APPLICATION NOTE

SH7268/SH7269 Group

Boot From the Serial Flash Memory

Using SPI Multi I/O Bus Controller

R01AN0663EJ0101
Rev. 1.01
Feb 16, 2012

Summary

SH7268/SH7269 SPI multi I/O bus controller (SPIBSC) has the function to directly fetch the program data on a serial flash memory and execute them (external address space read mode). This application note offers explanations

Target Device

SH7268/SH7269 MCU (In this document, SH7268/SH7269 are described as "SH7269".)

When using this application note with other Renesas MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.

Contents

1. Introduction

1.1 Specifications

Boot mode 3 allows booting the SH7269 from its serial flash memory (herein after called serial flash boot). The serial flash boot progresses and executes the loader programs in the high-speed internal RAM. At this time the external address space read mode for the SPI multi I/O bus controller (SPIBSC) is invalid. This mode is enabled by the loader program.

This application note describes about the loader program and application program examples using the serial flash boot as well as about the downloader to write the loader program and application to serial flash memory.

1.2 Modules Used

- Boot mode (serial flash boot)
- Renesas Serial Peripheral Interface (RSPI)
- SPI multi I/O bus controller (SPIBSC)

1.3 Applicable Conditions

MCU	SH7268/SH7269
Operating Frequency	Internal clock (Iϕ) : 266.67 MHz Internal bus clock (Bϕ) : 133.33 MHz Peripheral clock 1 (P1ϕ) : 66.67 MHz Peripheral clock 0 (P0ϕ) : 33.33 MHz
Integrated Development Environment	Renesas Electronics Corporation High-performance Embedded Workshop Ver.4.07.00
C Compiler	Renesas Electronics SuperH RISC engine Family C/C++ compiler package Ver.9.03 Release 02
Compiler Options	Default setting in the High-performance Embedded Workshop (-cpu=sh2afpu -fpu=single -object="$(CONFIGDIR)\$(FILELEAF).obj" -debug -gbr=auto -chgincpath -errorpath -global_volatile=0 -opt_range=all -infinite_loop=0 -del_vacant_loop=0 -struct_alloc=1 –nologo)
Serial Flash Memory	S25FL032P (Spansion) x 1

1.4 Related Application Note

The application note relating to this application note is introduced below. Refer to it along with this application note.

- SPI multi I/O bus controller serial flash memory connection example.

1.5 About Active-low Pins (Signals)

The symbol "#" suffixed to the pin (or signal) names indicates that the pins (or signals) are active-low.

2. Overview of the Serial Flash Boot

This chapter describes an overview of the serial flash boot.

2.1 Words and Terms

Table 1 lists the words and terms used in this application note to describe the serial flash boot.

Table 1 Terms to Describe the Serial Flash Boot

Term	Description
Internal ROM program to boot	Transfers the loader program stored in the beginning of the serial flash memory to the high-speed internal RAM, and branches to the loader program when the MCU is booted in boot mode 3. This program does not need to be created as it is already stored in the internal ROM to boot in CPU.
Loader program	Enables the application program allocated in the external address space as SPI multi I/O bus space of the SPI multi I/O bus controller (SPIBSC). Branches into the entry function of the application program retaining the external address space read mode of SPIBSC enabled. The size of the loader program is fixed to 8 KB. Create it according to the user's system.
Application program	A program that is created by user according to the user's system. In this application note, the application program is supposed to be allocated in the external address space of SPIBSC as SPI multi I/O bus space.
Downloader	A program to write the loader program and the application program to the serial flash memory. Create it according to the user's system.

2.2 The Serial Flash Boot Operation

Table 2 lists the external pins (MD_BOOT2 to MD_BOOT0) to decide the boot mode.

Table 2 Relationship between the External Pin Settings and Serial Flash Boot Mode

MD_BOOT2	MD_ BOOT1	MD_BOOT0	Boot Mode	Description
1	0	1	Boot mode 3	Boots the MCU from serial flash memory connected to Renesas Serial Peripheral Interface channel 0

In boot mode 3, an internal ROM program to boot transfers the loader program from the serial flash memory connected to Renesas Serial Peripheral Interface channel 0 (RSPI0) to the high-speed internal RAM after the power-on reset is canceled. After transferring, SH7269 branches to the beginning of the loader program. Figure 1 shows the operation image of the ROM program. A series of the processing is automatically executed.

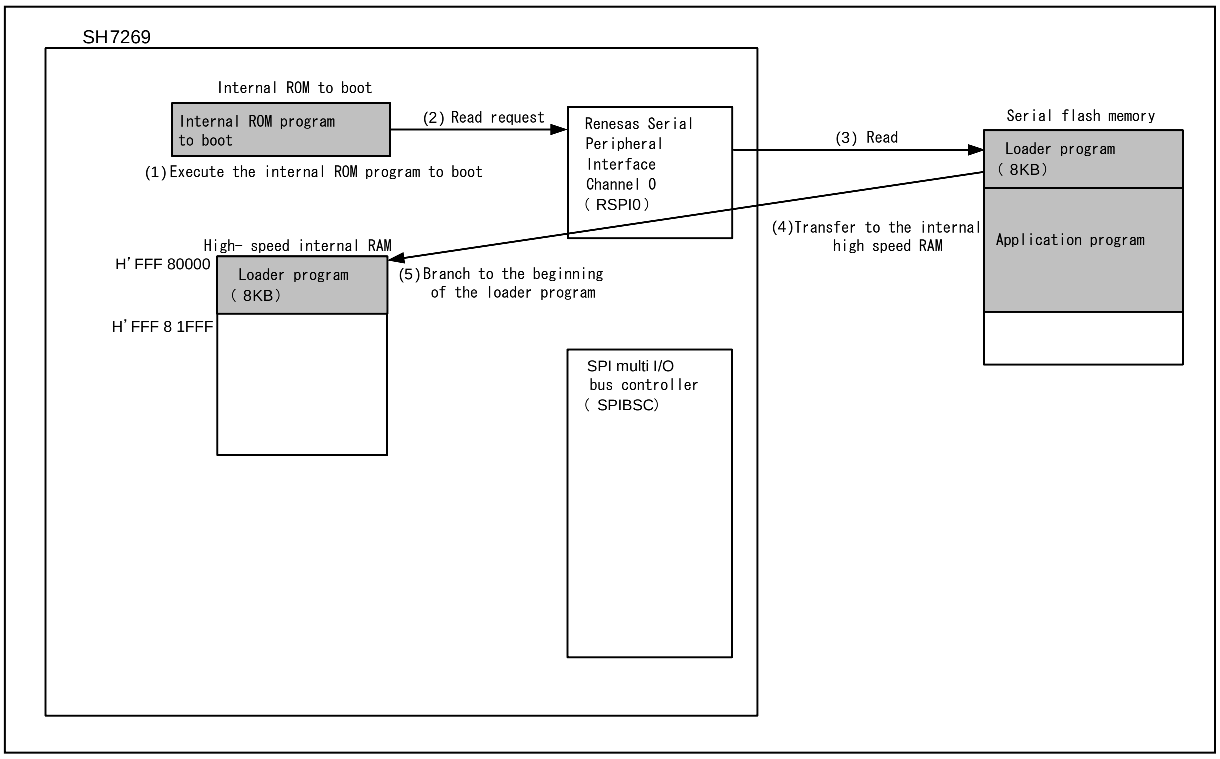

SH7269
Internal ROM to boot
Internal ROM program to boot
(1)Execute the internal ROM program to boot
(2) Read request
Renesas Serial Peripheral Interface Channel 0 (RSPI0)
(3) Read
Serial flash memory
Loader program (8KB)
Application program
(4)Transfer to the internal high speed RAM
High- speed internal RAM
H'FFF 80000
Loader program (8KB)
H'FFF 8 1FFF
(5)Branch to the beginning of the loader program
SPI multi I/O bus controller (SPIBSC)

Figure 1 Operation Image of the Internal ROM Program to Boot

The loader program enables the read mode in the external address space of SPIBSC. By this setting, the application program stored in the serial flash memory can be allocated in the external address space. After this setting, the SH7269 branches to the entry function of the application program. Figure 2 shows the operation image of the loader program.

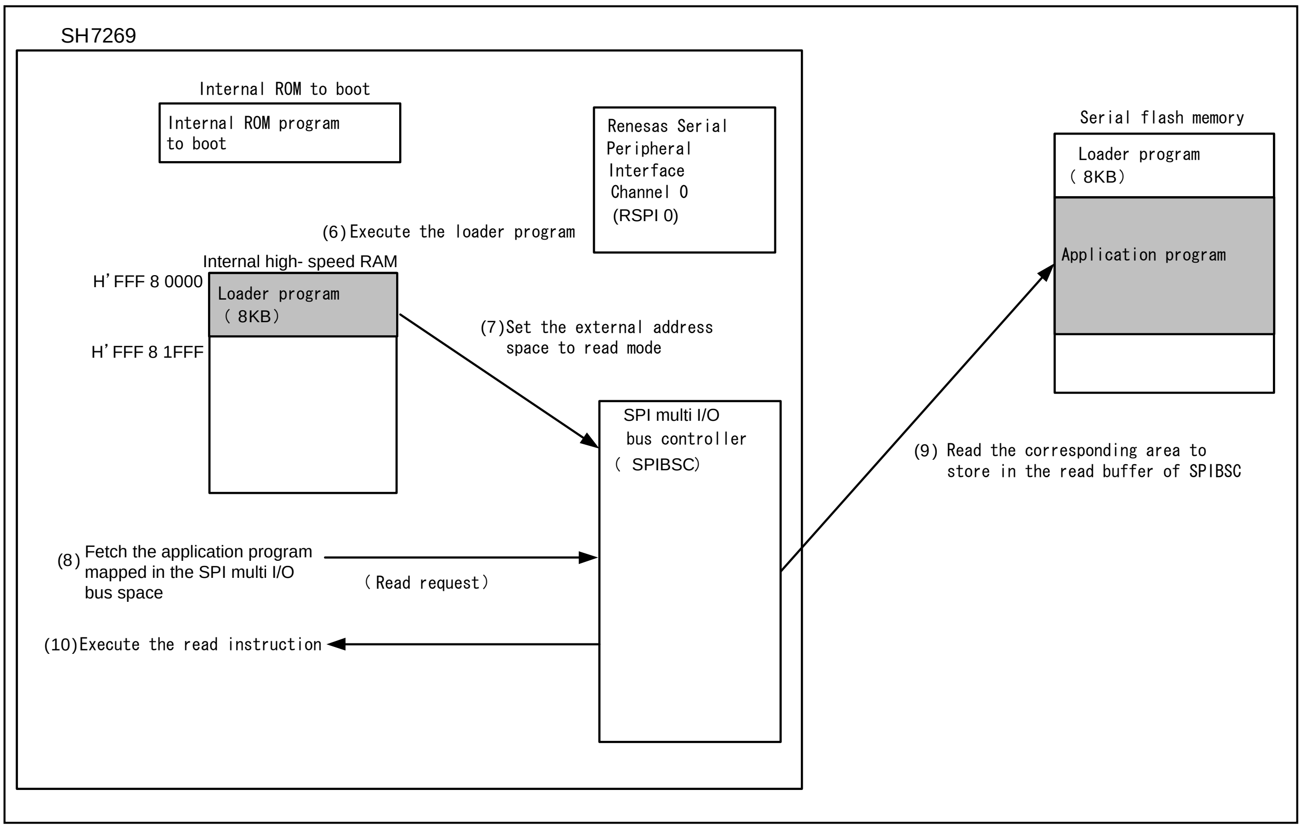

SH7269
Internal ROM to boot
Internal ROM program to boot
Renesas Serial Peripheral Interface Channel 0 (RSPI 0)
(6)Execute the loader program
Internal high- speed RAM
H'FFF 8 0000
Loader program (8KB)
H'FFF 8 1FFF
(7)Set the external address space to read mode
SPI multi I/O bus controller (SPIBSC)
(8) Fetch the application program mapped in the SPI multi I/O bus space
(Read request)
(10)Execute the read instruction
Serial flash memory
Loader program (8KB)
Application program
(9) Read the corresponding area to store in the read buffer of SPIBSC

Figure 2 Operation Image of the Loader Program

2.3 Dow nloader Operation

The downloader writes the loader program on the high-speed internal RAM and application program on RAM to the serial flash memory. Figure 3 shows the operation image of the downloader.

For more information, refer to “3.3 Downloader Example”.

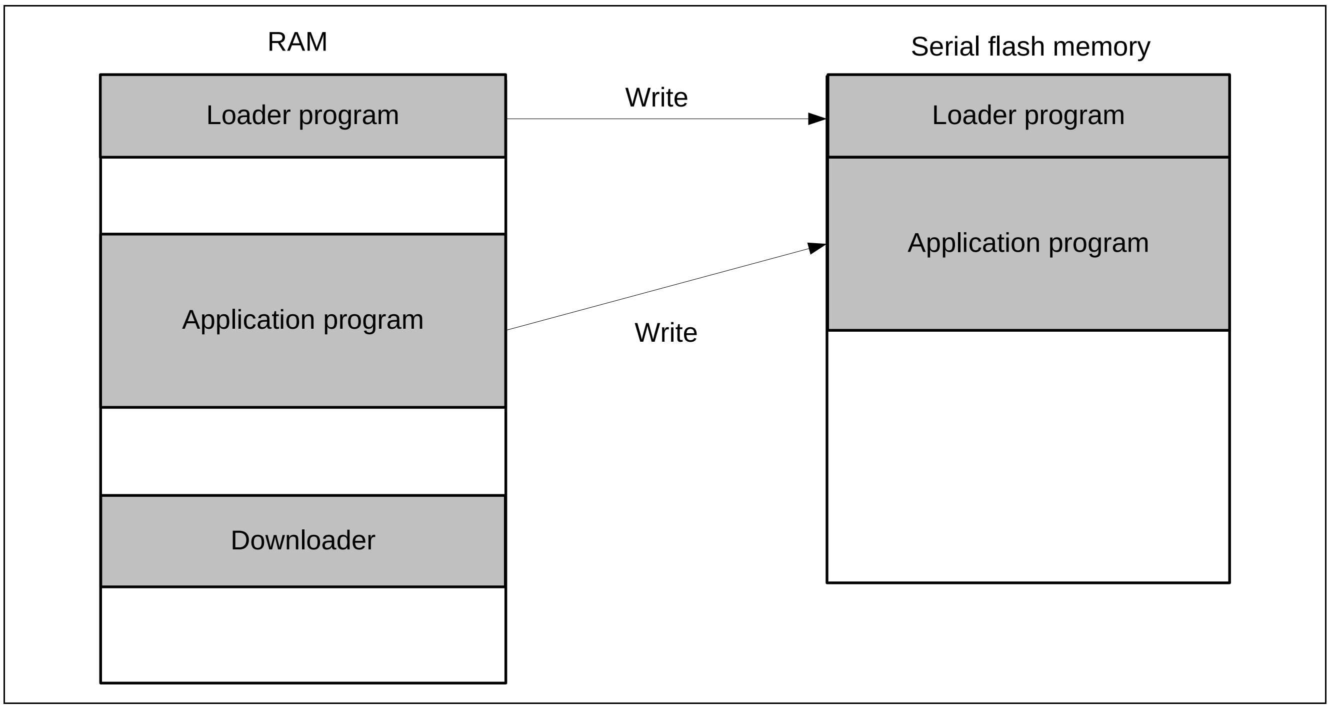


Figure 3 Operation Image of the Downloader

2.4 Serial Flash Memory Connection

Figure 4 shows an example of serial flash memory connection circuit and SPIBSC connection circuit. The internal ROM program to boot uses the Renesas Peripheral Interface channel 0 (RSPI0). Therefore the serial flash memory should be connected to RSPI0.

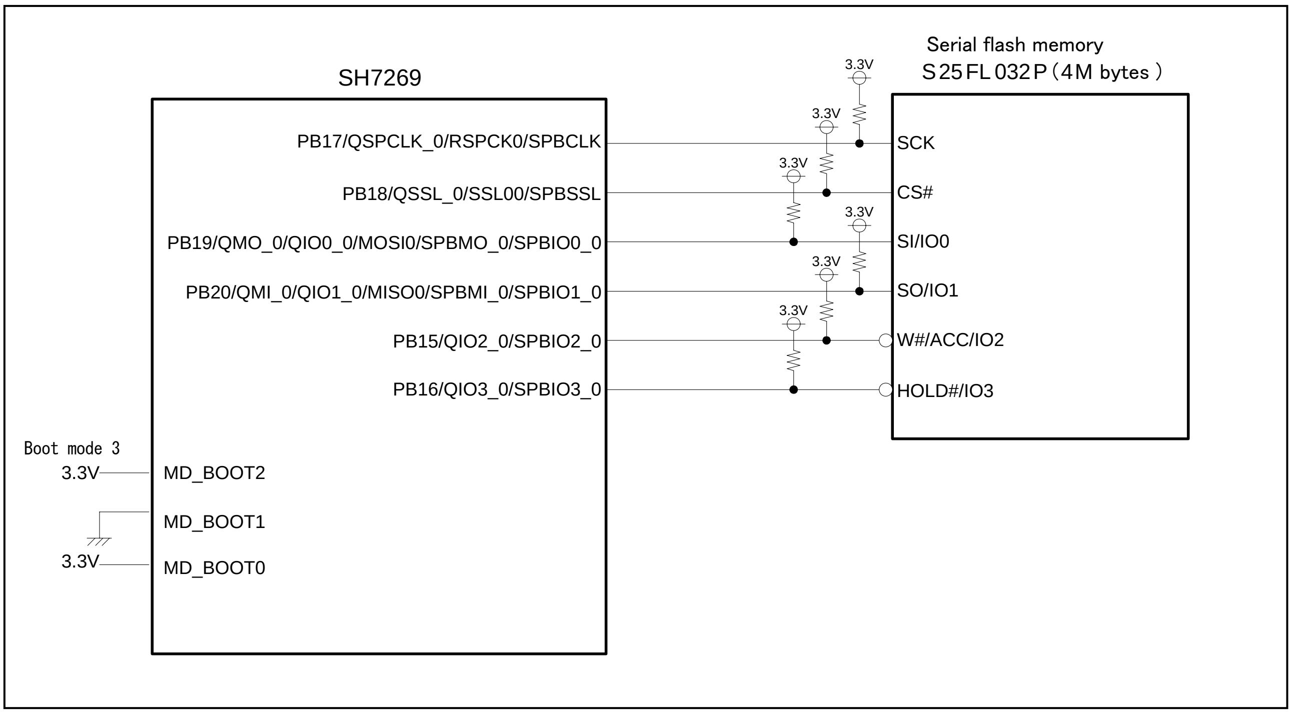


Figure 4 Serial Flash Memory Circuit

3. Applications

This chapter describes about the loader program, the application program, and the downloader.

3.1 Loader Program Specifications

The loader program enables read mode in the external address space of the SPI multi I/O bus controller (SPIBSC), and branches to the entry function in the application program.

3.1.1 Memor y Map

Figure 5 shows the memory map of the loader program.

The loader program is transferred from the serial flash memory to the high-speed external RAM by the internal ROM program to boot. The source area in the serial flash memory is for 8KB from address H’0000 0000 to H’0000 1FFF.

The destination area is from address H’FFF8 0000 to H’FFF8 1FFF in which high-speed internal RAM is allocated.

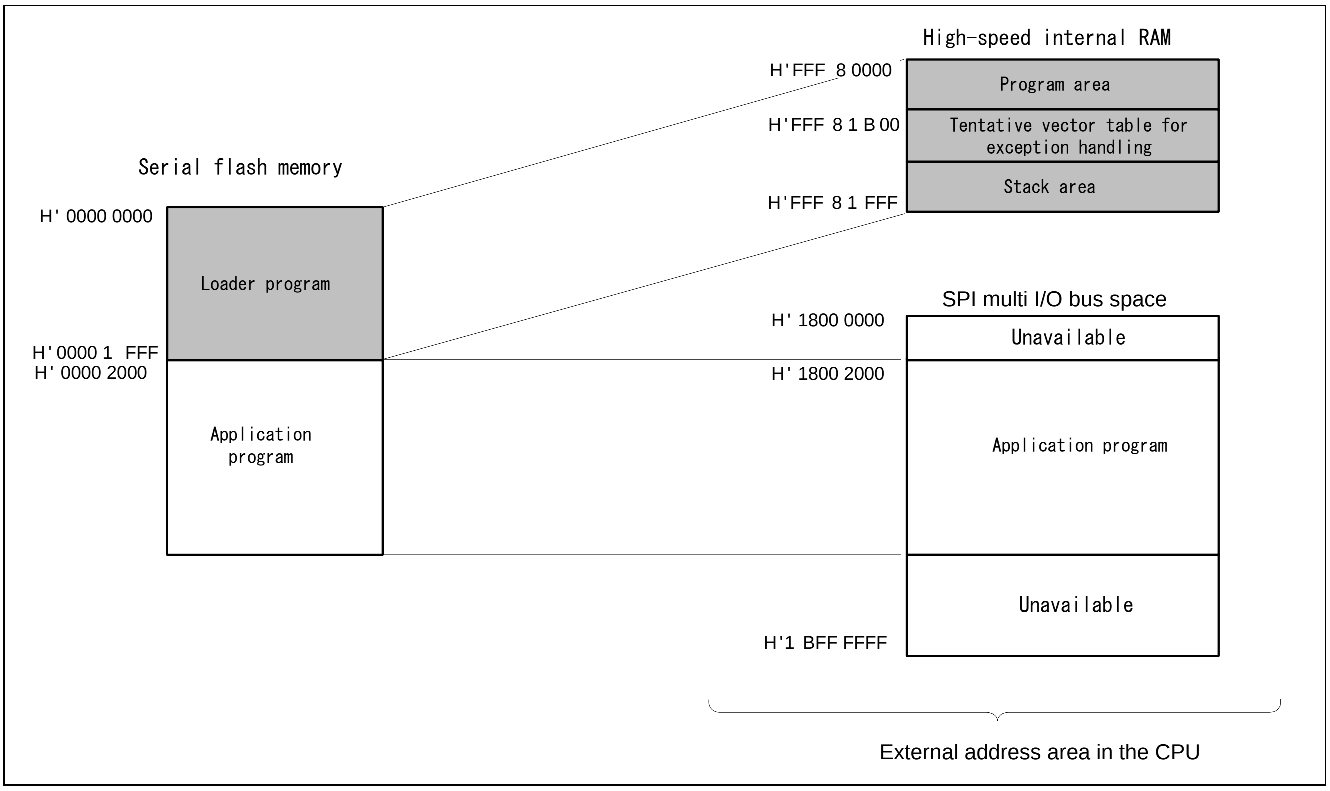


Figure 5 Loader Program Memory Map

3.1.2 Flow Chart of the Loader Program

Figure 6 shows the flow chart of the loader program. For more information, refer to sections 3.1.3 to 3.1.11.

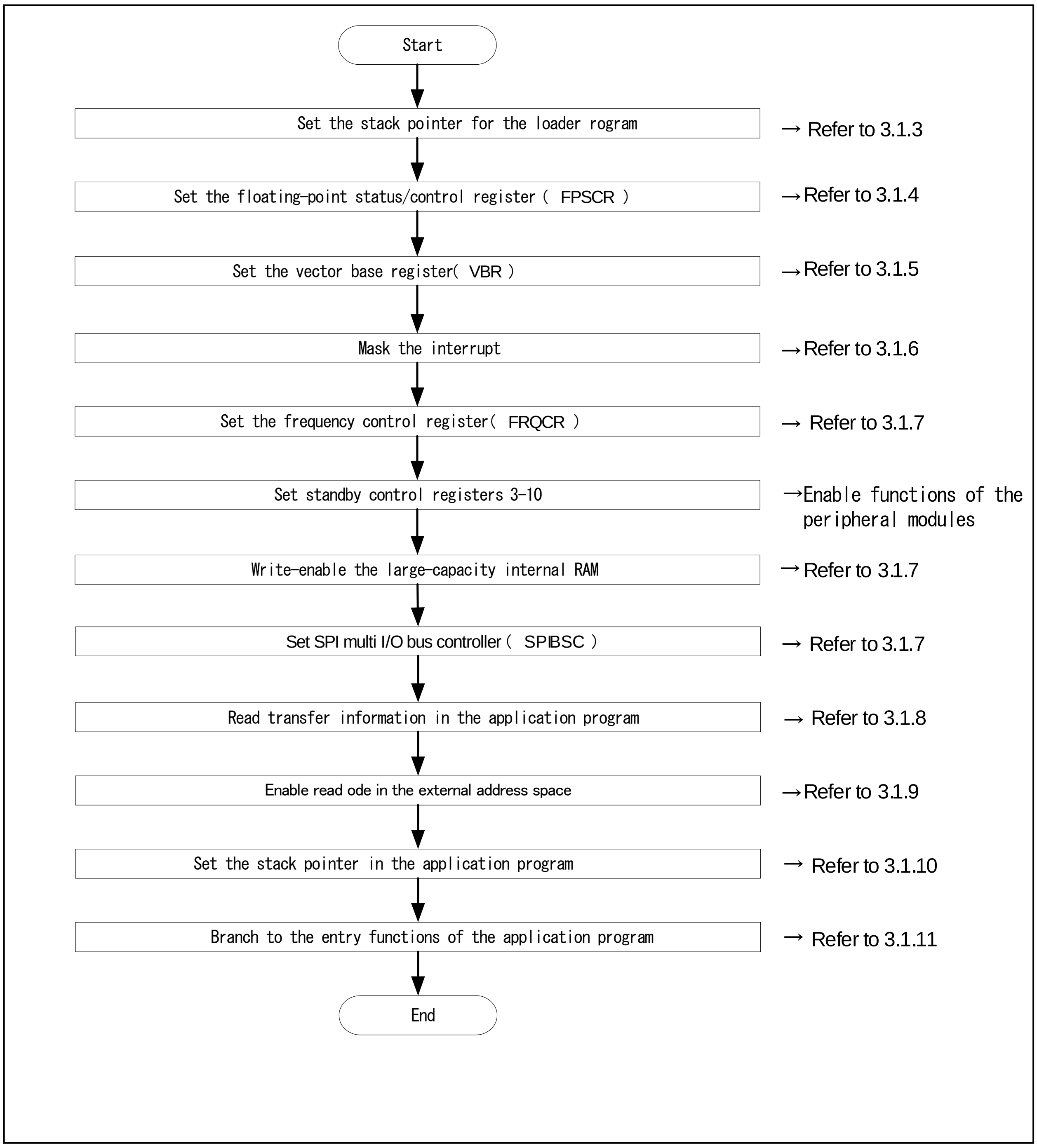

Start
Set the stack pointer for the loader rogram
→ Refer to 3.1.3
Set the floating-point status/control register (FPSCR)
→Refer to 3.1.4
Set the vector base register(VBR)
→Refer to 3.1.5
Mask the interrupt
→Refer to 3.1.6
Set the frequency control register(FRQCR)
→ Refer to 3.1.7
Set standby control registers 3-10
→Enable functions of the peripheral modules
Write-enable the large-capacity internal RAM
→Refer to 3.1.7
Set SPI multi I/O bus controller (SPIBSC)
→ Refer to 3.1.7
Read transfer information in the application program
→ Refer to 3.1.8
Enable read ode in the external address space
→Refer to 3.1.9
Set the stack pointer in the application program
→ Refer to 3.1.10
Branch to the entry functions of the application program
→ Refer to 3.1.11
End

Figure 6 Loader Program Flow Chart

3.1.3 Stack Pointer Setting

Set the address H'FFF8 2000 in the stack pointer (R15) . The loader program is allocated at the address H'FFF8 0000 in the assembly language to avoid the loader program using the undefined stack pointer. C can be used after configuring the stack pointer. Then, the loader program jumps to the entry function.

3.1.4 Floating-Point Status/Control Register (FPSCR) Setting

In the FPSCR, H'0004 0001 is set (single-precision operation, round to zero).

3.1.5 Vector Base Register (VBR) Setting

The loader program sets the tentative exception vector table in VBR to support the exceptional operation during the loader program is running. The exceptions or interrupts should not be generated before setting the VBR as the exception vector table is undefined., Only vector numbers 0 to 18 are defined in the tentative exception vector table as the loader program does not use interrupts. To embed exceptional operations such as an external interrupt during the loader program is operating, the tentative exception vector table need to be extended.

Note: Before executing exceptions, the exception vector table should be stored in the memory to allow the CPU to access the memory. For more information, refer to 6.9.4 "Note before Exception Handling Begins Running" in the SH7268 Group, SH7269 Group User's Manual for Hardware.

3.1.6 Interrupt Mask

B'1111 is specified in the interrupt mask level bit of the Status register (SR) as the loader program does not support interrupts in operation.

3.1.7 Initial Configuration

Initial configuration is necessary in the peripheral functions to read the application program from the serial flash memory.

3.1.8 Reading Transfer Information of Application Programs

The loader program refers to the transfer information (appinfo) of application program in the serial flash memory to obtain the external address in which the application program is allocated. Table 3 lists the detailed transfer information (appinfo) of application program .It is allocated at the address H'0000 2000 in the serial flash memory. The loader program handles the information in the address H'0000 2000 to H'0000 2007 in the serial flash memory as the transfer information of application program.

To access to the serial flash memory, SPI mode of SPIBSC is adopted.

Table 3 Transfer Information (appinfo) of Application Program

Item	Address	Size
Start address in transfer destination	H'0000 2000	4
End address in transfer destination	H'0000 2004	4

3.1.9 Enabling Read Mode in External Address Space

Read mode should be enabled in SPIBSC external address space to allocate application program in the SPI multi I/O bus space.

3.1.10 Application Program Stack Pointer Setting

The loader program specifies the value stored in the first 12 to 15 bytes in the application program in the stack pointer (R15).

3.1.11 Application Program Jump to the Entry Function Address

The loader program jumps to the entry function address stored in the first 8 to 11 bytes in the application program.

3.1.12 Register State after Executing Loader Program

Table 4 lists the each register state after executing the loader program. The unlisted registers are set the same value as described in SH7268 Group, SH7269 Group User's Manual for Hardware.

Table 4 Resister State after Executing Loader Programs (1)

Register Abbr	eviated	Setting value	Remarks
General register	R0 to R14	Indefinite	
Program counter	PC	Depends on the setting	Entry function address of the application program
Stack pointer	SP(R15)	Depends on the setting	Stack pointer setting value of the application program
Status register	SR	Indefinite	IMASK bit is B'1111
Vector base register	VBR	H'FFF8 1B00	
Floating-point status/ Control register	FPSCR	H'0004 0001	Single precision operation Round mode : to 0
Frequency control register	FRQCR	H'1015	
Standby control register 3	STBCR3	H'1A	
Standby control register 4	STBCR4	H'00	
Standby control register 5	STBCR5	H'00	
Standby control register 6	STBCR6	H'00	
Standby control register 7	STBCR7	H'12	
Standby control register 8	STBCR8	H'09	
System control register 5	SYSCR5	H'0F	Enables writing in large - capacity internal RAM
Common control register	CMNCR	H'00FF F320	
Bit rate setting register	SPBCR	H'0000 0100	
Data read control register	DRCR	H'0001 0101	
Data read command register	DRCMR	H'00EB 0000	
Data read enable setting register	DRENR	H'0222 47E0	
SPI mode control register	SMCR	H'0000 0004	
SPI mode command register	SMCMR	H'006B 0000	

Table 5 Resister State after Executing Loader Programs (2)

Register Abbr	eviated	Setting Value	Remarks
SPI mode address setting register	SMADR	H'0002 0004	
SPI mode option setting register	SMOPR	H'0003 0000	
SPI mode inable setting register	SMENR	H'0002 000F	
Common status register	CMNSR	H'0000 0003	
Port B control register 5	PBCR5	H'0006	
Port B control register 4	PBCR4	H'6666	
Port B control register 3	PBCR3	H'6000	

3.2 Application Program Example

Application program should be allocated in the SPI multi I/O bus space as it reads in external address space read mode. Also note that the application program must include the address information that is referred to by the loader program.

This section explains the procedure to create an application program special for serial flash boot.

3.2.1 Section Alignment

The section alignment in the application program is explained in this section.

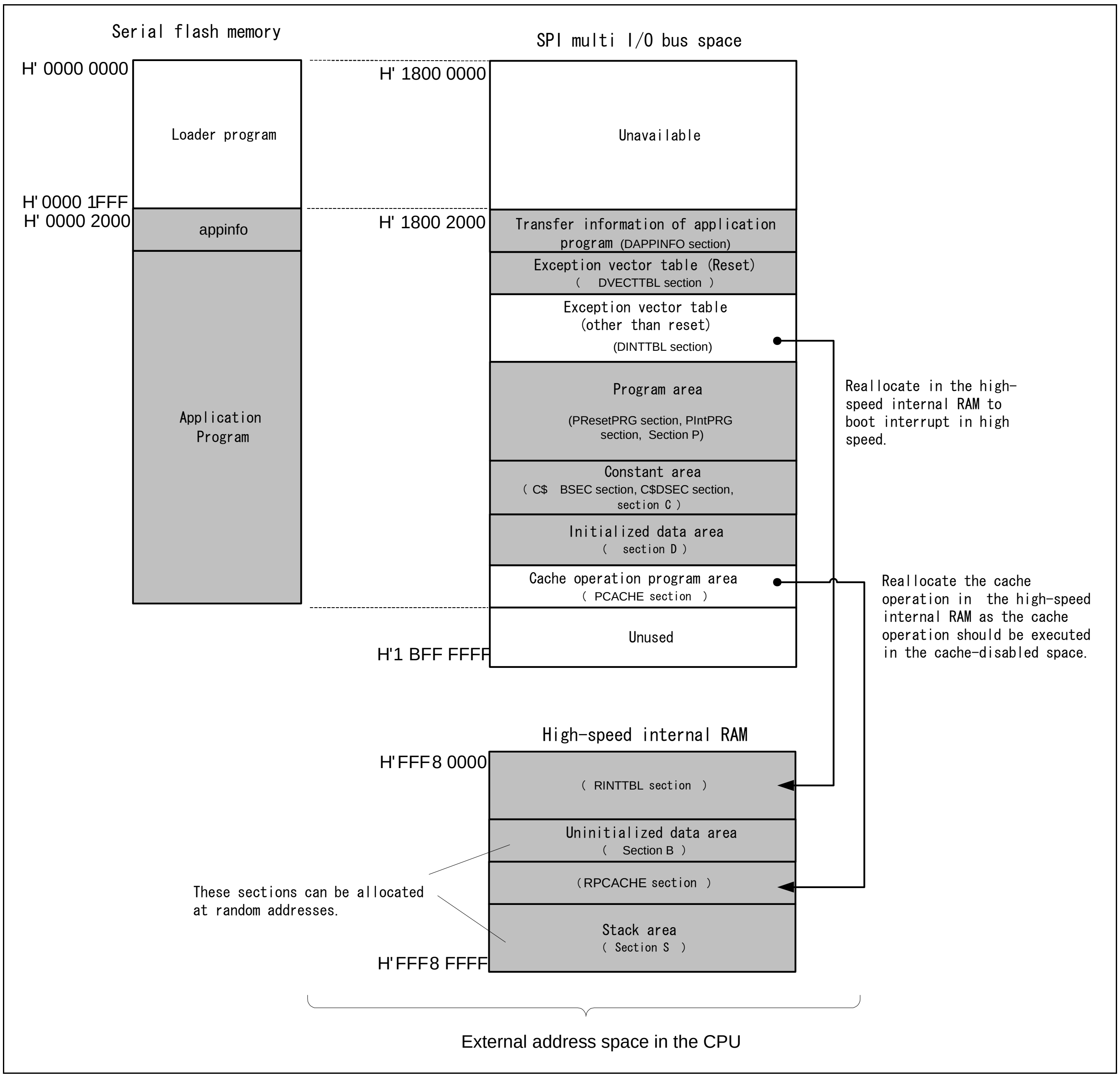
1. Application program is executed using external address space read mode. Therefore in this application program example, the section of the application program is allocated in the SPI multi I/O bus space.

2. The application program transfer information (appinfo) referred to by loader program, the entry function address of the application program and the setting value of stack pointer should be aligned the section in the fixed address. The transfer information of application program (appinfo) should be aligned in DAPPINF section and the entry function address of the application program in DVECTTBL section.. DAPPINFO section and DVECTTBL section should be allocated in turn from the beginning of the application program.

3. The area in the serial flash memory corresponding to H'1800 0000 to H'1800 1FFF in the SPI multi I/O bus space is used by the loader program. The program area of the application program, defined area, initialized data area should be allocated later than H'1800 2000 .

4. A reset vector table RESET_Vectors should be located in the start address of DVECTTBL section.

Figure 7 shows an example of the section alignment.



Serial flash memory
H' 0000 0000
Loader program
H' 0000 1FFF
H' 0000 2000
appinfo
Application Program
SPI multi I/O bus space
H' 1800 0000
Unavailable
H' 1800 2000
Transfer information of application program (DAPPINFO section)
Exception vector table (Reset) (DVECTTBL section)
Exception vector table (other than reset) (DINTTBL section)
Program area (PResetPRG section, PIntPRG section, Section P)
Constant area (C$ BSEC section, C$DSEC section, section C)
Initialized data area (section D)
Cache operation program area (PCACHE section)
Unused
H'1 BFF FFFF
Reallocate in the high-speed internal RAM to boot interrupt in high speed.
Reallocate the cache operation in the high-speed internal RAM as the cache operation should be executed in the cache-disabled space.
High-speed internal RAM
H'FFF8 0000
(RINTTBL section)
Uninitialized data area (Section B)
(RPCACHE section)
Stack area (Section S)
H'FFF8 FFFF
These sections can be allocated at random addresses.
External address space in the CPU

Figure 7 Application Program Section Alignment

3.2.2 Flo w Chart

The application program in this application transmits character strings to channel 2 in the Serial Communication Interface with FIFO (SCIF2), then sets switching on LED using the channel 0 in compare match timer.

Figure 8 shows the flow chart of the application program.

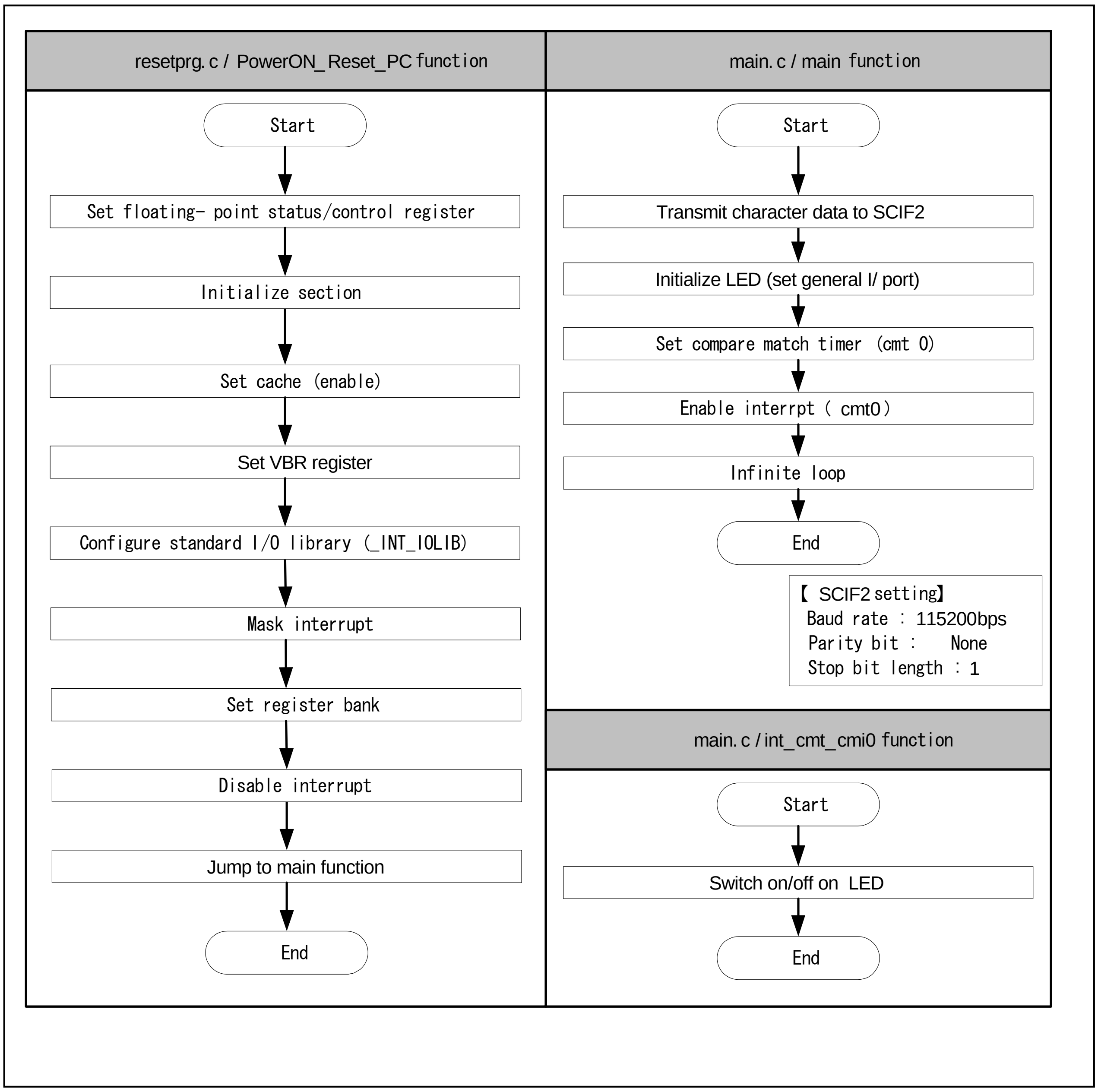


Figure 8 Application Program Flow Chart

3.2.3 Entry Function Setting

The entry function address of the application program is set to table number 0 of the reset vector table RESET_Vectors. Table 6 lists the entry function settings.

Table 6 Entry Function Address Settings

Item	Description
File Name	vecttbl.c
Placement section name	DVECTTBL
Table name	RESET_Vectors
Table number	0
Default	PowerON_Reset_PC

Note: PowerON_Reset_PC is an entry function of the application program.

3.2.4 Stack Pointer Setting

The stack pointer of the application program is set to table number 1 of the reset vector table RESET_Vectors. Table 7 lists the setting.

Table 7 Stack Pointer Settings

Item	Description
File name	vecttbl.c
Placement section name	DVECTTBL
Table name	RESET_Vectors
Table number	1
Default _se	cend ("S")

3.2.5 Section Initialization

The section is initialized by executing the section initialization routine (INITSCT function) using values stored in section initialization tables (DTBL and BTBL) described in the file dbsct.c. After the execution, write-back operation of the cache to guarantee the coherency between the cache memory and the large-capacity internal RAM.

3.2.6 Cache Setting (Enable)

Setting cache control register 1 validates the instruction cache and the operand cache..

3.2.7 Vector Base Register (VBR) Setting

The vector table of exceptional operation of the application program is set in VBR.

3.2.8 Generating the Application Program Transfer Information (appinfo)

Table 8 describes about the structure to generate the transfer information (appinfo) of application program. The beginning and the end address of the application program is obtained by section address operators (sectop, secend). The following structure is allocated in section DAPPINFO. The start address of the application program (program area, constant area, and initialized data area) should be registered in the app_top, and the end address of the application program in the app_end.

Table 8 Application Program Transfer Information (appinfo)

Item	**Description**		
File name	appinfo.c		
Structure name	appinfo		
Structure member	**Member Name**	**Value**	**Description**
	void *app_top	_sectop ("DAPPINFO")	Start address of the application program
	void *app_end	_secend ("PCACHE")	End address of the application program +1
The section to place	DAPPINFO		

Note: The amount of size of the loader program (8 KB) and application program must not exceed the capacity of the serial flash memory.

Figure 9 shows the generation image of transfer information of application program (appinfo).

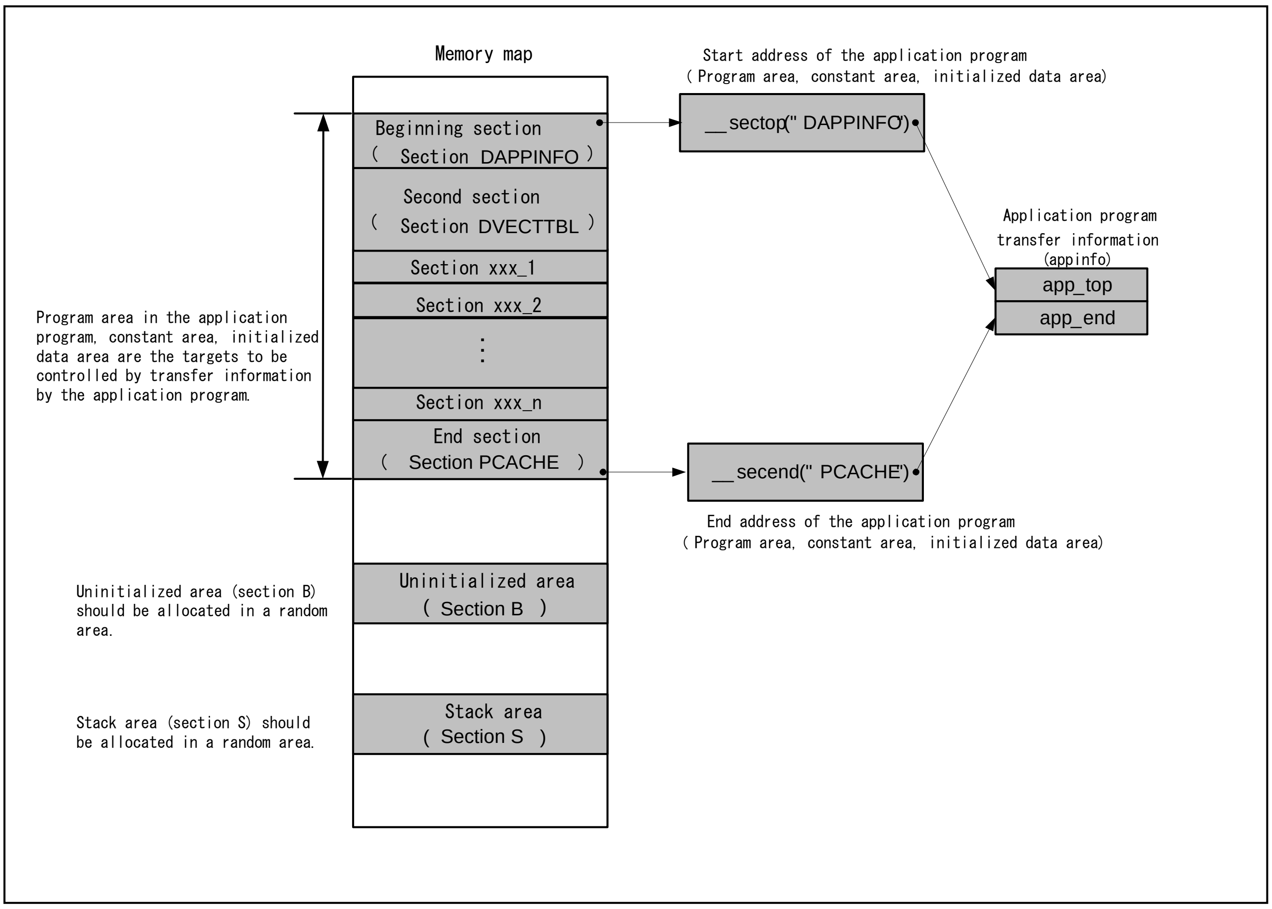

Memory map
Beginning section (Section DAPPINFO)
Second section (Section DVECTTBL)
Section xxx_1
Section xxx_2
:
Section xxx_n
End section (Section PCACHE)
Uninitialized area (Section B)
Stack area (Section S)
Program area in the application program, constant area, initialized data area are the targets to be controlled by transfer information by the application program.
Uninitialized area (section B) should be allocated in a random area.
Stack area (section S) should be allocated in a random area.
Start address of the application program (Program area, constant area, initialized data area)
__sectop(" DAPPINFO")
__secend(" PCACHE")
End address of the application program (Program area, constant area, initialized data area)
Application program transfer information (appinfo)
app_top
app_end

Figure 9 Generation image of Transfer Information (appinfo) of Application Program

3.3 Dow nloader Example

This section describes the downloader in this application.

3.3.1 Opera tion Overview

Before executing downloader, the downloader and the loader program should be transferred from the development environment to the high-speed internal RAM on system, and the application program to the large-capacity internal RAM by using the debugger. Figure 10 shows an operation image of the downloader.

The application program is allocated in SPI multi I/O bus space area, but the debugger cannot transfer programs to SPI multi I/O bus space. So only the debug information of abs file in the SPI multi I/O bus area while mot file is downloaded in the large-capacity RAM.

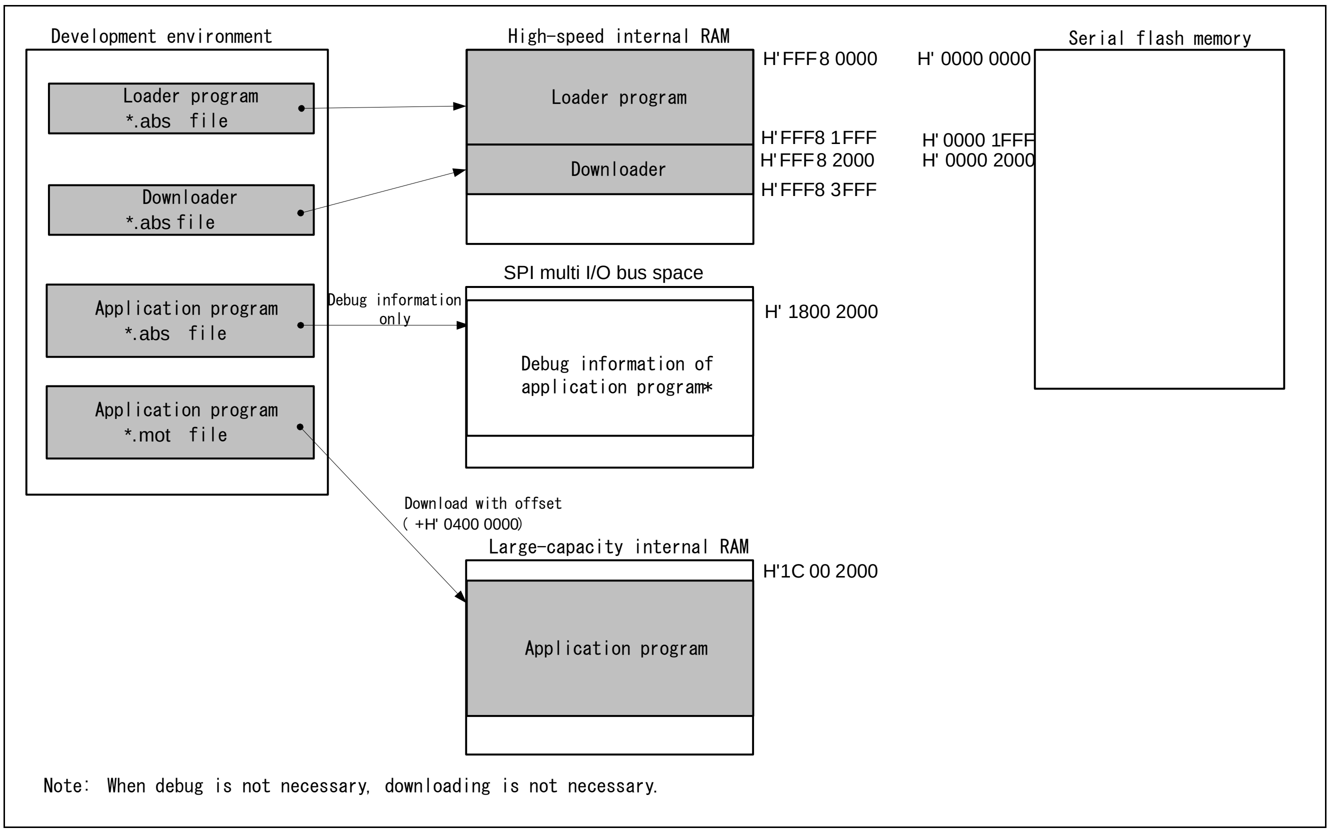


Figure 10 Downloader Operation Image (1)

The loader program and the application program are written in the serial flash memory by executing downloader. The downloader writes the loader program from H'0000 0000 to H'0000 1FFF address in the serial flash memory, and the application program from H'0000 2000.

Figure 11 shows another operation image of the downloader.

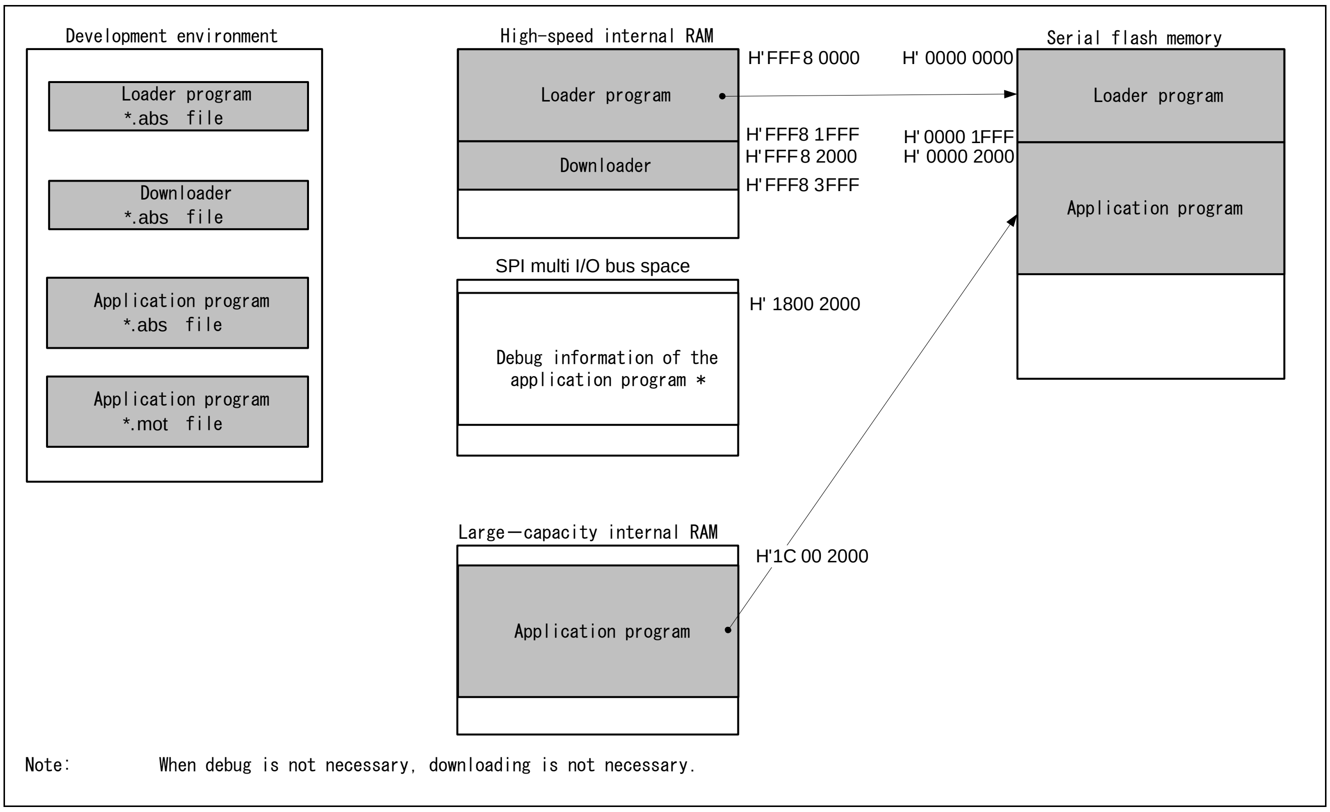


Figure 11 Downloader Operation Image

3.3.2 Areas Used by the Downloader

The downloader occupies the addresses from H'FFF8 2000 to H'FFF8 3FFF. When the loader program, application program and downloader occupy the same section, the programs do not operate properly.

3.3.3 Flow Chart

Figure 12 shows the flow chart of the downloader. Executing the downloader placed in the high-speed internal RAM enables writing in the serial flash memory. For more information, refer to the sections 3.3.4 to 3.3.8.

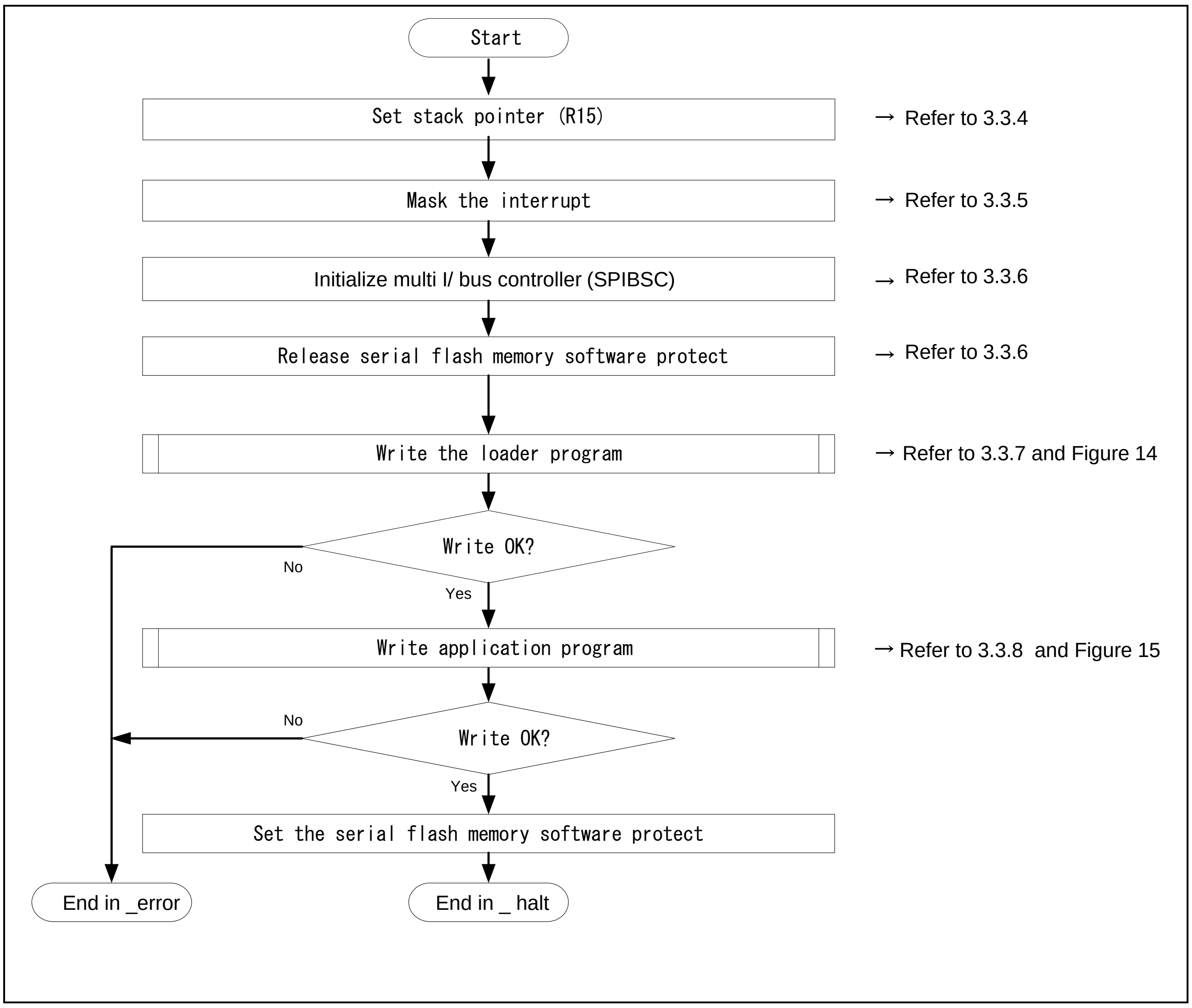

Start
Set stack pointer (R15)
→ Refer to 3.3.4
Mask the interrupt
→ Refer to 3.3.5
Initialize multi I/ bus controller (SPIBSC)
→ Refer to 3.3.6
Release serial flash memory software protect
→ Refer to 3.3.6
Write the loader program
→ Refer to 3.3.7 and Figure 14
Write OK?
No
Yes
Write application program
→ Refer to 3.3.8 and Figure 15
No
Write OK?
Yes
Set the serial flash memory software protect
End in _error
End in _ halt

Figure 12 Downloader Flow Chart

Figure 13 shows the flow chart of writing the loader program and application program.

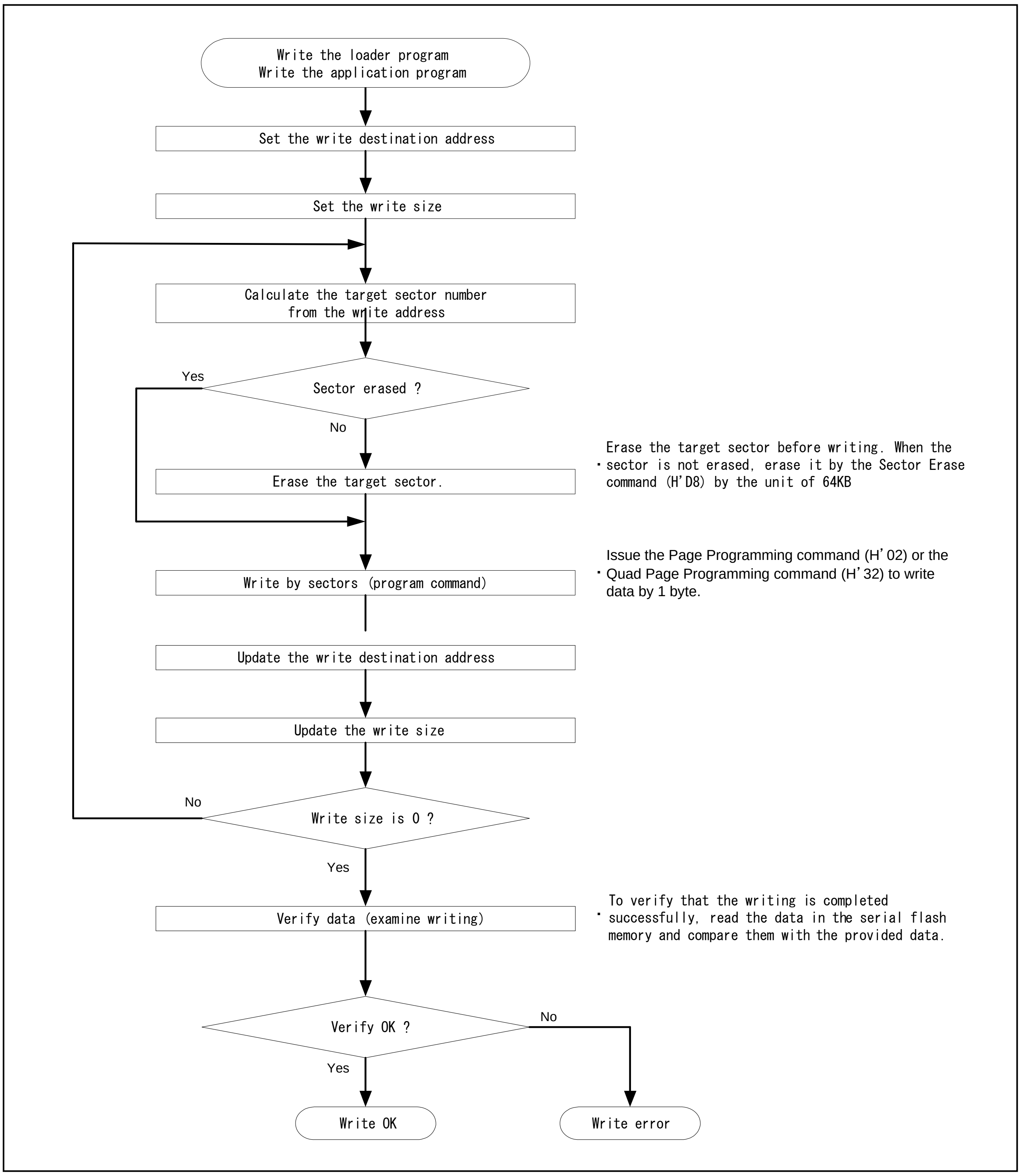

Write the loader program
Write the application program
Set the write destination address
Set the write size
Calculate the target sector number from the write address
Sector erased ?
Yes
No
Erase the target sector.
・Erase the target sector before writing. When the sector is not erased, erase it by the Sector Erase command (H'D8) by the unit of 64KB
Write by sectors (program command)
・Issue the Page Programming command (H'02) or the Quad Page Programming command (H'32) to write data by 1 byte.
Update the write destination address
Update the write size
Write size is 0 ?
No
Yes
Verify data (examine writing)
・To verify that the writing is completed successfully, read the data in the serial flash memory and compare them with the provided data.
Verify OK ?
No
Yes
Write OK
Write error

Figure 13 Flow Chart of Writing

3.3.4 Stack Pointer Setting

H'FF8 4000 is allocated in the number in the stack pointer(R15). This processing should be allocated at the address H'FFF8 2000, and use the assembly language to avoid the downloader using the undefined stack pointer. C can be used after configuring the stack pointer. Then, the downloader jumps to the entry function of the downloader.

3.3.5 Interrup t Mask

B'1111 is specified in the interrupt mask level bit of the Status Register (SR) as the downloader does not support the interrupt in an operation.

3.3.6 Initializ ation

The following initial setting should be given to the serial flash memory before accessing.

1. Configure SPIBSC
2. Issue the Write Enable command to the serial flash memory to cancel the software protection.

3.3.7 Writing the Loader Program

The downloader reads the loader program that has been transferred at the address from H'FFF8 0000 to H'FFF8 1FFF in the high-speed internal RAM, and writes the loader program at the address from H'0000 0000 to H'0000 1FFF in the serial flash memory. Table 9 lists the loader program writing.

Table 9 Loader Program Writing

Item	Description
Loader program transfer source address (high-speed internal RAM)	H'FFF8 0000 (fixed)
Loader program transfer destination address (serial flash memory)	H'0000 0000 (fixed)
Transfer size	H'2000 (fixed)
Writing procedures	1. Check if the destination address has already been erased 2. Erase the data when the address has not been erased 3. Issue the program command to write

3.3.8 Writing the Application Program

The downloader writes the application program from the address H'0000 2000 in the large-capacity internal RAM. Table 10 lists the application program writing.

Table 10 Application Program Writing

Item	Description
Application program transfer source address (large-capacity internal RAM)	H'1C00 2000 (download address in mot file)
Application program transfer destination address (serial flash memory)	H'0000 2000 (fixed)
Transfer size	Extracts from the appinfo in the application program (depends on the application program)
Writing procedures	1. Check if the destination address has already been erased 2. Erase the data when the address has not been erased 3. Issue the program command to write.

3.3.9 Ba tch File

Before executing the downloader, the loader program and the downloader must be transferred to the high-speed internal RAM, and the application program must be transferred to the large-capacity internal RAM to write the loader program and the application program in the serial flash memory.

This application note uses the command batch file in the High-performance Embedded Workshop to execute a series of processing automatically though manual process possible.

Figure 14 shows the flow chart of the command batch file. The command batch file is used to transfer programs to the high-speed internal RAM and the large-capacity internal RAM, and write programs in the serial flash memory.

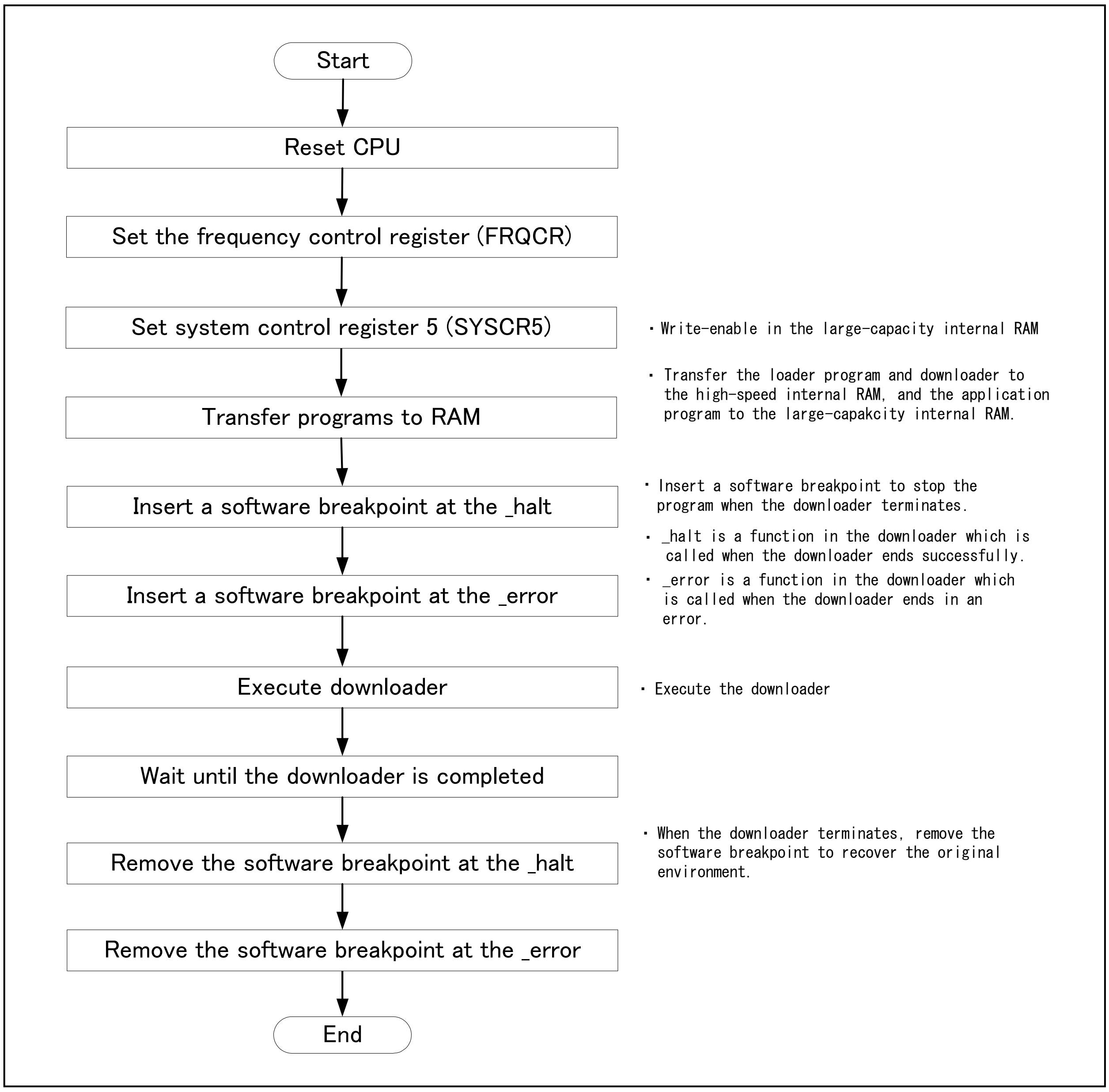

Start
Reset CPU
Set the frequency control register (FRQCR)
Set system control register 5 (SYSCR5)
Transfer programs to RAM
Insert a software breakpoint at the _halt
Insert a software breakpoint at the _error
Execute downloader
Wait until the downloader is completed
Remove the software breakpoint at the _halt
Remove the software breakpoint at the _error
End
・Write-enable in the large-capacity internal RAM
・Transfer the loader program and downloader to the high-speed internal RAM, and the application program to the large-capakcity internal RAM.
・Insert a software breakpoint to stop the program when the downloader terminates.
・_halt is a function in the downloader which is called when the downloader ends successfully.
・_error is a function in the downloader which is called when the downloader ends in an error.
・Execute the downloader
・When the downloader terminates, remove the software breakpoint to recover the original environment.

Figure 14 Command Batch File Flow Chart

4. Sample Program Listing

4.1 Loader Program

4.1.1 Loader Program Listing “loader.src” (1)

```
;/*******************************************************************************
;*   DISCLAIMER
;*
;*   This software is supplied by Renesas Electronics Corporation and is only
;*   intended for use with Renesas products. No other uses are authorized.
;*
;*   This software is owned by Renesas Electronics Corporation 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 CORPORATION 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) 2011 Renesas Electronics Corporation. All rights reserved.
;**************************** Technical reference data *************************
;*   System Name : SH7268/SH7269 Firm Update Sample Program
;*   File Name   : ld_loader.src
;*   Abstract    : Loader program preprocessing/jump processing to the application
;*               : program
;*   Version     : 1.00.00
;*   Device      : SH7268/SH7269
;*   Tool-Chain  : High-performance Embedded Workshop (Ver.4.07.00).
;*               : C/C++ compiler package for the SuperH RISC engine family
;*               :                              (Ver.9.03Release02).
;*   OS          : None
;*   H/W Platform: R0K57269(CPU board)
;*   Description :
;********************************************************************************
;*   History     : Jul.06,2011 Ver.1.00.00
;*""FILE COMMENT END""**************************************************/
  .SECTION LOADER_ENTRY,CODE,ALIGN = 4
 .IMPORT  _main
 .EXPORT  _jmp_app_prog

```

4.1.2 Loader Program Listing "loader.src" (2)

```
_loader_prog:
  MOV.L L2,R15 ; Sets the stack pointer
  MOV.L L1,R0      ; Retrieves the entry function of the loader program
  JMP @R0          ; Jumps to the entry function of the loader program
 NOP


;/*""FUNC COMMENT""**************************************************************
; * ID          :
; * Outline     : Jump to the application program
; *-------------------------------------------------------------------------------
; * Include     :
; *-------------------------------------------------------------------------------
; * Declaration : _jmp_app_prog
; *-------------------------------------------------------------------------------
; * Description : 1. Retrieves the stack pointer value stored in the first 12 to
; *             :    15 bytes in the application program.
; *             : 2. Specifies the stack pointer (R15).
; *             : 3. Retrieves the entry function address stored in the first 8 to
; *             :    11 bytes in the application program.
; *             : 4. Jumps to the entry function.
; *-------------------------------------------------------------------------------
; * Argument    :  R4  ; I : Start address of the application program
; *-------------------------------------------------------------------------------
; * Return Value: none
; *""FUNC COMMENT END""**********************************************************/
_jmp_app_prog:

  MOV.L R4,R0      ; Substitutes the start address of the application program for R0
  ADD #12,R0       ; Calculates the address storing the stack pointer value and
                   ; substitutes the address for R0
  MOV.L @R0,R15    ; Sets the stack pointer

  MOV.L R4,R0      ; Substitutes the start address of the application program for R0
 ADD  #8,R0        ; Calculates the address storing the entry function of the application
                   ; program and substitutes the address for R0
  MOV.L @R0,R0     ; Substitutes the entry function address of the application
                   ; program for R0
  JMP @R0          ; Jumps to the entry function of the application program
 NOP


 .ALIGN  4
L1:
  .DATA.L _main               ; Entry function address of the loader program

L2:
  .DATA.L H'FFF82000          ; Stack pointer (R15) value of the loader program

 .pool
 .end

;/* End of File */
```

4.1.3 Loader Program Listing "ld_main.c" (1)

```
/*******************************************************************************
*   DISCLAIMER
*
*   This software is supplied by Renesas Electronics Corporation and is only
*   intended for use with Renesas products. No other uses are authorized.
*
*   This software is owned by Renesas Electronics Corporation 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 CORPORATION 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) 2011 Renesas Electronics Corporation. All rights reserved.
**************************** Technical reference data *************************
*   System Name : SH7268/SH7269 Firm Update Sample Program
*   File Name   : ld_main.c
*   Abstract    : loader main
*   Version     : 1.00.00
*   Device      : SH7268/SH7269
*   Tool-Chain  : High-performance Embedded Workshop (Ver.4.07.00).
*               : C/C++ compiler package for the SuperH RISC engine family
*               :                             (Ver.9.03 Release02).
*   OS          : None
*   H/W Platform: R0K57269(CPU board)
*   Description :
********************************************************************************
*   History     : Jul.06,2011 Ver.1.00.00
*******************************************************************************/
#include <stdio.h>
#include <string.h>
#include <machine.h>
#include "iodefine.h"
#include "serial_flash.h"

```

4.1.4 Loader Program Listing "ld_main.c" (2)

```
/* ==== macro defined ==== */
#define FPSCR_INIT     0x00040001         /* Value to set in the FPSCR register */
#define INT_MASK       0x000000F0         /* Value to set in the SR register
                                          (for masking the interrupt) */

#define APROG_TOP_SFLASH   0x00002000     /* Start address of the application program */
                                          /* (serial flash memory) */

#define APPINFO_TOP    APROG_TOP_SFLASH       /* Address the appinfo.app_top is located */
#define APPINFO_END    (APROG_TOP_SFLASH + 4) /* Address the appinfo.app_end is located */


/* ==== prototype declaration ==== */
void main(void);
void get_appinfo( unsigned long *app_top_addr,unsigned long *app_end_addr);
void app_prog_transfer(unsigned long app_top_addr,unsigned long app_end_addr);
void system_down(void);

extern void jmp_app_prog(unsigned long app_top_addr);
extern void io_set_cpg(void);
extern void sf_byte_read_long(unsigned long addr, unsigned long *buf, int size);

/* ==== external data ==== */
extern unsigned long DUMMY_Vectors;


```

4.1.5 Loader Program Listing "ld_main.c" (3)

```
/******************************************************************************
 * ID          :
 * Outline     : Loader program main
 * Include     : #include "serial_flash.h"
 * Declaration : void main(void);
 * Description : Refers the data in the appinfo to transfer the application program
 *             : to the large-capacity internal RAM, and jumps to the entry function
 *             : of the application program.
 * Argument    : void
 * Return Value: void
 ******************************************************************************/
void main(void)
{
  unsigned long app_top,app_end;


   /* Sets the FPSCR */
  set_fpscr(FPSCR_INIT);

   /* Sets the tentative VBR */
  set_vbr((void  *)(&DUMMY_Vectors));

   /* Masks the interrupt */
  set_cr(INT_MASK);

   /* Sets the CPG */
  io_set_cpg();

   /* Sets the SPIBSC */
  sf_init_serial_flash_spibsc();

   /* Retrieves the appinfo */
  get_appinfo(&app_top,&app_end);

  sf_allocate_exspace_spibsc();

   /* Jumps to the application program */
  jmp_app_prog(app_top);

  while(1){
     /* LOOP */
  }
}

```

4.1.6 Loader Program Listing "ld_main.c" (4)

```
/*******************************************************************************
 * ID          :
 * Outline     : Retrieve the appinfo
 * Include     : #include "serial_flash.h"
 * Declaration : void get_appinfo (unsigned long *app_top_addr,
 *             :                  unsigned long *app_end_addr);
 * Description : Retrieves the appinfo.
 *             : Retrieves the appinfo.top from H'2000 to H'2003 in serial flash
 *             : memory, and stores it in the address specified by the first
 *             : argument. This function also retrieves the appinfo.end from
 *             : H'2004 to H'2007 in serial flash memory, and stores it in the
 *             : address specified by the second argument.
 * Argument    : unsigned long app_top_addr  ; O : Start address of the application
 *             :                                   program at destination
 *             : unsigned long app_end_addr  ; O : End address of the application
 *             :                                   program at destination
 * Return Value: void
 ******************************************************************************/
void get_appinfo( unsigned long *app_top_addr,unsigned long *app_end_addr)
{

  /* Retrieves the appinfo.top */
  sf_byte_read_spibsc(APPINFO_TOP, (unsigned char *)app_top_addr, 4);

  /* Retrieves the appinfo.end */
  sf_byte_read_spibsc(APPINFO_END, (unsigned char *)app_end_addr, 4);

}

```

4.1.7 Loader Program Listing "ld_main.c" (5)

```
/*******************************************************************************
 * ID          :
 * Outline     : Transfer the application program
 * Include     : #include "serial_flash.h"
 * Declaration : void app_prog_transfer(unsigned long app_top_addr,
 *             :                       unsigned long app_end_addr);
 * Description : Calculates the size of the application program, and transfers
 *             : the application program from serial flash memory to the
 *             : large-capacity internal RAM. (Rounds up the allocation size of the
 *             : application program to multiples of 4 to transfer in longword.)
 * Argument    : unsigned long app_top_addr  ; I : Start address of the application
 *             :                                   program at destination
 *             : unsigned long app_end_addr  ; I : End address of the application
 *             :                                   at destination
 * Return Value: void
 ******************************************************************************/
void app_prog_transfer(unsigned long app_top_addr,unsigned long app_end_addr)
{
  unsigned long app_prog_size;

  /* Calculates the size of the application program */
  app_prog_size = app_end_addr - app_top_addr;
  if( ( app_prog_size & 0x00000003 ) != 0 ){
    app_prog_size &= 0xFFFFFFFC;
    app_prog_size += 4;       /* Rounds up the allocation size of the application
                                 program to multiples of 4. */
 }

  /* Loads the application program in the large-capacity internal RAM */
  sf_byte_read_spibsc(APROG_TOP_SFLASH, (unsigned char *)app_top_addr, app_prog_size);

}

/*******************************************************************************
 * ID          :
 * Outline     : Terminate the system
 * Include     :
 * Declaration : void system_down(void);
 * Description : This function contains the infinite loop.
 *             : As this is registered in the DUMMY_Vectors table, this is
 *             : called when an exception occurs while the loader program
 *             : is operating.
 * Argument    : void
 * Return Value: void
 ******************************************************************************/
void system_down(void)
{
 while(1){
    /* System error */
 }
}

/* End of File */
```

4.2 Application Program

4.2.1 Application Program Listing “main.c”(1)

```
/*******************************************************************************
*   DISCLAIMER
*
*   This software is supplied by Renesas Electronics Corporation and is only
*   intended for use with Renesas products. No other uses are authorized.
*
*   This software is owned by Renesas Electronics Corporation 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 CORPORATION 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) 2011 Renesas Electronics Corporation. All rights reserved.
*************************** Technical reference data **************************
*   System Name : SH7268/SH7269 Sample Program
*   File Name   : main.c
*   Abstract    : Sample Program Main
*   Version     : 1.00.00
*   Device      : SH7268/SH7269
*   Tool-Chain  : High-performance Embedded Workshop (Ver.4.07.00).
*               : C/C++ compiler package for the SuperH RISC engine family
*               :                              (Ver.9.03Release02).
*   OS          : None
*   H/W Platform: R0K57269(CPU board)
*   Description :
********************************************************************************
*   History     : Jul.06,2011 Ver.1.00.00
*******************************************************************************/
#include <stdio.h>
#include <string.h>
#include <machine.h>
#include "iodefine.h"
#include "r0k57269.h"

```

4.2.2 Application Program Listing “main.c” (2)

```
/* ==== prototype declaration ==== */
void main(void);
void io_init_cmt0(void);
void int_cmt_cmi0(void);

/* ==== Global variable ==== */
int g_led_onoff;       /* LED lighting/turning off */

/*******************************************************************************
 * ID          :
 * Outline     : main
 *------------------------------------------------------------------------------
 * Include     :
 *------------------------------------------------------------------------------
 * Declaration : void main(void);
 *------------------------------------------------------------------------------
 * Description :
 *------------------------------------------------------------------------------
 * Argument    : void
 *------------------------------------------------------------------------------
 * Return Value: void
 *------------------------------------------------------------------------------
 * Note        : None
 ******************************************************************************/
void main(void)
{
  puts("\nSH7269 CPU Board Sample Program. Ver.0.02.00");
  puts("Copyright (C) 2010(2011) Renesas Electronics Corporation. All rights eserved.");
 puts("\n");

  /* ==== initial LED port ==== */
  g_led_onoff = 1;
  led_init();                         /* LED Initialization */

  /* ==== start timer ==== */
  io_init_cmt0();                    /* CMT Initialization */

  /* ==== Setting of interrupt priority level ==== */
  INTC.IPR12.BIT._CMT0 = 0x1;     /* CMI Priority level of interrupt = 1 */

  while(1){
        /* loop */
    }
}

```

4.2.3 Application Program Listing "main.c"(3)

```
/********************************************************************************
 * ID          :
 * Outline     : CMT0 setting
 *-------------------------------------------------------------------------------
 * Include     : #include "iodefine.h"
 *-------------------------------------------------------------------------------
 * Declaration : void io_init_cmt0(void) ;
 *-------------------------------------------------------------------------------
 * Description : CMT0 is set as a fixed cycle of about 500ms timer.
 *-------------------------------------------------------------------------------
 * Argument    : void
 *-------------------------------------------------------------------------------
 * Return Value: void
 *-------------------------------------------------------------------------------
 * Note        : None
 *******************************************************************************/
void io_init_cmt0(void)
{

    /* ---- STBCR7 setting ---- */
    CPG.STBCR7.BIT.MSTP72 = 0;        /* Module standby clear */

    /* ==== CMT0 setting ==== */
    /* ---- CMSTR setting ---- */
    CMT.CMSTR.BIT.STR0 = 0;               /* Count stop */
    /* ---- CMCSR0 setting ---- */
    CMT.CMCSR0.WORD = 0x0043;           /* Pclock/512 */

    /* ---- CMCNT0 setting ---- */
    CMT.CMCNT0.WORD = 0x0000;           /* Timer counter clear */

    /* ---- CMCOR0 setting ---- */
    CMT.CMCOR0.WORD = 0x7f08/5;         /* 500/5=100ms */

  /* ---- CMSTR setting ---- */
    CMT.CMSTR.BIT.STR0 = 0x1;           /* Count start */

}

```

4.2.4 Application Program Listing “main.c”(4)

```
/******************************************************************************
 * ID          :
 * Outline     : CMI interrupt
 *-----------------------------------------------------------------------------
 * Include     : #include "iodefine.h"
 *-----------------------------------------------------------------------------
 * Declaration : void int_cmt_cmi0(void);
 *-----------------------------------------------------------------------------
 * Description : The CMF flag is cleared, and the output of
 *             : LED of each 0.5sec is reversed.
 *-----------------------------------------------------------------------------
 * Argument    : void
 *-----------------------------------------------------------------------------
 * Return Value: void
 *-----------------------------------------------------------------------------
 * Note        : None
 ******************************************************************************/
void int_cmt_cmi0(void)
{

  /* ====CMF Clearness of flag ==== */
  CMT.CMCSR0.BIT.CMF = 0;

  /* ==== PORT Reversing output(LED blinking) ==== */
  g_led_onoff ^= 1;
  if(g_led_onoff == 0){
    led_on(ID_LED1);
    led_on(ID_LED2);
 }
 else{
    led_off(ID_LED1);
    led_off(ID_LED2);
 }
}

/* End of File */
```

4.2.5 Application Program Listing “appinfo.c”(1)

```
/*******************************************************************************
*   DISCLAIMER
*
*   This software is supplied by Renesas Electronics Corporation and is only
*   intended for use with Renesas products. No other uses are authorized.
*
*   This software is owned by Renesas Electronics Corporation 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 CORPORATION 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) 2011 Renesas Electronics Corporation. All rights reserved.
**************************** Technical reference data **************************
*   System Name : SH7268/SH7269 Sample Program
*   File Name   : appinfo.c
*   Abstract    : Generate the application program transfer information (appinfo).
*   Version     : 1.00.00
*   Device      : SH7268/SH7269
*   Tool-Chain  : High-performance Embedded Workshop (Ver.4.07.00).
*               : C/C++ compiler package for the SuperH RISC engine family
*               :                            (Ver.9.03 Release02).
*   OS          : None
*   H/W Platform: R0K57269(CPU board)
*   Description :
********************************************************************************
*   History     : Jul.06,2011 Ver.1.00.00
*******************************************************************************/
```

4.2.6 Application Program Listing “appinfo.c”(2)

```
#include "appinfo.h"

#pragma section APPINFO

static APPINFO appinfo = {
  __sectop("DAPPINFO"),    /* Start address in the start section of the application */
                           /* program (program area, constant area, and initialized */
                           /* data area). */

  __secend("PCACHE")       /* End address in the end section of the application */
                           /* program (program area, constant area, and initialized */
                           /* data area) */
};

/* End of File */

```

4.2.7 Application Program Listing “appinfo.h”

```
/*******************************************************************************
*   DISCLAIMER
*
*   This software is supplied by Renesas Electronics Corporation and is only
*   intended for use with Renesas products. No other uses are authorized.
*
*   This software is owned by Renesas Electronics Corporation 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 CORPORATION 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) 2011 Renesas Electronics Corporation. All rights reserved.
**************************** Technical reference data *************************
*   System Name : SH7268/SH7269 Sample Program
*   File Name   : appinfo.h
*   Abstract    : Header file of the application program transfer information (appinfo).
*   Version     : 1.00.00
*   Device      : SH7268/SH7269
*   Tool-Chain  : High-performance Embedded Workshop (Ver.4.07.00).
*               : C/C++ compiler package for the SuperH RISC engine family
*               :                             (Ver.9.03 Release02).
*   OS          : None
*   H/W Platform: R0K57269(CPU board)
*   Description :
********************************************************************************
*   History     : Jul.06,2011 Ver.1.00.00
*******************************************************************************/
#ifndef __APPINFO_H__
#define __APPINFO_H__

typedef struct appinfo_t {
  void *app_top;          /* Start address of the application program */
  void *app_end;          /* End address of the application program */
} APPINFO;

#endif  /* __APPINFO_H__ */
/* End of File */
```

4.3 Dow nloader

4.3.1 Downloader Program Listing “downloader.hdc” (1)

```
#/******************************************************************************
#*   DISCLAIMER
#*
#*   This software is supplied by Renesas Electronics Corporation and is only
#*   intended for use with Renesas products. No other uses are authorized.
#*
#*   This software is owned by Renesas Electronics Corporation 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 CORPORATION 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) 2011 Renesas Electronics Corporation. All rights reserved.
#**************************** Technical reference data *************************
#*   System Name : SH7268/SH7269 Firm Update Sample Program
#*   File Name   : downloader.hdc
#*   Abstract    : ダウンローダ用バッチファイル
#*   Version     : 1.00.00
#*   Device      : SH7269/SH7269
#*   Tool-Chain  : High-performance Embedded Workshop (Ver.4.07.00).
#*               : C/C++ compiler package for the SuperH RISC engine family
#*               :                              (Ver.9.03Release02).
#*   OS          : None
#*   H/W Platform: R0K57269(CPU board)
#*   Description :
#*******************************************************************************
#*   History     : Jul.06,2011 Ver.1.00.00
#*""FILE COMMENT END""**********************************************************/


tcl enable


```

4.3.2 Downloader Program Listing “downloader.hdc” (2)

```
#Macro downloader -Start
proc init_hardware {} {

  # set CPG
  MF H'FFFE0010 H'FFFE0011 H'1015 WORD

  # set SYSCR5
  MF H'FFFE0428 H'FFFE0428 H'0F BYTE
}


proc downloader {} {
  # reset CPU
 reset

  # init_hardware call routines
 init_hardware

  #download all [Download modules] in High-performance Embedded Workshop
 file_load_all

  #permit user stack (to use software breakpoint)
 sh2a_sbstk  enable

  # set software break point in _halt(refer to main.c)
  set_disassembly_soft_break _halt set

  # set software break point in _error(refer to main.c)
  set_disassembly_soft_break _error set

  # execute _downloader(refer to downloader.src). Wait till stops
  go wait _downloader

  # clear software break point set in _halt
  set_disassembly_soft_break _halt clear

  # clear the software break point set in _error
  set_disassembly_soft_break _error clear

}

downloader
#Macro downloader -End



#Note: "tcl","reset","file_load","sh2a_sbstk","set_disassembly_soft_break","go" are、the
commands of #High-performance Embedded Workshop and E10A-USB emulator. For the details of
commands, see the Manual.。

# /* End of File */
```

4.3.3 Downloader Program Listing “dl_entry.src” (1)

```
;/*******************************************************************************
;*   DISCLAIMER
;*
;*   This software is supplied by Renesas Electronics Corporation and is only
;*   intended for use with Renesas products. No other uses are authorized.
;*
;*   This software is owned by Renesas Electronics Corporation 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 CORPORATION 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(2011) Renesas Electronics Corporation. All rights reserved.
;***************************** Technical reference data *************************
;*   System Name : SH7268/SH7269 Firm Update Sample Program
;*   File Name   : dl_entry.src
;*   Abstract    : downloader start up
;*   Version     : 0.03.00
;*   Device      : SH7268/SH7269
;*   Tool-Chain  : High-performance Embedded Workshop (Ver.4.07.00).
;*               : C/C++ compiler package for the SuperH RISC engine family
;*               :                              (Ver.9.03Release02).
;*   OS          : None
;*   H/W Platform: R0K57269(CPU board)
;*   Description :
;********************************************************************************
;*   History     : Sep.06,2010 Ver.0.01.00
;*               : Apr.27,2011 Ver.0.02.00 change comment
;*               : May.08,2011 Ver.0.03.00 change comment
;*""FILE COMMENT END""**************************************************************/
```

4.3.4 Downloader Program Listing “dl_entry.src” (2)

```
   .SECTION ENTRY,CODE,ALIGN = 4
  .IMPORT  _PowerON_Reset_PC

_downloader:
   MOV.L STACK_POINTER,R15     ; setting stack pointer
   MOV.L MAIN_PROGRAM,R0       ; get entry address of downloader
   JMP @R0                     ; jump to entry address of downloader
  NOP

  .ALIGN  4
MAIN_PROGRAM:
   .DATA.L _PowerON_Reset_PC   ; entry address of downloader

STACK_POINTER:
   .DATA.L H'FFF84000          ; stack pointer of downloader(R15)

  .pool
  .end

;/* End of File */
```

4.3.5 Downloader Program List “dl_main.c” (1)

```
/*******************************************************************************
*   DISCLAIMER
*
*   This software is supplied by Renesas Electronics Corporation and is only
*   intended for use with Renesas products. No other uses are authorized.
*
*   This software is owned by Renesas Electronics Corporation 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 CORPORATION 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) 2011 Renesas Electronics Corporation. All rights reserved.
**************************** Technical reference data **************************
*   System Name : SH7268/SH7269 Firm Update Sample Program
*   File Name   : dl_main.c
*   Abstract    : downloader main
*   Version     : 1.00.00
*   Device      : SH7268/SH7269
*   Tool-Chain  : High-performance Embedded Workshop (Ver.4.07.00).
*               : C/C++ compiler package for the SuperH RISC engine family
*               :                             (Ver.9.03 Release02).
*   OS          : None
*   H/W Platform: R0K57269(CPU board)
*   Description :
********************************************************************************
*   History     : Jul.06,2011 Ver.1.00.00
*******************************************************************************/
#include <stdio.h>
#include <string.h>
#include <machine.h>
#include "iodefine.h"
#include "serial_flash.h"

```

4.3.6 Downloader Program List “dl_main.c” (2)

```
/* ==== macro defined ==== */
#define SECTOR_SIZE       SF_SECTOR_SIZE     /* Sector size */
#define SECTOR_NUM        SF_NUM_OF_SECTOR   /* Total number of sectors in the device  */
#define DEVICE_SIZE       (SECTOR_SIZE * SECTOR_NUM) /* Device size  */

#if (SFLASH_DUAL == 0)
#define L_PROG_SIZE       8192               /* Loader program size         */
#else
#define L_PROG_SIZE       4096               /* Loader program size         */
#endif
#define L_PROG_SRC        0xFFF80000         /* Source address of the loader program */
#define L_PROG_DST        0x00000000     /* Destination address of the loader program */

#define APROG_TOP_SFLASH  0x00002000    /* Start address of the application program */

#define APROG_TOP_RAM     0x1C002000    /* Start address of the application program */
                                        /* When changing the start section of the */
                                        /* application program, change this definition */

#define APPINFO_TOP       APROG_TOP_RAM      /* Address the appinfo.app_top is located */
#define APPINFO_END    ( APROG_TOP_RAM + 4 )  /* Address the appinfo.app_end is located */

/* ==== prototype declaration ==== */
/*** User API ****/
void main(void);

static void halt(void);
static void error(void);
static void init_erase_flag(void);
static int Is_erased_sector(unsigned long sector_no);
static void set_erase_flag(unsigned long sector_no);
static int write_prog_data(unsigned char *program_data, unsigned long sflash_addr,
                      unsigned long size);

/*** data ***/
static unsigned char sflash_erase_flag[SECTOR_NUM];/* 0: sector not erased,1: sector
erased */
```

4.3.7 Downloader Program List “dl_main.c” (3)

```

/*******************************************************************************
 * ID          :
 * Outline     : Downloader main
 * Include     :
 * Declaration : void main(void);
 * Description : Writes the loader program and application program in serial
 *             : flash memory as the following procedures.
 *             : 1. Mask the interrupt while the downloader is operating.
 *             : 2. Initialize the RSPI0.
 *             : 3. Disable the software protection in serial flash memory.
 *             : 4. Write the loader program in serial flash memory.
 *             : 5. Write the application program in serial flash memory.
 * Argument    : void
 * Return Value: void
 ******************************************************************************/
void main(void)
{
  unsigned long app_top_addr,app_end_addr,app_prog_size;


   /* Initializes the erase flag */
  init_erase_flag();

   /* Initializes the SPIBSC */
  sf_init_serial_flash_spibsc();

   /* Disables the software protection in serial flash memory */
  sf_protect_ctrl_spibsc(SF_REQ_UNPROTECT);

   /* Writes the loader program */
   if( write_prog_data( (unsigned char *)L_PROG_SRC, L_PROG_DST, L_PROG_SIZE) < 0 ){
     error();
  }

   /* Retrieves the start address and end address from the application program
     transfer information (appinfo) */
   app_top_addr = *(volatile unsigned long *)APPINFO_TOP;
   app_end_addr = *(volatile unsigned long *)APPINFO_END;

   /* Calculates the size of the application program */
   app_prog_size = app_end_addr - app_top_addr;
```

4.3.8 Downloader Program List “dl_main.c” (4)

```

     /* Writes the application program */
    if( write_prog_data((unsigned char *)APROG_TOP_RAM,APROG_TOP_SFLASH, app_prog_size)<0){
        error();
    }

     /* Enables the software protection in serial flash memory */
    sf_protect_ctrl_spibsc(SF_REQ_PROTECT);

     /* Exits the downloader */
    halt();
  }
```

4.3.9 Downloader Program List “dl_main.c” (5)

```

/*******************************************************************************
 * ID          :
 * Outline     : Write the program data
 * Include     :
 * Declaration : int write_prog_data(unsigned char *program_data,
 *             :                    unsigned long sflash_addr, unsigned long size);
 * Description : Writes the program data as the following procedures.
 *             : 1. Erase the target sector when it is not erased.
 *             : 2. Write the program data in serial flash memory.
 *             : 3. Reads the data in serial flash memory and compare it with the
 *             :    provided data.
 * Argument    : unsigned char *program_data ; I : Start address of the program data
 *             : unsigned long sflash_addr   ; I : Start address at the destination in
 *                                                 serial flash memory
 *             : unsigned long size          ; I : Write size
 * Return Value: Equal or bigger than 0: Success
 *             : Less than 0: Error
 *******************************************************************************/
int write_prog_data(unsigned char *program_data, unsigned long sflash_addr, unsigned long
size)
{
  unsigned long sector_no;
  unsigned long saddr;
  unsigned long sz;
  unsigned char read_data[2];
  unsigned char *w_p;
 int  wr_size;
 int  rd_size;
 int  bsz;

  bsz = 1;

  /* ==== Copies the value from the argument to the local variable ==== */
  saddr = sflash_addr;
  sz = size;
  w_p = program_data;

  /* ==== Writes data in serial flash memory ==== */
  while( sz > 0){
    if(sz > ((256 * bsz) - (saddr % (256 * bsz)))){
        wr_size = (int)((256 * bsz) - (saddr % (256 * bsz)));
    }
    else{
        wr_size = (int)sz;
    }

    sector_no = saddr / (SECTOR_SIZE * bsz);
```

4.3.10 Downloader Program List “dl_main.c” (6)

```

           if( Is_erased_sector(sector_no) == 0 ){ /* When it is not erased */
               sf_sector_erase_spibsc(sector_no);/* Erase */
               set_erase_flag(sector_no);        /* When it is erased, set the erase flag */
           }

           sf_byte_program_spibsc(saddr, w_p, wr_size );
                                                 /* Writes data in units of */
                                                 /* single byte */
           w_p += wr_size;
           saddr += wr_size;
           sz -= wr_size;
        }

         /* ==== Verifies data (serial flash memory is programmed successfully) ==== */
         saddr = sflash_addr;
         sz = size;
         w_p = program_data;

         rd_size = 1;
         while( sz > 0){
           sf_byte_read_spibsc(saddr,read_data, rd_size);
                                             /* Reads the data written in */
                                          /*    serial flash memory */

           if( w_p[0] != read_data[0] ){
               return -1;                    /* Returns an error when the data */
                                          /*    unmatched */
           }
           w_p += rd_size;
           saddr += rd_size;
           sz -= rd_size;
        }

        return  0;
       }

```

4.3.11 Downloader Program List “dl_main.c” (7)

```
/******************************************************************************
 * ID          :
 * Outline     : Initialize the Erase Flag
 * Include     :
 * Declaration : static void init_erase_flag(void);
 * Description : Initializes the table sflash_erase_flag[].
 * Argument    : void
 * Return Value: void
 ******************************************************************************/
static void init_erase_flag(void)
{
 int  i;

  for( i=0; i < SECTOR_NUM ;i++){
    sflash_erase_flag[i] = 0;
 }
}

/******************************************************************************
 * ID          :
 * Outline     : Retrieve the Sector Erase Status
 * Include     :
 * Declaration : static int Is_erased_sector(unsigned long sector_no);
 * Description : Returns the information (not erased or eraser) of the
 *             : sector specified by the sector number.
 * Argument    : unsigned long sector_no   ; I : Sector number
 * Return Value: 1 : Sector in the specified address is already erased
 *             : 0 : Sector in the specified address is not erased
 ******************************************************************************/
static int Is_erased_sector(unsigned long sector_no)
{
 return  sflash_erase_flag[sector_no];
}

/******************************************************************************
 * ID          :
 * Outline     : Set the Erase Flag
 * Include     :
 * Declaration : static void set_erase_flag(unsigned long sector_no);
 * Description : Sets the erase flag to modify the information of the specified
 *             : sector as erased.
 * Argument    : unsigned long sector_no   ; I : Sector number
 * Return Value: void
 ******************************************************************************/
static void set_erase_flag(unsigned long sector_no)
{
  sflash_erase_flag[sector_no] = 1;
}

```

4.3.12 Downloader Program List “dl_main.c” (8)

```
/******************************************************************************
 * ID          :
 * Outline     : Program stops (successful).
 * Include     :
 * Declaration : static void halt(void);
 * Description : When the downloader ends successfully, this function is called
 *             : to stop the program.
 * Argument    : void
 * Return Value: void
 ******************************************************************************/
static void halt(void)
{
    while(1){
        /* When the downloader ends successfully, this function stops the program. */
    }
}

/******************************************************************************
 * ID          :
 * Outline     : Program stops (error).
 * Include     :
 * Declaration : static void error(void);
 * Description : When the downloader ends in error, this function is called
 *             : to stop the program.
 * Argument    : void
 * Return Value: void
 ******************************************************************************/
static void error(void)
{
    while(1){
    /* When the downloader ends in error, this function stops the program */
    }
}

/* End of File */

```

5. Using the Downloader

The downloader in this application is designed to operate with the combination of the High-performance embedded Workshop and the E10A-USB emulator. When using the downloader with other development environments, alter the program according to the usage environments.

Programs cannot be written in the serial flash memory by selecting the downloader module in the **Download** dialog box on the Debug menu. This section explains the procedures to write programs in the serial flash memory using the downloader in the applicable examples.

5.1 Sample Program Configuration

The sample program consists of three workspaces as listed in Table 11.

Table 11 Sample Program Configuration

Workspace Name	Description
sh7269_spibsc_downloader	Builds the downloader in the project of this workspace
sh7269_spibsc_loader_prog	Builds the loader program in the project of this workspace
sh7269_spibsc_app	Builds the application program in the project of this workspace. The downloader created in the [sh7269_spibsc_downloader] workspace, a batch file to boot the downloader, and the loader program created in the [sh7269_spibsc_loader_prog] workspace are registered in the project of this workspace. Use these items to write the loader program and application program in the serial flash memory.

5.2 Writing Programs in the Serial Flash Memory

This section describes how to write the loader program and application program in the serial flash memory using the [sh7269_spibsc_app] workspace.

5.2.1 Registering the Download Module and the Batch File

Figure 15 shows the directory configuration of the [sh7269_spibsc_app] workspace. Download modules (A, B, Cand E) and a batch file (D) in the figure are registered in the project. An attention is required to register the download modules A and B. A as abs file is downloaded only its debug information and B as mot file is downloaded on large-capacity RAM using offset specification as SPI multi I/O bus space does not allow downloading into it directly.

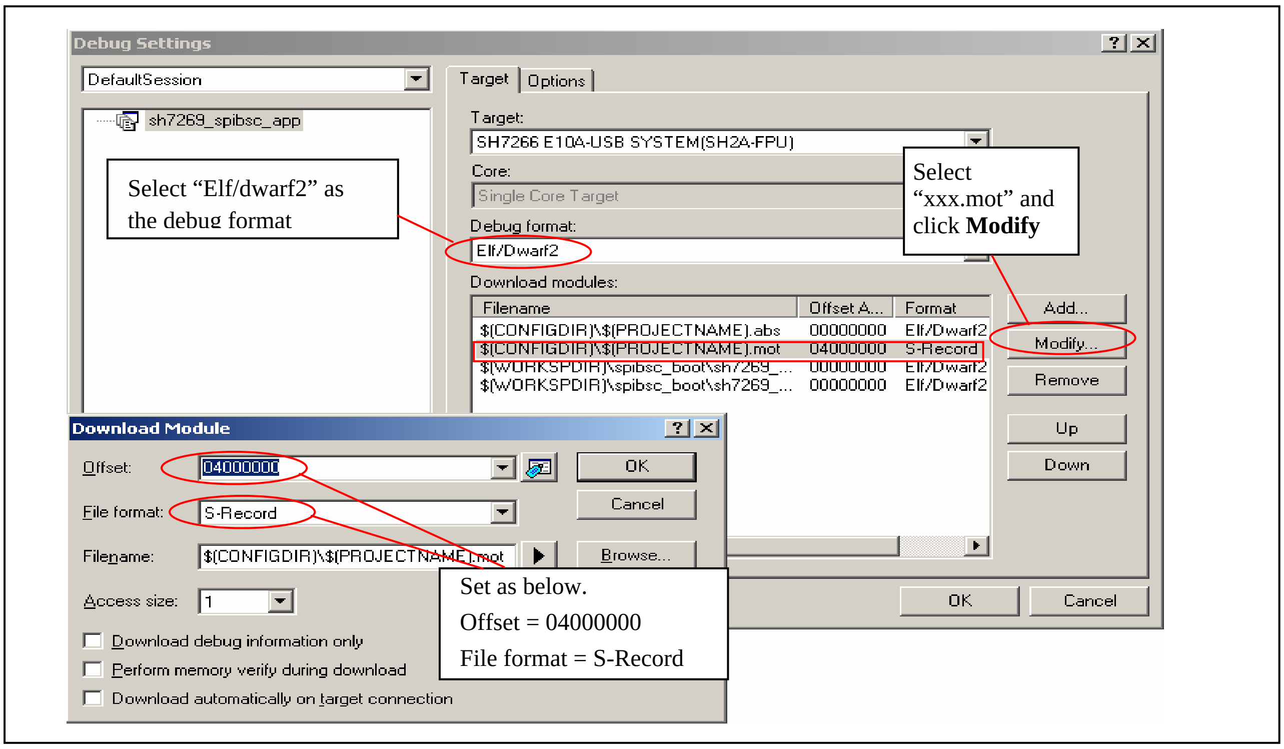
Figure 16 and Figure 17 show the download examples using in the sample programs.

```
¥sh7269_spibsc_app                    :  Workspace directory
  |-sh7269_spibsc_app                 :  Project directory
  |   |-debug                         :
  |       |-sh7269_spibsc_app.abs     :  Application program execute file 1       ------------------------A
  |       |-sh7269_spibsc_app.mot     :--Application program execute file 2            -----------------B
  |                                   :
  |-inc                               :  Directory to store the common include files
  |-src                               :  Directory to store the source files
  |-sflash_boot                       :  Directory to store the downloader and loader programs
    |-sh7269_spibsc_downloader.abs    :  Downloader execute file    ---------------------------------------------C
    |-downloader.hdc                  :  Batch file to boot the downloader  ----------------------------------D
    |-sh7269_spibsc_loader_prog.abs   :  Loader program execute file    ----------------------------------------E
```

Figure 15 [sh7269_spibsc_app] Workspace Directory Configuration

1. Changing the download module
   The download module setting is changed in the **Debug Settings** dialog box which is opened by selecting **Debug Setting** in the **Debug** menu of the High-performance Embedded Workshop.
   For the procedure to register the download modules, refer to the High-performance Embedded Workshop User’s Manual.

2. Changing the batch file
   The batch file setting registered in the project is changed in the **Set Batch File** dialog box. The following procedure will open the **Set Batch File** dialog box. On the View menu in the High-performance Embedded Workshop, click the **Command Line** to open the **Command Line** window. Open the **Set Batch File** dialog box from the **Batch File** pop-up menu on the **Command Line** window.
   For the procedure to register the batch file, refer to the High-performance Embedded Workshop User’s Manual.



Debug Settings

DefaultSession

sh7269_spibsc_app

Target | Options

Target: SH7266 E10A-USB SYSTEM(SH2A-FPU)

Core: Single Core Target

Debug format: Elf/Dwarf2

Download modules:

Filename	Offset A...	Format
$(CONFIGDIR)\$(PROJECTNAME).abs	00000000	Elf/Dwarf2
$(CONFIGDIR)\$(PROJECTNAME).mot	04000000	S-Record
$(WORKSPDIR)\spibsc_boot\sh7269_...	00000000	Elf/Dwarf2
$(WORKSPDIR)\spibsc_boot\sh7269_...	00000000	Elf/Dwarf2

Download Module

Offset: 04000000

File format: S-Record

Filename: $(CONFIGDIR)\$(PROJECTNAME).mot

Access size: 1

☐ Download debug information only

☐ Perform memory verify during download

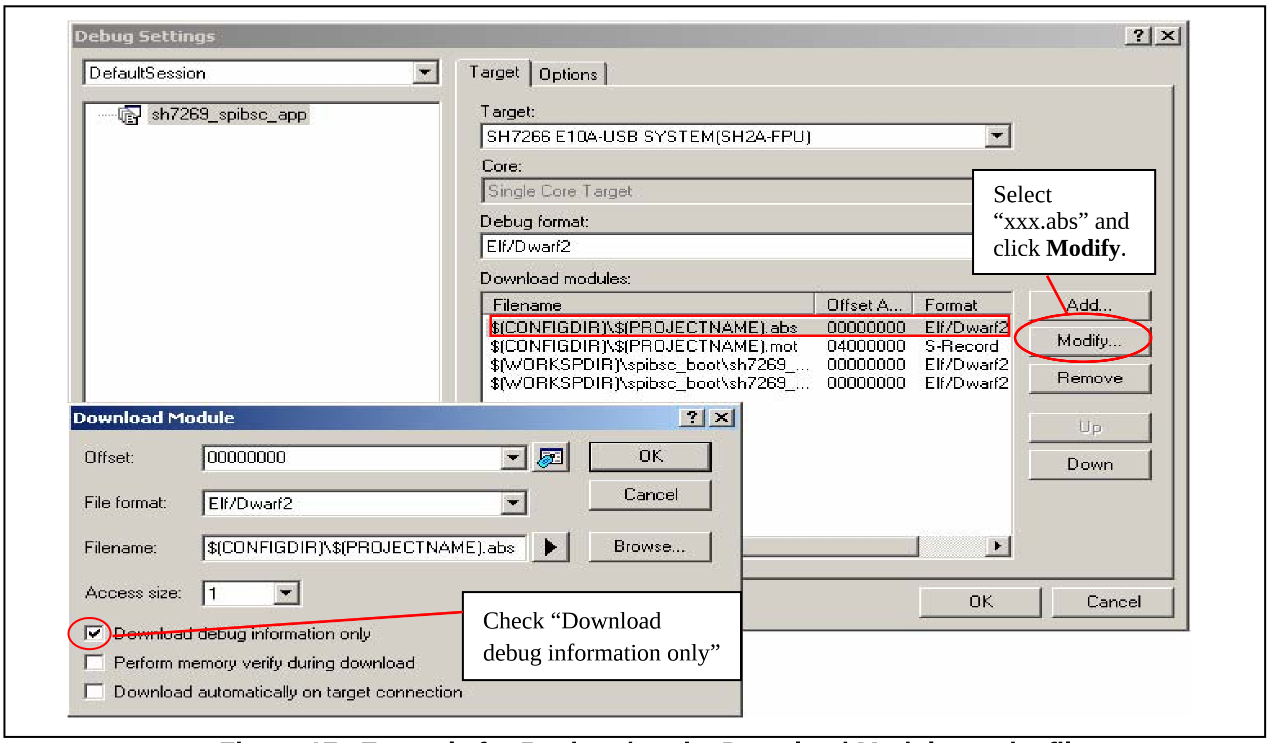
☐ Download automatically on target connection

Select "Elf/dwarf2" as the debug format

Select "xxx.mot" and click **Modify**

Set as below.
Offset = 04000000
File format = S-Record

Figure 16 Example for Registering the Download Module as mot file



Debug Settings

DefaultSession

sh7269_spibsc_app

Target | Options

Target: SH7266 E10A-USB SYSTEM(SH2A-FPU)

Core: Single Core Target

Debug format: Elf/Dwarf2

Download modules:

Filename	Offset A...	Format
$(CONFIGDIR)\$(PROJECTNAME).abs	00000000	Elf/Dwarf2
$(CONFIGDIR)\$(PROJECTNAME).mot	04000000	S-Record
$(WORKSPDIR)\spibsc_boot\sh7269_...	00000000	Elf/Dwarf2
$(WORKSPDIR)\spibsc_boot\sh7269_...	00000000	Elf/Dwarf2

Download Module

Offset: 00000000

File format: Elf/Dwarf2

Filename: $(CONFIGDIR)\$(PROJECTNAME).abs

Access size: 1

☑ Download debug information only

☐ Perform memory verify during download

☐ Download automatically on target connection

Select "xxx.abs" and click **Modify.**

Check "Download debug information only"

Figure 17 Example for Registering the Download Module as abs file

5.2.2 Procedures to Writing Programs

This section describes how to write the loader program and the application program in the serial flash memory using the [sh7269_spibsc_app] workspace.

1. Copy the [sh7269_spibsc_app] workspace directory in C:\Workspace.
2. Double-click the [sh7269_spibsc_app].hws in the workspace directory to activate the High-performance Embedded Workshop.
3. On the **Build** menu in High-performance Embedded Workshop, select the **Build All** to build the project. The application program is generated.
4. On the **Debug** menu in High-performance Embedded Workshop, select the **Go** to connect with the target device.
5. After the connection establishment, select the **Command Line** on the **View** menu in High-performance Embedded Workshop to open the **Command Line** window as shown in Figure 18.
6. Click the **Run Batch** button in the **Command Line** window to execute the registered batch file [downloader.hdc].

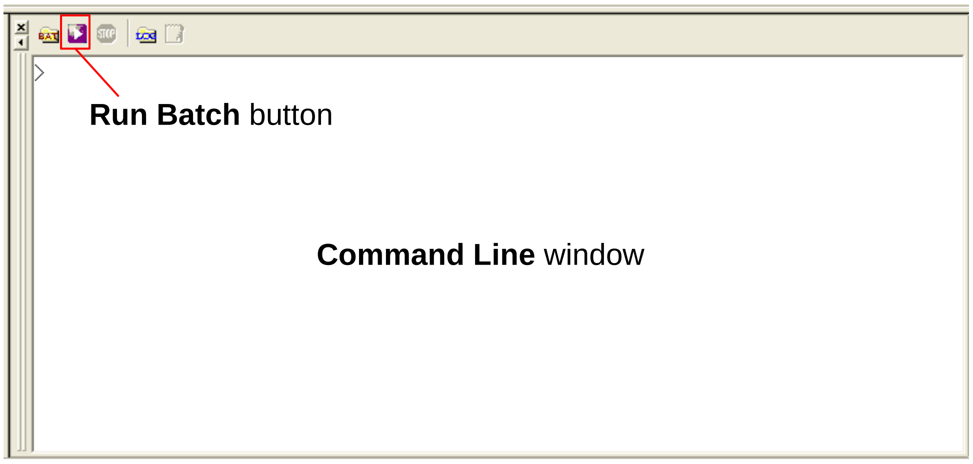
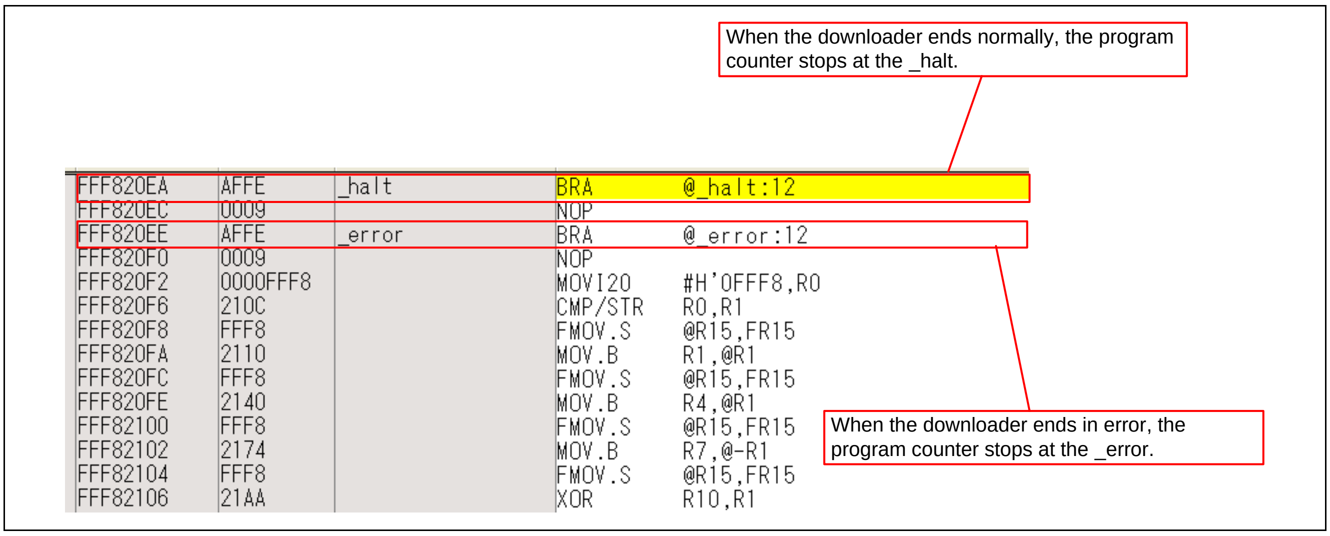


Figure 18 Command Line Window and Run Batch Button

7. When the batch file [downloader.hdc] is executed, all of the download modules registered in the workspace (loader program, application program, and downloader) are transferred to RAM to execute the downloader. As shown in Figure 19, the program counter stops at the _halt, when the downloader ends normally. The program counter stops at the _error, when the downloader ends in error. A source file may appear when the [sh7269_spibsc_downloader] workspace directory is copied in C:\Workspace.
8. When writing is completed successfully, the loader program and the application program can be executed after **Reset Go**.


When the downloader ends normally, the program counter stops at the _halt.

```
FFF820EA   AFFE       _halt     BRA       @_halt:12
FFF820EC   0009                 NOP
FFF820EE   AFFE       _error    BRA       @_error:12
FFF820F0   0009                 NOP
FFF820F2   0000FFF8             MOVI20    #H'0FFF8,R0
FFF820F6   210C                 CMP/STR   R0,R1
FFF820F8   FFF8                 FMOV.S    @R15,FR15
FFF820FA   2110                 MOV.B     R1,@R1
FFF820FC   FFF8                 FMOV.S    @R15,FR15
FFF820FE   2140                 MOV.B     R4,@R1
FFF82100   FFF8                 FMOV.S    @R15,FR15
FFF82102   2174                 MOV.B     R7,@-R1
FFF82104   FFF8                 FMOV.S    @R15,FR15
FFF82106   21AA                 XOR       R10,R1
```

When the downloader ends in error, the program counter stops at the _error.

Figure 19 High-performance Embedded Workshop Window When the Downloader Ends

6. References

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

- User's Manual for Hardware
  SH7268 Group, SH7269 Group User's Manual: Hardware Rev. 1.00
  The latest version can be downloaded from the Renesas Electronics website.

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	Jul 11.11	—	First edition issued
1.01	Feb 16.12	—	Added sample code of SH726B

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. Clo ck 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: +49-211-65030, 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.
13F, No. 363, Fu Shing North Road, Taipei, Taiwan
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 1.1