To our customers,

Old Company Name in Catalogs and Other Documents

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

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

April 1st, 2010
Renesas Electronics Corporation

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

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

Notice

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

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

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

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

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

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

SH7262/SH7264 Group

Boot from the Serial Flash Memory

Summary

This application note describes the boot from the SH7262/SH7264 microcomputers (MCUs) internal serial flash memory.

Target Device

SH7262/SH7264 (In this document, SH7264/SH7262 are described as "SH7264".)

Contents

1. Introduction

1.1 Specifications

SH7264 MCU boots from an internal serial flash memory (serial boot) in boot mode 1 and boot mode 3. This application note describes the loader program and application program examples when using the serial boot. The downloader to write the loader program and application program to serial flash memory is also described.

1.2 Modules Used

- Boot mode 1
- Renesas Serial Peripheral Interface (RSPI)

1.3 Applicable Conditions

MCU	SH7262/SH7264
Operating Frequency	Internal clock: 144 MHz Bus clock: 72 MHz Peripheral clock: 36 MHz
Integrated Development Environment	Renesas Technology Corp. High-performance Embedded Workshop Ver.4.04.01
C compiler	Renesas Technology SuperH RISC engine Family C/C++ compiler package Ver.9.02 Release 00
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)

1.4 Related Application Note

Refer to the related application notes as follows:

- SH7262/SH7264 Group Example of Initialization
- SH7262/SH7264 Group Interfacing Serial Flash Memory Using the Renesas Peripheral Serial Interface

2. Overview of the Serial boot

This chapter describes the overview of the serial boot.

2.1 Glossary of Terms

The following table lists the terms used in this application note to describe the serial 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, and not required to create.
Loader program	A program to transfer the application program from 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 the system
Downloader	A program to write the loader program and application program to 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 boot Mode

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

In boot mode 1 or boot mode 3, the internal ROM program to boot transfers the loader program from serial flash memory connected to the 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.

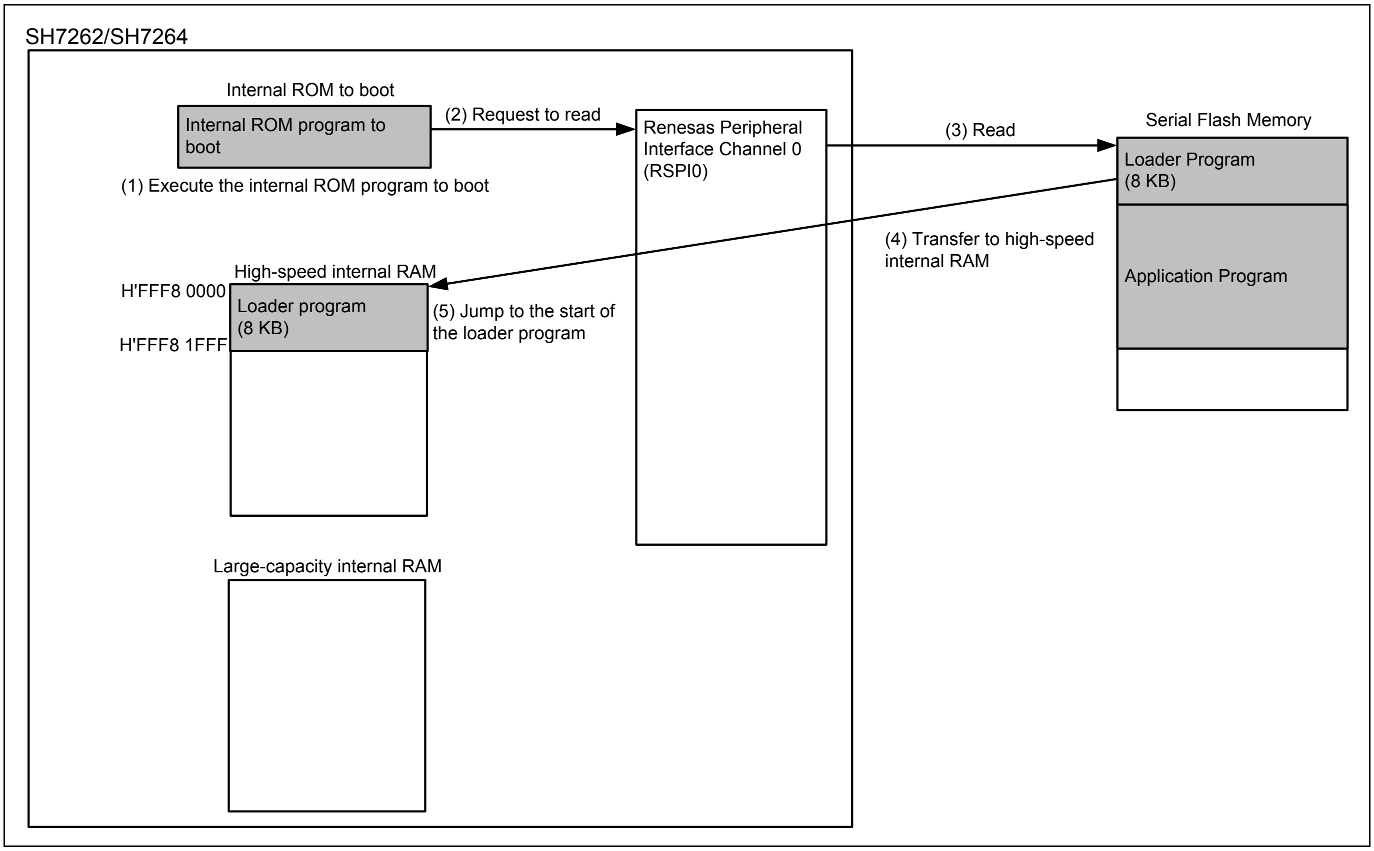

SH7262/SH7264
Internal ROM to boot
Internal ROM program to boot
(2) Request to read
(1) Execute the internal ROM program to boot
Renesas Peripheral Interface Channel 0 (RSPI0)
(3) Read
Serial Flash Memory
Loader Program (8 KB)
Application Program
(4) Transfer to high-speed internal RAM
High-speed internal RAM
H'FFF8 0000
H'FFF8 1FFF
Loader program (8 KB)
(5) Jump to the start of the loader program
Large-capacity internal RAM

Figure 1 Operation Image of the Internal ROM Program to Boot

The loader program transfers the application program from 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.

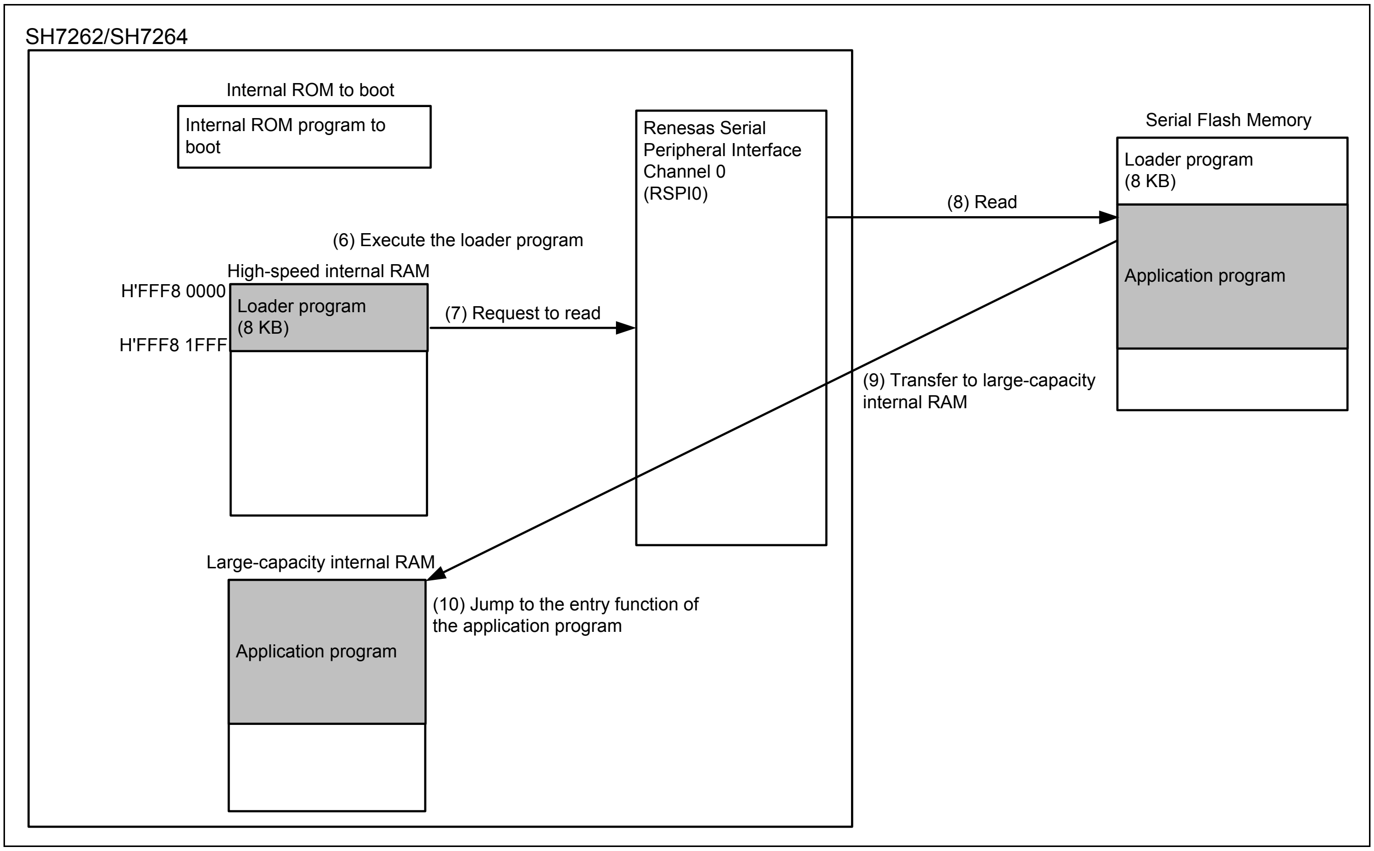

SH7262/SH7264
Internal ROM to boot
Internal ROM program to boot
Renesas Serial Peripheral Interface Channel 0 (RSPI0)
Serial Flash Memory
Loader program (8 KB)
(8) Read
(6) Execute the loader program
High-speed internal RAM
H'FFF8 0000
Loader program (8 KB)
H'FFF8 1FFF
(7) Request to read
Application program
(9) Transfer to large-capacity internal RAM
Large-capacity internal RAM
(10) Jump to the entry function of the application program
Application program

Figure 2 Operation Image of the Loader Program

Note: Application program can be transferred to external RAM such as SDRAM.

2.3 Downloader Operation

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

Refer to 3.3 Downloader Example for details.

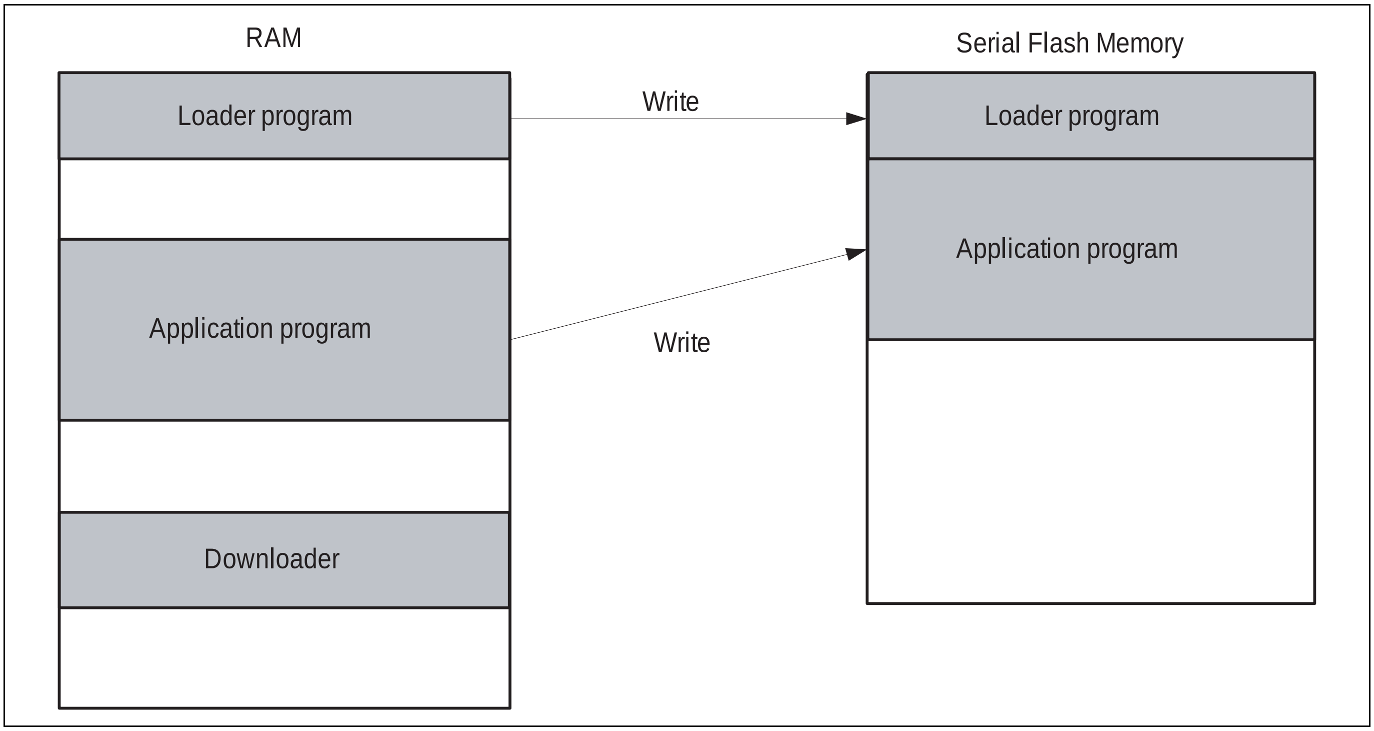


Figure 3 Operation Image of the Downloader

2.4 Serial Flash Memory Connection

The figure below shows the connection circuit for the SH7264 MCU and serial flash memory. Connect serial flash memory to the Renesas Peripheral Interface Channel 0 (RSPI0) to use the serial boot.

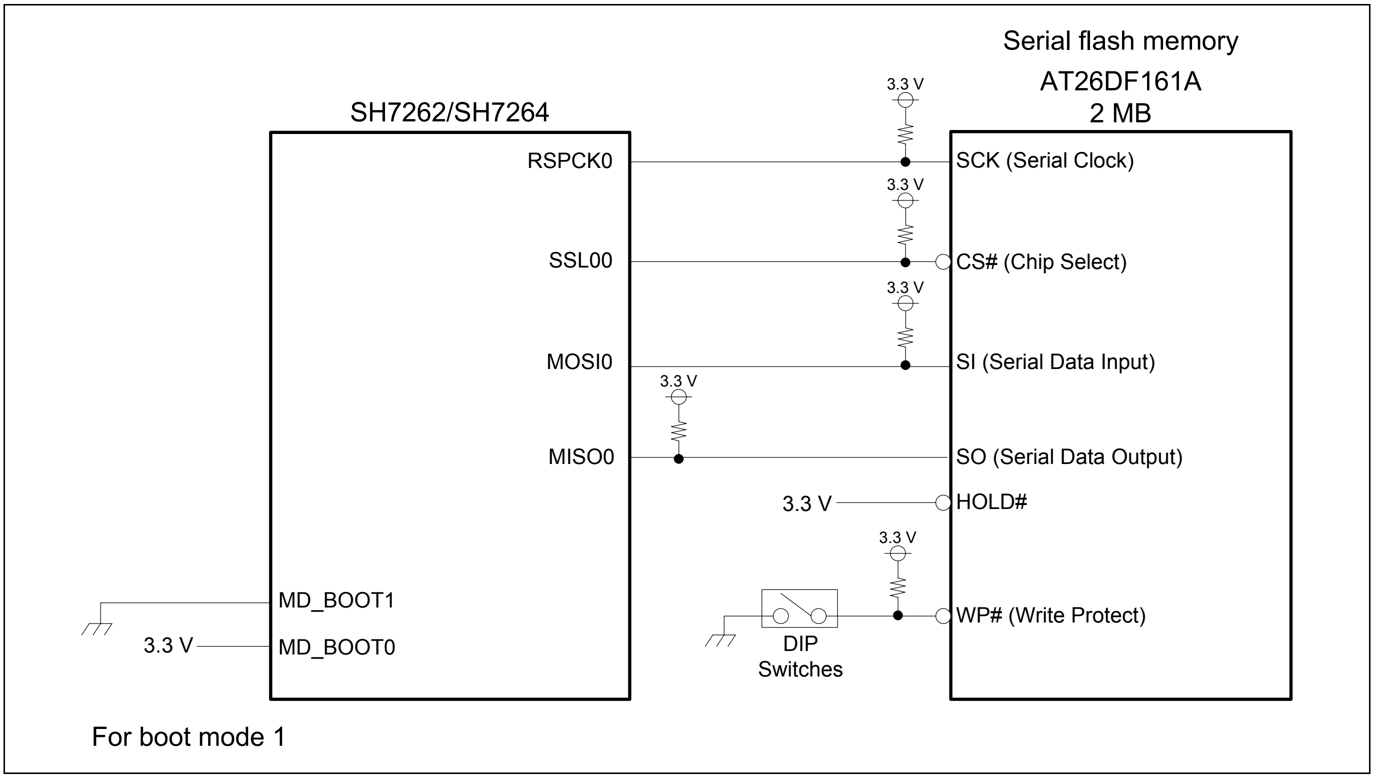


Figure 4 Connection Circuit for the SH7264 and Serial Flash Memory

Note: The SH7264 MCU 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 fulfill the AC characteristics of 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 serial flash memory to the large-capacity internal RAM, and jumps to the entry function of the application program.

3.1.1 Memory Map

The figure below 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 (Refer to 3.1.5 for details).
3. The loader program (the stack area) is allocated to the address from H'FFF8 1C00 to H'FFF8 1FFF (Refer to 3.1.3 for details).

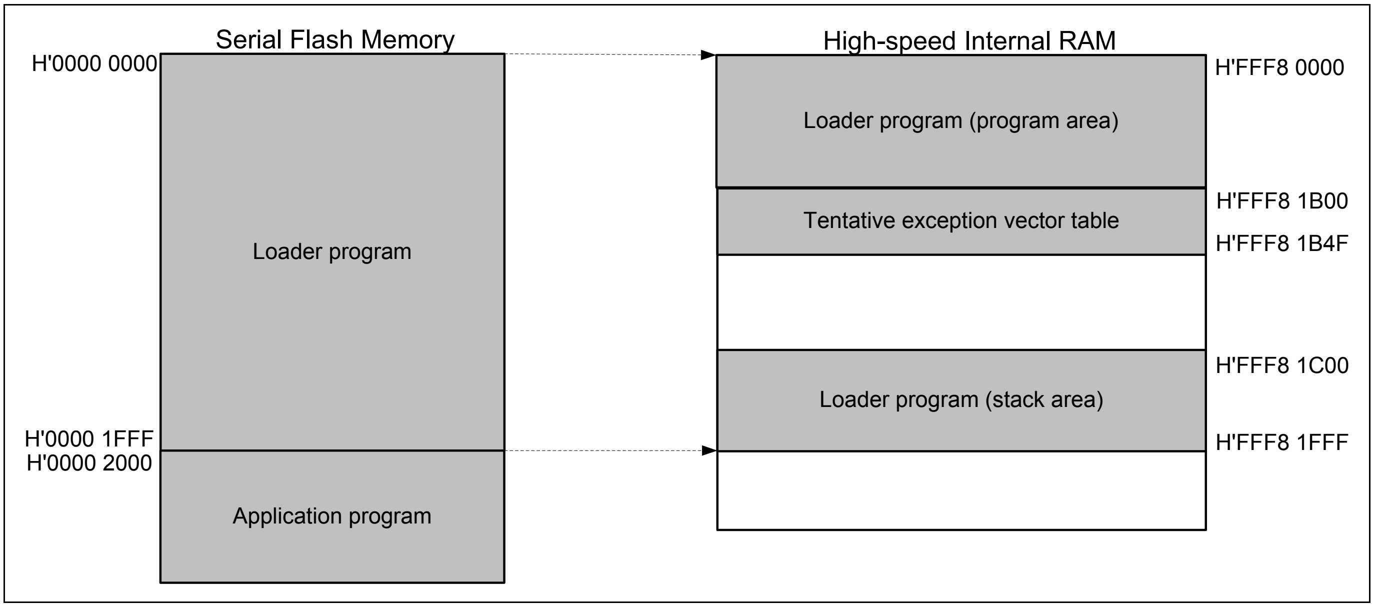


Figure 5 Loader Program Memory Map

3.1.2 Flow Chart of the Loader Program

The following figure shows the flow chart of the loader program. For details, refer to sections 3.1.3 to 3.1.11.

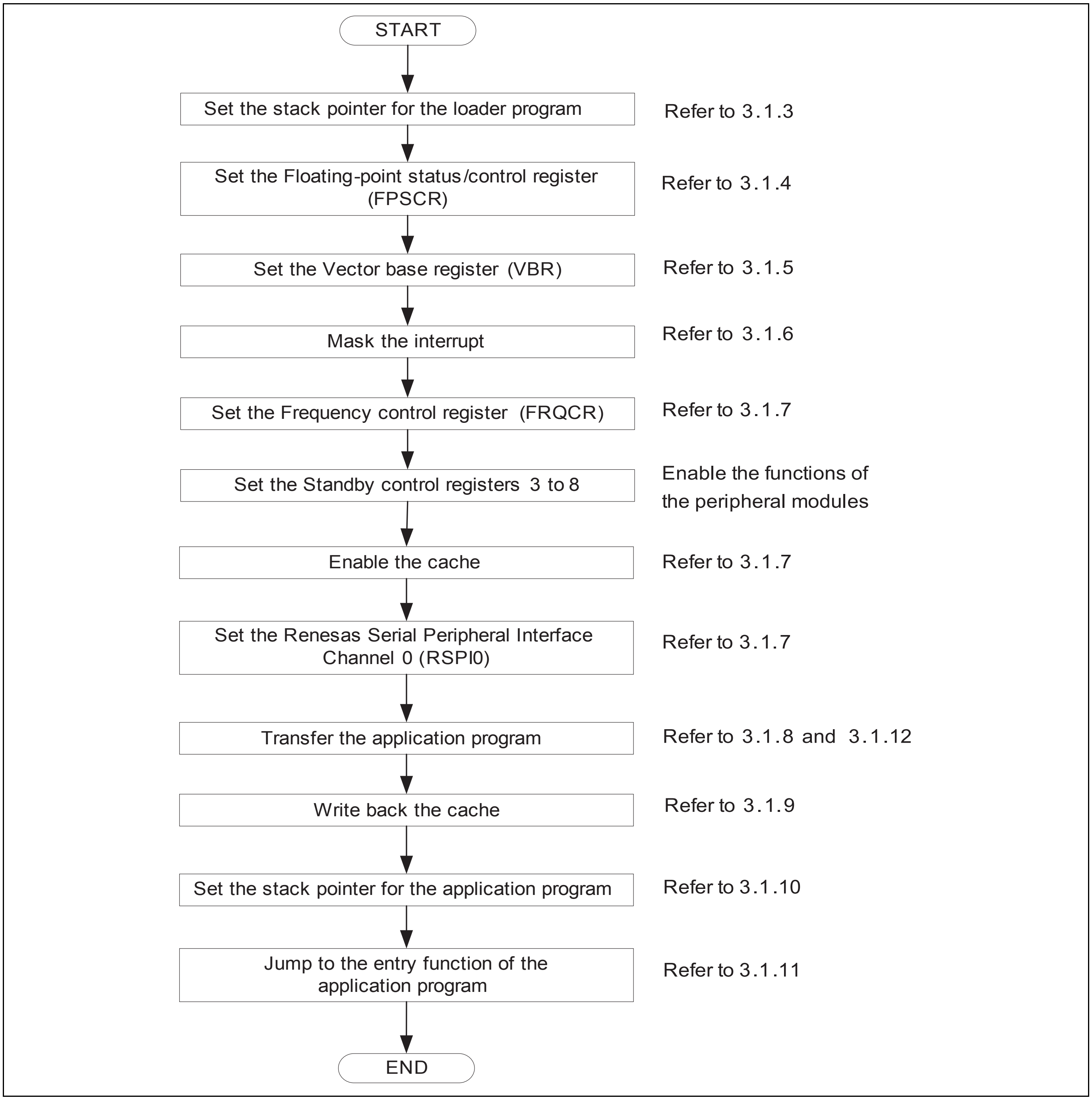

START
Set the stack pointer for the loader program
Refer to 3.1.3
Set the Floating-point status/control register (FPSCR)
Refer to 3.1.4
Set the Vector base register (VBR)
Refer to 3.1.5
Mask the interrupt
Refer to 3.1.6
Set the Frequency control register (FRQCR)
Refer to 3.1.7
Set the Standby control registers 3 to 8
Enable the functions of the peripheral modules
Enable the cache
Refer to 3.1.7
Set the Renesas Serial Peripheral Interface Channel 0 (RSPI0)
Refer to 3.1.7
Transfer the application program
Refer to 3.1.8 and 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 of H'FFF8 2000. Allocate the loader program processing at the address of 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 of 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 details, refer to 6.9.4 “Note before Exception Handling Begins Running” in the SH7262 Group, SH7264 Group 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 serial flash memory.

3.1.8 Transferring the Application Program

The loader program refers the application program transfer information (appinfo) in 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 in the address of H'0000 2000 in serial flash memory. The loader program handles the information from H'0000 2000 to H'0000 2007 in 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. For the procedures to generate the appinfo, refer to 3.2.7.

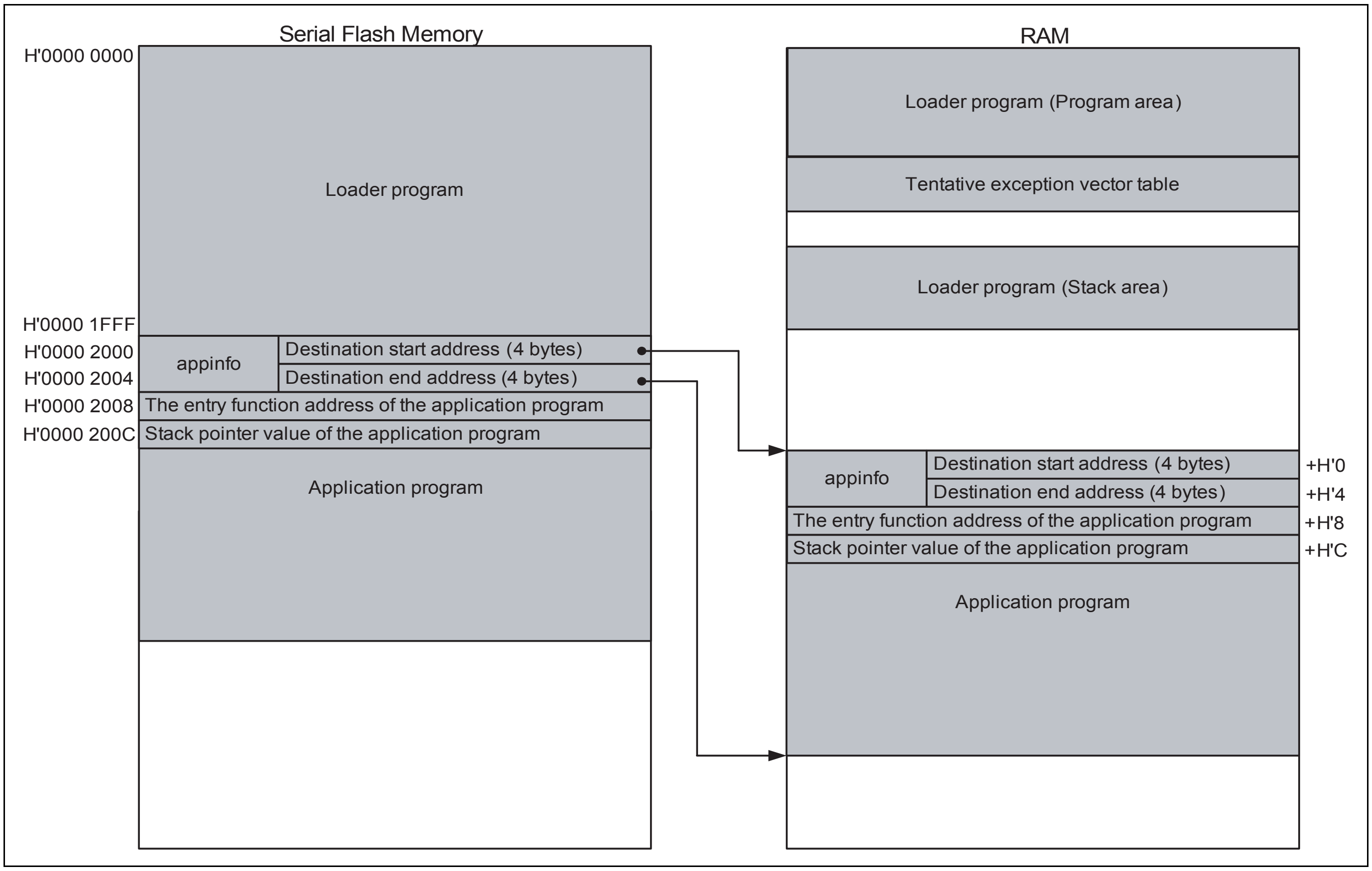

Serial Flash Memory
H'0000 0000
Loader program
H'0000 1FFF
H'0000 2000
H'0000 2004
H'0000 2008
H'0000 200C
appinfo
Destination start address (4 bytes)
Destination end address (4 bytes)
The entry function address of the application program
Stack pointer value of the application program
Application program
RAM
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.

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 serial flash memory. The loader program use the Read Array command in serial flash memory to read the application program from serial flash memory, and transfer the program to the large-capacity internal RAM. The following table lists the serial flash memory command used in the loader program.

Table 4 Serial Flash Memory Command

Command Name	Opcode	Function
Read Array	H'0B	Reads the data

Note: Although this application refers the commands of the ATMEL AT26DF161A, 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

The table below lists the register status after executing the loader program. Registers not included in the table are set as default in the SH7262 Group, SH7264 Group Hardware Manual.

Table 5 Register Status

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	The stack pointer value in the application program
Status register	SR	Undefined	Note: 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 to: zero
Frequency control register	FRQCR	H'1003	
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 register 8	STBCR8	H'7E	
Cache control register 1	CCR1	H'0000 0101	Instruction cache is valid Operand cache is valid
Control register_0	SPCR_0	H'48	
Slave select polarity register_0	SSLP_0	H'00	
Pin control register_0	SPPCR_0	H'30	
Status register_0	SPSR_0	H'60	
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'01	
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'E700	
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 setting register_0	SPBFDR_0	H'0000	

3.2 Application Program Example

As the loader program transfers the application program from serial flash memory to the large-capacity internal RAM, the memory map of the application program must be allocated as the loader program can read. Also, the application program must include the address information that the loader program refers.

This section describes the procedure to create the application program for the serial 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.

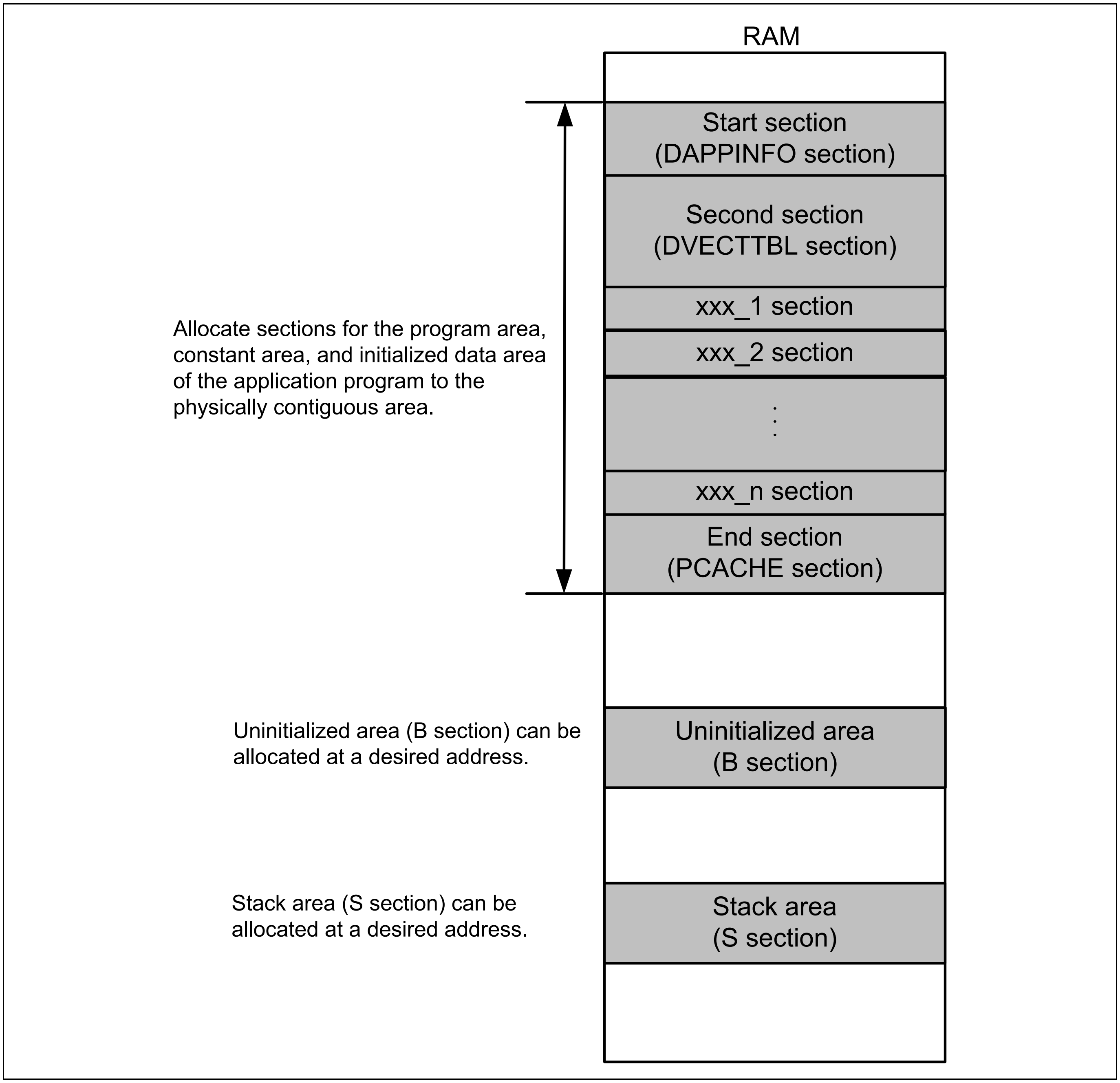
2. As the loader program uses the start address and end address of the application program to transfer the application program from serial flash memory to the large-capacity internal RAM, allocate the program area, constant area and initialized data area of the application program to the physically contiguous 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 DAPPINFO section, application program entry function address, and stack pointer value in DVECTTBL section. Allocate DAPPINFO section at the start on RAM, and then allocate DVECTTBL section.

4. As the loader program uses from H'FFF8 0000 to H'FFF8 1FFF in the high-speed internal RAM, do not allocate the program area, the constant area, and the initialized data area of the application program to that address.

5. Allocate the reset vector table RESET_Vectors in the start address of the DVECTTBL section.

The figure below shows an example of the section alignment in RAM.



Allocate sections for the program area, constant area, and initialized data area of the application program to the physically contiguous area.

Uninitialized area (B section) can be allocated at a desired address.

Stack area (S section) can be allocated at a desired address.

Figure 8 Application Program Section Alignment

3.2.2 Flow Chart

The application program in this application example transmits the strings of characters to channel 0 of the serial communication interface with FIFO (SCIF0). The following figure shows the flow chart of the application program.

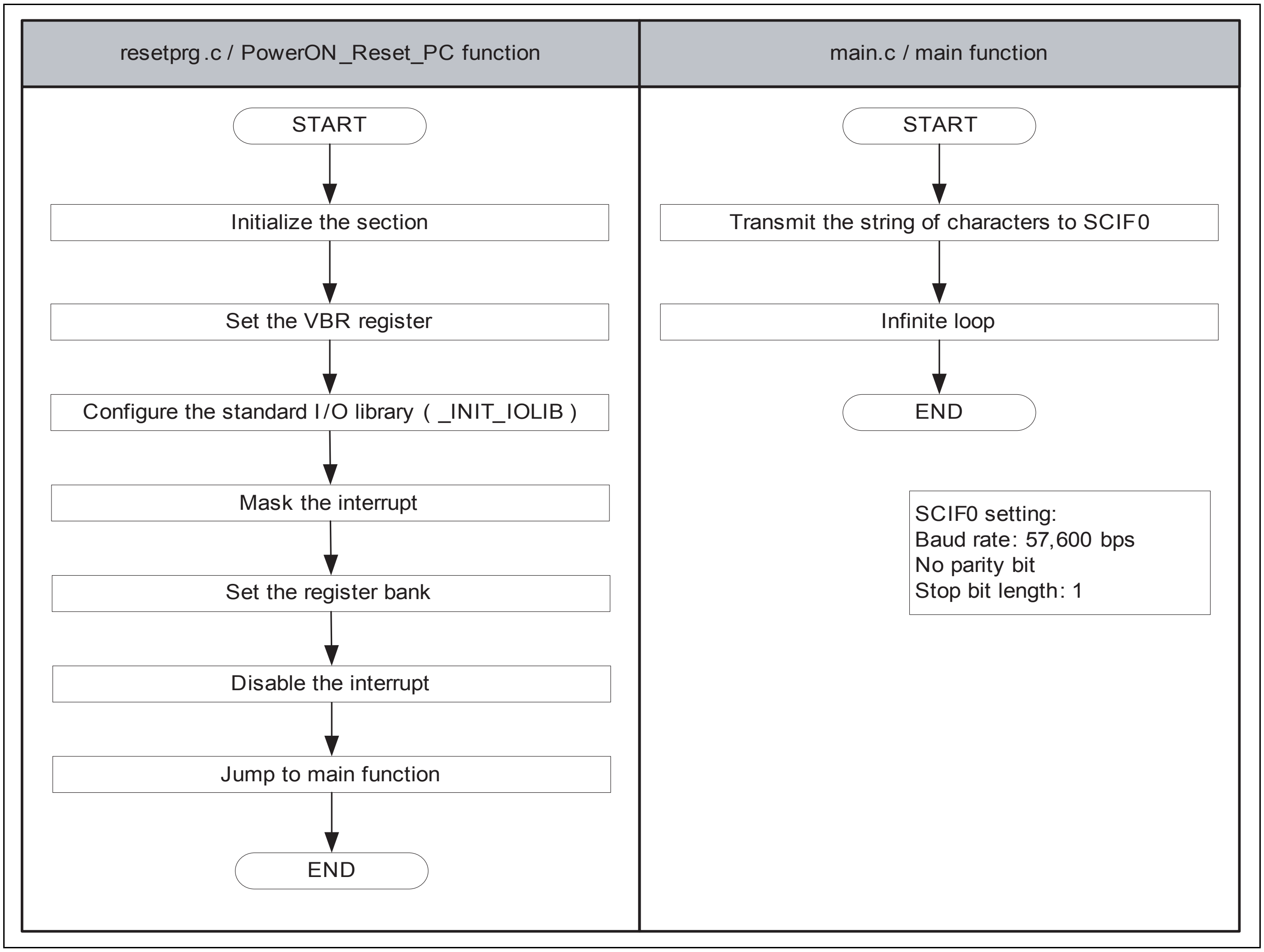


Figure 9 Application Program Flow Chart

3.2.3 Entry Function Setting

Set the entry function address of the application program to the table number 0 of the reset vector table RESET_Vectors. The following table lists its settings.

Table 6 Entry Function Address Settings

Item	Setting
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 the table number 1 of the reset vector table RESET_Vectors. The following table lists its settings.

Table 7 Stack Pointer Setting

Item	Setting
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)

The table below lists the structure to generate the application program transfer information (appinfo). Retrieve the start and end address of the application program by the section address operators (_sectop, _secend). Allocate the following structure in the DAPPINFO section. Register the start address of the application program (the 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 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
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 serial flash memory.

The following figure shows the Application Program Transfer Information (appinfo) generated image.

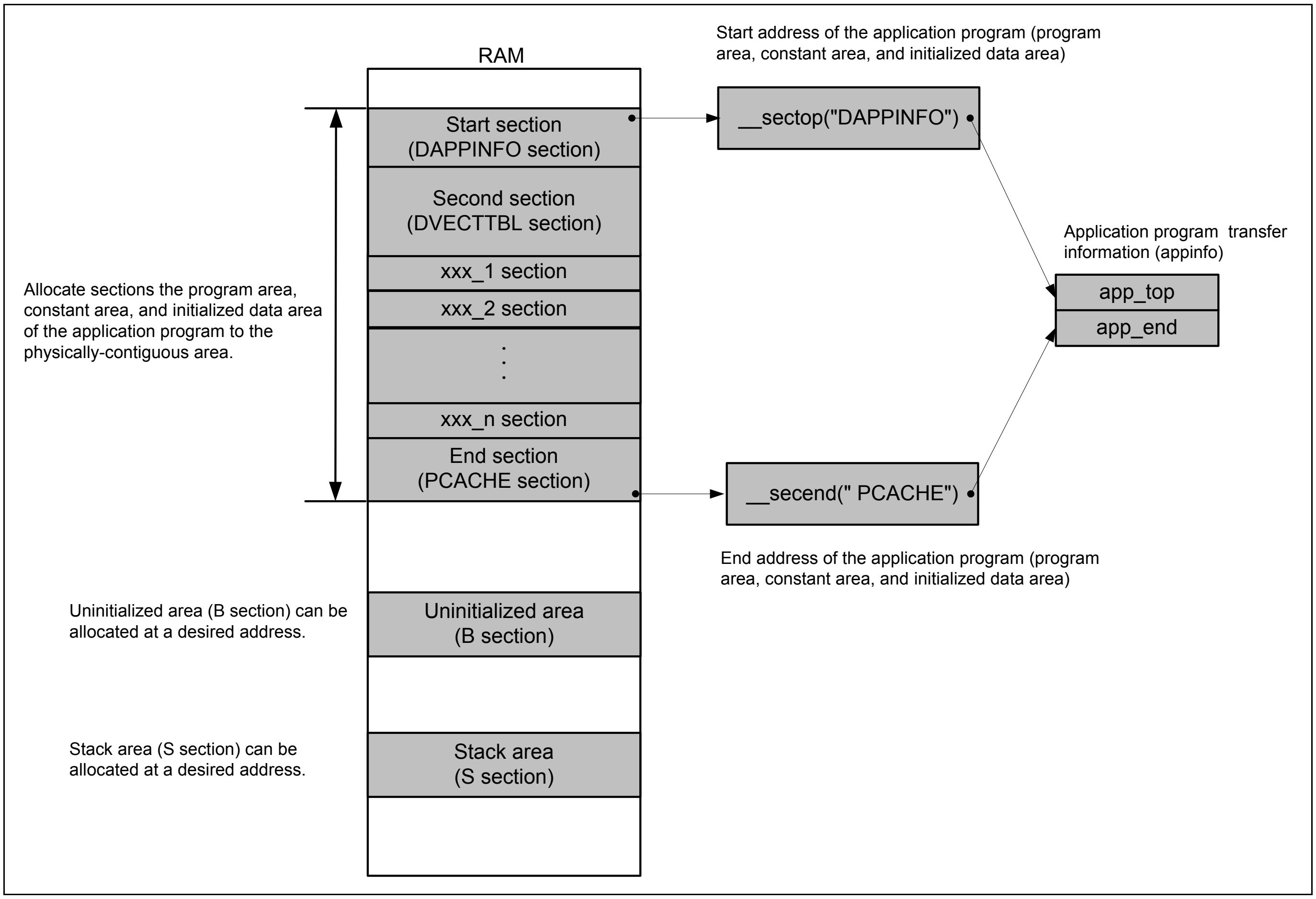

RAM
Start section
(DAPPINFO section)
Second section
(DVECTTBL section)
xxx_1 section
xxx_2 section
xxx_n section
End section
(PCACHE section)
Uninitialized area
(B section)
Stack area
(S section)
Allocate sections the program area, constant area, and initialized data area of the application program to the physically-contiguous area.
Uninitialized area (B section) can be allocated at a desired address.
Stack area (S section) 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. The following figure shows an operation image of the downloader.

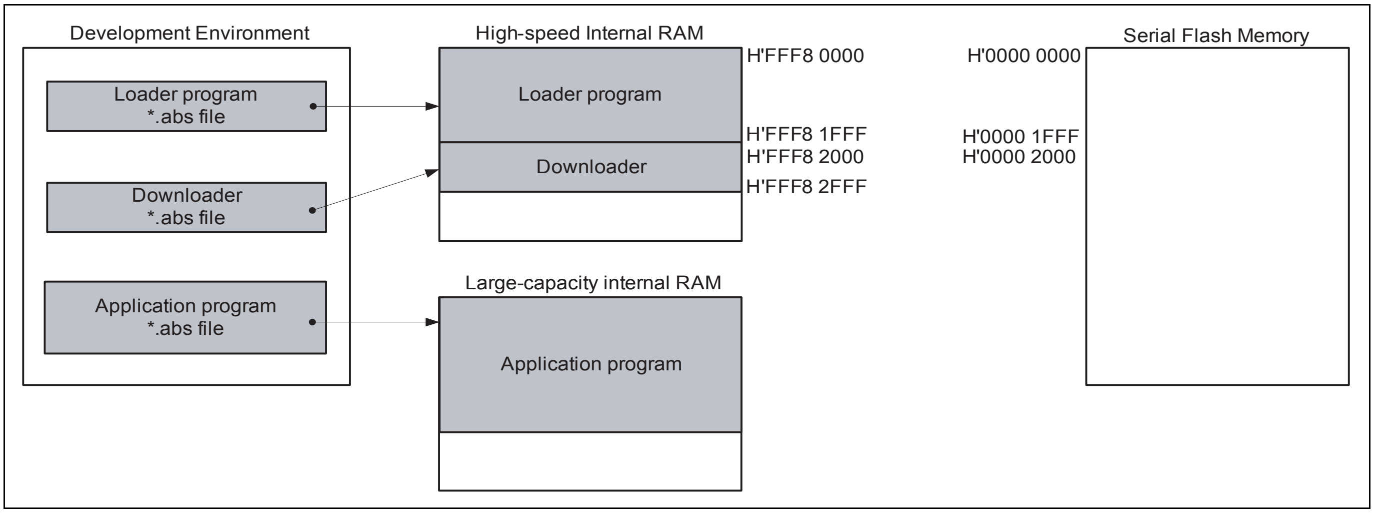


Figure 11 Downloader Operation Image (1/2)

Execute the downloader to write the loader program and the application program in 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. The following figure shows the operation image.

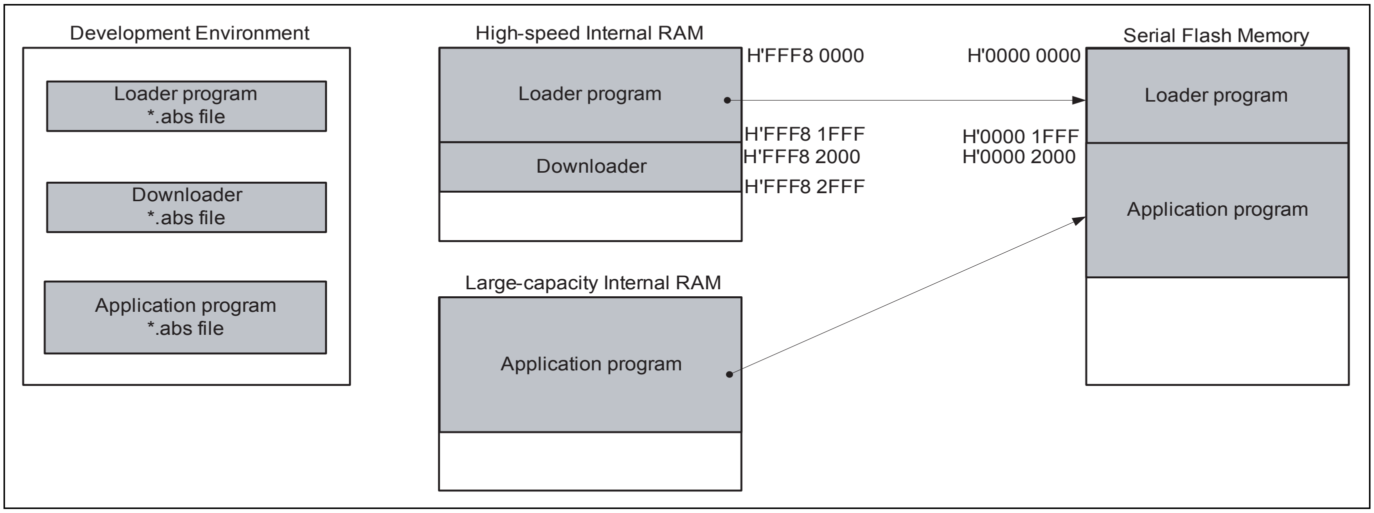


Figure 12 Downloader Operation Image (2/2)

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, the application program and the downloader occupy the same section, the programs do not operate properly.

3.3.3 Flow Chart of the Downloader

The figure below shows the flow chart of the downloader. Execute the downloader placed on the high-speed internal RAM to write the loader program and the application program in serial flash memory. For details, refer to sections 3.3.4 to 3.3.8.

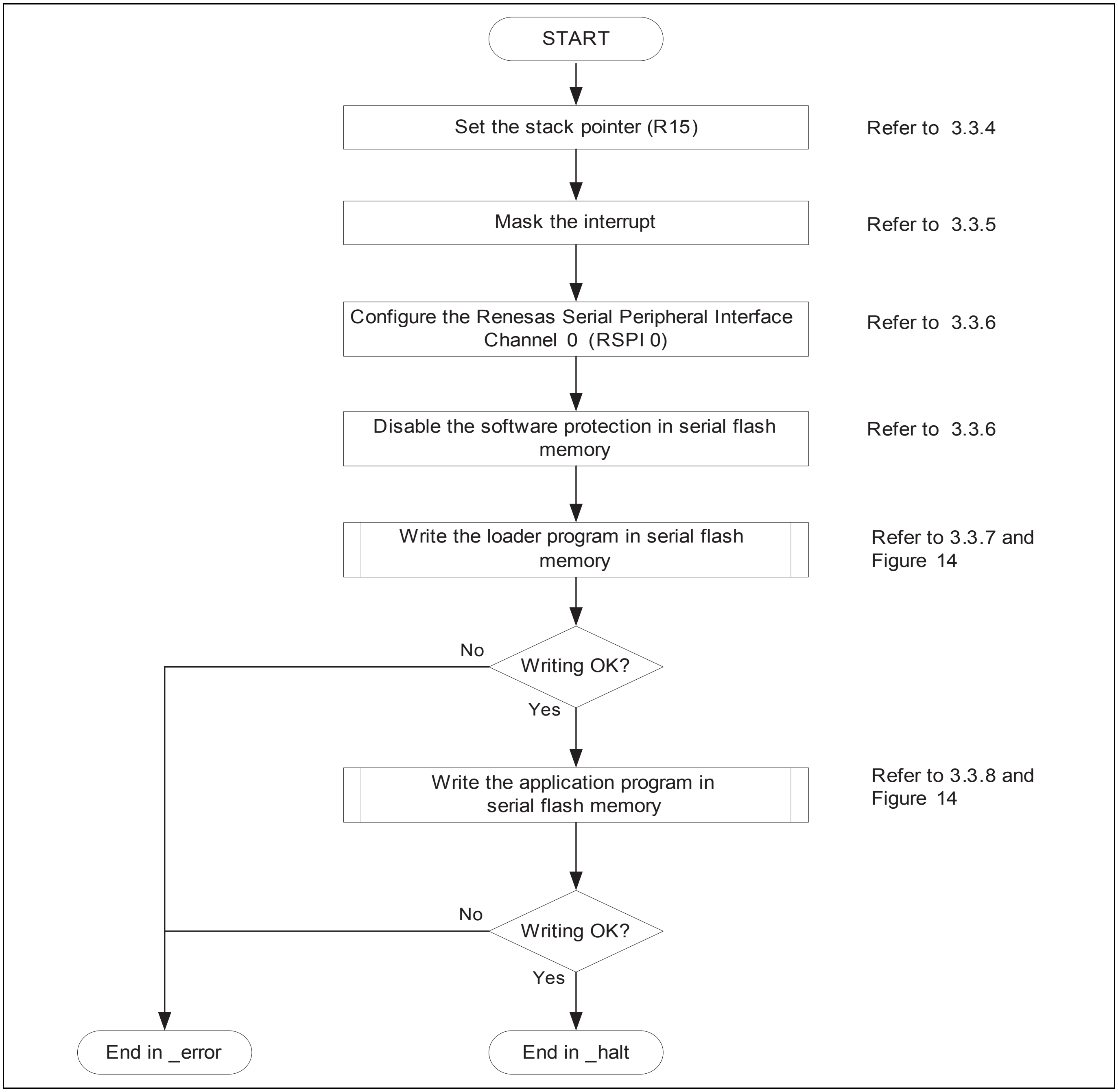

START
Set the stack pointer (R15)
Refer to 3.3.4
Mask the interrupt
Refer to 3.3.5
Configure the Renesas Serial Peripheral Interface Channel 0 (RSPI 0)
Refer to 3.3.6
Disable the software protection in serial flash memory
Refer to 3.3.6
Write the loader program in serial flash memory
Refer to 3.3.7 and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

Figure 13 Downloader Flow Chart

The following figure shows the flow chart of writing the loader program and the application program.

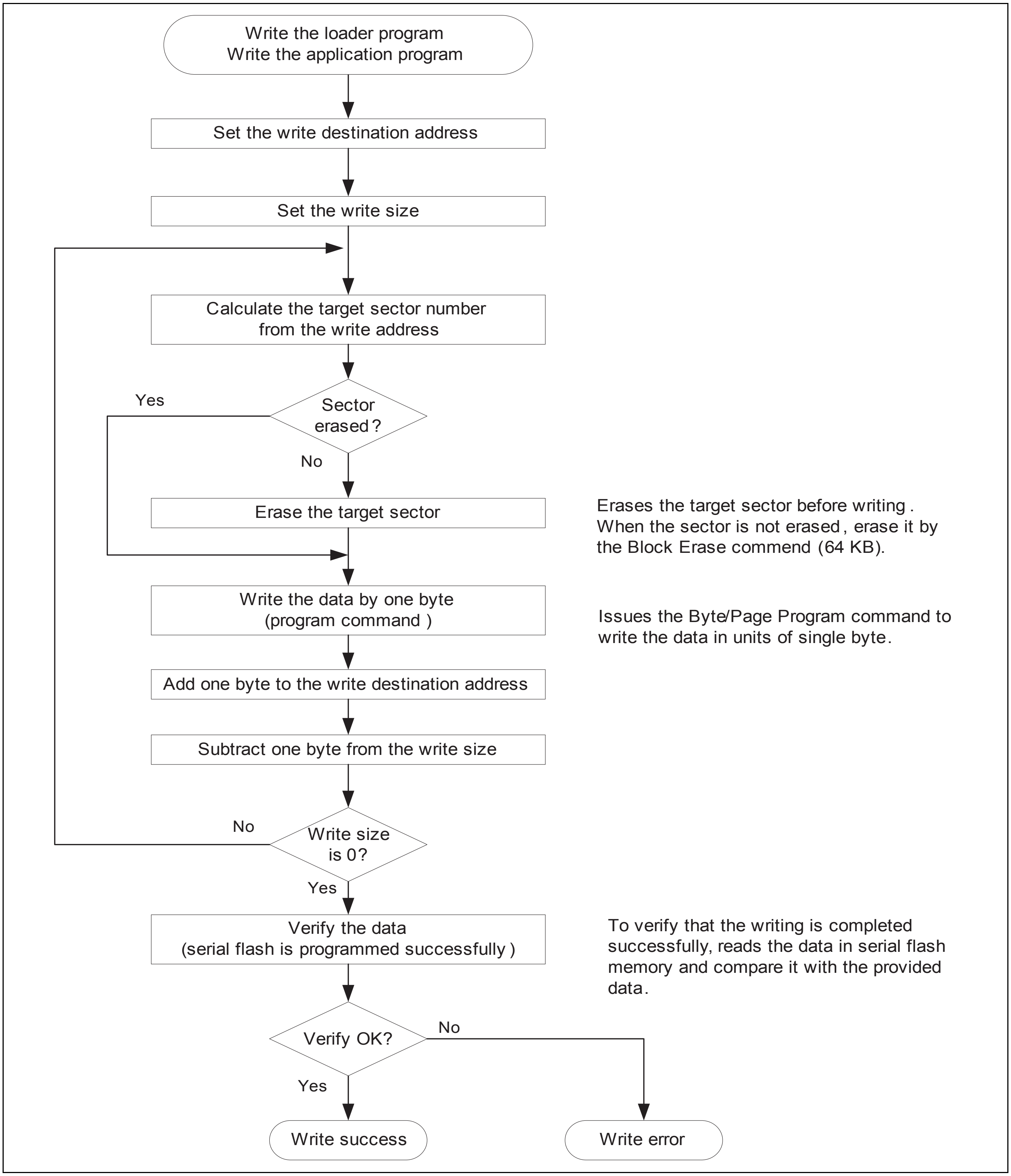

Write the loader program
Write the application program
Set the write destination address
Set the write size
Calculate the target sector number from the write address
Sector erased ?
Yes
No
Erase the target sector
Erases the target sector before writing . When the sector is not erased , erase it by the Block Erase commend (64 KB).
Write the data by one byte (program command)
Issues the Byte/Page 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, reads the data in serial flash memory and compare it with the provided data .
Verify OK?
No
Yes
Write success
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 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 interrupts during it is operating.

3.3.6 Initialization

Initialize serial flash memory before accessing.

1. Initialize the RSPI0
2. Issue the Write Status Register command to 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 serial flash memory. The following table lists the items for writing the loader program.

Table 9 Writing the Loader Program

Item	Description
Source Address of the Loader Program (High-speed internal RAM)	H'FFF8 0000 (fixed)
Destination Address of the Loader Program (Serial flash memory)	H'0000 0000 (fixed)
Transfer Size	H'2000 (fixed)
Writing Procedures	1. Checks if the destination address is already erased. 2. Erases the data when the address is not erased. 3. Issues the program command to write the loader program in units of single byte.

3.3.8 Writing the Application Program

The downloader reads the application program in the large-capacity internal RAM, and writes it at the address from H'0000 2000. The following table lists the items for writing the application program.

Table 10 Writing the Application Program

Item	Description
Source Address of the Application Program (Large-capacity internal RAM)	Retrieves from the appinfo in the application program (Application program dependent)
Destination Address of the Application Program (Serial flash memory)	H'0000 2000 (fixed)
Transfer Size	Extracts from the appinfo in the application program (Application program dependent)
Writing Procedures	1. Checks if the destination address is already erased. 2. Erases the data when the address is not erased. 3. Issues the program command to write the application program in units of single byte.

3.3.9 Serial Flash Memory Commands

The table below lists the serial flash memory commands used in the downloader. Issue these commands via the Renesas Serial Peripheral Interface Channel 0 (RSPI0) to read, write, and erase serial flash memory.

Table 11 Serial Flash Memory Commands

Command Name	Opcode	Function
Read Array	H'0B	Reads the data
Write Enable	H'06	Enables to execute the program, erase, write status register commands
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)
Block Erase (64Kbytes)	H'D8	Erases the data in blocks (64 KB)
Byte/Page Program	H'02	Programs the data (1 to 256 bytes)

Notes: 1. Although this application refers the commands of the ATMEL AT26DF161A, serial flash memory commands depend on the type of the serial flash memory. Refer to the datasheet provided by the serial flash memory manufacturer.

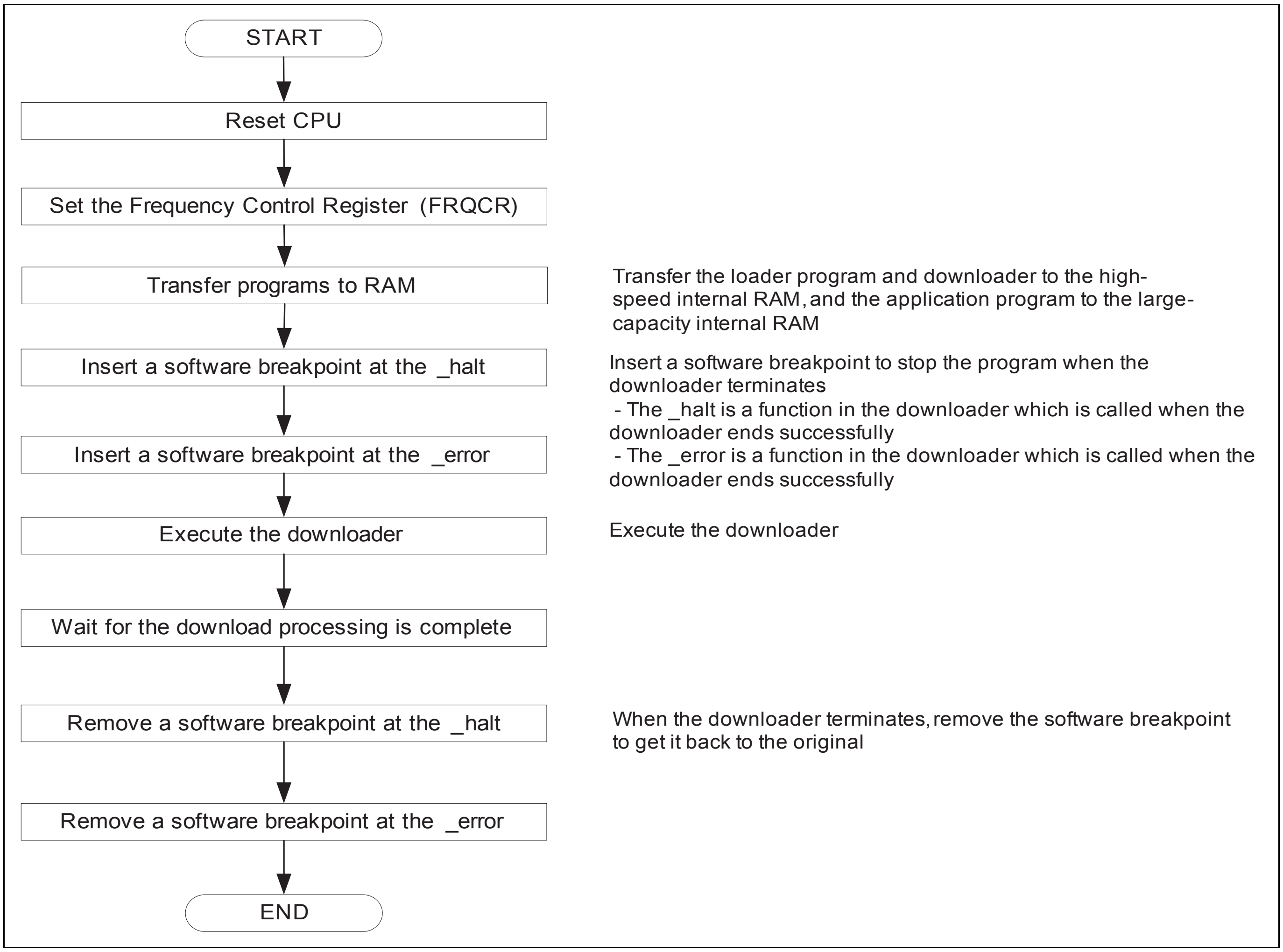
2. Erase the data in the destination address in serial flash memory before writing.

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 serial flash memory.

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

The figure below 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 serial flash memory.


START
Reset CPU
Set the Frequency Control Register (FRQCR)
Transfer programs to RAM
Insert a software breakpoint at the _halt
Insert a software breakpoint at the _error
Execute the downloader
Wait for the download processing is complete
Remove a software breakpoint at the _halt
Remove a software breakpoint at the _error
END

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 to stop the program when the downloader terminates
- The _halt is a function in the downloader which is called when the downloader ends successfully
- The _error is a function in the downloader which is called when the downloader ends successfully

Execute the downloader

When the downloader terminates, remove the software breakpoint to get it back to the original

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)

```
;/*""FILE COMMENT""*********** Technical reference data *************************
;*
;*       System Name : SH7264 Sample Program
;*       File Name   : loader.src
;*       Abstract    : Loader program preprocessing/jump processing to the application
;*                   : program
;*       Version     : 1.00.00
;*       Device      : SH7264/SH7262
;*       Tool-Chain  : High-performance Embedded Workshop (Ver.4.04.01).
;*                   : C/C++ compiler package for the SuperH RISC engine family
;*                   :                             (Ver.9.02 Release00).
;*       OS          : None
;*       H/W Platform: M3A-HS64G50 (CPU board)
;*       Disclaimer  :
;*
;*       The information described here may contain technical inaccuracies or
;*       typographical errors. Renesas Technology Corporation and Renesas Solutions
;*       assume no responsibility for any damage, liability, or other loss rising
;*       from these inaccuracies or errors.
;*
;*       Copyright (C) 2008 Renesas Technology Corp. All Rights Reserved
;*       AND Renesas Solutions Corp. All Rights Reserved
;*
;*       History     : Dec.19,2008 Ver.1.00.00
;*""FILE COMMENT END""**********************************************************/
  .SECTION LOADER_ENTRY,CODE,ALIGN = 4
  .IMPORT _main
  .EXPORT _jmp_app_prog

_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

```

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

```

;/*""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)

```
/*""FILE COMMENT""*********** Technical reference data *************************
*
*       System Name : SH7264 Sample Program
*       File Name   : main.c
*       Abstract    : Loader program
*       Version     : 1.00.00
*       Device      : SH7264/SH7262
*       Tool-Chain  : High-performance Embedded Workshop (Ver.4.04.01).
*                   : C/C++ compiler package for the SuperH RISC engine family
*                   :                             (Ver.9.02 Release00).
*       OS          : None
*       H/W Platform: M3A-HS64G50 (CPU board)
*       Disclaimer  :
*
*       The information described here may contain technical inaccuracies or
*       typographical errors. Renesas Technology Corporation and Renesas Solutions
*       assume no responsibility for any damage, liability, or other loss rising
*       from these inaccuracies or errors.
*
*       Copyright (C) 2008 Renesas Technology Corp. All Rights Reserved
*       AND Renesas Solutions Corp. All Rights Reserved
*
*       History     : Dec.19,2008 Ver.1.00.00
*""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) */

#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 */

```

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

```

/* ==== 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;


/*""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""**********************************************************/
```

4.1.5 Loader Program Listing "main.c" (3/6)

```
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);


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

```

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

```
/*""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)

```
/*""FILE COMMENT""*********** Technical reference data *************************
*
*       System Name : SH7264 Sample Program
*       File Name   : main.c
*       Abstract    : Application program example
*       Version     : 1.00.00
*       Device      : SH7264/SH7262
*       Tool-Chain  : High-performance Embedded Workshop (Ver.4.04.01).
*                   : C/C++ compiler package for the SuperH RISC engine family
*                   :                             (Ver.9.02 Release00).
*       OS          : None
*       H/W Platform: M3A-HS64G50 (CPU board)
*       Disclaimer  :
*
*       The information described here may contain technical inaccuracies or
*       typographical errors. Renesas Technology Corporation and Renesas Solutions
*       assume no responsibility for any damage, liability, or other loss rising
*       from these inaccuracies or errors.
*
*       Copyright (C) 2008 Renesas Technology Corp. All Rights Reserved
*       AND Renesas Solutions Corp. All Rights Reserved
*
*       History     : Dec.19,2008 Ver.1.00.00
*""FILE COMMENT END""**********************************************************/
#include <stdio.h>

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


/*""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""**********************************************************/
```

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

```
void main(void)
{
  puts("==== Serial Flash Boot Done. ====");
  fflush(stdout);

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

/* End of File */
```

4.2.3 Application Program Listing "appinfo.c"

```
/*""FILE COMMENT""*********** Technical reference data *************************
*
*       System Name : SH7264 Sample Program
*       File Name   : appinfo.c
*       Abstract    : Generate the application program transfer information (appinfo).
*       Version     : 1.00.00
*       Device      : SH7264/SH7262
*       Tool-Chain  : High-performance Embedded Workshop (Ver.4.04.01).
*                   : C/C++ compiler package for the SuperH RISC engine family
*                   :                             (Ver.9.02 Release00).
*       OS          : None
*       H/W Platform: M3A-HS64G50 (CPU board)
*       Disclaimer  :
*
*       The information described here may contain technical inaccuracies or
*       typographical errors. Renesas Technology Corporation and Renesas Solutions
*       assume no responsibility for any damage, liability, or other loss rising
*       from these inaccuracies or errors.
*
*       Copyright (C) 2008 Renesas Technology Corp. All Rights Reserved
*       AND Renesas Solutions Corp. All Rights Reserved
*
*       History     : Dec.19,2008 Ver.1.00.00
*""FILE COMMENT END""**********************************************************/
#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 */
```

4.2.4 Application Program Listing "appinfo.h"

```
/*""FILE COMMENT""*********** Technical reference data *************************
*
*       System Name : SH7264 Sample Program
*       File Name   : appinfo.h
*       Abstract    : Header file of the application program transfer information
(appinfo).
*       Version     : 1.00.00
*       Device      : SH7264/SH7262
*       Tool-Chain  : High-performance Embedded Workshop (Ver.4.04.01).
*                   : C/C++ compiler package for the SuperH RISC engine family
*                   :                            (Ver.9.02 Release00).
*       OS          : None
*       H/W Platform: M3A-HS64G50 (CPU board)
*       Disclaimer  :
*
*       The information described here may contain technical inaccuracies or
*       typographical errors. Renesas Technology Corporation and Renesas Solutions
*       assume no responsibility for any damage, liability, or other loss rising
*       from these inaccuracies or errors.
*
*       Copyright (C) 2008 Renesas Technology Corp. All Rights Reserved
*       AND Renesas Solutions Corp. All Rights Reserved
*
*       History     : Dec.19,2008 Ver.1.00.00
*""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)

```
#/*""FILE COMMENT""*********** Technical reference data *************************
#*
#*       System Name : SH7264 Sample Program
#*       File Name   : downloader.hdc
#*       Abstract    : Batch File for the Downloader
#*       Version     : 1.00.00
#*       Device      : SH7264/SH7262
#*       Tool-Chain  : High-performance Embedded Workshop (Ver.4.04.01).
#*                   : C/C++ compiler package for the SuperH RISC engine family
#*                   :                             (Ver.9.02 Release00).
#*       OS          : None
#*       H/W Platform: M3A-HS64G50 (CPU board)
#*       Disclaimer  :
#*
#*       The information described here may contain technical inaccuracies or
#*       typographical errors. Renesas Technology Corporation and Renesas Solutions
#*       assume no responsibility for any damage, liability, or other loss rising
#*       from these inaccuracies or errors.
#*
#*       Copyright (C) 2008 Renesas Technology Corp. All Rights Reserved
#*       AND Renesas Solutions Corp. All Rights Reserved
#*
#*       History     : Dec.19,2008 Ver.1.00.00
#*""FILE COMMENT END""*********************************************************/


tcl enable


#Macro downloader -Start
proc init_hardware {} {

  # Set the CPG
  # FRQCR I=144MHz/B=72MHz/P=36MHz/CLK MODE2
  MF H'FFFE0010 H'FFFE0011 H'1003 WORD

}


proc downloader {} {
  # Reset CPU
  reset

  # Calls the init_hardware routine
  init_hardware

```

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

```
  # 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"

```
;/*""FILE COMMENT""*********** Technical reference data *************************
;*
;*       System Name : SH7264 Sample Program
;*       File Name   : downloader.src
;*       Abstract    : Downloader
;*       Version     : 1.00.00
;*       Device      : SH7264/SH7262
;*       Tool-Chain  : High-performance Embedded Workshop (Ver.4.04.01).
;*                   : C/C++ compiler package for the SuperH RISC engine family
;*                   :                             (Ver.9.02 Release00).
;*       OS          : None
;*       H/W Platform: M3A-HS64G50 (CPU board)
;*       Disclaimer  :
;*
;*       The information described here may contain technical inaccuracies or
;*       typographical errors. Renesas Technology Corporation and Renesas Solutions
;*       assume no responsibility for any damage, liability, or other loss rising
;*       from these inaccuracies or errors.
;*
;*       Copyright (C) 2008 Renesas Technology Corp. All Rights Reserved
;*       AND Renesas Solutions Corp. All Rights Reserved
;*
;*       History     : Dec.19,2008 Ver.1.00.00
;*""FILE COMMENT END""**************************************************************/
  .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
```

4.3.4 Downloader Program Listing "main.c" (1/7)

```
/*""FILE COMMENT""*********** Technical reference data *************************
*
*       System Name : SH7264 Sample Program
*       File Name   : main.c
*       Abstract    : Downloader
*       Version     : 1.00.00
*       Device      : SH7264/SH7262
*       Tool-Chain  : High-performance Embedded Workshop (Ver.4.04.01).
*                   : C/C++ compiler package for the SuperH RISC engine family
*                   :                             (Ver.9.02 Release00).
*       OS          : None
*       H/W Platform: M3A-HS64G50 (CPU board)
*       Disclaimer  :
*
*       The information described here may contain technical inaccuracies or
*       typographical errors. Renesas Technology Corporation and Renesas Solutions
*       assume no responsibility for any damage, liability, or other loss rising
*       from these inaccuracies or errors.
*
*       Copyright (C) 2008 Renesas Technology Corp. All Rights Reserved
*       AND Renesas Solutions Corp. All Rights Reserved
*
*       History     : Dec.19,2008 Ver.1.00.00
*""FILE COMMENT END""**********************************************************/
#include <stdio.h>
#include <string.h>
#include <machine.h>
#include "iodefine.h"
#include "serial_flash.h"

/* ==== 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 */
```

4.3.5 Downloader Program Listing "main.c" (2/7)

```

#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 */



/*""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;

```

4.3.6 Downloader Program Listing "main.c" (3/7)

```
  /* 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;


  /* 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();
}

```

4.3.7 Downloader Program Listing "main.c" (4/7)

```
/*""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""**********************************************************/
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--;
```

4.3.8 Downloader Program Listing "main.c" (5/7)

```
  }

  /* ==== 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;
}

/*""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;
  }
}

```

4.3.9 Downloader Program Listing "main.c" (6/7)

```
/*""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];
}

/*""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;
}

```

4.3.10 Downloader Program Listing "main.c" (7/7)

```
/*""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. */
    }
}

/*""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.

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

5.1 Sample Program Configuration

The sample program consists of three workspaces as listed in the following table.

Table 12 Sample Program Configuration

Workspace Name	Description
sh7264_sflash_downloader	Build the downloader in the project of this workspace
sh7264_sflash_loader_prog	Build the loader program in the project of this workspace
sh7264_sflash_app	Build the application program in the project of this workspace. The downloader which is created in the [sh7264_sflash_downloader] workspace, a batch file to boot the downloader, and the loader program which is created in the [sh7264_sflash_loader_prog] workspace are registered in the project of this workspace. Use these items to write the loader program and application program in serial flash memory.

5.2 Writing Programs in Serial Flash Memory

This section describes the procedures to write the loader program and application program in serial flash memory using the [sh7264_sflash_app] workspace.

5.2.1 Register the Download Module and Batch File

The figure below lists the directory configuration of the [sh7264_sflash_app] workspace. Download modules (A, B, and D) and a batch file (C) in the figure are registered in the project.

```
¥sh7264_sflash_app                        : Workspace directory
  |-sh7264_sflash_app                     : Project directory
  |   |-debug                             :
  |       |-sh7264_sflash_app.abs         : Application program run 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
     |-sh7264_sflash_downloader.abs       : Downloader run file--------------------------------------------------B
     |-downloader.hdc                     : Batch file to boot the downloader----------------------------------C
     |-sh7264_sflash_loader_prog.abs      : Loader program run file---------------------------------------------D
```

Figure 16 [sh7264_sflash_app] Workspace Directory Configuration

1. Changing the download module
Change the download module registered in the project in the **Debug setting** 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 the procedures to write the loader program and application program in serial flash memory using the [sh7264_sflash_app] workspace.

1. Copy the [sh7264_sflash_app] workspace directory in C:¥WorkSpace.
2. Double-click the [sh7264_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].

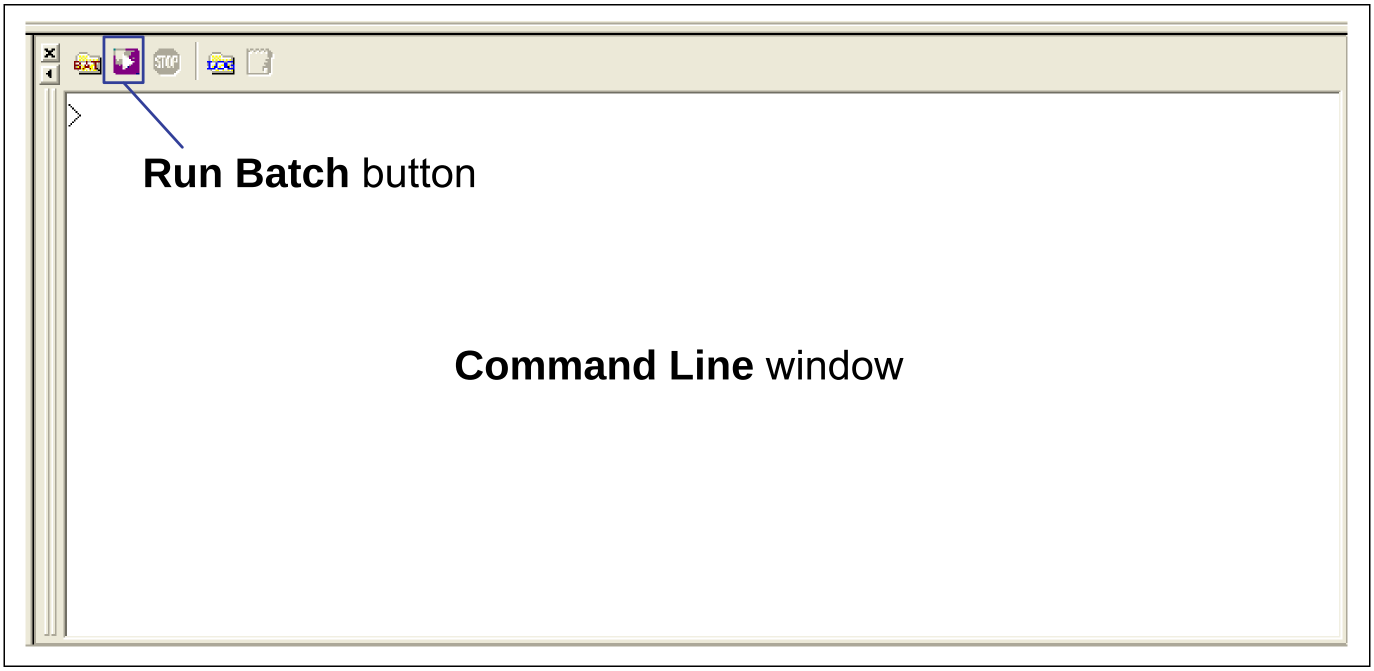
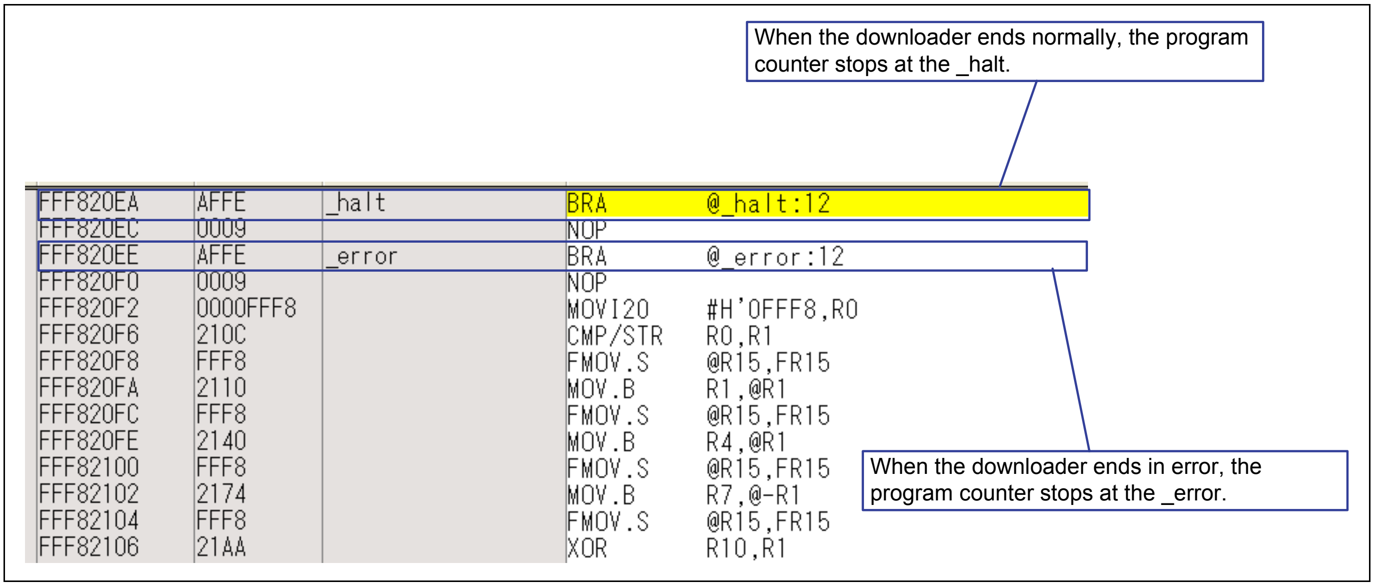


Figure 17 Command Line Window and Run Batch Button

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 shown in the figure below, 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 [sh7264_sflash_downloader] workspace directory is copied in C:¥WorkSpace.



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 18 High-performance Embedded Workshop Window When the Downloader Ends

6. References

- Software Manual
  SH-2A/SH-2A-FPU Software Manual Rev. 3.00
  (Download the latest version from the Renesas website.)

- Hardware Manual
  SH7262 Group, SH7264 Group Hardware Manual Rev. 1.00
  (Download the latest version from the Renesas website.)

Website and Support

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

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

Revision History

		Description	
Rev.	Date	Page	Summary
1.00	Apr 14, 2009	—	First edition issued

All trademarks and registered trademarks are the property of their respective owners.

Notes regarding these materials

1. This document is provided for reference purposes only so that Renesas customers may select the appropriate Renesas products for their use. Renesas neither makes warranties or representations with respect to the accuracy or completeness of the information contained in this document nor grants any license to any intellectual property rights or any other rights of Renesas or any third party with respect to the information in this document.
2. Renesas shall have no liability for damages or infringement of any intellectual property or other rights arising out of the use of any information in this document, including, but not limited to, product data, diagrams, charts, programs, algorithms, and application circuit examples.
3. You should not use the products or the technology described in this document for the purpose of military applications such as the development of weapons of mass destruction or for the purpose of any other military use. When exporting the products or technology described herein, you should follow the applicable export control laws and regulations, and procedures required by such laws and regulations.
4. All information included in this document such as product data, diagrams, charts, programs, algorithms, and application circuit examples, is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas products listed in this document, please confirm the latest product information with a Renesas sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas such as that disclosed through our website. (http://www.renesas.com)
5. Renesas has used reasonable care in compiling the information included in this document, but Renesas assumes no liability whatsoever for any damages incurred as a result of errors or omissions in the information included in this document.
6. When using or otherwise relying on the information in this document, you should evaluate the information in light of the total system before deciding about the applicability of such information to the intended application. Renesas makes no representations, warranties or guaranties regarding the suitability of its products for any particular application and specifically disclaims any liability arising out of the application and use of the information in this document or Renesas products.
7. With the exception of products specified by Renesas as suitable for automobile applications, Renesas products are not designed, manufactured or tested for applications or otherwise in systems the failure or malfunction of which may cause a direct threat to human life or create a risk of human injury or which require especially high quality and reliability such as safety systems, or equipment or systems for transportation and traffic, healthcare, combustion control, aerospace and aeronautics, nuclear power, or undersea communication transmission. If you are considering the use of our products for such purposes, please contact a Renesas sales office beforehand. Renesas shall have no liability for damages arising out of the uses set forth above.
8. Notwithstanding the preceding paragraph, you should not use Renesas products for the purposes listed below:
   (1) artificial life support devices or systems
   (2) surgical implantations
   (3) healthcare intervention (e.g., excision, administration of medication, etc.)
   (4) any other purposes that pose a direct threat to human life
   Renesas shall have no liability for damages arising out of the uses set forth in the above and purchasers who elect to use Renesas products in any of the foregoing applications shall indemnify and hold harmless Renesas Technology Corp., its affiliated companies and their officers, directors, and employees against any and all damages arising out of such applications.
9. You should use the products described herein within the range specified by Renesas, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas shall have no liability for malfunctions or damages arising out of the use of Renesas products beyond such specified ranges.
10. Although Renesas endeavors to improve the quality and reliability of its products, IC products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Please be sure to implement safety measures to guard against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other applicable measures. Among others, since the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
11. In case Renesas products listed in this document are detached from the products to which the Renesas products are attached or affixed, the risk of accident such as swallowing by infants and small children is very high. You should implement safety measures so that Renesas products may not be easily detached from your products. Renesas shall have no liability for damages arising out of such detachment.
12. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written approval from Renesas.
13. Please contact a Renesas sales office if you have any questions regarding the information contained in this document, Renesas semiconductor products, or if you have any other inquiries.