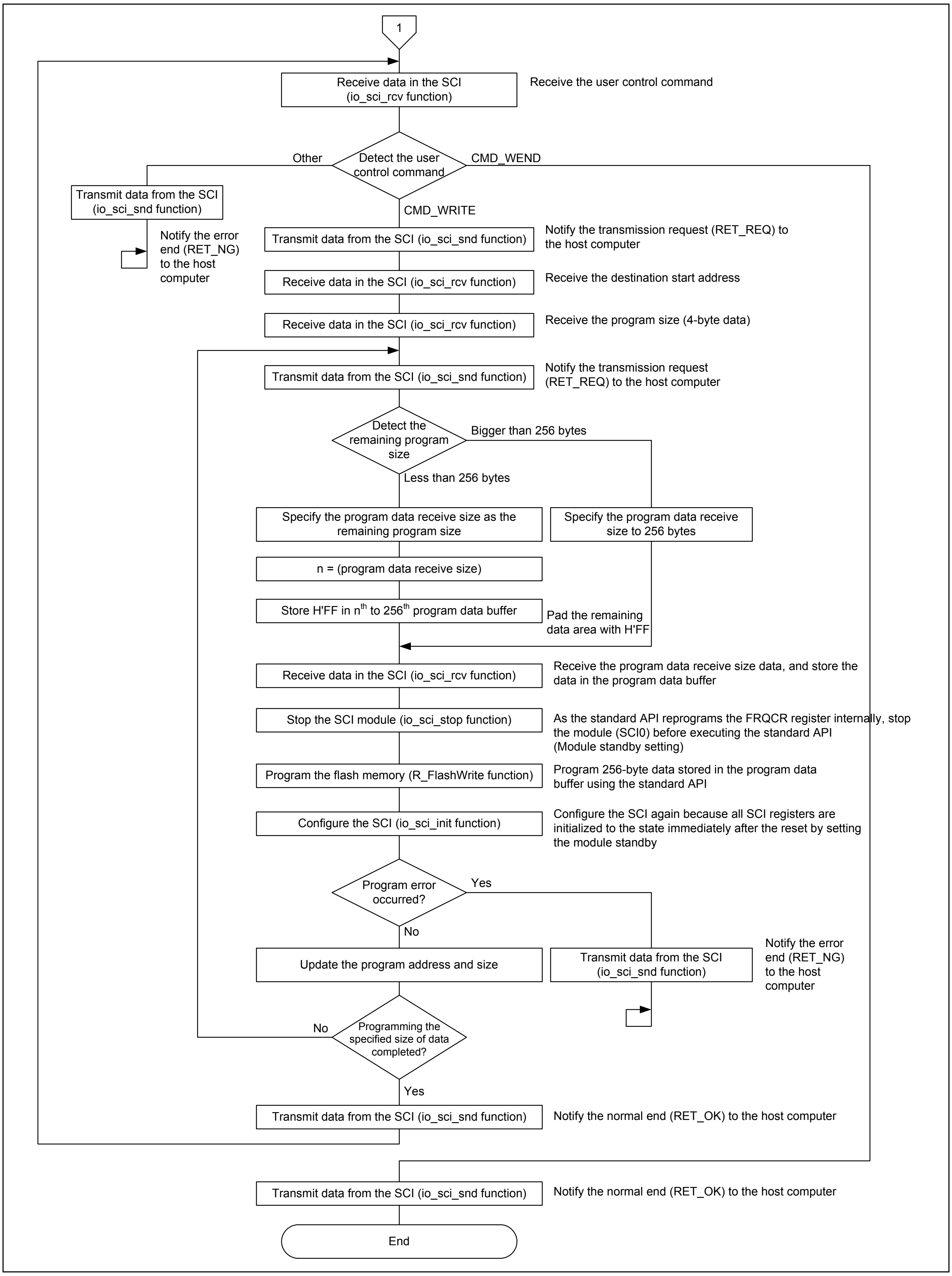

1
Receive data in the SCI (io_sci_rcv function)
Receive the user control command
Detect the user control command
Other
CMD_WEND
CMD_WRITE
Transmit data from the SCI (io_sci_snd function)
Notify the error end (RET_NG) to the host computer
Transmit data from the SCI (io_sci_snd function)
Notify the transmission request (RET_REQ) to the host computer
Receive data in the SCI (io_sci_rcv function)
Receive the destination start address
Receive data in the SCI (io_sci_rcv function)
Receive the program size (4-byte data)
Transmit data from the SCI (io_sci_snd function)
Notify the transmission request (RET_REQ) to the host computer
Detect the remaining program size
Bigger than 256 bytes
Less than 256 bytes
Specify the program data receive size as the remaining program size
Specify the program data receive size to 256 bytes
n = (program data receive size)
Store H'FF in nth to 256th program data buffer
Pad the remaining data area with H'FF
Receive data in the SCI (io_sci_rcv function)
Receive the program data receive size data, and store the data in the program data buffer
Stop the SCI module (io_sci_stop function)
As the standard API reprograms the FRQCR register internally, stop the module (SCI0) before executing the standard API (Module standby setting)
Program the flash memory (R_FlashWrite function)
Program 256-byte data stored in the program data buffer using the standard API
Configure the SCI (io_sci_init function)
Configure the SCI again because all SCI registers are initialized to the state immediately after the reset by setting the module standby
Program error occurred?
Yes
No
Update the program address and size
Transmit data from the SCI (io_sci_snd function)
Notify the error end (RET_NG) to the host computer
Programming the specified size of data completed?
No
Yes
Transmit data from the SCI (io_sci_snd function)
Notify the normal end (RET_OK) to the host computer
Transmit data from the SCI (io_sci_snd function)
Notify the normal end (RET_OK) to the host computer
End

Figure 15 Programming/Erasing the Flash Memory (2/2)

4.4.3 Reading the Flash Memory

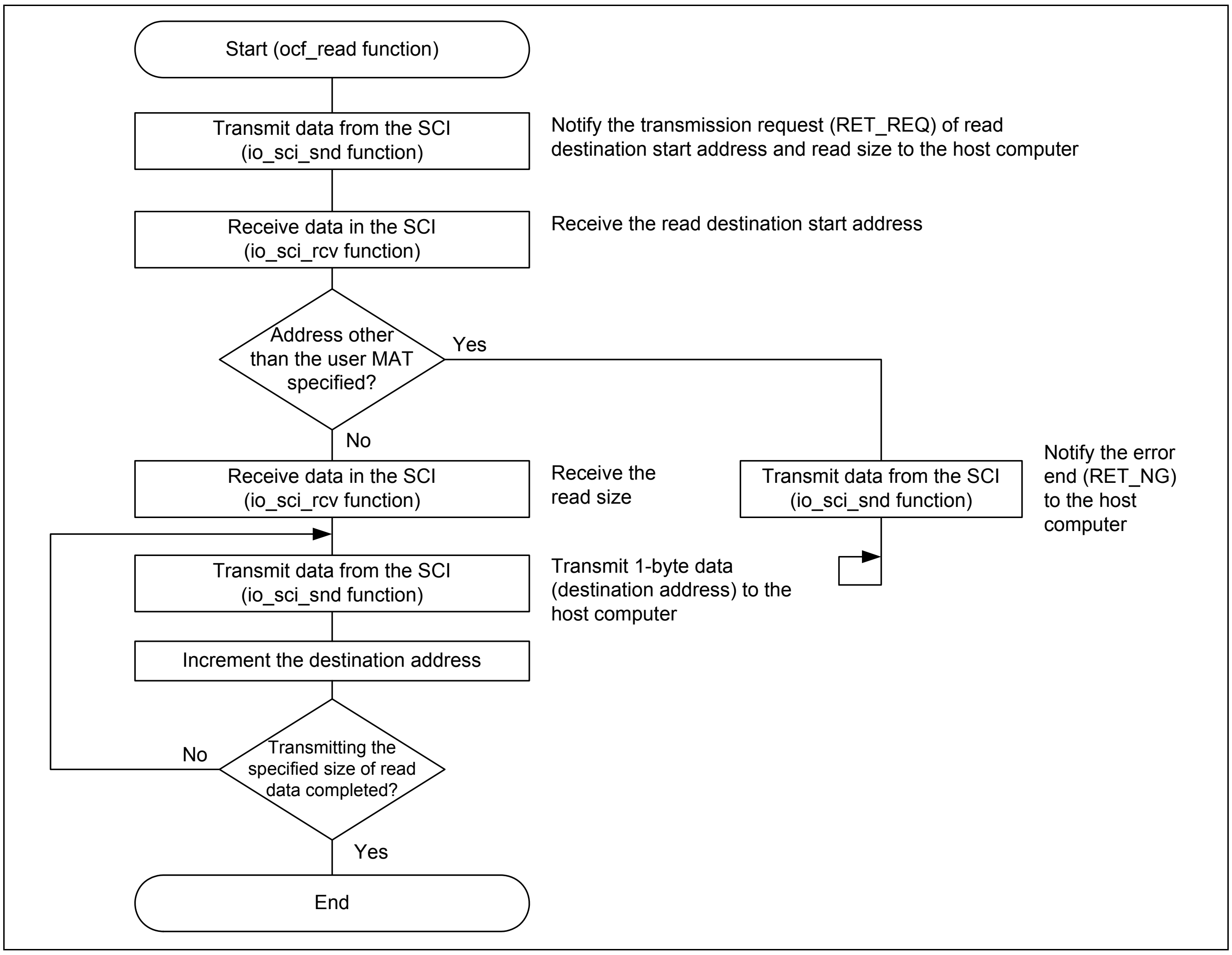

Start (ocf_read function)
Transmit data from the SCI (io_sci_snd function)
Notify the transmission request (RET_REQ) of read destination start address and read size to the host computer
Receive data in the SCI (io_sci_rcv function)
Receive the read destination start address
Address other than the user MAT specified?
Yes
No
Receive data in the SCI (io_sci_rcv function)
Receive the read size
Transmit data from the SCI (io_sci_snd function)
Notify the error end (RET_NG) to the host computer
Transmit data from the SCI (io_sci_snd function)
Transmit 1-byte data (destination address) to the host computer
Increment the destination address
Transmitting the specified size of read data completed?
No
Yes
End

Figure 16 Reading the Flash Memory

4.4.4 Checking the Flash Memory is Program-/Erase-enabled

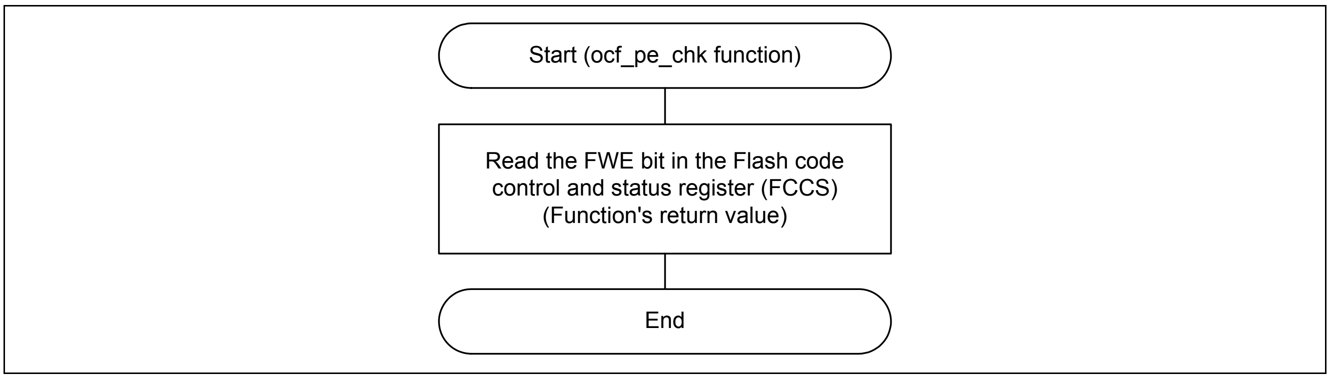

Start (ocf_pe_chk function)
Read the FWE bit in the Flash code control and status register (FCCS) (Function's return value)
End

Figure 17 Checking the Flash Memory is Program-/Erase-enabled

5. Sample Program Listing

5.1 Sample Program Listing "main.c" (1/10)

```
/*******************************************************************************
*   DISCLAIMER
*
*   This software is supplied by Renesas Electronics Corp. and is only
*   intended for use with Renesas products.  No other uses are authorized.
*
*   This software is owned by Renesas Electronics Corp. and is protected under
*   all applicable laws, including copyright laws.
*
*   THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
*   REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
*   INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
*   PARTICULAR PURPOSE AND NON-INFRINGEMENT.  ALL SUCH WARRANTIES ARE EXPRESSLY
*   DISCLAIMED.
*
*   TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
*   ELECTRONICS CORP. NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
*   FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
*   FOR ANY REASON RELATED TO THIS SOFTWARE, EVEN IF RENESAS OR ITS
*   AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
*
*   Renesas reserves the right, without notice, to make changes to this
*   software and to discontinue the availability of this software.
*   By using this software, you agree to the additional terms and
*   conditions found by accessing the following link:
*   http://www.renesas.com/disclaimer
********************************************************************************
*   Copyright (C) 2010 Renesas Electronics Corporation. All rights reserved.
*******************************************************************************/
/*""FILE COMMENT""*********** Technical reference data ************************
*   System Name : SH7286 Sample Program
*   File Name   : main.c
*   Abstract    : Using user program mode
*   Version     : 1.00.00
*   Device      : SH7286
*   Tool-Chain  : High-performance Embedded Workshop (Ver.4.04.01).
*               : C/C++ compiler package for the SuperH RISC engine family
*               :                             (Ver.9.01 Release01).
*   OS          : None
*   H/W Platform: M3A-HS87 (CPU board)
*   Description :
********************************************************************************
*   History     : Sep.10,2010 Ver.1.00.00
*""FILE COMMENT END""**********************************************************/
#include <machine.h>
#include "iodefine.h"
#include "Flash_API_SH7280.h"

```

5.2 Sample Program Listing "main.c" (2/10)

```
/* ==== Macro definition ==== */
#define FLASH_PE_ENABLE       1   /* Flash program/erase enabled       */
#define FLASH_PE_DISABLE   0   /* Flash program/erase disabled      */
#define PROGRAM_SIZE      256 /* Flash programming size unit       */

#define CMD_GO        0x55    /* Flash memory programming/erasing start command */
#define CMD_READ      0xaa    /* Flash memory reading command */
#define CMD_ERASE     0x77    /* Flash memory erasing command */
#define CMD_WRITE     0x88    /* Flash memory programming command */
#define CMD_WEND      0x99    /* Flash memory programming/erasing command */
#define RET_OK        0x00    /* Normal end */
#define RET_NG        0x01    /* Error end */
#define RET_REQ       0x11    /* Transmission request */

/* ==== Prototype declaration ==== */
void main(void);              /* main function                           */
void _section_init(void);     /* section initialization function         */
void io_cpg_init(void);           /* FRQCR setting function                  */
int ocf_pe_chk(void);         /* Flash P/E check function                */
void ocf_write(void);         /* Flash program/erase processing function */
void ocf_read(void);          /* Flash reading function                  */
void dummy_f(void);               /* Dummy interrupt function                */
/* ---- External reference ---- */
extern void io_sci_init(void);
extern int io_sci_chk_rcv(void);
extern void io_sci_snd(unsigned char data);
extern void io_sci_rcv(unsigned char *data, unsigned long num);
extern void io_sci_stop(void);

/* ==== Global variable ==== */
#pragma section WriteDATA /* Program data buffer area */
unsigned char WriteBuff[PROGRAM_SIZE];
#pragma section

```

5.3 Sample Program Listing "main.c" (3/10)

```
/*""FUNC COMMENT""**************************************************************
 * ID           :
 * Outline      : Sample program main
 *------------------------------------------------------------------------------
 * Include      :
 *------------------------------------------------------------------------------
 * Declaration  : void main(void);
 *------------------------------------------------------------------------------
 * Description  :
 *------------------------------------------------------------------------------
 * Argument     : void
 *------------------------------------------------------------------------------
 * Return Value : void
 *------------------------------------------------------------------------------
 * Note         : None
 *""FUNC COMMENT END""**********************************************************/
void main(void)
{
  unsigned char RcvData;
  int pe_ok;

  /* ==== Initializes sections ==== */
  _section_init();

  /* ==== Initializes the FRQCR ==== */
  io_cpg_init();

  /* ==== Configures the SCI ==== */
  io_sci_init();

  /* ==== Checks the flash memory is program-/erase-enabled ==== */
  do{
    pe_ok = ocf_pe_chk();      /* FWE pin = High ? */
  }while(pe_ok != FLASH_PE_ENABLE);

  /* ==== Notifies the transmission request to the host computer ==== */
  io_sci_snd(RET_REQ);

  /* ==== Programs/erases the flash memory or reads the flash memory ==== */
  while(1){
    /* ---- Checks the user control command ---- */
    if(io_sci_chk_rcv() != 0){
        io_sci_rcv(&RcvData, 1);
        if(RcvData == CMD_GO){
            ocf_write();           /* Programs or erases the flash memory */
        }
        else if(RcvData == CMD_READ){
            ocf_read();            /* Reads the flash memory */
        }
    }
  }
}

```

5.4 Sample Program Listing "main.c" (4/10)

```
/*""FUNC COMMENT""**************************************************************
 * ID             :
 * Outline        : Section initialization
 *-----------------------------------------------------------------------------
 * Include        : <machine.h>
 *-----------------------------------------------------------------------------
 * Declaration    : void _section_init(void);
 *-----------------------------------------------------------------------------
 * Description    : Clears section B and program data buffer area to 0, and copies
 *                : section D to the on-chip RAM (section R).
 *-----------------------------------------------------------------------------
 * Argument       : void
 *-----------------------------------------------------------------------------
 * Return Value : void
 *-----------------------------------------------------------------------------
 * Note           : None
 *""FUNC COMMENT END""*********************************************************/
void _section_init(void)
{
  unsigned char *src, *dst, *end;

  /* Zero our all un-initialized (BSS) RAM data as specified by ANSI. */
  src = (unsigned char *)(__sectop("B"));
  end = (unsigned char *)(__secend("B"));
  while(src < end){
    *src++ = 0x00;
  }

  src = (unsigned char *)(__sectop("BWriteDATA"));
  end = (unsigned char *)(__secend("BWriteDATA"));
  while(src < end){
    *src++ = 0x00;
  }

  /* Copy in our all initialized (DATA) RAM data as specified by ANSI. */
  src = (unsigned char *)(__sectop("D"));
  dst = (unsigned char *)(__sectop("R"));
  end = (unsigned char *)(__secend("D"));
  while(src < end){
    *dst++ = *src++;
  }
}

```

5.5 Sample Program Listing "main.c" (5/10)

```
/*""FUNC COMMENT""***************************************************************
 * ID            :
 * Outline       : FRQCR initialization
 *-------------------------------------------------------------------------------
 * Include       : <machine.h> and "iodefine.h"
 *-------------------------------------------------------------------------------
 * Declaration   : void io_cpg_init(void);
 *-------------------------------------------------------------------------------
 * Description   : Initializes the FRQCR register.
 *-------------------------------------------------------------------------------
 * Argument      : void
 *-------------------------------------------------------------------------------
 * Return Value  : void
 *-------------------------------------------------------------------------------
 * Note          : None
 *""FUNC COMMENT END""**********************************************************/
void io_cpg_init(void)
{
  volatile unsigned short dummy;

  WDT.WRITE.WTCSR = 0xa51e;   /* WDT stop, WDT count clock: 1/4096 x P-clock */
                             /* (Overflow cycle = 41.94 ms at P-clock 25 MHz) */
  WDT.WRITE.WTCNT = 0x5ac2;   /* WDT counter for 10 ms */
  CPG.FRQCR.WORD = 0x0333;/* Clock-in = 12.5 MHz */
                             /* I-clock = 25 MHz */
                             /* B-clock = 25 MHz */
                             /* P-clock = 25 MHz */
                             /*  -> (I : B : P) = (1/4 : 1/4 : 1/4) */
  dummy = CPG.FRQCR.WORD;     /* FRQCR readout */
  /* ---- 64 NOPs for 32 x P-clock (I:P = 1:1) ---- */
  nop(); nop(); nop(); nop(); nop(); nop(); nop(); nop();
  nop(); nop(); nop(); nop(); nop(); nop(); nop(); nop();
  nop(); nop(); nop(); nop(); nop(); nop(); nop(); nop();
  nop(); nop(); nop(); nop(); nop(); nop(); nop(); nop();
  nop(); nop(); nop(); nop(); nop(); nop(); nop(); nop();
  nop(); nop(); nop(); nop(); nop(); nop(); nop(); nop();
  nop(); nop(); nop(); nop(); nop(); nop(); nop(); nop();
  nop(); nop(); nop(); nop(); nop(); nop(); nop(); nop();
}

```

5.6 Sample Program Listing "main.c" (6/10)

```
/*""FUNC COMMENT""*************************************************************
 * ID              :
 * Outline         : Flash memory program-/erase-enabled state check
 *-----------------------------------------------------------------------------
 * Include         : "iodefine.h"
 *-----------------------------------------------------------------------------
 * Declaration     : int ocf_pe_chk(void);
 *-----------------------------------------------------------------------------
 * Description     : Reads the FWE bit in the Flash code control and status register
 *                 : (FCCS) and returns the value.
 *-----------------------------------------------------------------------------
 * Argument        : void
 *-----------------------------------------------------------------------------
 * Return Value : 0 ; Flash memory is program-/erase-disabled
 *                 : 1 ; Flash memory is program-/erase-enabled
 *-----------------------------------------------------------------------------
 * Note            : None
 *""FUNC COMMENT END""*********************************************************/
int ocf_pe_chk(void)
{
  return FLASH.FCCS.BIT.FWE;
}

/*""FUNC COMMENT""*************************************************************
 * ID              :
 * Outline         : Programming/erasing the flash memory
 *-----------------------------------------------------------------------------
 * Include         : "Flash_API_SH7280.h"
 *-----------------------------------------------------------------------------
 * Declaration     : void ocf_write(void);
 *-----------------------------------------------------------------------------
 * Description     : Erases the program/erase destination block (other than EB0)
 *                 : which is specified by the host computer, and programs the
 *                 : specified size of data from the destination start address.
 *-----------------------------------------------------------------------------
 * Argument        : void
 *-----------------------------------------------------------------------------
 * Return Value : void
 *-----------------------------------------------------------------------------
 * Note            : None
 *""FUNC COMMENT END""*********************************************************/
void ocf_write(void)
{
  unsigned char error;      /* Function return value */
  unsigned char RcvData;        /* Receive data */
  unsigned char EraseBlkNum;    /* Erase block number */
  unsigned short EraseBlkSelect;  /* Specified erase block number by bit field */
  unsigned long WriteAddr;/* Start address to be programmed */
  unsigned long WriteSize;/* Data size to be programmed */
  unsigned long RcvSize;        /* Receiving size for data to be programmed */
  unsigned long i;          /* Loop counter */

```

5.7 Sample Program Listing "main.c" (7/10)

```
    /* ==== Transmission request ==== */
    io_sci_snd(RET_REQ);

    /* ==== Receives the flash memory erasing command ==== */
    io_sci_rcv(&RcvData, 1);
    if(RcvData != CMD_ERASE){    /* Received the command other than the CMD_ERASE? */
      io_sci_snd(RET_NG);        /* Error end */
      while(1){
      }
    }

    /* ==== Transmission request ==== */
    io_sci_snd(RET_REQ);

    /* ==== Receives the erase block number data ==== */
    io_sci_rcv((unsigned char *)&EraseBlkSelect, 2);
    if( (EraseBlkSelect == 0x0000) ||
      ((EraseBlkSelect & 0x0001) != 0) ){
      /* Block number is not specified or EB0 is specified? */
      io_sci_snd(RET_NG);            /* Error end */
      while(1){
      }
    }

    /* ==== Erases the flash memory ==== */
    EraseBlkNum = BLOCK_1;
    do{
      EraseBlkSelect >>= 1;
      if((EraseBlkSelect & 0x0001) != 0){
          /* ---- Sets the SCI in module standby ---- */
          io_sci_stop();
          /* ---- Erases a block ---- */
          error = R_FlashErase((uint8_t)EraseBlkNum);
          /* ---- Configures the SCI ---- */
          io_sci_init();
          if(error != 0){            /* Erase error occurred? */
              io_sci_snd(RET_NG);        /* Error end */
              while(1){
              }
          }
      }
    } while(EraseBlkNum++ <= BLOCK_15);

    io_sci_snd(RET_OK);              /* Normal end */


```

5.8 Sample Program Listing "main.c" (8/10)

```
    /* ==== Programs the flash memory ==== */
    while(1){
      io_sci_rcv(&RcvData, 1);      /* Receives the user control command */
      if(RcvData == CMD_WRITE){     /* Received the CMD_WRITE? */
          io_sci_snd(RET_REQ);      /* Transmission request */
      }
      else if(RcvData == CMD_WEND){ /* Received the CMD_WEND? */
          io_sci_snd(RET_OK);           /* Normal end */
          break;
      }
      else{
          io_sci_snd(RET_NG);           /* Error end */
          while(1){
          }
      }

      /* ---- Receives the destination start address ---- */
      io_sci_rcv((unsigned char *)&WriteAddr, 4);

      /* ---- Receives the program data size ---- */
      io_sci_rcv((unsigned char *)&WriteSize, 4);

      while(WriteSize > 0){
          io_sci_snd(RET_REQ);          /* Transmission request */

          if(WriteSize > PROGRAM_SIZE){
              RcvSize = PROGRAM_SIZE;
          }
          else{
              RcvSize = WriteSize;
              for(i = RcvSize; i < PROGRAM_SIZE; i++){
                  WriteBuff[i] = 0xff;
              }
          }

```

5.9 Sample Program Listing "main.c" (9/10)

```
            /* ---- Receives the program data ---- */
            io_sci_rcv(WriteBuff, RcvSize);   /* Stores the data in the program data buffer */

            /* ---- Sets the SCI in module standby ---- */
            io_sci_stop();
            /* ---- Programs the flash memory ---- */
            error = R_FlashWrite((uint32_t)WriteAddr, (uint32_t)WriteBuff, PROGRAM_SIZE);
            /* ---- Configures the SCI ---- */
            io_sci_init();
            if(error != 0){               /* Program error occurred? */
                io_sci_snd(error);        /* Error end */
                while(1){
                }
            }

            WriteAddr += PROGRAM_SIZE;
            WriteSize -= RcvSize;
        }
        io_sci_snd(RET_OK);                   /* Normal end */
    }

}

/*""FUNC COMMENT""***************************************************************
 * ID            :
 * Outline       : Reading the flash memory
 *-------------------------------------------------------------------------------
 * Include       :
 *-------------------------------------------------------------------------------
 * Declaration   : void ocf_read(void);
 *-------------------------------------------------------------------------------
 * Description   : Reads the specified size of data from the read destination
 *               : start address and transmits the data to the host computer.
 *-------------------------------------------------------------------------------
 * Argument      : void
 *-------------------------------------------------------------------------------
 * Return Value : void
 *-------------------------------------------------------------------------------
 * Note          : None
 *""FUNC COMMENT END""***********************************************************/
void ocf_read(void)
{
  unsigned char *ReadData;/* Pointer for readout data */
  unsigned long ReadAddr;     /* Start address to be read */
  unsigned long ReadSize;     /* Reading size */
  unsigned long i;        /* Loop counter */

  /* ==== Transmission request ==== */
  io_sci_snd(RET_REQ);

```

5.10 Sample Program Listing "main.c" (10/10)

```
    /* ==== Receives the read destination start address ==== */
    io_sci_rcv((unsigned char *)&ReadAddr, 4);
    if(ReadAddr >= 0x00100000){
      /* Specified the address other than the user MAT? */
      io_sci_snd(RET_NG);           /* Error end */
      while(1){
      }
    }

    /* ==== Receives the read data size ==== */
    io_sci_rcv((unsigned char *)&ReadSize, 4);

    /* ==== Transmits the data which is read from ROM ==== */
    ReadData = (unsigned char *)ReadAddr;
    for(i = 0; i < ReadSize; i++){
      io_sci_snd(*ReadData++);
    }
  }

  /*""FUNC COMMENT""*************************************************************
   * ID           :
   * Outline      : Interrupt handling (dummy function).
   *-----------------------------------------------------------------------------
   * Include      :
   *-----------------------------------------------------------------------------
   * Declaration  : void dummy_f(void);
   *-----------------------------------------------------------------------------
   * Description  :
   *-----------------------------------------------------------------------------
   * Argument     : void
   *-----------------------------------------------------------------------------
   * Return Value : void
   *-----------------------------------------------------------------------------
   * Note         : None
   *""FUNC COMMENT END""*********************************************************/
  #pragma interrupt dummy_f
  void dummy_f(void)
  {
    while(1){
      /* Infinite loop */
    }
  }

  /* End of File */
```

6. References

- Software Manual
  SH-2A/SH2A-FPU Software Manual Rev. 3.00
  The latest version of the software manual can be downloaded from the Renesas Electronics website.

- Hardware Manual
  SH7280 Group Hardware Manual Rev. 2.00
  The latest version of the hardware manual can be downloaded from the Renesas Electronics website.

- Technical update
  Prohibition of Interrupts during Programming/Erasing in User Program Mode (TN-SH7-A657A/E)

Website and Support

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

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

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Nov.30.10	–	First edition issued

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this manual, refer to the relevant sections of the manual. If the descriptions under General Precautions in the Handling of MPU/MCU Products and in the body of the manual differ from each other, the description in the body of the manual takes precedence.

1. Handling of Unused Pins
   Handle unused pins in accord with the directions given under Handling of Unused Pins in the manual.
   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.
2. Processing at Power-on
   The state of the product is undefined at the moment when power is supplied.
   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.
3. Prohibition of Access to Reserved Addresses
   Access to reserved addresses is prohibited.
   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.
4. Clock Signals
   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.
   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.
5. Differences between Products
   Before changing from one product to another, i.e. to one with a different type number, confirm that the change will not lead to problems.
   — The characteristics of MPU/MCU in the same group but having different type numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different type numbers, implement a system-evaluation test for each of the products.

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
7F, No. 363 Fu Shing North Road Taipei, Taiwan, R.O.C.
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
1 harbourFront Avenue, #06-10, keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 1.0