SH7266/SH7267 Group

Boot From the Serial Flash Memory

R01AN0214EJ0101
Rev. 1.01
Mar. 31, 2011

Summary

This application note describes how to boot the SH7266/SH7267 Microcomputers (MCUs) from its internal serial flash memory.

Target Device

SH7266/SH7267 MCU (In this document, SH7266/SH7267 are described as "SH7267".)

Contents

1. Introduction

1.1 Specifications

Boot modes 1 and 3 allow the engineer to boot the SH7267 from its internal flash memory (serial flash boot). This application note describes the loader program and application program examples when using the serial flash boot. The downloader to write the loader program and application to serial flash memory is also described.

1.2 Modules Used

- Boot mode (serial flash boot)
- Renesas Serial Peripheral Interface (RSPI)

1.3 Applicable Conditions

MCU	SH7266/SH7267
Operating Frequency	Internal clock: 144 MHz Bus clock: 72 MHz Peripheral clock: 36 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)

1.4 Related Application Note

For more information, refer to the following application note:

- SH7266/SH7267 Group Interfacing Serial Flash Memory Using the Renesas Quad Serial Peripheral Interface

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 Glossary of Terms

The following table lists the terms used in this application note to describe the serial flash boot.

Table 1 Glossary

Item	Description
Internal ROM program to boot	A program to transfer the loader program stored in the beginning of the serial flash memory to the high-speed internal RAM, and jump to the loader program when the MCU is booted in boot mode 1 or 3. As this program is already stored in the internal ROM to boot in CPU, user is not required to create it.
Loader program	A program to transfer the application program from the serial flash memory to RAM, and jump to the entry function of the application program. The size of the loader program is fixed to 8-KB. Create it according to the system.
Application program	A program that is created by user according to the system
Downloader	A program to write the loader program and application program to the serial flash memory. Create it according to the system.

2.2 Operation

The following table lists the external pins (MD_BOOT1 to MD_BOOT0) to decide the boot mode.

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

MD_BOOT1	MD_BOOT0	Boot Mode	Description
0	1	Boot mode 1	Boots the MCU from serial flash memory connected to Renesas Serial Peripheral Interface channel 0 (RSPI0) in low-speed. Low-speed communication is the communication at 1/4 of the bus clock rate (Bφ)
1	1	Boot mode 3	Boots the MCU from serial flash memory connected to Renesas Serial Peripheral Interface channel 0 (RSPI0) in high-speed. High-speed communication is the communication at 1/2 of the bus clock rate (Bφ)

In boot modes 1 or 3, the 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 the transfer is complete, it jumps to the start of the loader program. The following figure shows the operation image of the internal ROM program to boot. A series of processing is automatically executed.

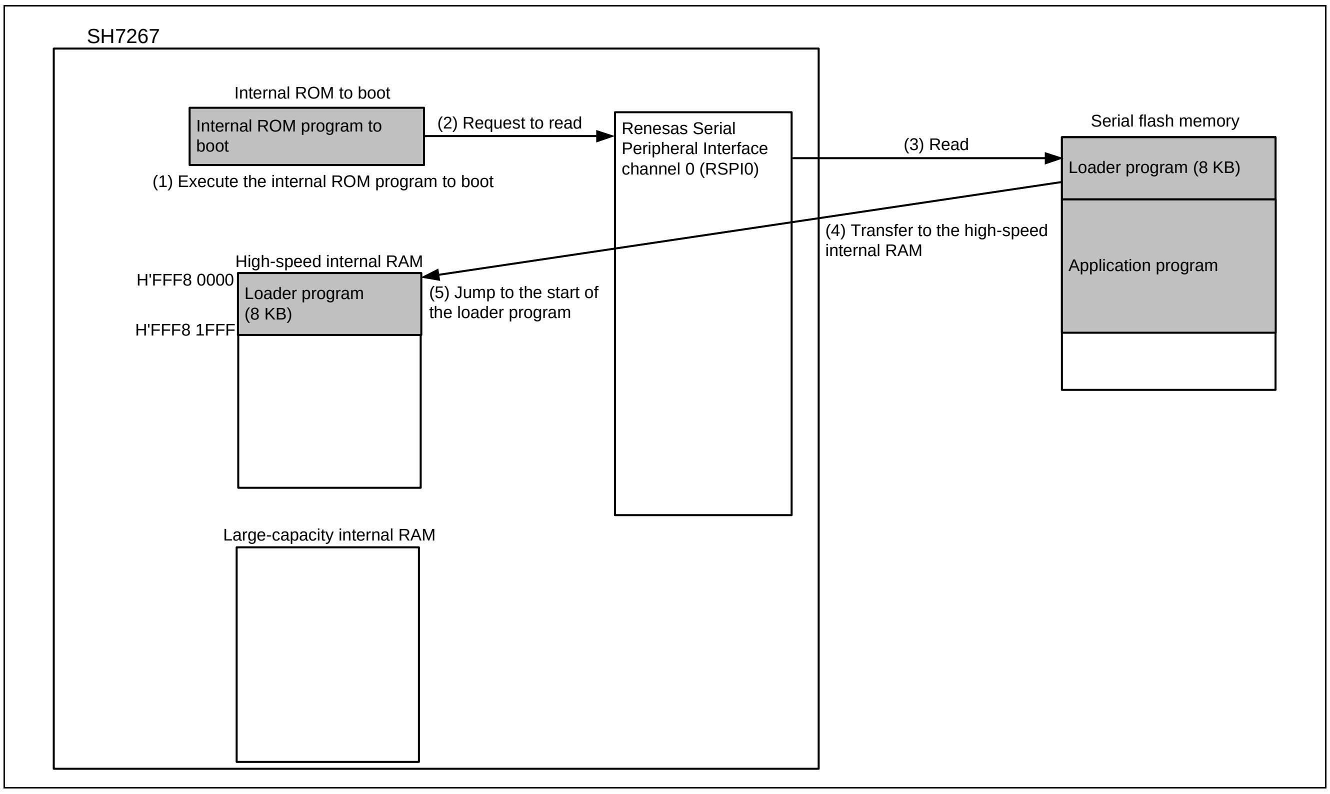

SH7267
Internal ROM to boot
Internal ROM program to boot
(2) Request to read
Renesas Serial Peripheral Interface channel 0 (RSPI0)
(3) Read
Serial flash memory
Loader program (8 KB)
(1) Execute the internal ROM program to boot
(4) Transfer to the high-speed internal RAM
High-speed internal RAM
H'FFF8 0000
H'FFF8 1FFF
Loader program (8 KB)
(5) Jump to the start of the loader program
Application program
Large-capacity internal RAM

Figure 1 Operation Image of the Internal ROM Program to Boot

The loader program transfers the application program from the serial flash memory connected to the Renesas Serial Peripheral Interface channel 0 (RSPI0) to the large-capacity internal RAM. After the transfer is complete, the loader program jumps to the entry function of the application program. The following figure shows the operation image of the loader program.

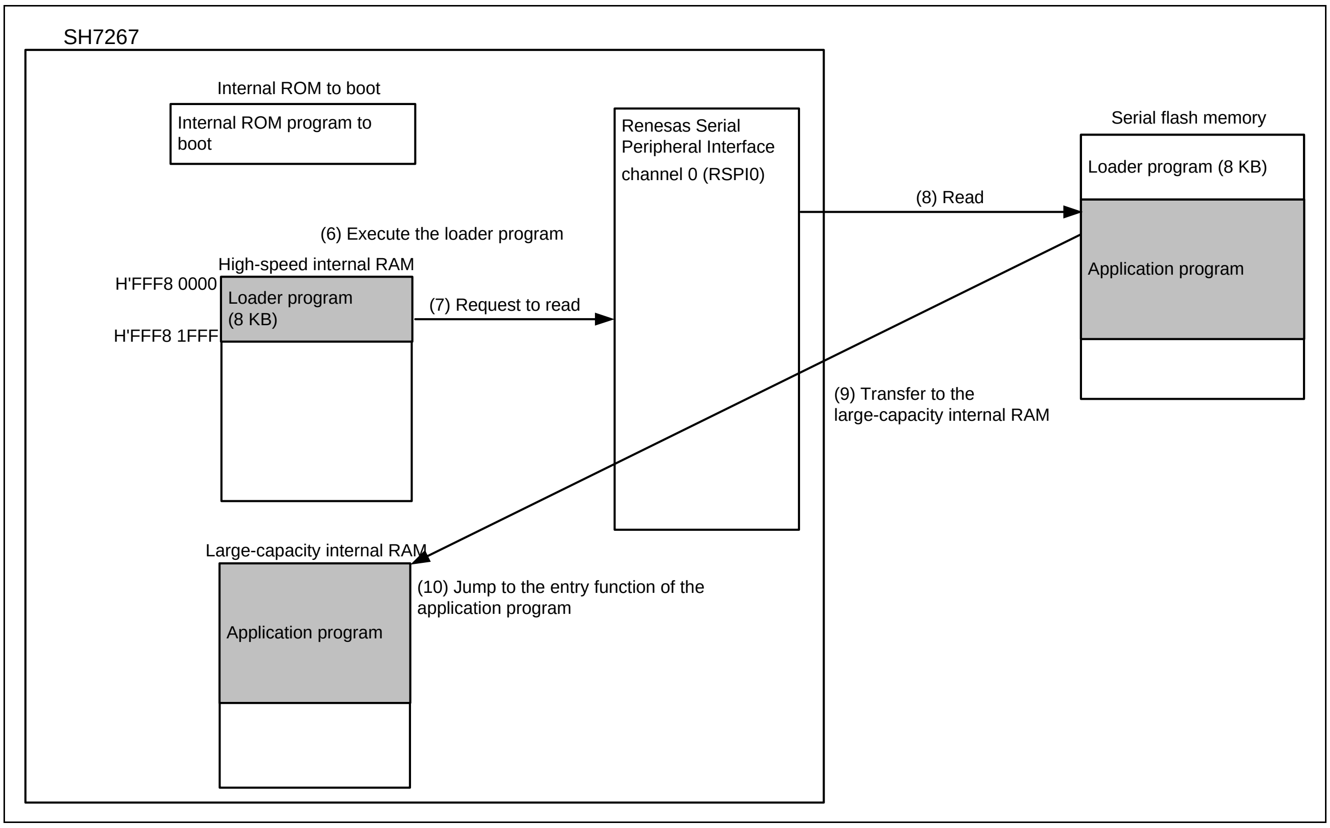

SH7267
Internal ROM to boot
Internal ROM program to boot
Renesas Serial Peripheral Interface channel 0 (RSPI0)
Serial flash memory
Loader program (8 KB)
(8) Read
(6) Execute the loader program
High-speed internal RAM
H'FFF8 0000
Loader program (8 KB)
H'FFF8 1FFF
(7) Request to read
Application program
(9) Transfer to the large-capacity internal RAM
Large-capacity internal RAM
Application program
(10) Jump to the entry function of the application program

Figure 2 Operation Image of the Loader Program

Note: Application program can be transferred to external RAM such as SDRAM by modifying the loader program.

2.3 Downloader Operation

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

For more information, refer to 3.3 Downloader Example.

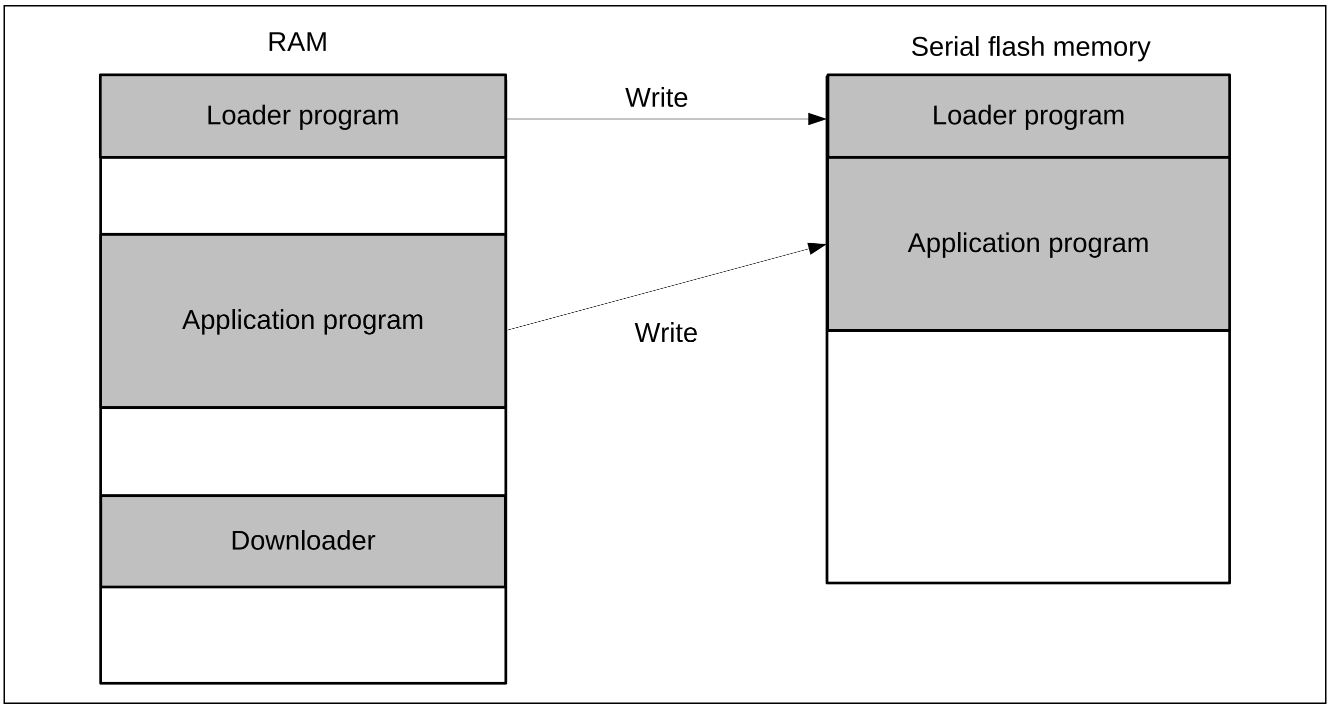


Figure 3 Operation Image of the Downloader

2.4 Serial Flash Memory Connection

The figure below shows an example of serial flash memory connection circuit. Connect the serial flash memory to the Renesas Peripheral Interface channel 0 (RSPI0) to use the serial flash boot.

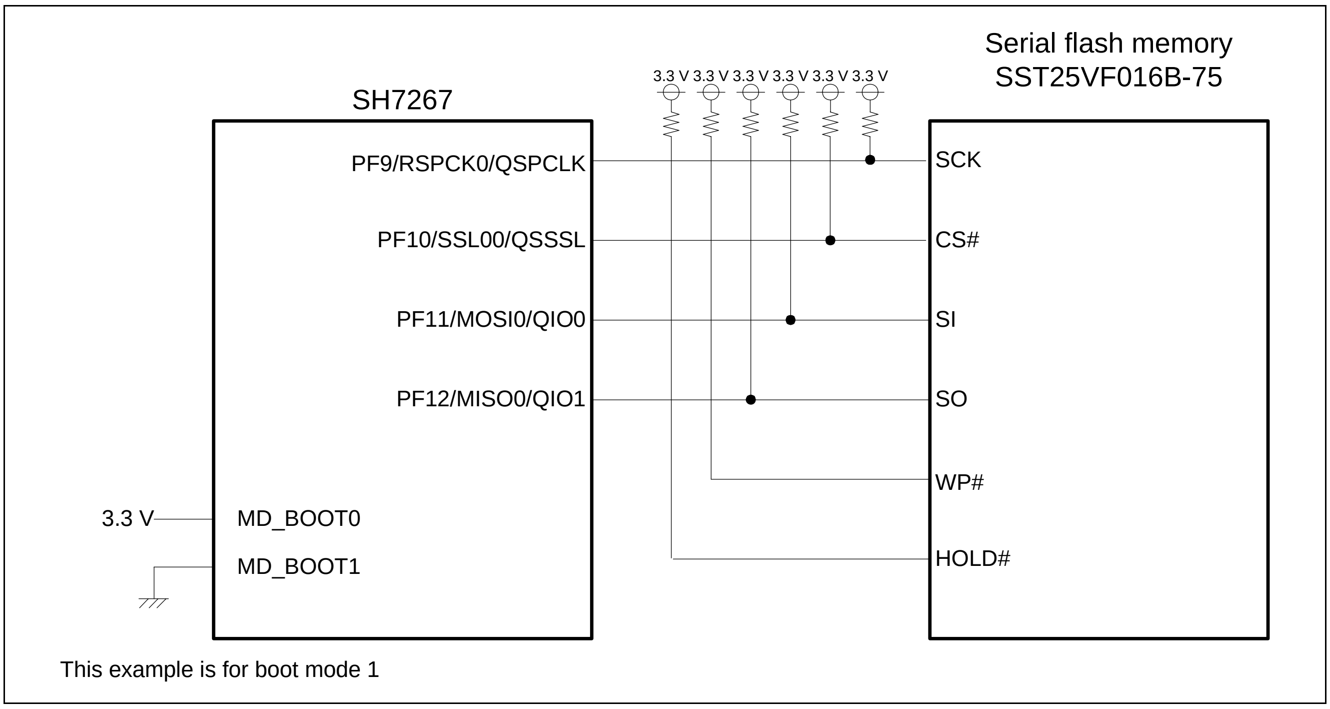


Figure 4 Serial Flash Memory Circuit

Note: The SH7267 uses the RSPI clock at 1/2 of the bus clock rate (Bφ) in boot mode 1, and uses the RSPI clock at 1/4 of the bus clock rate in boot mode 3. Select the boot mode to satisfy the AC characteristics of the serial flash memory and the RSPI.

3. Applications

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

3.1 Loader Program Specifications

The loader program transfers the application program from the serial flash memory to the large-capacity internal RAM, and jumps to the entry function of the application program.

3.1.1 Memory Map

Figure 5 shows the memory map example of the loader program.

1. The loader program (the program area) is allocated to the address from H'FFF8 0000 to H'FFF8 1AFF.
2. Tentative exception vector table is allocated to the address from H'FFF8 1B00 to H'FFF8 1B4F (For more information, refer to 3.1.5).
3. The loader program (the stack area) is allocated to the address from H'FFF8 1C00 to H'FFF8 1FFF (For more information, refer to 3.1.3).

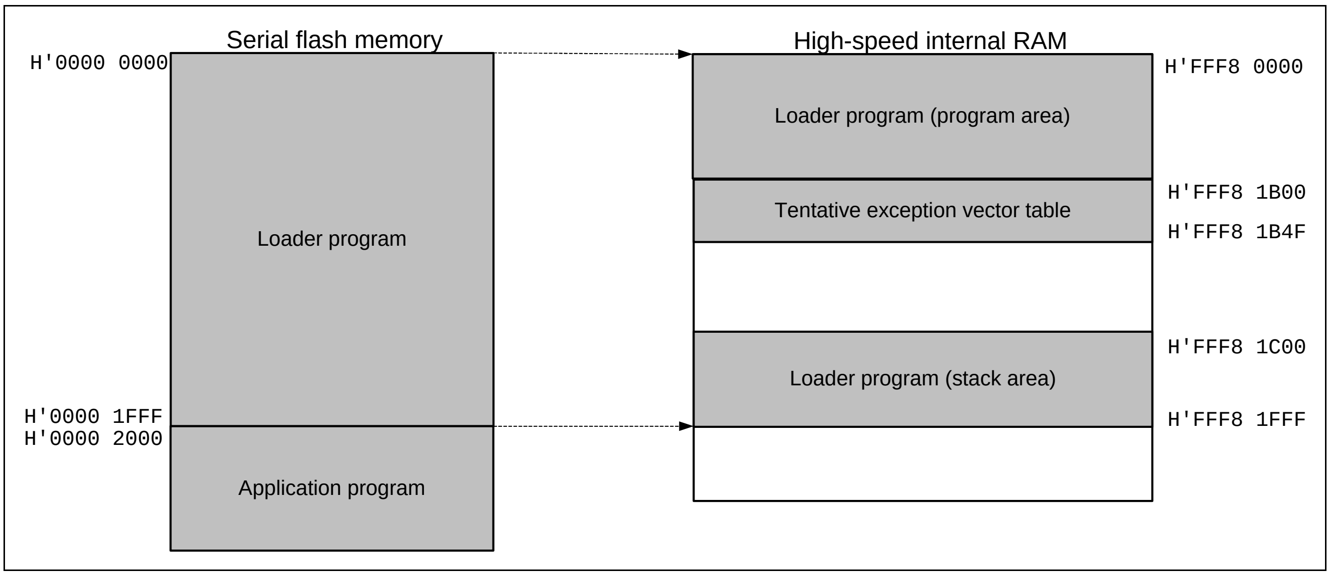


Figure 5 Loader Program Memory Map

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.

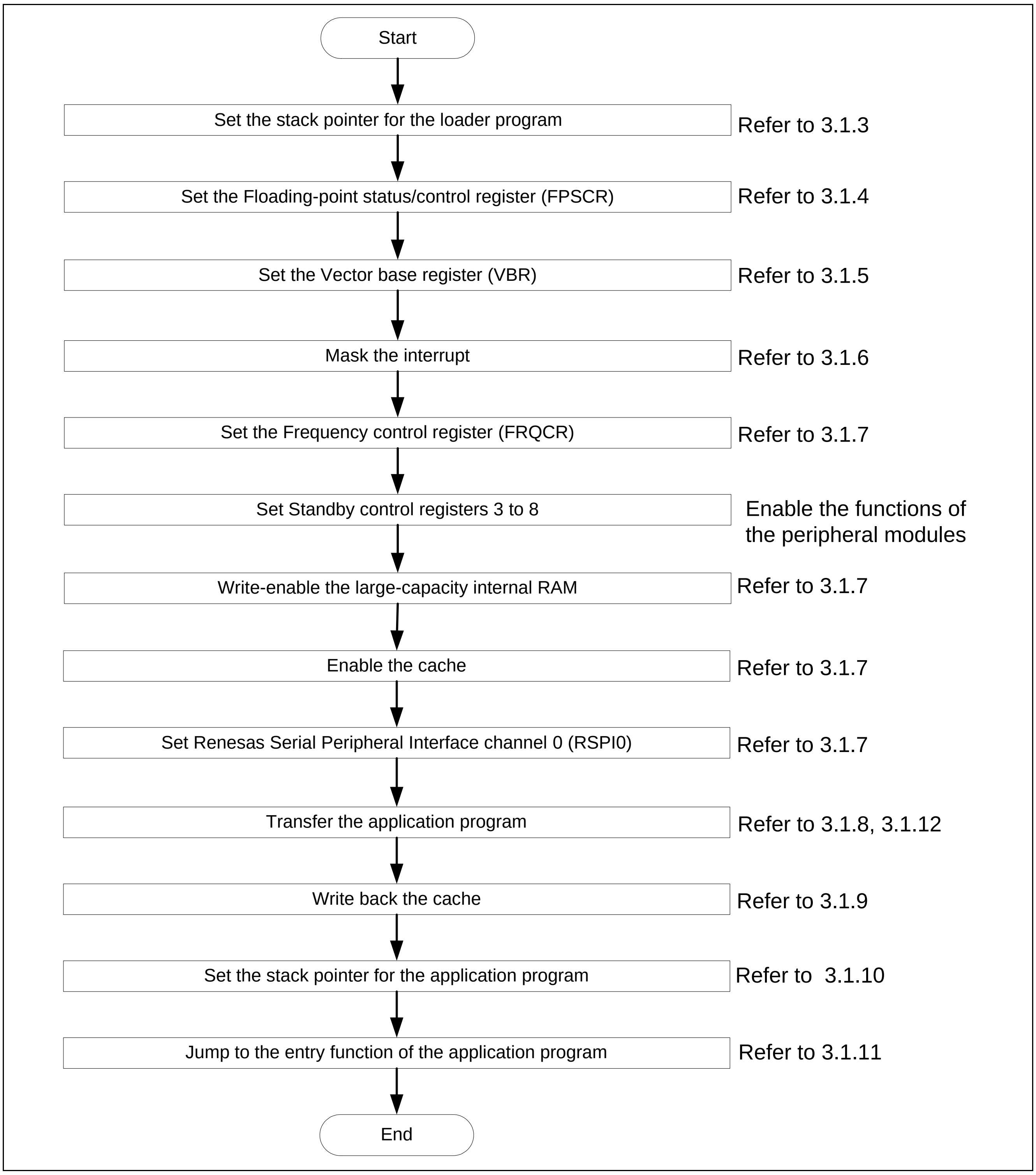

Start
Set the stack pointer for the loader program
Refer to 3.1.3
Set the Floading-point status/control register (FPSCR)
Refer to 3.1.4
Set the Vector base register (VBR)
Refer to 3.1.5
Mask the interrupt
Refer to 3.1.6
Set the Frequency control register (FRQCR)
Refer to 3.1.7
Set Standby control registers 3 to 8
Enable the functions of the peripheral modules
Write-enable the large-capacity internal RAM
Refer to 3.1.7
Enable the cache
Refer to 3.1.7
Set Renesas Serial Peripheral Interface channel 0 (RSPI0)
Refer to 3.1.7
Transfer the application program
Refer to 3.1.8, 3.1.12
Write back the cache
Refer to 3.1.9
Set the stack pointer for the application program
Refer to 3.1.10
Jump to the entry function of the application program
Refer to 3.1.11
End

Figure 6 Loader Program Flow Chart

3.1.3 Stack Pointer Setting

Set the stack pointer (R15) to the address H'FFF8 2000. Allocate the loader program processing at the address H'FFF8 0000, and use 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 of the loader program.

3.1.4 Floating-Point Status Control Register (FPSCR) Setting

Specify the FPSCR at the address H'0004 0001 (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 exception during the loader program is operating. Do not generate exceptions or interrupts before setting the VBR, as the exception vector table is undefined. As the loader program does not use the interrupt, only vector numbers 0 to 18 are defined in the tentative exception vector table. To embed the exception such as the external interrupt during the loader program is operating, extend the tentative exception vector table.

Note: Store the exception vector table on memory and allow the CPU to access the memory before executing exception. For more information, refer to 6.9.4 "Note before Exception Handling Begins Running" in the SH7266 Group, SH7267 Group User’s Hardware Manual.

3.1.6 Interrupt Mask

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

3.1.7 Configuration

Configure the peripheral functions to read the application program from the serial flash memory.

3.1.8 Transferring the Application Program

The loader program refers the application program transfer information (appinfo) in the serial flash memory, and transfers the application program to the large-capacity internal RAM. The table below lists the appinfo in detail. Allocate the appinfo at the address H'0000 2000 in the serial flash memory. The loader program handles the information from H'0000 2000 to H'0000 2007 in the serial flash memory as the appinfo.

Table 3 Application Program Transfer Information (appinfo)

Item	Address	Size
Destination start address	H'0000 2000	4
Destination end address	H'0000 2004	4

The figure below shows the transfer image of the application program using the appinfo. Refer to 3.2.7 for the procedures to generate the appinfo,

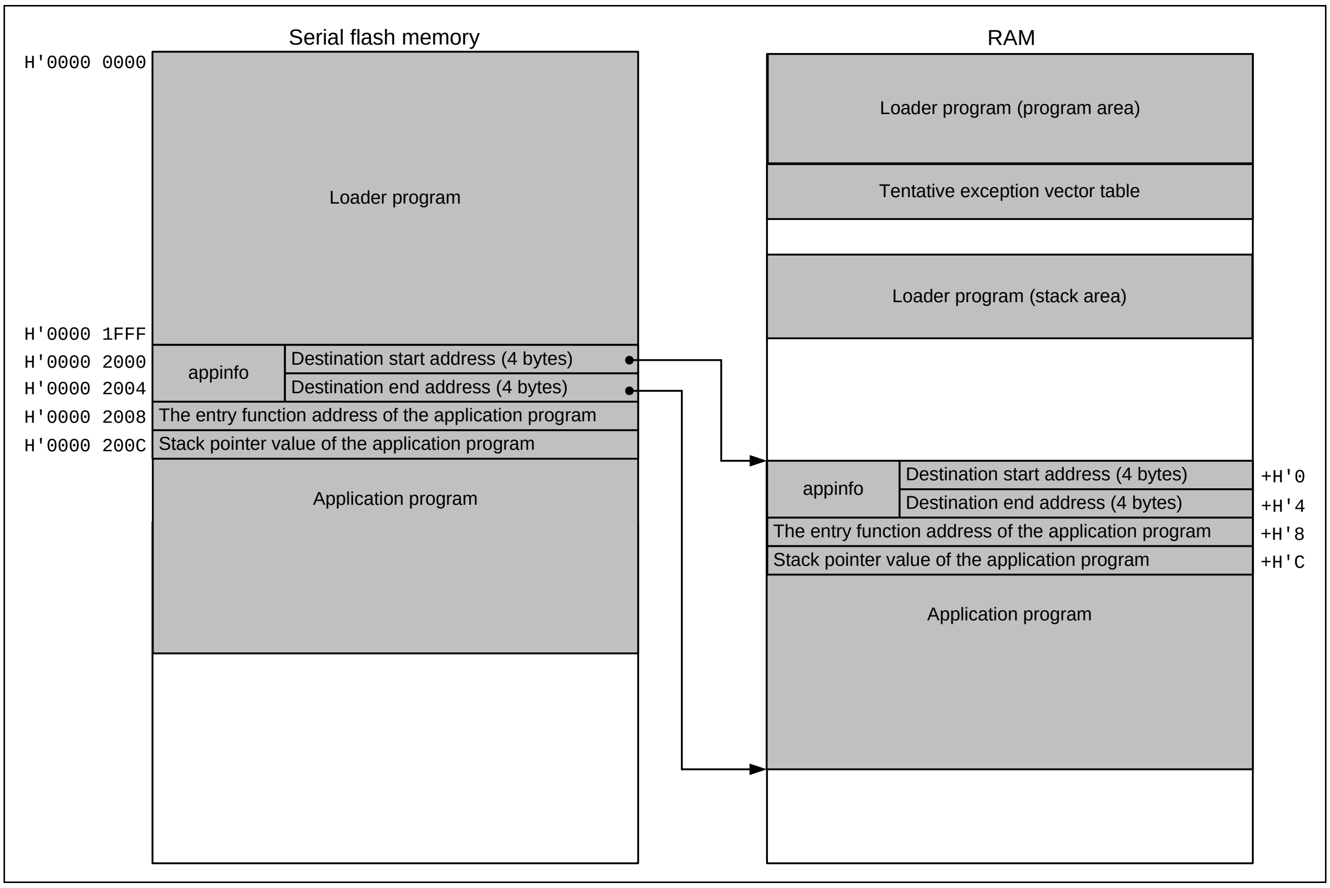

Serial flash memory
RAM
H'0000 0000
Loader program
H'0000 1FFF
H'0000 2000
H'0000 2004
H'0000 2008
H'0000 200C
appinfo
Destination start address (4 bytes)
Destination end address (4 bytes)
The entry function address of the application program
Stack pointer value of the application program
Application program
Loader program (program area)
Tentative exception vector table
Loader program (stack area)
appinfo
Destination start address (4 bytes)
Destination end address (4 bytes)
The entry function address of the application program
Stack pointer value of the application program
Application program
+H'0
+H'4
+H'8
+H'C

Figure 7 Application Program Transfer Image

3.1.9 Writing Back the Cache

After transferring the application program to the large-capacity internal RAM, the loader program writes back the cache to guarantee the coherency between the cache memory and the large-capacity internal RAM.

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 Serial Flash Memory Commands

A set of commands are used to access the serial flash memory. The loader program uses the High-Speed Read command in the serial flash memory to read the application program from the serial flash memory, and transfers the program to the large-capacity internal RAM. Table 4 lists the serial flash memory command used in the loader program.

Table 4 Serial Flash Memory Command

Command Name	Opcode	Function
High-Speed Read	H'0B	Reads the data

Note: This application refers the commands of the SST SST25VF016B. As the serial flash memory commands depend on the type of the serial flash memory, refer to the datasheet provided by the serial flash memory manufacturer.

3.1.13 Register Status After Executing the Loader Program

Table 5 and Table 6 list the register status after executing the loader program. Registers not included in the tables are set as default in the SH7266 Group, SH7267 Group User's Hardware Manual.

Table 5 Register Status (1/2)

Register Name	Abbreviation	Value	Remarks
General registers	R0 to R14	Undefined	
Program counter	PC	Depends on the setting	Application program entry function address
Stack pointer	SP (R15)	Depends on the setting	Stack pointer value in the application program
Status register	SR	Undefined	IMASK bit = B'1111
Vector base register	VBR	H'FFF8 1B00	
Floating-point status/control register	FPSCR	H'0004 0001	Single precision operation, round to 0
Frequency control register	FRQCR	H'1103	
Standby control register 3	STBCR3	H'02	
Standby control register 4	STBCR4	H'00	
Standby control register 5	STBCR5	H'10	
Standby control register 6	STBCR6	H'00	
Standby control register 7	STBCR7	H'2A	
Standby control register8	STBCR8	H'7F	
System control register 5	SYSCR5	H'0F	Write-enable the large-capacity internal RAM
Cache control register1	CCR1	H'0000 0101	Instruction cache and operand cache are valid
Control register_0	SPCR_0	H'88	
Slave select polarity register_0	SSLP_0	H'00	
Pin control register_0	SPPCR_0	H'30	
Status register_0	SPSR_0	H'E0	
Data register_0	SPDR_0	Undefined	
Sequence control register_0	SPSCR_0	H'00	
Sequence status register_0	SPSSR_0	H'00	
Bit rate register_0	SPBR_0	H'00	
Data control register_0	SPDCR_0	H'20	
Clock delay register_0	SPCKD_0	H'00	
Slave select negation delay register_0	SSLND_0	H'00	
Command register_00	SPCMD_00	H'E781	
Command register_01	SPCMD_01	H'070D	
Command register_02	SPCMD_02	H'070D	
Command register_03	SPCMD_03	H'070D	
Buffer control register_0	SPBFCR_0	H'00	
Buffer data count set register_0	SPBFDR_0	H'0002	

Table 6 Register Status (2/2)

Register Name	Abbreviation	Value	Remarks
DMA source address register_0	SAR_0	H'FFFF 8004	
DMA destination address register_0	DAR_0	H'1C00 1ED0	
DMA channel control register_0	CHCR_0	H'0008 0000	
DMA operation register	DMAOR	H'0001	
DMA extension resource selector 0	DMARS0	H'0052	
Port F control register 3	PFCR3	H'0003	QMI/QIO1 pin
Port F control register 2	PFCR2	H'3330	QMO/QIO0 pin, QSSL pin, QSPCLK pin

3.2 Application Program Example

As the loader program transfers the application program from the serial flash memory to the large-capacity internal RAM, the memory map of the application program must be allocate to allow the loader program to read. Also note that the application program must include the address information that is referred by the loader program.

This section describes how to create the application program for the serial flash boot.

3.2.1 Section Alignment

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

1. As an application program is executed on RAM, sections of the application program are located on the large-capacity internal RAM in this example.

2. As the loader program uses the start address and end address of the application program to transfer the application program from the serial flash memory to the large-capacity internal RAM, allocate the program area, constant area and initialized data area of the application program to the physically contiguous area. Uninitialized data area and stack area can be allocated at a desired address.

3. Allocate the appinfo, application program entry function address, and stack pointer value at fixed address. Place the appinfo in section DAPPINFO, application program entry function address, and stack pointer value in section DVECTTBL. Allocate section DAPPINFO at the start on RAM, and then allocate section DVECTTBL.

4. As the loader program uses from H'FFF8 0000 to H'FF8 1FFF in the high-speed internal RAM, do not allocate the program area, constant area, and initialized data area of the application program to that address.
5. Allocate the reset vector table RESET_Vectors in the start address of section DVECTTBL.

6. As the cache operation program area is executed on the cache-disabled space, allocate section PCACHE again to section RPCACHE on the high-speed internal RAM, and execute the section.

Figure 8 shows an example of the section alignment in RAM.

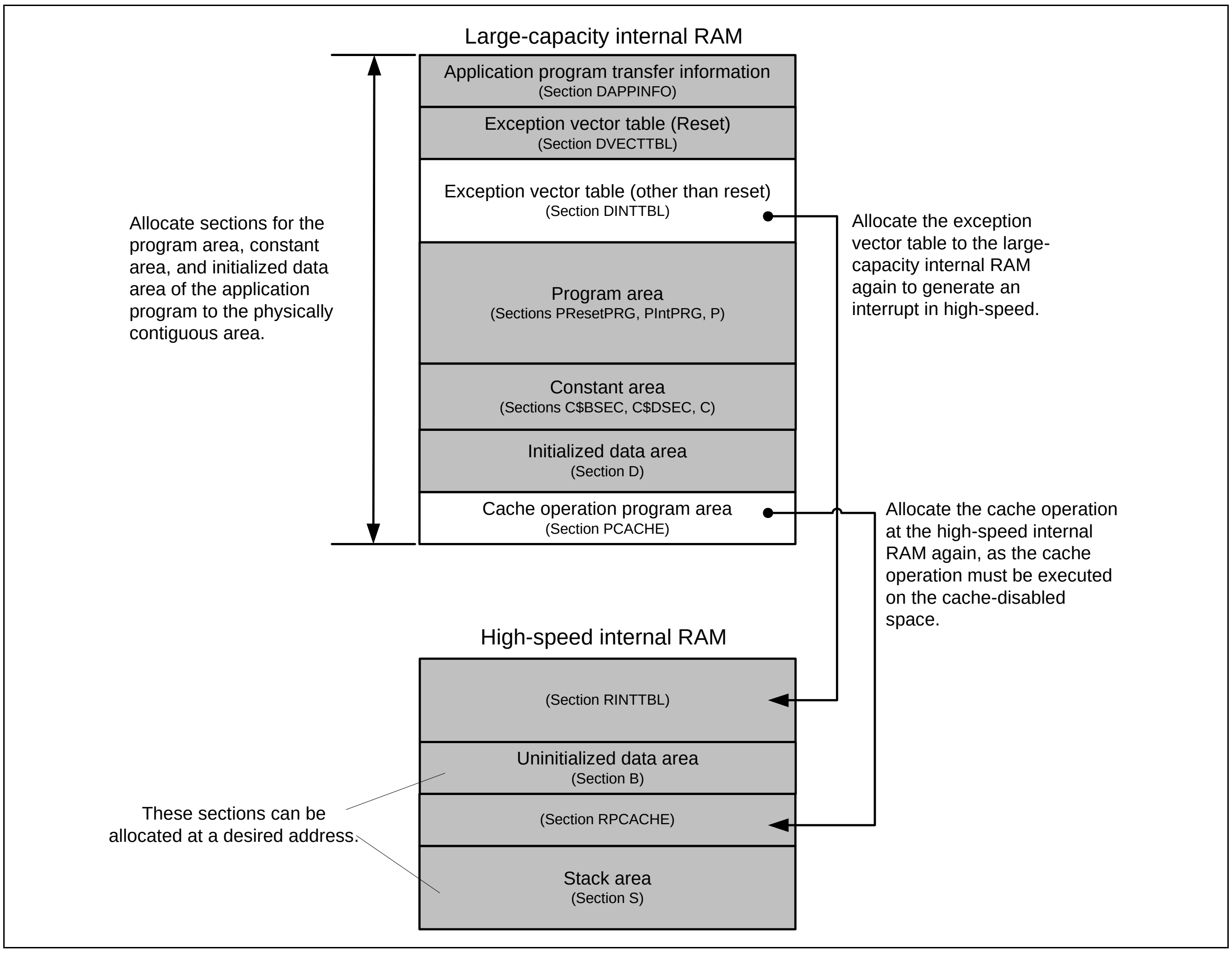

Large-capacity internal RAM
Application program transfer information
(Section DAPPINFO)
Exception vector table (Reset)
(Section DVECTTBL)
Exception vector table (other than reset)
(Section DINTTBL)
Program area
(Sections PResetPRG, PIntPRG, P)
Constant area
(Sections C$BSEC, C$DSEC, C)
Initialized data area
(Section D)
Cache operation program area
(Section PCACHE)
Allocate sections for the program area, constant area, and initialized data area of the application program to the physically contiguous area.
Allocate the exception vector table to the large-capacity internal RAM again to generate an interrupt in high-speed.
Allocate the cache operation at the high-speed internal RAM again, as the cache operation must be executed on the cache-disabled space.
High-speed internal RAM
(Section RINTTBL)
Uninitialized data area
(Section B)
(Section RPCACHE)
Stack area
(Section S)
These sections can be allocated at a desired address.

Figure 8 Application Program Section Alignment

3.2.2 Flow Chart

The application program in this application transmits the strings of characters to channel 0 of the Serial Communication Interface with FIFO (SCIF0). Figure 9 shows the flow chart of the application program.

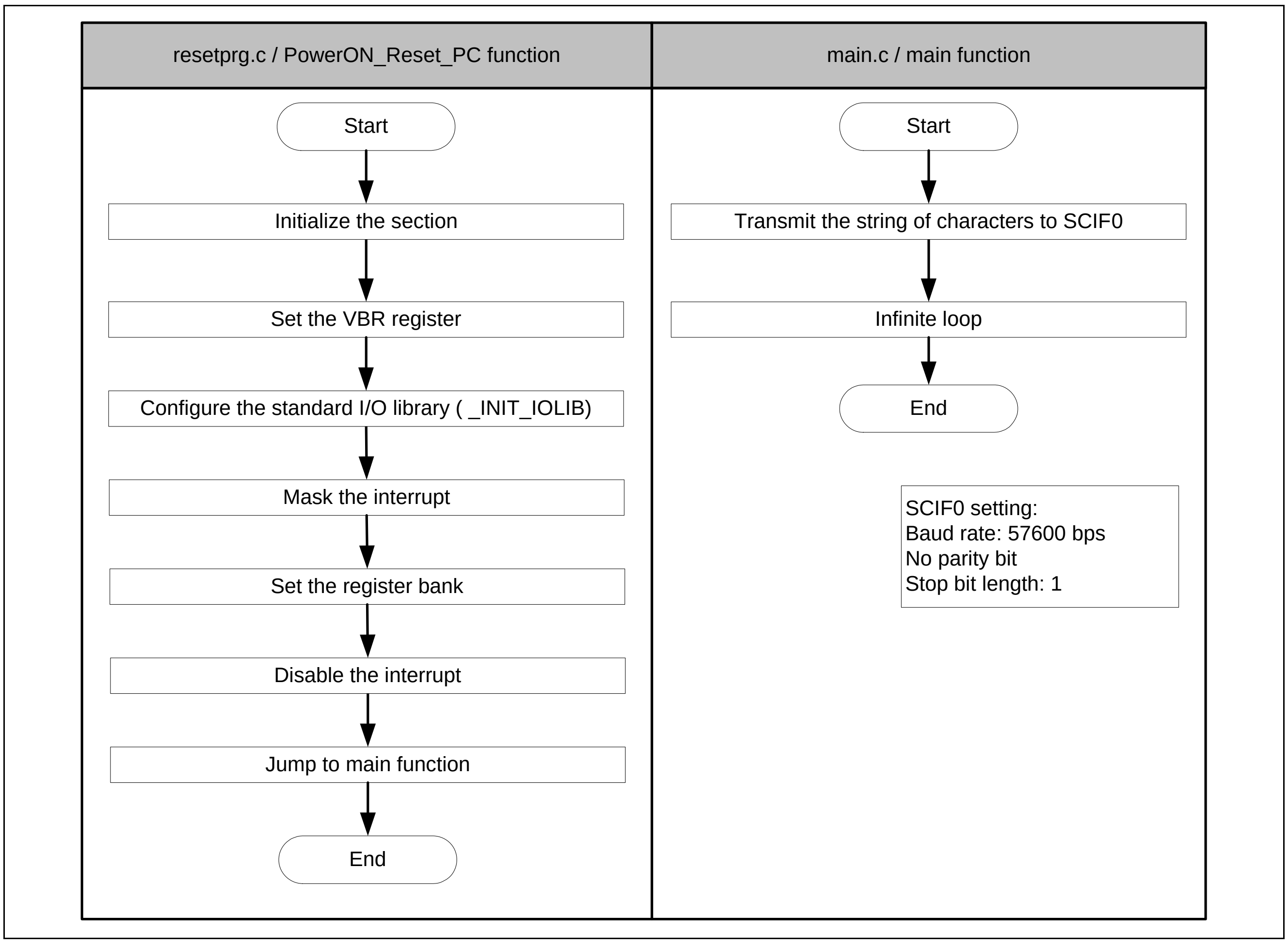


Figure 9 Application Program Flow Chart

3.2.3 Entry Function Setting

Set the entry function address of the application program to table number 0 of the reset vector table RESET_Vectors. Table 7 lists the setting.

Table 7 Entry Function Address Setting

Item	Description
File Name	vecttbl.c
Name of section to place	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

Set the stack pointer of the application program to table number 1 of the reset vector table RESET_Vectors. Table 8 lists the setting.

Table 8 Stack Pointer Setting

Item	Description
File name	vecttbl.c
Name of section to place	DVECTTBL
Table name	RESET_Vectors
Table number	1
Default	_secend ("S")

3.2.5 Initializing the Section

Initialize the section by executing the section initialization routine (_INITSCT function). To execute the _INITSCT function, use values stored in section initialization tables (DTBL and BTBL) described in the file dbsct.c. After executing the _INITSCT function, write back the cache to guarantee the coherency between the cache memory and the large-capacity internal RAM.

3.2.6 Vector Base Register (VBR) Setting

Set the exception vector table of the application program to VBR.

3.2.7 Generating the Application Program Transfer Information (appinfo)

Table 9 lists the structure to generate the application program transfer information (appinfo). Retrieve the start and end address of the application program by section address operators (_sectop, _secend). Allocate the following structure in section DAPPINFO. Register the start address of the application program (program area, constant area, and initialized data area) in the app_top, and the end address of the application program in the app_end.

Table 9 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
Name of section to place	DAPPINFO		

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

Figure 10 shows the application program transfer information (appinfo) generated image.

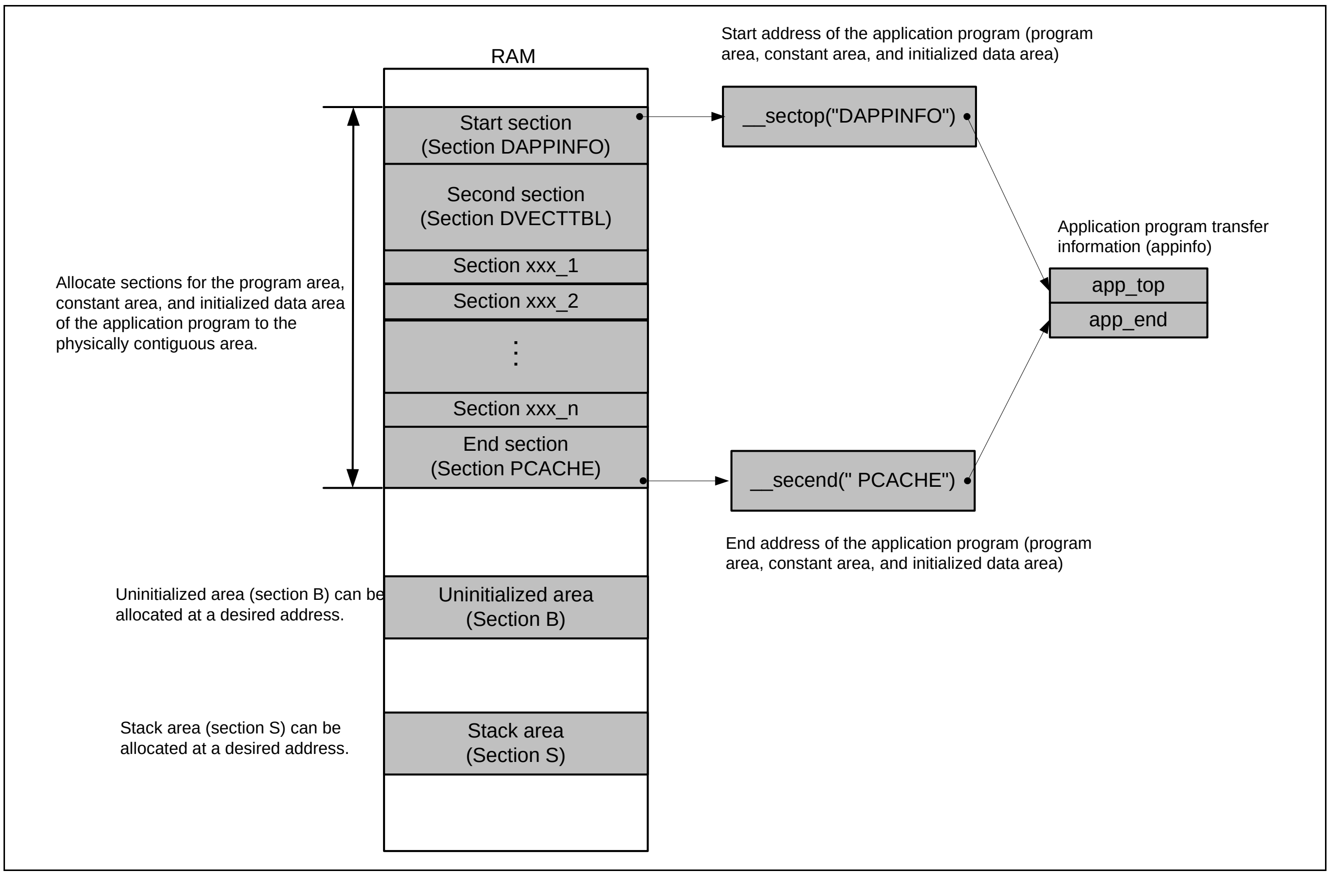

RAM
Start section
(Section DAPPINFO)
Second section
(Section DVECTTBL)
Section xxx_1
Section xxx_2
Section xxx_n
End section
(Section PCACHE)
Uninitialized area
(Section B)
Stack area
(Section S)
Allocate sections for the program area, constant area, and initialized data area of the application program to the physically contiguous area.
Uninitialized area (section B) can be allocated at a desired address.
Stack area (section S) can be allocated at a desired address.
Start address of the application program (program area, constant area, and initialized data area)
__sectop("DAPPINFO")
Application program transfer information (appinfo)
app_top
app_end
__secend(" PCACHE")
End address of the application program (program area, constant area, and initialized data area)

Figure 10 Application Program Transfer Information Generated Image

3.3 Downloader Example

This section describes the downloader in this application.

3.3.1 Operation

Transfer the downloader and the loader program from the development environment to the high-speed internal RAM on system by the debugger, and the application program to the large-capacity internal RAM. Figure 11 shows an operation image of the downloader.

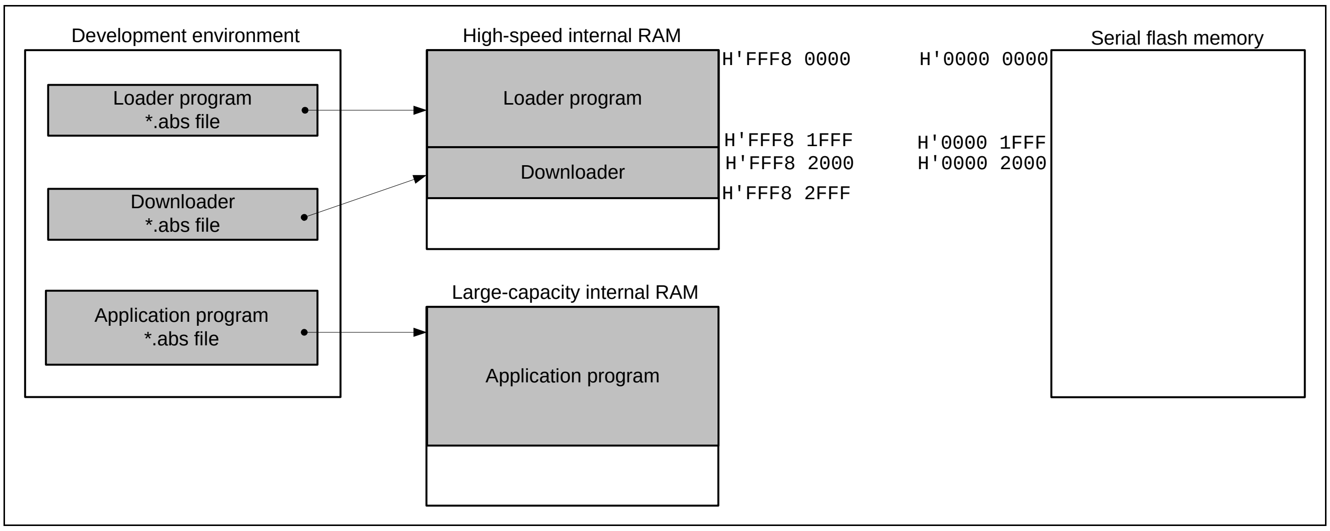


Figure 11 Downloader Operation Image (1/2)

Execute the downloader to write the loader program and the application program in the serial flash memory. The downloader allocates the loader program from H'0000 0000 to H'0000 1FFF, and the application program from H'0000 2000. Figure 12 shows another operation image of the downloader.

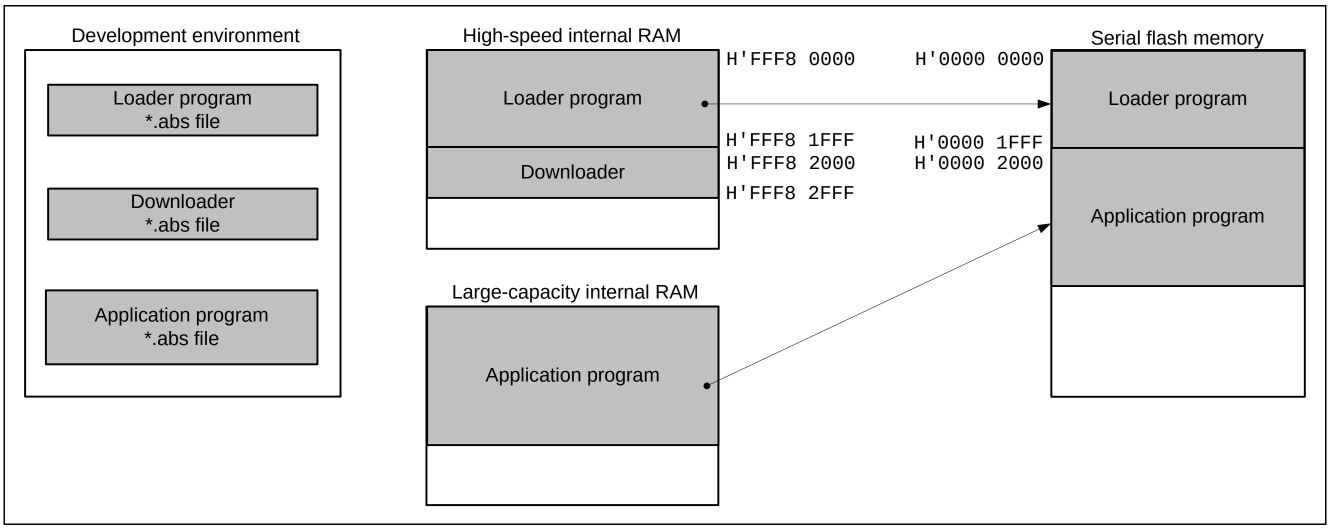


Figure 12 Downloader Operation Image (2/2)

3.3.2 Area Used by the Downloader

The downloader occupies the address from H'FFF8 2000 to H'FFF8 2FFF. When the loader program, application program and downloader occupy the same section, programs do not operate properly.

3.3.3 Flow Chart

Figure 13 shows the flow chart of the downloader. Execute the downloader placed on the high-speed internal RAM to write the loader program and application program in the serial flash memory. For more information, refer to sections 3.3.4 to 3.3.8.

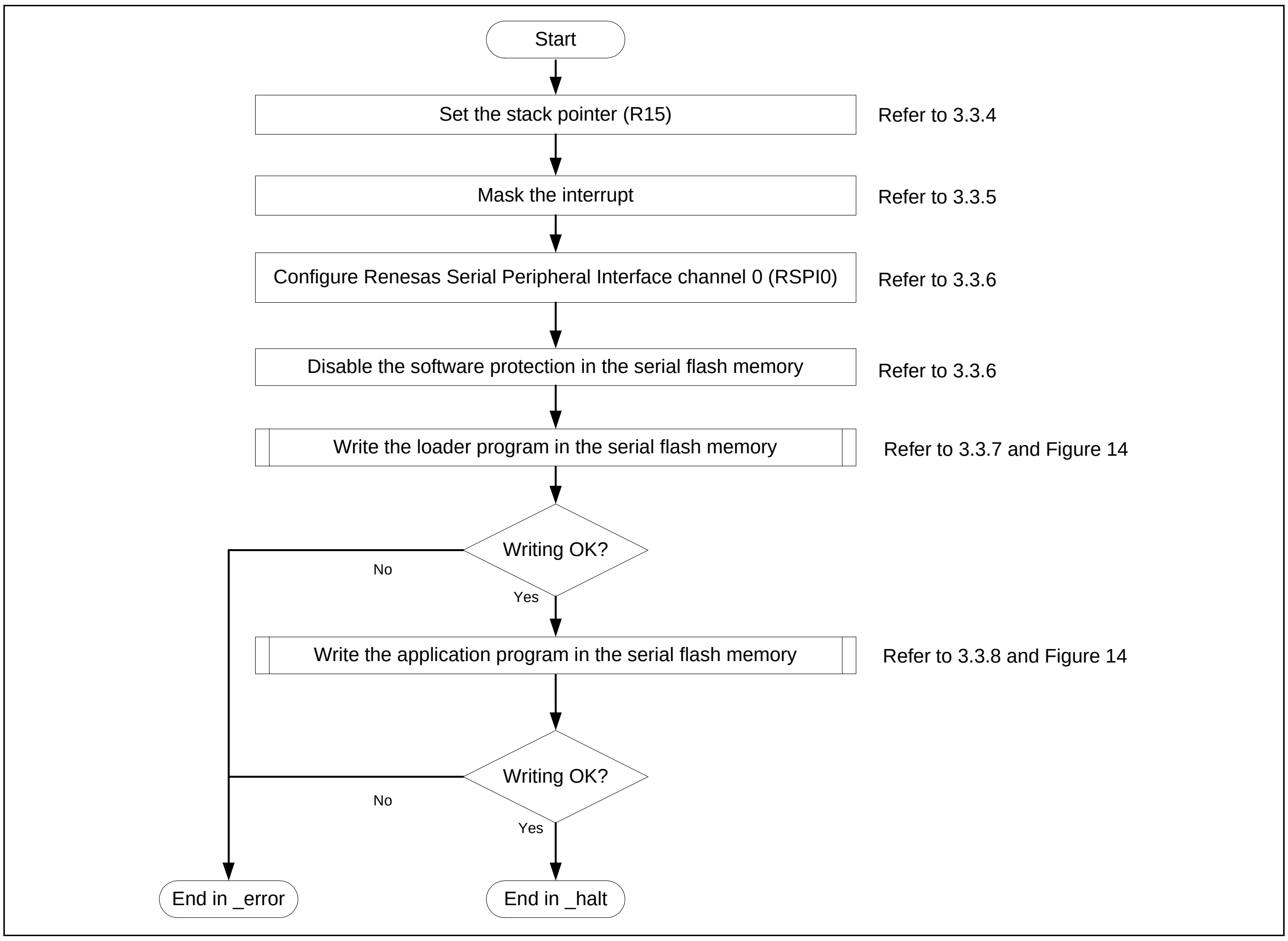

Start
Set the stack pointer (R15)
Refer to 3.3.4
Mask the interrupt
Refer to 3.3.5
Configure Renesas Serial Peripheral Interface channel 0 (RSPI0)
Refer to 3.3.6
Disable the software protection in the serial flash memory
Refer to 3.3.6
Write the loader program in the serial flash memory
Refer to 3.3.7 and Figure 14
Writing OK?
No
Yes
Write the application program in the serial flash memory
Refer to 3.3.8 and Figure 14
Writing OK?
No
Yes
End in _error
End in _halt

Figure 13 Downloader Flow Chart

Figure 14 shows the flow chart of writing the loader program and application program.

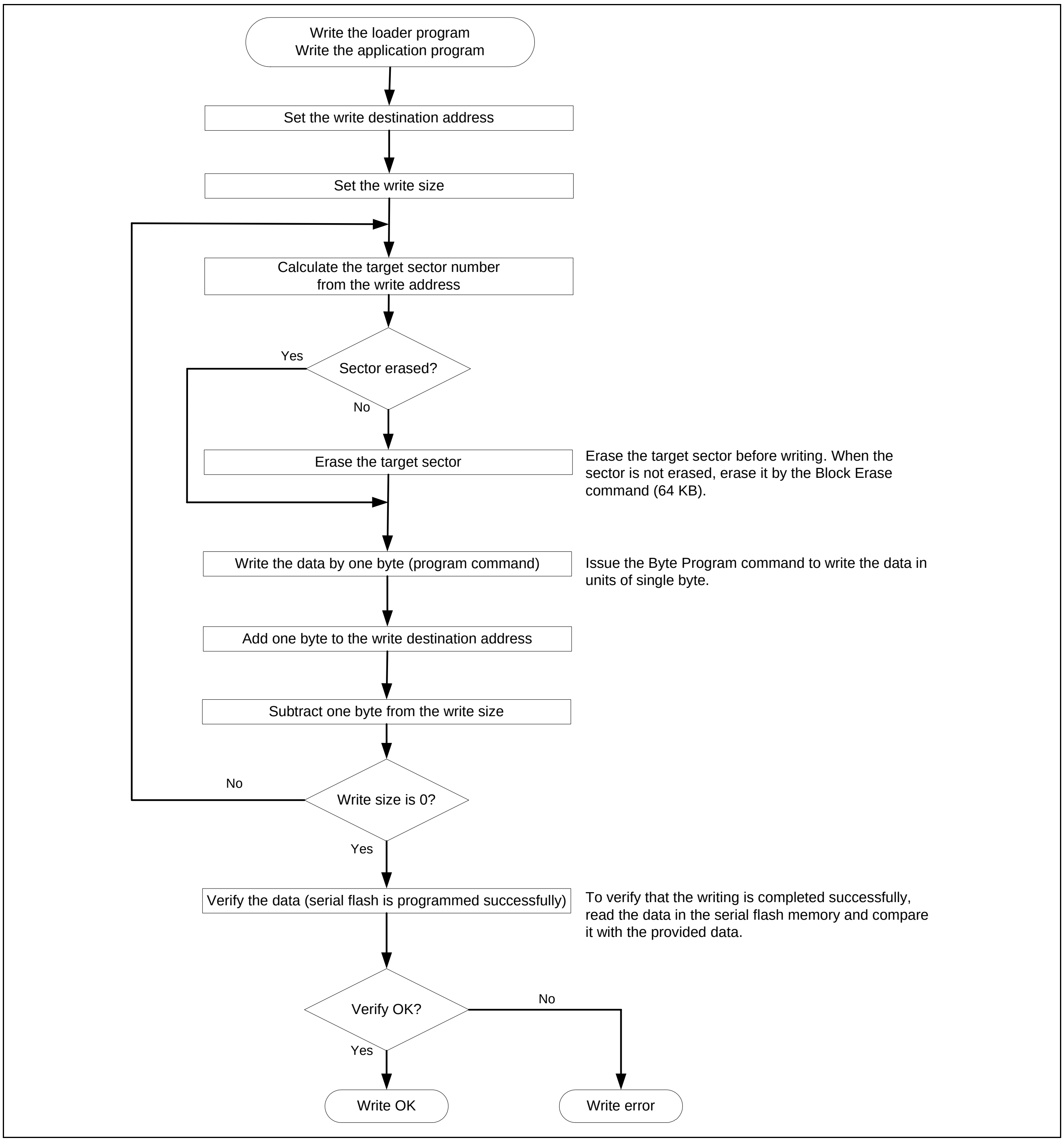

Write the loader program
Write the application program
Set the write destination address
Set the write size
Calculate the target sector number from the write address
Sector erased?
Yes
No
Erase the target sector
Erase the target sector before writing. When the sector is not erased, erase it by the Block Erase command (64 KB).
Write the data by one byte (program command)
Issue the Byte Program command to write the data in units of single byte.
Add one byte to the write destination address
Subtract one byte from the write size
Write size is 0?
No
Yes
Verify the data (serial flash is programmed successfully)
To verify that the writing is completed successfully, read the data in the serial flash memory and compare it with the provided data.
Verify OK?
No
Yes
Write OK
Write error

Figure 14 Flow Chart of Writing

3.3.4 Stack Pointer Setting

Specify the address at H'FFF8 3000 to the stack pointer (R15). Allocate this processing 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 Interrupt Mask

Specify B'1111 in the interrupt mask level bit of the Status register (SR) as the downloader does not support during it is operation.

3.3.6 Initialization

Initialize the serial flash memory before accessing.

1. Configure RSPI0
2. Issue the Write Status Register command to the serial flash memory to disable the software protection (global unprotect).

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 10 lists the loader program writing.

Table 10 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 is already erased 2. Erase the data when the address is not erased 3. Issue the program command to write the loader program in units of single byte

3.3.8 Writing the Application Program

The downloader writes the application program in the large-capacity internal RAM, and writes it at the address from H'0000 2000. Table 11 lists the application program writing.

Table 11 Application Program Writing

Item	Description
Application program transfer source address (large-speed internal RAM)	Retrieve from the appinfo in the application program (Application program dependent)
Application program transfer destination address (serial flash memory)	H'0000 2000 (fixed)
Transfer size	Extracts from the appinfo in the application program
Writing procedures	1. Check if the destination address is already erased 2. Erase the data when the address is not erased 3. Issue the program command to write the application program in units of single byte

3.3.9 Serial Flash Memory Commands

Table 12 lists the serial flash memory commands used in the downloader. Issue these commands via Renesas Serial Peripheral Interface channel 0 (RSPI0) to read, write, and erase the serial flash memory.

Table 12 Serial Flash Memory Commands

Command Name	Opcode	Function
High-Speed Read	H'0B	Reads the data
Write Enable	H'06	Enables to execute the program, erase, write status register command
Write Disable	H'04	Disables to execute the program, erase, write status register command
Read Status Register	H'05	Reads the Status register
Write Status Register	H'01	Writes the data in the Status register (disable the software protection)
64-Kbyte Block Erase	H'D8	Erases the data in blocks (64 KB)
Byte Program	H'02	Programs the data (1 byte)

Notes: 1. This application refers the commands of the SST SST25VF016B. As the serial flash memory commands depend on the type of the serial flash memory, refer to the datasheet provided by the serial flash memory manufacturer.

2. Erase the data in the destination address in the serial flash memory before writing.

3.3.10 Batch 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.

Figure 15 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.

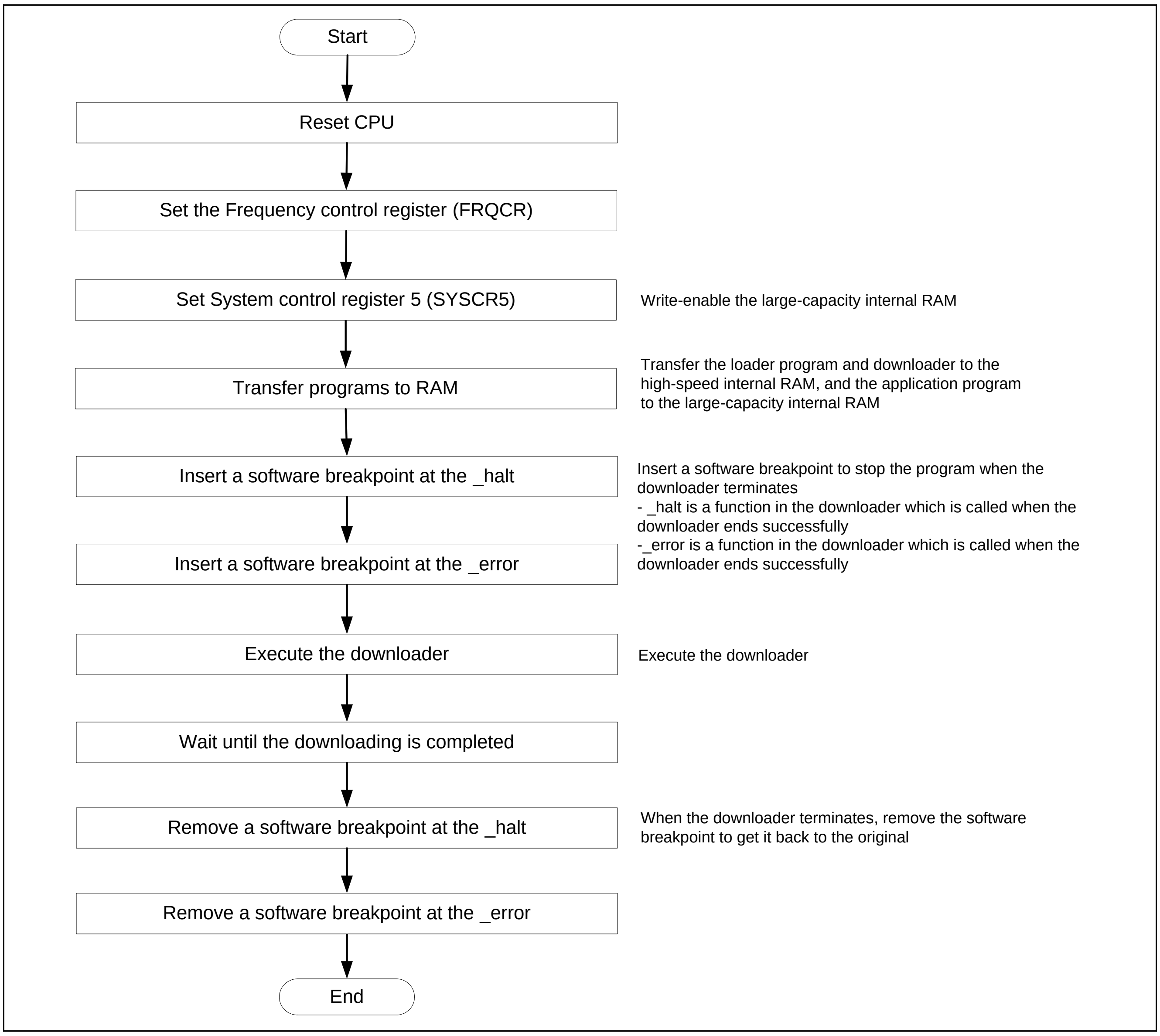

Start
Reset CPU
Set the Frequency control register (FRQCR)
Set System control register 5 (SYSCR5)
Write-enable the large-capacity internal RAM
Transfer programs to RAM
Transfer the loader program and downloader to the high-speed internal RAM, and the application program to the large-capacity internal RAM
Insert a software breakpoint at the _halt
Insert a software breakpoint at the _error
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 successfully
Execute the downloader
Execute the downloader
Wait until the downloading is completed
Remove a software breakpoint at the _halt
When the downloader terminates, remove the software breakpoint to get it back to the original
Remove a software breakpoint at the _error
End

Figure 15 Command Batch File Flow Chart

4. Sample Program Listing

4.1 Loader Program

4.1.1 Loader Program Listing "loader.src" (1/2)

```
;/*******************************************************************************
;*   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 Renesas Electronics Corporation. All rights reserved.
;*""FILE COMMENT""*********** Technical reference data **************************
;*   System Name : SH7266/SH7267 Sample Program
;*   File Name   : loader.src
;*   Abstract    : Loader program preprocessing/jump processing to the application
;*   Version     : 1.00.00
;*   Device      : SH7266/SH7267
;*   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: R0K57267(CPU board)
;*   Description :
;********************************************************************************
;*   History     : Aug.17,2010 Ver.1.00.00 First Release
;*""FILE COMMENT END""**********************************************************/
  .SECTION LOADER_ENTRY,CODE,ALIGN = 4
  .IMPORT _main
  .EXPORT _jmp_app_prog

```

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

```
_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 "main.c" (1/6)

```
/*******************************************************************************
*   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 Renesas Electronics Corporation. All rights reserved.
*""FILE COMMENT""*********** Technical reference data ***************************
*   System Name : SH7266/SH7267 Sample Program
*   File Name   : main.c
*   Abstract    : Loader program
*   Version     : 1.00.00
*   Device      : SH7266/SH7267
*   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: R0K57267(CPU board)
*   Description :
*********************************************************************************
*   History     : Aug.17,2010 Ver.1.00.00 First Release
*""FILE COMMENT END""***********************************************************/
#include <stdio.h>
#include <string.h>
#include <machine.h>
#include "iodefine.h"
#include "serial_flash.h"

/* ==== 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) */
```

4.1.4 Loader Program Listing "main.c" (2/6)

```

#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 io_init_cache(void);
extern int io_cache_writeback(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 "main.c" (3/6)

```
/*""FUNC COMMENT""***************************************************************
 * 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
 *""FUNC COMMENT END""*********************************************************/
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();

  /* Enables the cache */
  io_init_cache();

  /* Sets the RSPI0 */
  sf_init_serial_flash();

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

  /* Transfers the application program to the large-capacity internal RAM */
  app_prog_transfer(app_top, app_end);

  /* Writes back the cache */
  io_cache_writeback();

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

4.1.6 Loader Program Listing "main.c" (4/6)

```

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

/*""FUNC COMMENT""***************************************************************
 * 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
 *""FUNC COMMENT END""**********************************************************/
void get_appinfo( unsigned long *app_top_addr,unsigned long *app_end_addr)
{
  /* Retrieves the appinfo.top */
  sf_byte_read(APPINFO_TOP, (unsigned char *)app_top_addr, 4);

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

```

4.1.7 Loader Program Listing "main.c" (5/6)

```
/*""FUNC COMMENT""***************************************************************
 * 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
 *""FUNC COMMENT END""**********************************************************/
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_long(APROG_TOP_SFLASH, (unsigned long *)app_top_addr, app_prog_size);
}

```

4.1.8 Loader Program Listing "main.c" (6/6)

```
/*""FUNC COMMENT""***************************************************************
 * 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
 *""FUNC COMMENT END""**********************************************************/
void system_down(void)
{
  while(1){
    /* System error */
  }
}

/* End of File */
```

4.2 Application Program

4.2.1 Application Program Listing "main.c" (1/2)

```
/*******************************************************************************
*   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 Renesas Electronics Corporation. All rights reserved.
*""FILE COMMENT""*********** Technical reference data *************************
*   System Name : SH7266/SH7267 Sample Program
*   File Name   : main.c
*   Abstract    : Application program example
*   Version      : 1.00.00
*   Device       : SH7266/SH7267
*   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: R0K57267(CPU board)
*   Description :
********************************************************************************
*   History      : Aug.17,2010 Ver.1.00.00 First Release
*""FILE COMMENT END""**********************************************************/
#include <stdio.h>

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

```

4.2.2 Application Program Listing "main.c" (2/2)

```
/*""FUNC COMMENT""*************************************************************
 * ID           :
 * Outline      : Application program main function
 *-----------------------------------------------------------------------------
 * Include      :
 *-----------------------------------------------------------------------------
 * Declaration : void main(void);
 *-----------------------------------------------------------------------------
 * Description : Transmits the strings of characters to the SCIF0.
 *               : (Baud rate: 57600 bps, no parity, stop bit length: 1).
 *-----------------------------------------------------------------------------
 * Argument     : void
 *-----------------------------------------------------------------------------
 * Return Value: void
 *""FUNC COMMENT END""*********************************************************/
void main(void)
{
    puts("\nSH7267 CPU Board Sample Program. Ver.1.00.00");
    puts("Copyright (C) 2010 Renesas Electronics Corporation. All rights reserved.\n");
    puts("SH7266/SH7267 Serial-flash boot done.\n");
    fflush(stdout);

  while(1){
    /* loop */
  }
}
/* End of File */
```

4.2.3 Application Program Listing "appinfo.c" (1/2)

```
/******************************************************************************
*   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 Renesas Electronics Corporation. All rights reserved.
*""FILE COMMENT""*********** Technical reference data *************************
*   System Name : SH7266/SH7267 Sample Program
*   File Name    : appinfo.c
*   Abstract     : Generate the application program transfer information (appinfo).
*   Version      : 1.00.00
*   Device       : SH7266/SH7267
*   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: R0K57267(CPU board)
*   Description :
*******************************************************************************
*   History      : Aug.17,2010 Ver.1.00.00 First Release
*""FILE COMMENT END""**********************************************************/
```

4.2.4 Application Program Listing "appinfo.c" (2/2)

```
#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 */

```

4.2.5 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) 2010 Renesas Electronics Corporation. All rights reserved.
*""FILE COMMENT""*********** Technical reference data **************************
*   System Name : SH7266/SH7267 Sample Program
*   File Name    : appinfo.h
*   Abstract     : Header file of the application program transfer information (appinfo).
*   Version      : 1.00.00
*   Device       : SH7266/SH7267
*   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: R0K57267(CPU board)
*   Description :
********************************************************************************
*   History      : Aug.17,2010 Ver.1.00.00 First Release
*""FILE COMMENT END""**********************************************************/
#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 Downloader

4.3.1 Downloader Program Listing "downloader.hdc" (1/2)

```
#/*******************************************************************************
#*   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 Renesas Electronics Corporation. All rights reserved.
#*""FILE COMMENT""*********** Technical reference data *************************
#*   System Name : SH7266/SH7267 Sample Program
#*   File Name   : downloader.hdc
#*   Abstract    : Batch File for the Downloader
#*   Version     : 1.00.00
#*   Device      : SH7266/SH7267
#*   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: R0K57267(CPU board)
#*   Description :
#********************************************************************************
#*   History     : Aug.17,2010 Ver.1.00.00 First Release
#*""FILE COMMENT END""**********************************************************/


tcl enable


```

4.3.2 Downloader Program Listing "downloader.hdc" (2/2)

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

  # Set the CPG
  # FRQCR I=144MHz/B=72MHz/P=36MHz/CLK MODE0
  MF H'FFFE0010 H'FFFE0011 H'1103 WORD

  # On-Chip Large-Capacity RAM write enable
  # CPG.SYSCR5.BYTE = 0x0fu;
  MF H'FFFE0428 H'FFFE0428 H'0F BYTE
}


proc downloader {} {
  # Reset CPU
  reset

  # Calls the init_hardware routine
  init_hardware

  # Downloads all modules registered in the High-performance Embedded Workshop
  file_load_all

  # Enables the user stack (to use the software breakpoint)
  sh2a_sbstk enable

  # Inserts a software breakpoint at the _halt (refer to main.c)
  set_disassembly_soft_break _halt set

  # Inserts a software breakpoint at the _error (refer to main.c)
  set_disassembly_soft_break _error set

  # Executes the _downloader (refer to downloader.src) to wait until it terminates
  go wait _downloader

  # Removes a software breakpoint at the _halt
  set_disassembly_soft_break _halt clear

  # Removes a software breakpoint at the _error
  set_disassembly_soft_break _error clear

}

downloader
#Macro downloader -End



# Note: "tcl", "reset", "file_load", "sh2a_sbstk", "set_disassembly_soft_break",
# and "go" are commands used in the High-performance Embedded Workshop and the
# E10A-USB emulator. For details, refer to manuals.

# /* End of File */
```

4.3.3 Downloader Program Listing "downloader.src" (1/2)

```
;/*******************************************************************************
;*   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 Renesas Electronics Corporation. All rights reserved.
;*""FILE COMMENT""*********** Technical reference data *************************
;*   System Name : SH7266/SH7267 Sample Program
;*   File Name   : downloader.src
;*   Abstract     : Downloader
;*   Version      : 1.00.00
;*   Device       : SH7266/SH7267
;*   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: R0K57267(CPU board)
;*   Description :
;********************************************************************************
;*   History     : Aug.17,2010 Ver.1.00.00 First Release
;*""FILE COMMENT END""**********************************************************/
```

4.3.4 Downloader Program Listing "rownloader.src" (2/2)

```
   .SECTION DOWNLOADER_ENTRY,CODE,ALIGN = 4
   .IMPORT _main

_downloader:
   MOV.L L2,R15        ; Sets the stack pointer
   MOV.L L1,R0         ; Retrieves the entry function of the downloader
   JMP @R0             ; Jumps to the entry function of the downloader
   NOP

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

L2:
   .DATA.L H'FFF83000  ; Stack pointer (R15) value of the downloader

   .pool
   .end
```

4.3.5 Downloader Program Listing "main.c" (1/8)

```
/*******************************************************************************
*   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 Renesas Electronics Corporation. All rights reserved.
*""FILE COMMENT""*********** Technical reference data *************************
*   System Name : SH7266/SH7267 Sample Program
*   File Name   : main.c
*   Abstract    : Downloader
*   Version     : 1.00.00
*   Device       : SH7266/SH7267
*   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: R0K57267(CPU board)
*   Description :
********************************************************************************
*   History     : Aug.17,2010 Ver.1.00.00 First Release
*""FILE COMMENT END""**********************************************************/
#include <stdio.h>
#include <string.h>
#include <machine.h>
#include "iodefine.h"
#include "serial_flash.h"

```

4.3.6 Downloader Program Listing "main.c" (2/8)

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

#define SECTOR_SIZE    0x10000             /* Sector size: 64 KB */
#define SECTOR_NUM 32                     /* Total number of sectors in the device */
#define DEVICE_SIZE    (SECTOR_SIZE * SECTOR_NUM) /* Device size */

#define L_PROG_SIZE    8192               /* Loader program size */
#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      0x1C000000     /* 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}; /* 0: sector not erased, 1:
sector erased */
```

4.3.7 Downloader Program Listing "main.c" (3/8)

```
/*""FUNC COMMENT""*************************************************************
 * 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
 *""FUNC COMMENT END""**********************************************************/
void main(void)
{
  unsigned long app_top_addr,app_end_addr,app_prog_size;

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

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

  /* Initializes the RSPI0 */
  sf_init_serial_flash();

  /* Disables the software protection in serial flash memory */
  sf_protect_ctrl(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 Listing "main.c" (4/8)

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

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

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

    /*""FUNC COMMENT""***************************************************************
     * 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
     *""FUNC COMMENT END""**********************************************************/
```

4.3.9 Downloader Program Listing "main.c" (5/8)

```
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;
  unsigned char *w_p;

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

    sf_byte_program(saddr, w_p, 1);           /* Writes data in units of single byte */
    w_p++;
    saddr++;
    sz--;
  }

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

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

    if( *w_p != read_data ){
        return -1;                    /* Returns an error when the data unmatched */
    }

    w_p++;
    saddr++;
    sz--;
  }

  return 0;
}
```

4.3.10 Downloader Program Listing "main.c" (6/8)

```
/*""FUNC COMMENT""*************************************************************
 * 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
 *""FUNC COMMENT END""*********************************************************/
static void init_erase_flag(void)
{
  int i;

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

/*""FUNC COMMENT""*************************************************************
 * 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
 *""FUNC COMMENT END""*********************************************************/
static int Is_erased_sector(unsigned long sector_no)
{
  return sflash_erase_flag[sector_no];
}

```

4.3.11 Downloader Program Listing "main.c" (7/8)

```
/*""FUNC COMMENT""*************************************************************
 * 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
 *""FUNC COMMENT END""*********************************************************/
static void set_erase_flag(unsigned long sector_no)
{
  sflash_erase_flag[sector_no] = 1;
}

/*""FUNC COMMENT""*************************************************************
 * 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
 *""FUNC COMMENT END""*********************************************************/
static void halt(void)
{
    while(1){
        /* When the downloader ends successfully, this function stops the program. */
    }
}

```

4.3.12 Downloader Program Listing "main.c" (8/8)

```
/*""FUNC COMMENT""**************************************************************
 * 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
 *""FUNC COMMENT END""**********************************************************/
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 tools, alter the program according to the tool.

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

5.1 Sample Program Configuration

The sample program consists of three workspaces as listed in Table 13.

Table 13 Sample Program Configuration

Workspace Name	Description
sh7267_sflash_downloader	Build the downloader in the project of this workspace
sh7267_sflash_loader_prg	Build the loader program in the project of this workspace
sh7267_sflash_app	Build the application program in the project of this workspace. The downloader which is created in the [sh7267_sflash_downloader] workspace, a batch file to boot the downloader, and the loader program which is created in the [sh7267_sflash_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 [sh7267_sflash_app] workspace.

5.2.1 Registering the Download Module and Batch File

Figure 16 shows the directory configuration of the [sh7267_sflash_app] workspace. Download modules (A, B, and D) and a batch file (C) in the figure area registered in the project.

```
¥sh7267_sflash_app                   : Workspace directory
  |-sh7267_sflash_app                : Project directory
  |   |-debug                        :
  |       |-sh7267_sflash_app.abs    : Application program execute file------------------------------------------- A
  |                                  :
  |-inc                              : Directory to store the common include files
  |-src                              : Directory to store the source files
  |-sflash_boot                      : Directory to store the downloader and loader program
    |-sh7267_sflash_downloader.abs   : Downloader execute file---------------------------------------------------- B
    |-downloader.hdc                 : Batch file to boot the downloader ---------------------------------------- C
    |-sh7267_sflash_loader_prog.abs  : Loader program execute file ---------------------------------------------- D
```

Figure 16 [sh7267_sflash_app] Workspace Directory Configuration

1. Changing the download module

   Change the download module registered in the project in the **Debug Settings** dialog box. On the **Debug** menu in the High-performance Embedded Workshop, click **Debug Settings**, and the dialog box appears.

   For registering the download modules, refer to the High-performance Embedded Workshop User's Manual.

2. Changing the batch file

   Change the batch file registered in the project in the **Set Batch File** dialog box. On the View menu in the High-performance Embedded Workshop, click the **Command Line** command to show the **Command Line** window. Open the **Set Batch File** dialog box from the **Batch File** pop-up menu on the **Command Line** window.

   For registering the batch file, refer to the High-performance Embedded Workshop User's Manual.

5.2.2 Procedures to Writing Programs

This section describes how to write the loader program and application program in the serial flash memory using the [sh7267_sflash_app] workspace.

1. Copy the [sh7267_sflash_app] workspace directory in C:\Workspace.
2. Double-click the [sh7267_sflash_app].hws in the workspace directory to activate the High-performance Embedded Workshop.
3. On the **Build** menu, select the **Build All** command to build the project. The application program is generated.
4. On the **Debug** menu, select the **Go** command to connect with the target device.
5. After the connection is established, select the **Command Line** command to show the **Command Line** window.
6. Click the **Run Batch** button on the **Command Line** window to execute the registered batch file [downloader.hdc].

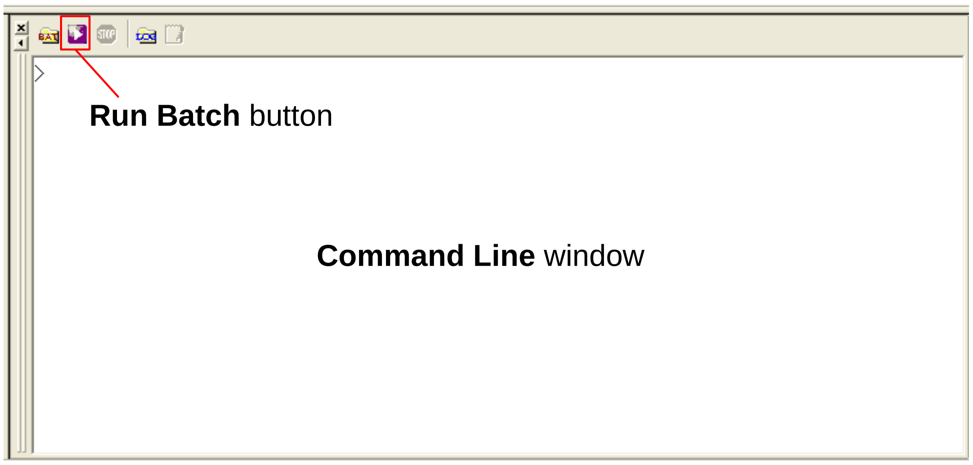
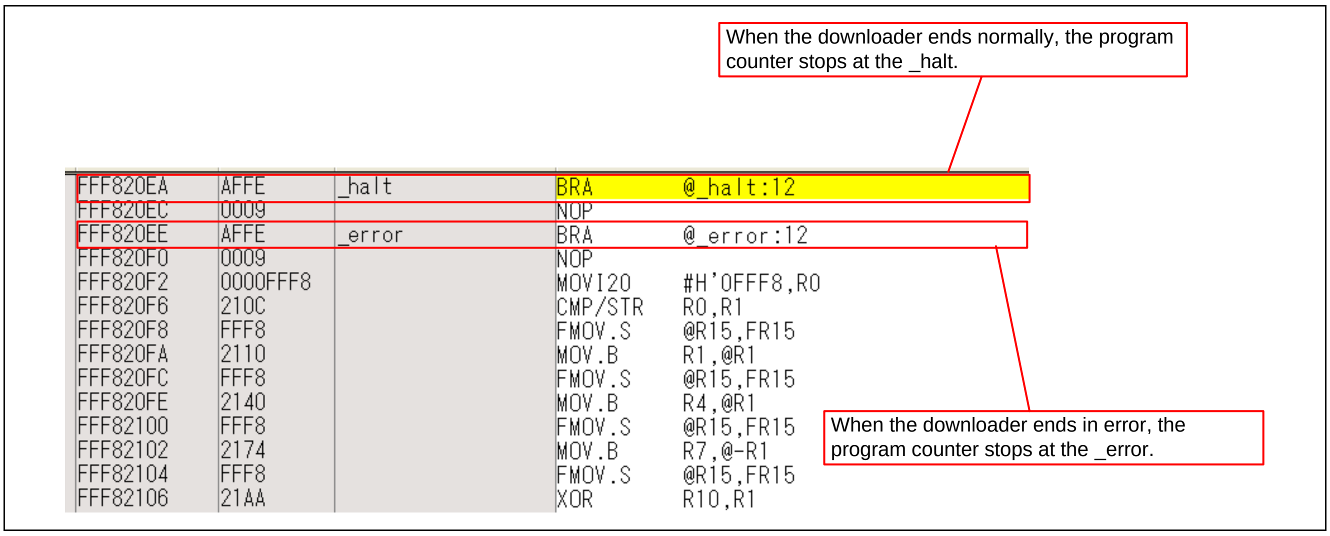


Figure 17 Command Line Window and Run Batch Button

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 show in Figure 18, the program counter stops at the _holt, 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 [sh7267_sflash_downloader] workspace directory is copied in C:\Workspace.
8. When writing is completed successfully, the loader program and application program can be executed after **Reset Go**.



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

Figure 18 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 Hardware Manual
  SH7266 Group, SH7267 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	Dec.27.10	—	First edition issued
1.01	Mar.31.11	4	Changed description

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