SH7268/SH7269 Group

Boot From the Serial Flash Memory

Using SPI Multi I/O Bus Controller

Summary

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

Target Device

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

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

Contents

1.   Introduction

1.1   Specifications

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

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

1.2   Modules Used

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

1.3   Applicable Conditions

MCU	SH72268/SH7269
Operating Frequency	Internal clock (Iϕ) : 266.67 MHz
	Internal bus clock (Bϕ) : 133.33 MHz
	Peripheral clock 1 (P1ϕ) : 66.67 MHz
	Peripheral clock 0 (P0ϕ) : 33.33 MHz
Integrated Development	Renesas Electronics Corporation
Environment	High-performance Embedded Workshop Ver.4.07.00
C Compiler	Renesas Electronics SuperH RISC engine Family
	C/C++ compiler package Ver.9.03 Release 02
Compiler Options	Default setting in the High-performance Embedded Workshop
	(-cpu=sh2afpu -fpu=single -object="$(CONFIGDIR)\$(FILELEAF).obj"
	-debug -gbr=auto -chgincpath -errorpath -global_volatile=0 -opt_range=all
	-infinite_loop=0 -del_vacant_loop=0 -struct_alloc=1 –nologo)
Serial Flash Memory	S25FL032P (Spansion) x 1

1.4   Related Application Note

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

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

1.5   About Active-low Pins (Signals)

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

2. Overview of the Serial Flash Boot

This chapter describes an overview of the serial flash boot.

2.1   Words and Terms

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

Table 1   Terms to Describe the Serial Flash Boot

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

2.2 The Serial Flash Boot Operation

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

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

MD_BOOT2	MD_BOOT1	MD_BOOT0	Boot Mode	Description
1	0	1	Boot mode 3	Boots the MCU from serial flash memory connected to Renesas Serial Peripheral Interface channel 0

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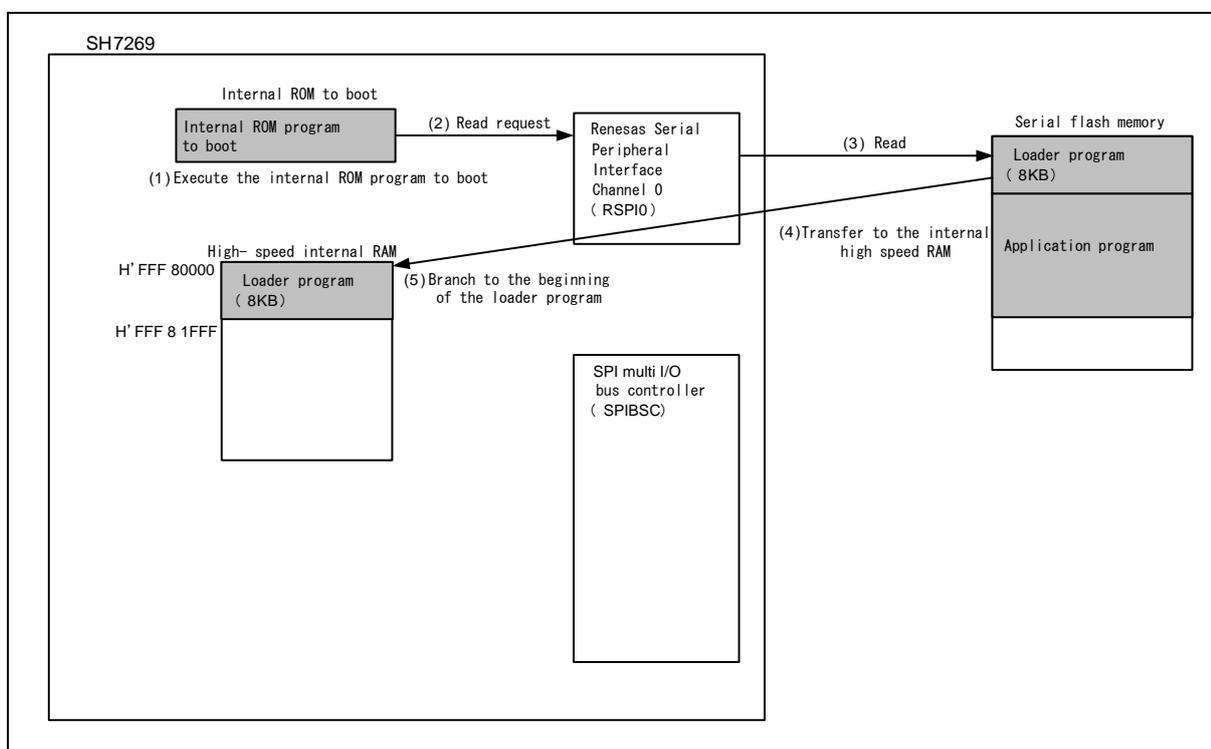
Figure 1 Operation Image of the Internal ROM Program to Boot

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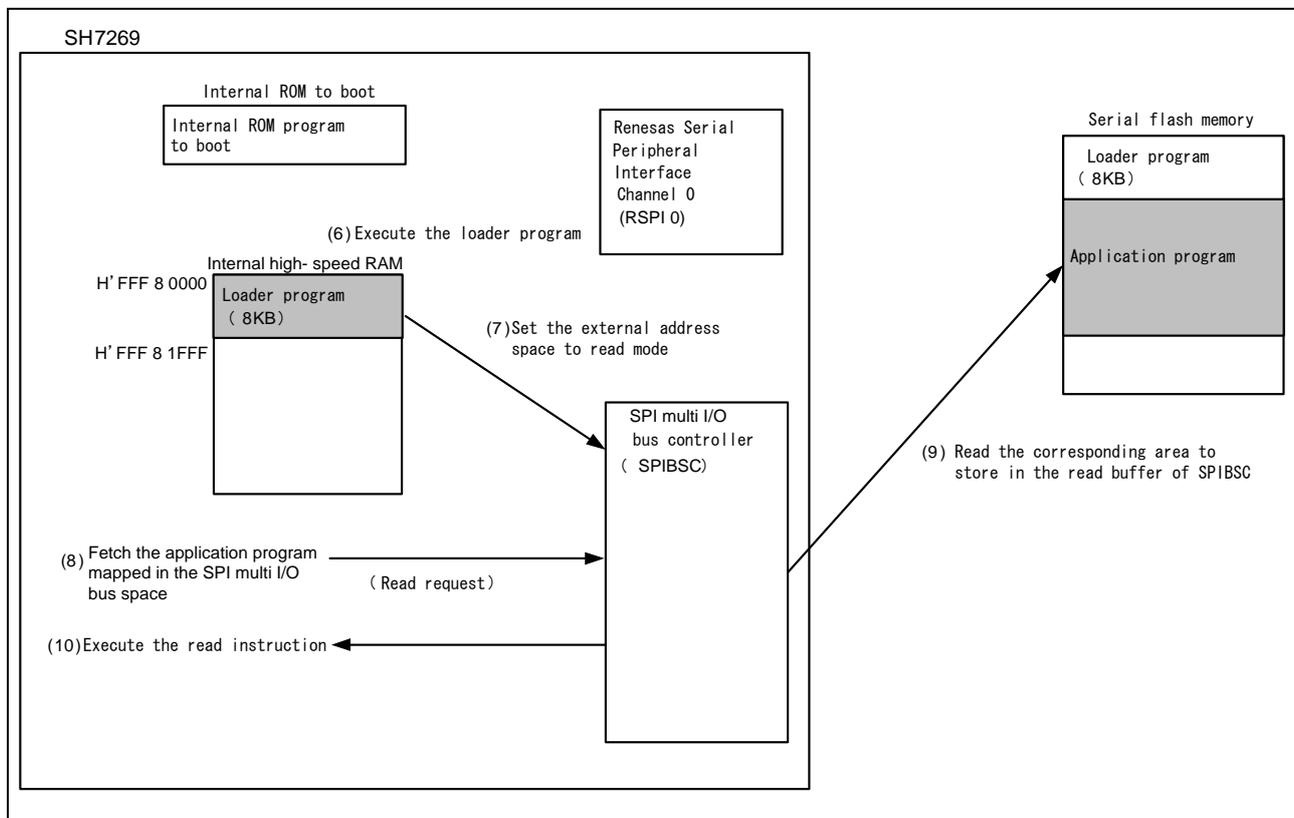
Figure 2   Operation Image of the Loader Program

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. Figure 3 shows the operation image of the downloader.

For more information, refer to "3.3 Downloader Example".
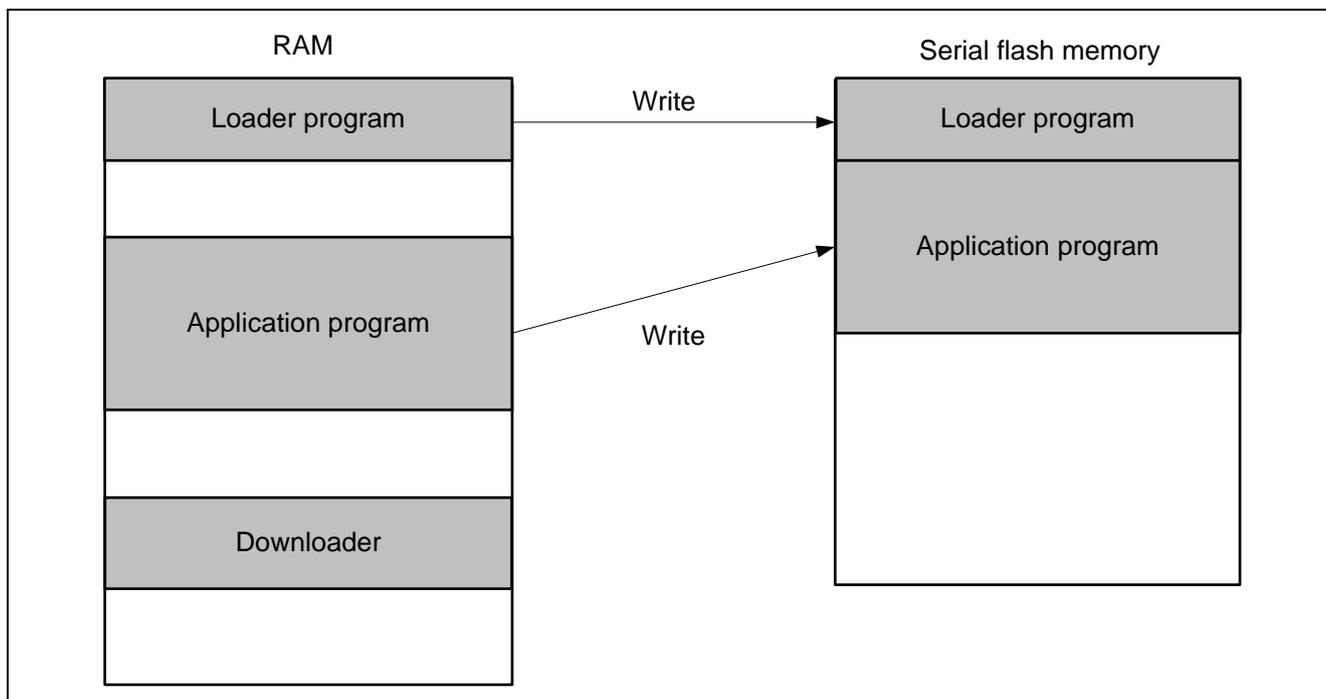
Figure 3　Operation Image of the Downloader

2.4   Serial Flash Memory Connection

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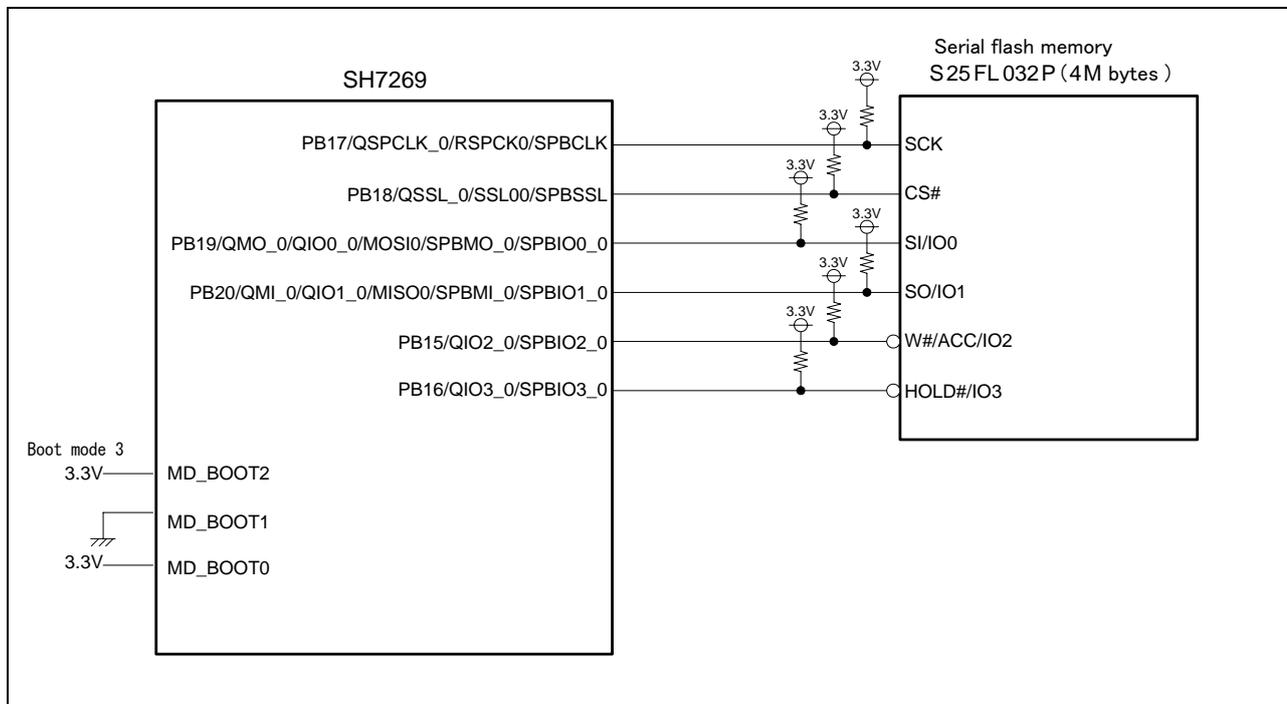
Figure 4   Serial Flash Memory Circuit

3. Applications

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

3.1 Loader Program Specifications

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

3.1.1 Memory Map

Figure 5 shows the memory map of the loader program.

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

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

Figure 5   Loader Program Memory Map

3.1.2 Flow Chart of the Loader Program

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

Figure 6   Loader Program Flow Chart

3.1.3　　　Stack Pointer Setting

Set the address H'FFF8 2000 in the stack pointer (R15) . The loader program is allocated at the address H'FFF8 0000 in the assembly language to avoid the loader program using the undefined stack pointer. C can be used after configuring the stack pointer. Then, the loader program jumps to the entry function.
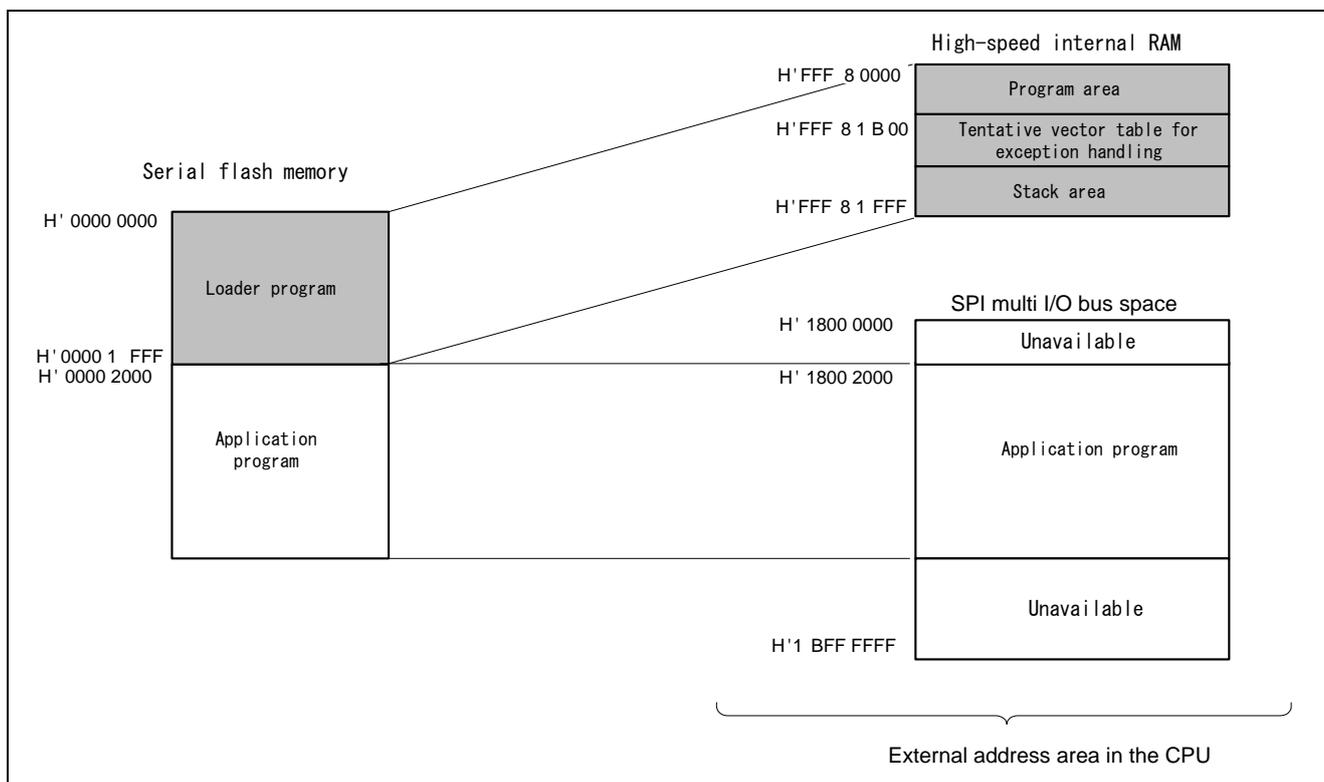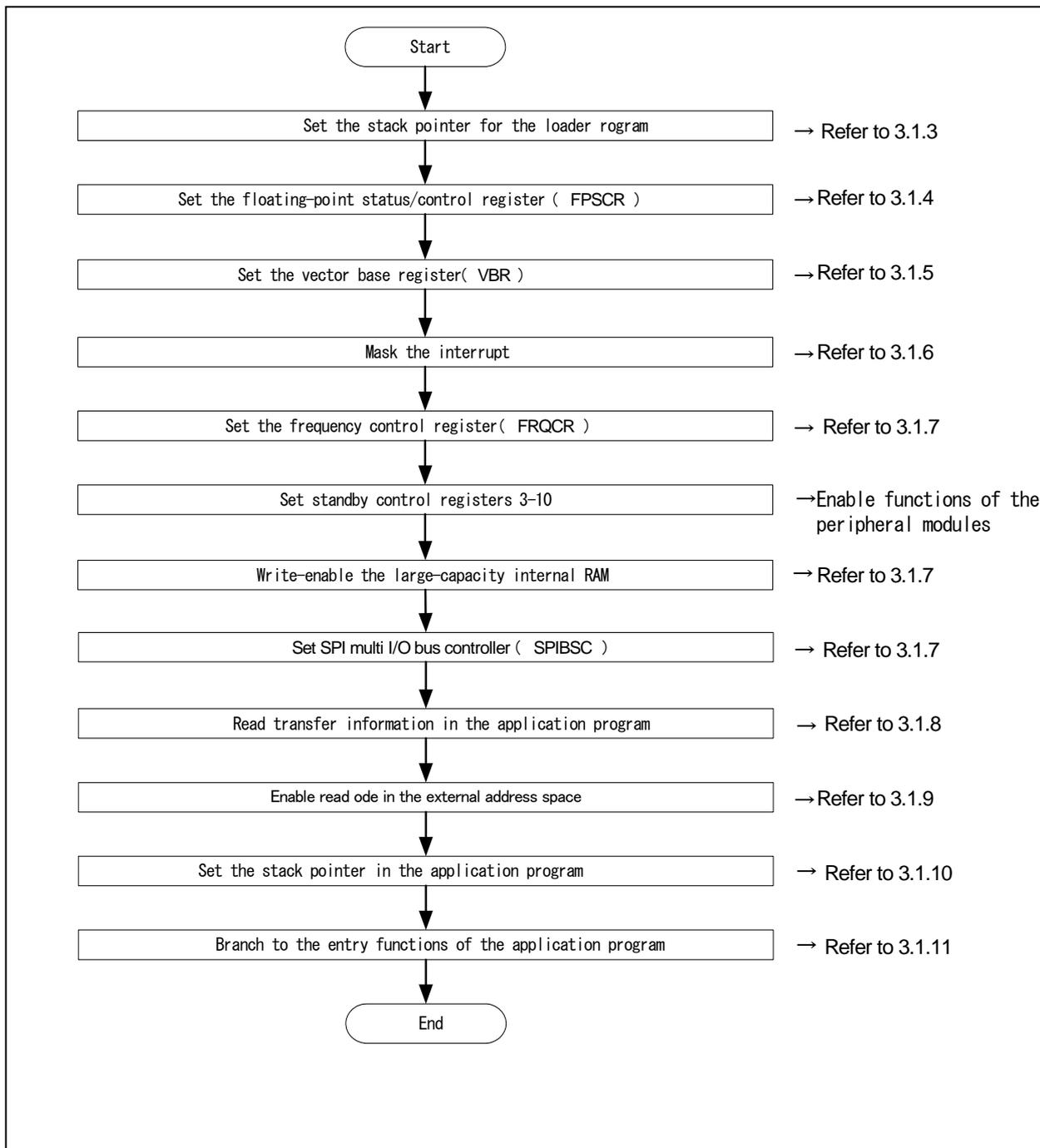
3.1.4　　　Floating-Point Status/Control Register (FPSCR) Setting

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

3.1.5　　　Vector Base Register (VBR) Setting

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

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

3.1.6　　　Interrupt Mask

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

3.1.7　　　Initial Configuration

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

3.1.8　　　Reading Transfer Information of Application Programs

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

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

Table 3　Transfer Information (appinfo) of Application Program

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

3.1.9 Enabling Read Mode in External Address Space

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

3.1.10 Application Program Stack Pointer Setting

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

3.1.11 Application Program Jump to the Entry Function Address

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

3.1.12 Register State after Executing Loader Program

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

Table 4   Resister State after Executing Loader Programs (1)

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

Table 5　Resister State after Executing Loader Programs (2)

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

3.2   Application Program Example

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

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

3.2.1   Section Alignment

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

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

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

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

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

Figure 7 shows an example of the section alignment.
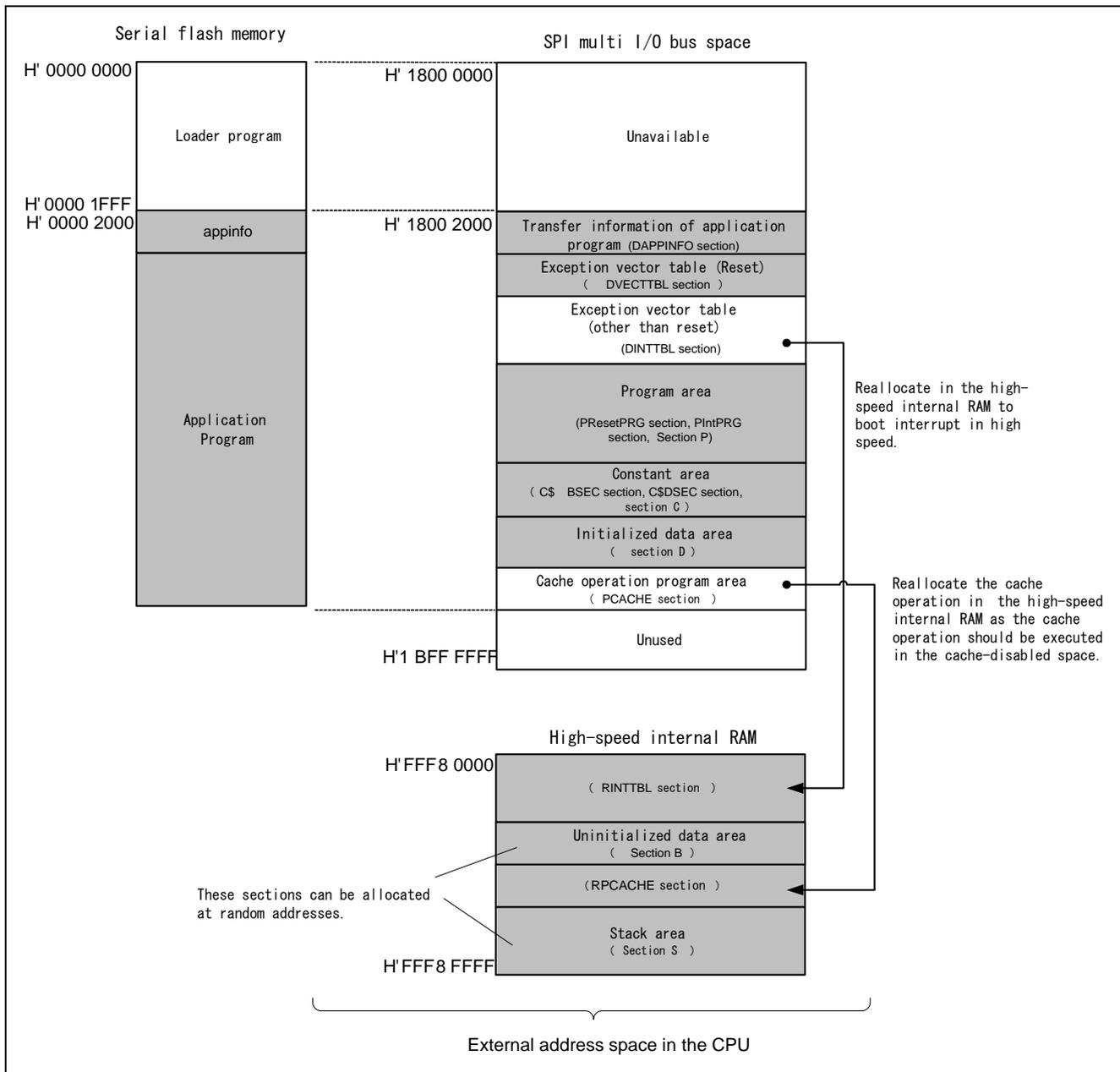
Figure 7   Application Program Section Alignment

3.2.2   Flow Chart

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

Figure 8 shows the flow chart of the application program.
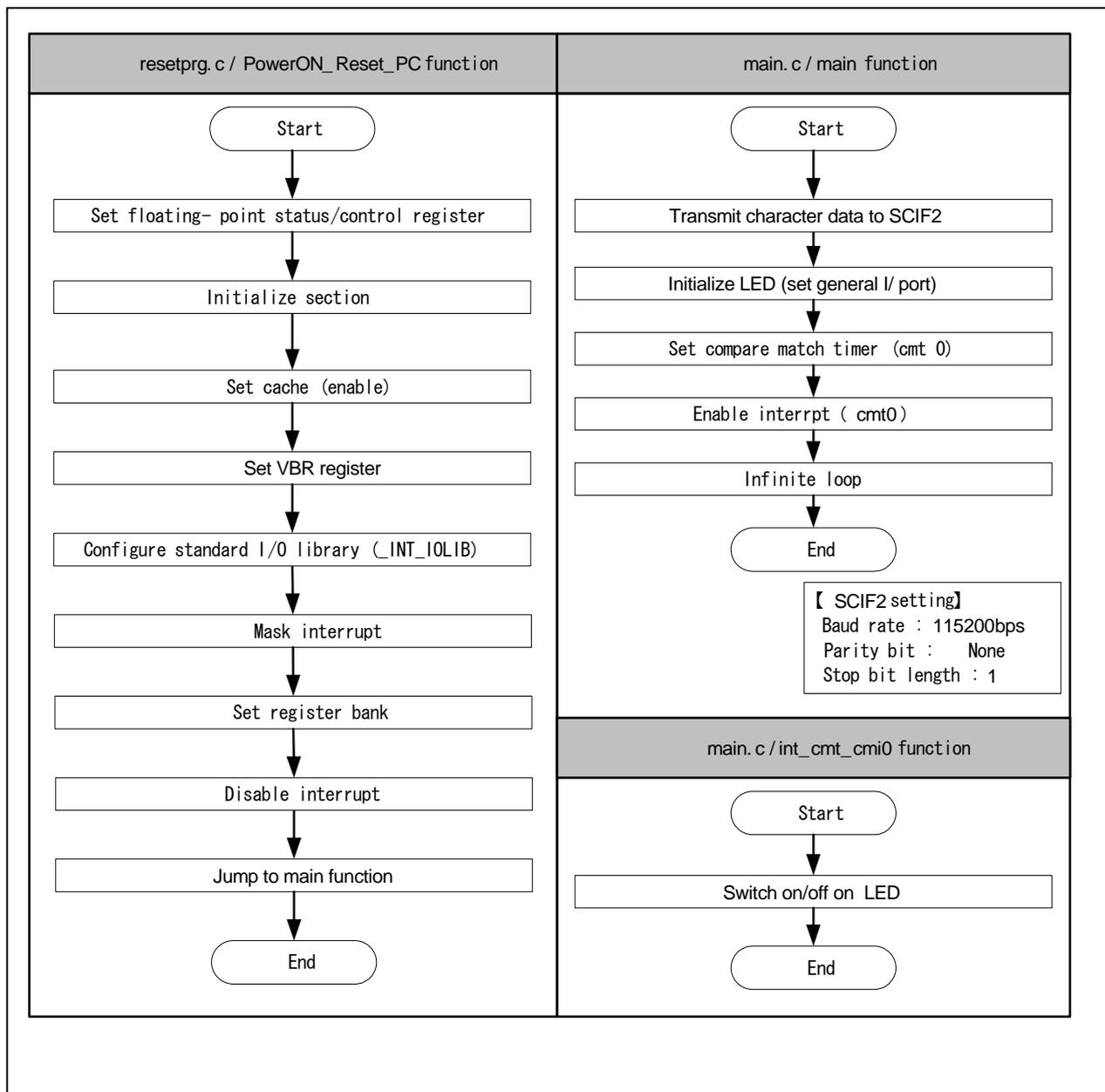
Figure 8   Application Program Flow Chart

3.2.3 Entry Function Setting

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

Table 6 Entry Function Address Settings

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

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

3.2.4 Stack Pointer Setting

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

Table 7 Stack Pointer Settings

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

3.2.5 Section Initialization

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

3.2.6 Cache Setting (Enable)

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

3.2.7 Vector Base Register (VBR) Setting

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

3.2.8 Generating the Application Program Transfer Information (appinfo)

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

Table 8 Application Program Transfer Information (appinfo)

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

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

Figure 9 shows the generation image of transfer information of application program (appinfo).
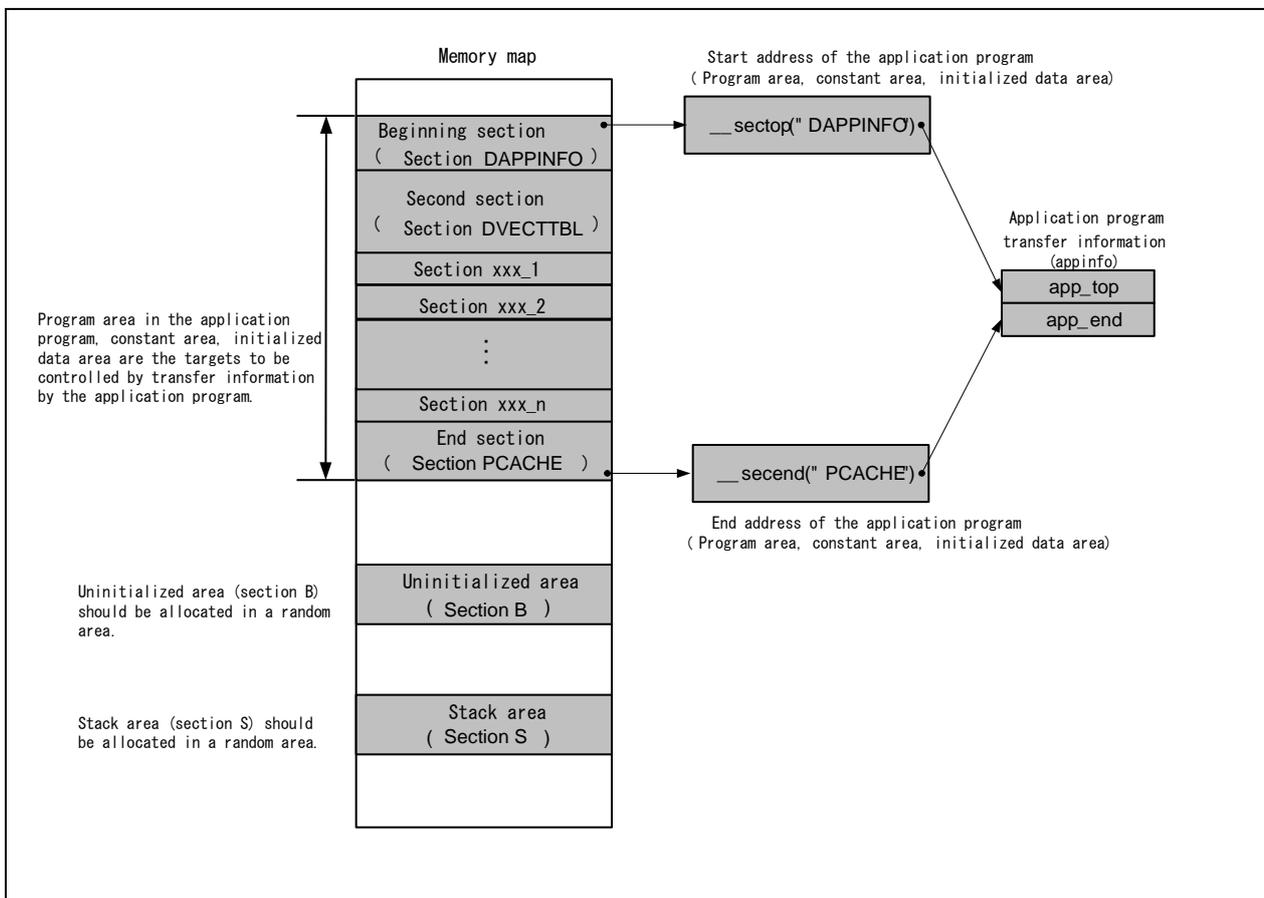
Figure 9   Generation image of Transfer Information (appinfo) of Application Program

3.3 Downloader Example

This section describes the downloader in this application.

3.3.1 Operation Overview

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

The application program is allocated in SPI multi I/O bus space area, but the debugger cannot transfer programs to SPI multi I/O bus space. So only the debug information of abs file in the SPI multi I/O bus area while mot file is downloaded in the large-capacity RAM.
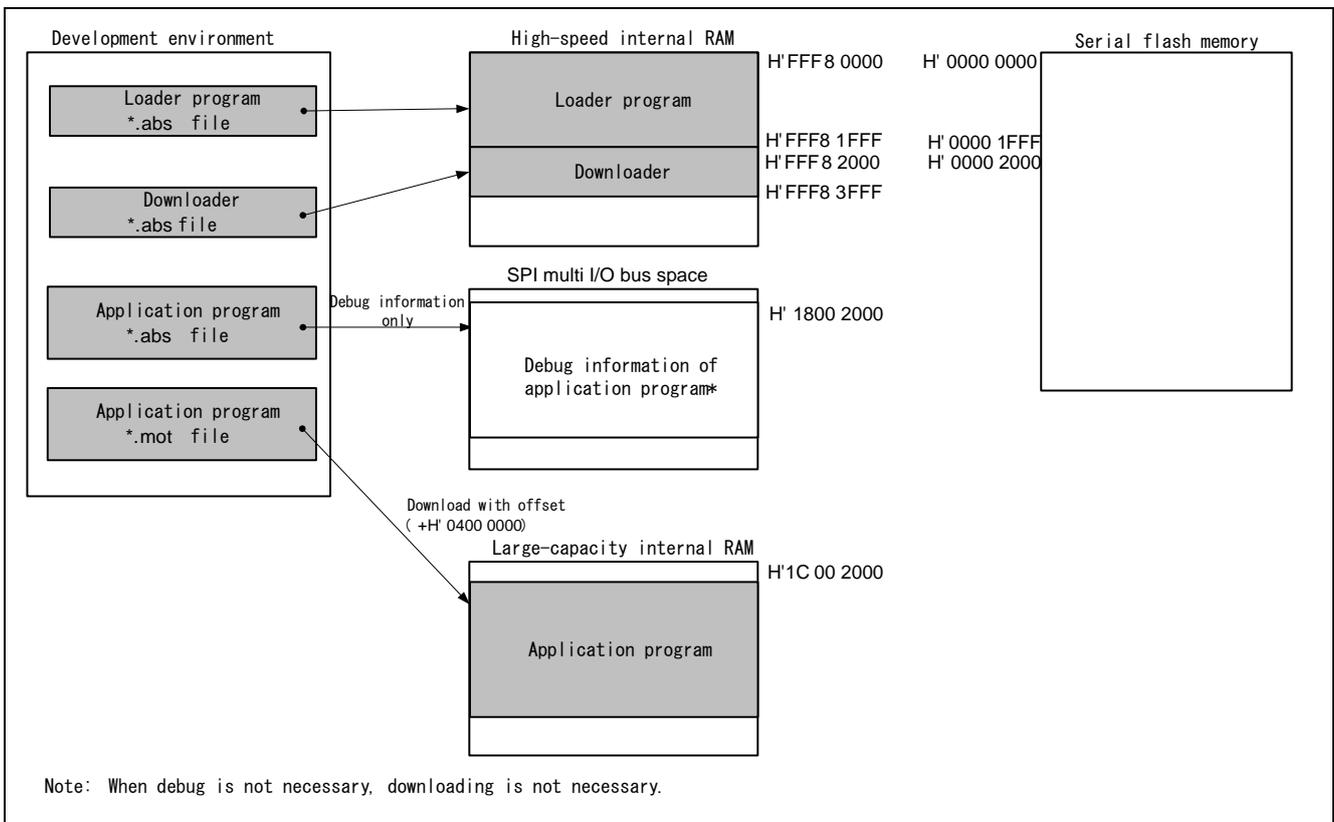
Figure 10   Downloader Operation Image (1)

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

Figure 11 shows another operation image of the downloader.
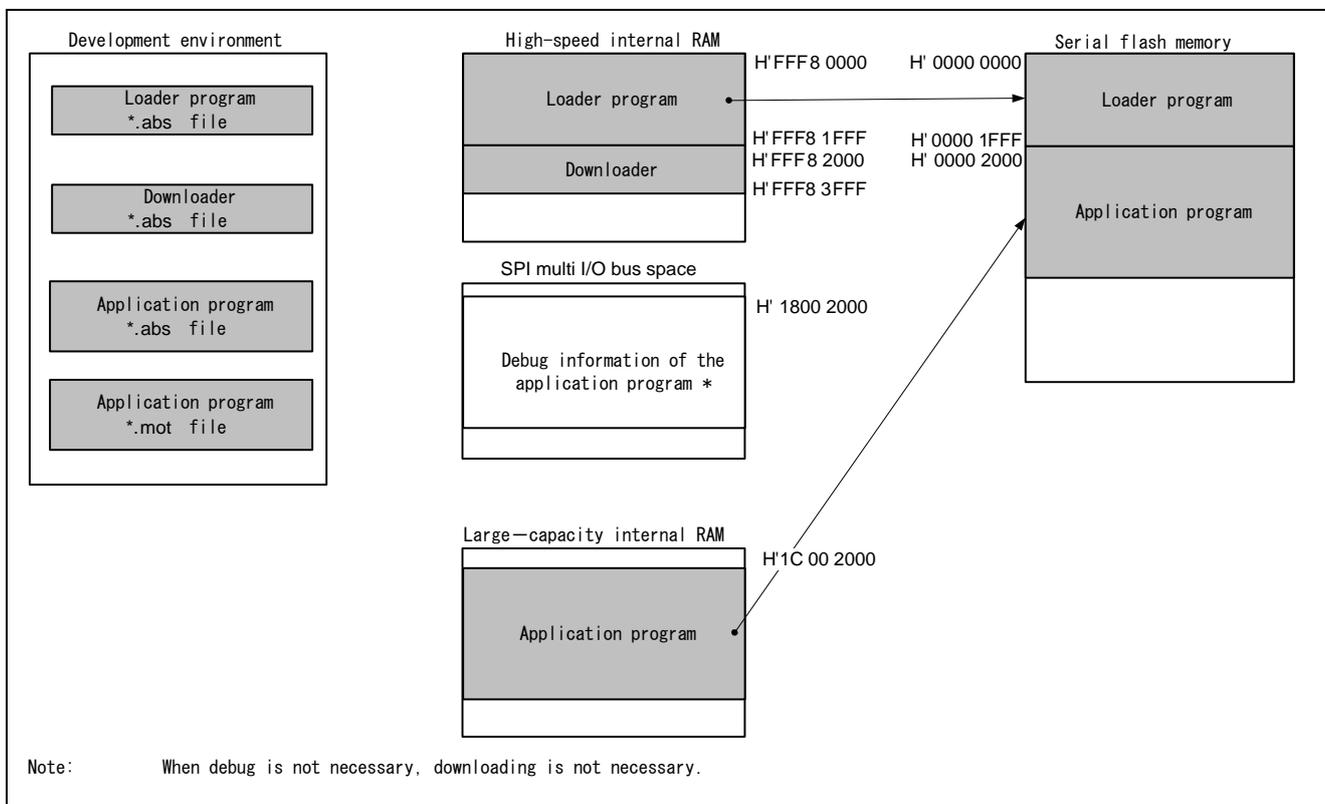
Figure 11   Downloader Operation Image

3.3.2   Areas Used by the Downloader

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

3.3.3   Flow Chart

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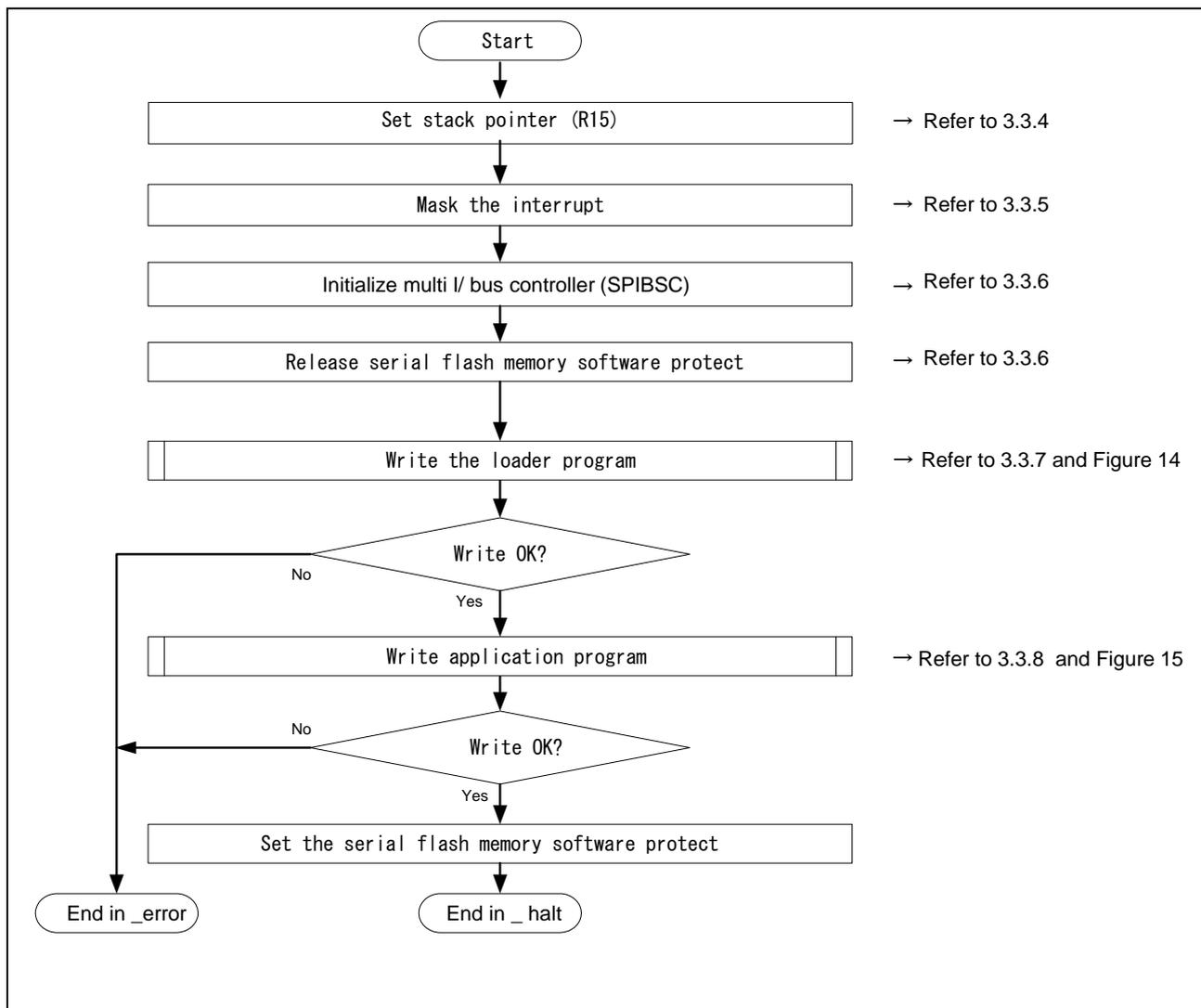
Figure 12   Downloader Flow Chart

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

```
                    ┌──────────────────────────────┐
                    │    Write the loader program    │
                    │  Write the application program │
                    └──────────────────────────────┘
                                   │
                                   ▼
                    ┌──────────────────────────────┐
                    │  Set the write destination address │
                    └──────────────────────────────┘
                                   │
                                   ▼
                    ┌──────────────────────────────┐
                    │       Set the write size       │
                    └──────────────────────────────┘
                                   │
            ┌──────────────────────▼
            │       ┌──────────────────────────────┐
            │       │  Calculate the target sector number │
            │       │      from the write address    │
            │       └──────────────────────────────┘
            │                      │
            │                      ▼
            │    Yes         ◇─────────────◇
            │  ◀─────────────  Sector erased ?
            │                ◇─────────────◇
            │                      │ No
            │                      ▼
            │       ┌──────────────────────────────┐
            │       │      Erase the target sector.  │
            │       └──────────────────────────────┘
            │                      │
            │                      ▼
            │       ┌──────────────────────────────┐
            │       │  Write by sectors (program command) │
            │       └──────────────────────────────┘
            │                      │
            │                      ▼
            │       ┌──────────────────────────────┐
            │       │ Update the write destination address │
            │       └──────────────────────────────┘
            │                      │
            │                      ▼
            │       ┌──────────────────────────────┐
            │       │      Update the write size     │
            │       └──────────────────────────────┘
            │                      │
            │   No                 ▼
            └◀────────────  ◇─────────────◇
                            Write size is 0 ?
                            ◇─────────────◇
                                   │ Yes
                                   ▼
                    ┌──────────────────────────────┐
                    │  Verify data (examine writing) │
                    └──────────────────────────────┘
                                   │
                                   ▼
                            ◇─────────────◇   No
                             Verify OK ?   ─────────┐
                            ◇─────────────◇         │
                                   │ Yes            │
                                   ▼                ▼
                          ┌──────────────┐  ┌──────────────┐
                          │   Write OK    │  │  Write error  │
                          └──────────────┘  └──────────────┘
```

- Erase the target sector before writing. When the sector is not erased, erase it by the Sector Erase command (H'D8) by the unit of 64KB

- Issue the Page Programming command (H'02) or the Quad Page Programming command (H'32) to write data by 1 byte.

- To verify that the writing is completed successfully, read the data in the serial flash memory and compare them with the provided data.

Figure 13   Flow Chart of Writing

3.3.4 Stack Pointer Setting

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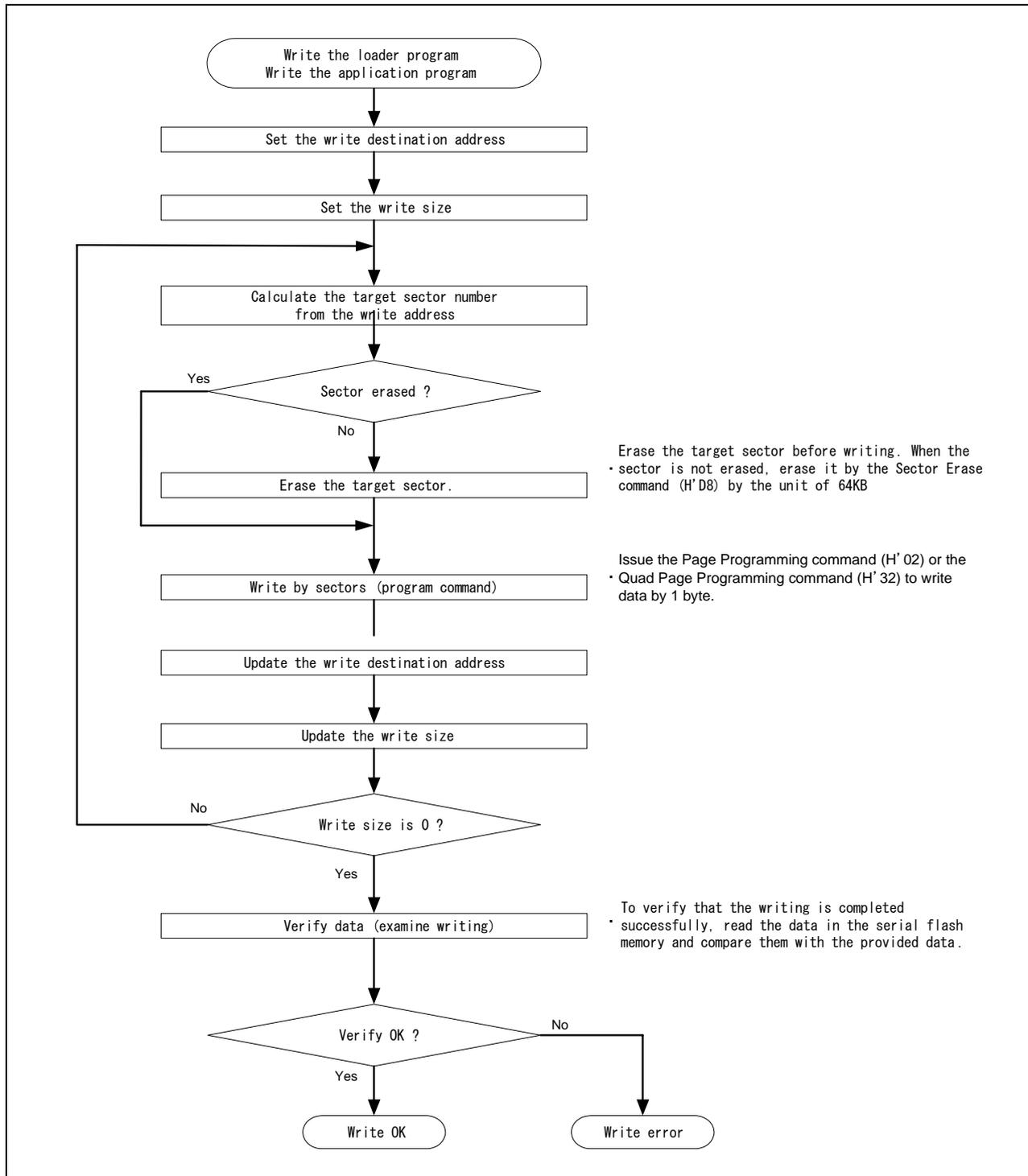er using the undefined stack pointer. C can be used after configuring the stack pointer. Then, the downloader jumps to the entry function of the downloader.

3.3.5 Interrupt Mask

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

3.3.6 Initialization

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

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

3.3.7 Writing the Loader Program

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

Table 9　Loader Program Writing

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

3.3.8　　Writing the Application Program

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

Table 10　Application Program Writing

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

3.3.9 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 though manual process possible.

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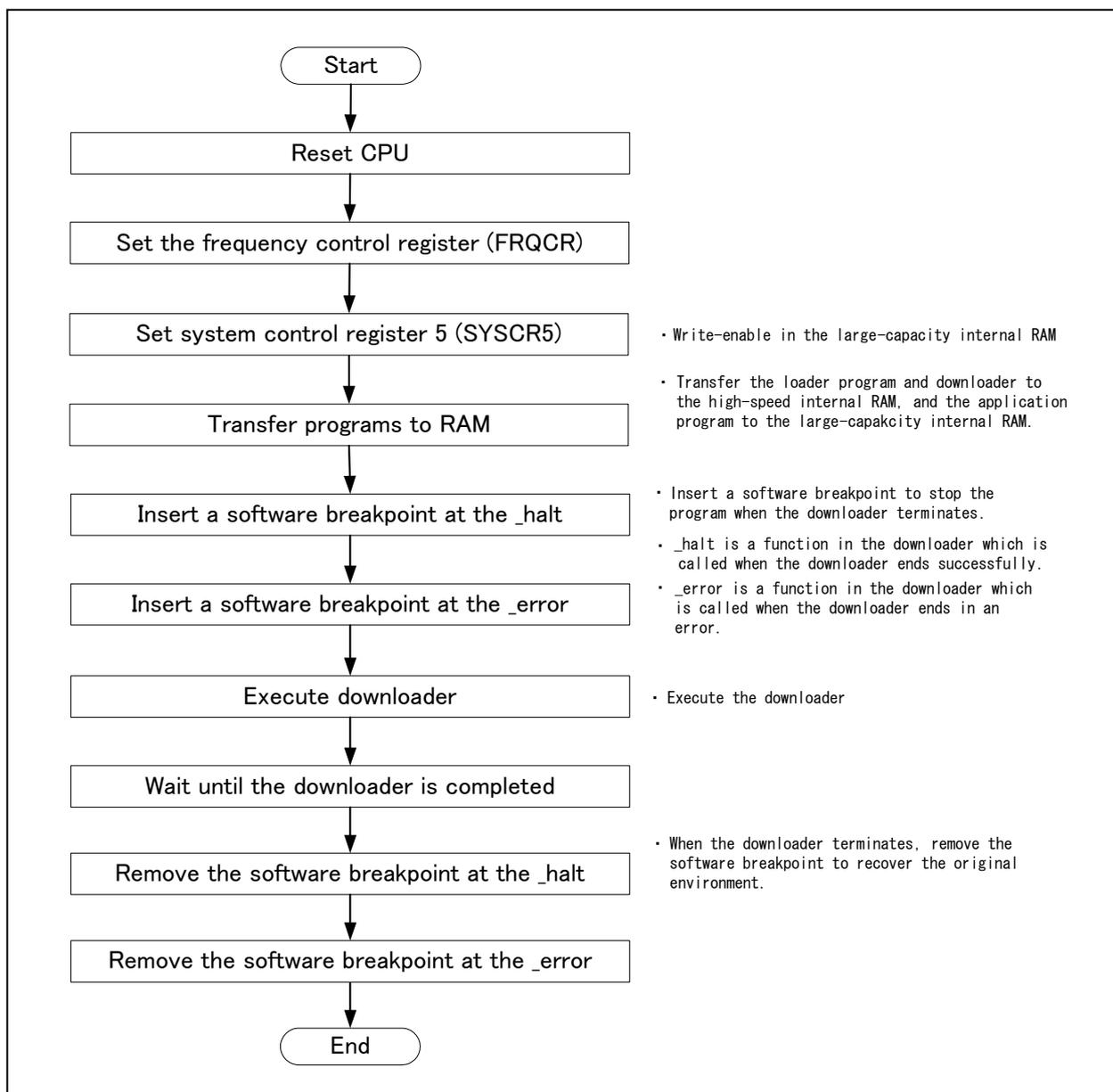

Figure 14   Command Batch File Flow Chart

4. Sample Program Listing

4.1 Loader Program

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

```
 1      ;/*******************************************************************************
 2      ;*    DISCLAIMER
 3      ;*
 4      ;*    This software is supplied by Renesas Electronics Corporation and is only
 5      ;*    intended for use with Renesas products. No other uses are authorized.
 6      ;*
 7      ;*    This software is owned by Renesas Electronics Corporation and is protected under
 8      ;*    all applicable laws, including copyright laws.
 9      ;*
10      ;*    THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
11      ;*    REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
12      ;*    INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
13      ;*    PARTICULAR PURPOSE AND NON-INFRINGEMENT.  ALL SUCH WARRANTIES ARE EXPRESSLY
14      ;*    DISCLAIMED.
15      ;*
16      ;*    TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
17      ;*    ELECTRONICS CORPORATION NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
18      ;*    FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
19      ;*    FOR ANY REASON RELATED TO THIS SOFTWARE, EVEN IF RENESAS OR ITS
20      ;*    AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
21      ;*
22      ;*    Renesas reserves the right, without notice, to make changes to this
23      ;*    software and to discontinue the availability of this software.
24      ;*    By using this software, you agree to the additional terms and
25      ;*    conditions found by accessing the following link:
26      ;*    http://www.renesas.com/disclaimer
27      ;*******************************************************************************
28      ;*    Copyright (C) 2011 Renesas Electronics Corporation. All rights reserved.
29      ;*************************** Technical reference data **************************
30      ;*    System Name : SH7268/SH7269 Firm Update Sample Program
31      ;*    File Name   : ld_loader.src
32      ;*    Abstract    : Loader program preprocessing/jump processing to the application
33      ;*                : program
34      ;*    Version     : 1.00.00
35      ;*    Device      : SH7268/SH7269
36      ;*    Tool-Chain  : High-performance Embedded Workshop (Ver.4.07.00).
37      ;*                : C/C++ compiler package for the SuperH RISC engine family
38      ;*                :                         (Ver.9.03Release02).
39      ;*    OS          : None
40      ;*    H/W Platform: R0K57269(CPU board)
41      ;*    Description :
42      ;*******************************************************************************
43      ;*    History     : Jul.06,2011 Ver.1.00.00
44      ;*""FILE COMMENT END""*********************************************************/
45        .SECTION LOADER_ENTRY,CODE,ALIGN = 4
46        .IMPORT _main
47        .EXPORT _jmp_app_prog
48
```

RENESAS

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

```
49    _loader_prog:
50      MOV.L L2,R15 ; Sets the stack pointer
51      MOV.L L1,R0      ; Retrieves the entry function of the loader program
52      JMP @R0          ; Jumps to the entry function of the loader program
53      NOP
54
55
56    ;/*""FUNC COMMENT""********************************************************
57    ; * ID         :
58    ; * Outline    : Jump to the application program
59    ; *-----------------------------------------------------------------------
60    ; * Include    :
61    ; *-----------------------------------------------------------------------
62    ; * Declaration : _jmp_app_prog
63    ; *-----------------------------------------------------------------------
64    ; * Description : 1. Retrieves the stack pointer value stored in the first 12 to
65    ; *              :    15 bytes in the application program.
66    ; *              : 2. Specifies the stack pointer (R15).
67    ; *              : 3. Retrieves the entry function address stored in the first 8 to
68    ; *              :    11 bytes in the application program.
69    ; *              : 4. Jumps to the entry function.
70    ; *-----------------------------------------------------------------------
71    ; * Argument   : R4  ; I : Start address of the application program
72    ; *-----------------------------------------------------------------------
73    ; * Return Value: none
74    ; *""FUNC COMMENT END""****************************************************/
75    _jmp_app_prog:
76
77      MOV.L R4,R0      ; Substitutes the start address of the application program for R0
78      ADD #12,R0       ; Calculates the address storing the stack pointer value and
79                       ; substitutes the address for R0
80      MOV.L @R0,R15     ; Sets the stack pointer
81
82      MOV.L R4,R0      ; Substitutes the start address of the application program for R0
83      ADD #8,R0        ; Calculates the address storing the entry function of the application
84                       ; program and substitutes the address for R0
85      MOV.L @R0,R0      ; Substitutes the entry function address of the application
86                       ; program for R0
87      JMP @R0          ; Jumps to the entry function of the application program
88      NOP
89
90
91      .ALIGN 4
92    L1:
93      .DATA.L _main               ; Entry function address of the loader program
94
95    L2:
96      .DATA.L H'FFF82000          ; Stack pointer (R15) value of the loader program
97
98      .pool
99      .end
100
101   ;/* End of File */
```

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

```
 1     /**************************************************************************
 2      *    DISCLAIMER
 3      *
 4      *    This software is supplied by Renesas Electronics Corporation and is only
 5      *    intended for use with Renesas products. No other uses are authorized.
 6      *
 7      *    This software is owned by Renesas Electronics Corporation and is protected under
 8      *    all applicable laws, including copyright laws.
 9      *
10      *    THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
11      *    REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
12      *    INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
13      *    PARTICULAR PURPOSE AND NON-INFRINGEMENT.  ALL SUCH WARRANTIES ARE EXPRESSLY
14      *    DISCLAIMED.
15      *
16      *    TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
17      *    ELECTRONICS CORPORATION NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
18      *    FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
19      *    FOR ANY REASON RELATED TO THIS SOFTWARE, EVEN IF RENESAS OR ITS
20      *    AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
21      *
22      *    Renesas reserves the right, without notice, to make changes to this
23      *    software and to discontinue the availability of this software.
24      *    By using this software, you agree to the additional terms and
25      *    conditions found by accessing the following link:
26      *    http://www.renesas.com/disclaimer
27      **************************************************************************
28      *    Copyright (C) 2011 Renesas Electronics Corporation. All rights reserved.
29      *************************** Technical reference data ***************************
30      *    System Name : SH7268/SH7269 Firm Update Sample Program
31      *    File Name   : ld_main.c
32      *    Abstract    : loader main
33      *    Version     : 1.00.00
34      *    Device      : SH7268/SH7269
35      *    Tool-Chain  : High-performance Embedded Workshop (Ver.4.07.00).
36      *                : C/C++ compiler package for the SuperH RISC engine family
37      *                :                              (Ver.9.03 Release02).
38      *    OS          : None
39      *    H/W Platform: R0K57269(CPU board)
40      *    Description :
41      **************************************************************************
42      *    History    : Jul.06,2011 Ver.1.00.00
43      **************************************************************************/
44     #include <stdio.h>
45     #include <string.h>
46     #include <machine.h>
47     #include "iodefine.h"
48     #include "serial_flash.h"
49
```

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

```
50      /* ==== macro defined ==== */
51      #define FPSCR_INIT      0x00040001          /* Value to set in the FPSCR register */
52      #define INT_MASK        0x000000F0          /* Value to set in the SR register
53                                                  (for masking the interrupt) */
54
55      #define APROG_TOP_SFLASH   0x00002000       /* Start address of the application program */
56                                                  /* (serial flash memory) */
57
58      #define APPINFO_TOP     APROG_TOP_SFLASH       /* Address the appinfo.app_top is located */
59      #define APPINFO_END     (APROG_TOP_SFLASH + 4) /* Address the appinfo.app_end is located */
60
61
62      /* ==== prototype declaration ==== */
63      void main(void);
64      void get_appinfo( unsigned long *app_top_addr,unsigned long *app_end_addr);
65      void app_prog_transfer(unsigned long app_top_addr,unsigned long app_end_addr);
66      void system_down(void);
67
68      extern void jmp_app_prog(unsigned long app_top_addr);
69      extern void io_set_cpg(void);
70      extern void sf_byte_read_long(unsigned long addr, unsigned long *buf, int size);
71
72      /* ==== external data ==== */
73      extern unsigned long DUMMY_Vectors;
74
75
```

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

```
76      /***************************************************************************
77       * ID         :
78       * Outline    : Loader program main
79       * Include    : #include "serial_flash.h"
80       * Declaration : void main(void);
81       * Description : Refers the data in the appinfo to transfer the application program
82       *             : to the large-capacity internal RAM, and jumps to the entry function
83       *             : of the application program.
84       * Argument    : void
85       * Return Value: void
86       ***************************************************************************/
87      void main(void)
88      {
89        unsigned long app_top,app_end;
90
91
92        /* Sets the FPSCR */
93        set_fpscr(FPSCR_INIT);
94
95        /* Sets the tentative VBR */
96        set_vbr((void *)(&DUMMY_Vectors));
97
98        /* Masks the interrupt */
99        set_cr(INT_MASK);
100
101       /* Sets the CPG */
102       io_set_cpg();
103
104       /* Sets the SPIBSC */
105       sf_init_serial_flash_spibsc();
106
107       /* Retrieves the appinfo */
108       get_appinfo(&app_top,&app_end);
109
110       sf_allocate_exspace_spibsc();
111
112       /* Jumps to the application program */
113       jmp_app_prog(app_top);
114
115       while(1){
116         /* LOOP */
117       }
118     }
119
```

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

```
120    /***************************************************************************
121     * ID          :
122     * Outline     : Retrieve the appinfo
123     * Include     : #include "serial_flash.h"
124     * Declaration : void get_appinfo (unsigned long *app_top_addr,
125     *             :                    unsigned long *app_end_addr);
126     * Description : Retrieves the appinfo.
127     *             : Retrieves the appinfo.top from H'2000 to H'2003 in serial flash
128     *             : memory, and stores it in the address specified by the first
129     *             : argument. This function also retrieves the appinfo.end from
130     *             : H'2004 to H'2007 in serial flash memory, and stores it in the
131     *             : address specified by the second argument.
132     * Argument    : unsigned long app_top_addr  ; O : Start address of the application
133     *             :                                   program at destination
134     *             : unsigned long app_end_addr  ; O : End address of the application
135     *             :                                   program at destination
136     * Return Value: void
137     ***************************************************************************/
138    void get_appinfo( unsigned long *app_top_addr,unsigned long *app_end_addr)
139    {
140
141      /* Retrieves the appinfo.top */
142      sf_byte_read_spibsc(APPINFO_TOP, (unsigned char *)app_top_addr, 4);
143
144      /* Retrieves the appinfo.end */
145      sf_byte_read_spibsc(APPINFO_END, (unsigned char *)app_end_addr, 4);
146
147    }
148
```

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

```
149    /*****************************************************************************
150     * ID        :
151     * Outline   : Transfer the application program
152     * Include   : #include "serial_flash.h"
153     * Declaration : void app_prog_transfer(unsigned long app_top_addr,
154     *               :                      unsigned long app_end_addr);
155     * Description : Calculates the size of the application program, and transfers
156     *               : the application program from serial flash memory to the
157     *               : large-capacity internal RAM. (Rounds up the allocation size of the
158     *               : application program to multiples of 4 to transfer in longword.)
159     * Argument    : unsigned long app_top_addr  ; I : Start address of the application
160     *               :                                program at destination
161     *               : unsigned long app_end_addr  ; I : End address of the application
162     *               :                                at destination
163     * Return Value: void
164     *****************************************************************************/
165    void app_prog_transfer(unsigned long app_top_addr,unsigned long app_end_addr)
166    {
167      unsigned long app_prog_size;
168
169      /* Calculates the size of the application program */
170      app_prog_size = app_end_addr - app_top_addr;
171      if( ( app_prog_size & 0x00000003 ) != 0 ){
172        app_prog_size &= 0xFFFFFFFC;
173        app_prog_size += 4;          /* Rounds up the allocation size of the application
174                                        program to multiples of 4. */
175      }
176
177      /* Loads the application program in the large-capacity internal RAM */
178      sf_byte_read_spibsc(APROG_TOP_SFLASH, (unsigned char *)app_top_addr, app_prog_size);
179
180    }
181
182    /*****************************************************************************
183     * ID        :
184     * Outline   : Terminate the system
185     * Include   :
186     * Declaration : void system_down(void);
187     * Description : This function contains the infinite loop.
188     *               : As this is registered in the DUMMY_Vectors table, this is
189     *               : called when an exception occurs while the loader program
190     *               : is operating.
191     * Argument    : void
192     * Return Value: void
193     *****************************************************************************/
194    void system_down(void)
195    {
196      while(1){
197        /* System error */
198      }
199    }
200
201    /* End of File */
```

RENESAS

4.2 Application Program

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

```
1    /*******************************************************************************
2    *    DISCLAIMER
3    *
4    *    This software is supplied by Renesas Electronics Corporation and is only
5    *    intended for use with Renesas products. No other uses are authorized.
6    *
7    *    This software is owned by Renesas Electronics Corporation and is protected under
8    *    all applicable laws, including copyright laws.
9    *
10   *    THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
11   *    REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
12   *    INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
13   *    PARTICULAR PURPOSE AND NON-INFRINGEMENT.  ALL SUCH WARRANTIES ARE EXPRESSLY
14   *    DISCLAIMED.
15   *
16   *    TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
17   *    ELECTRONICS CORPORATION NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
18   *    FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
19   *    FOR ANY REASON RELATED TO THIS SOFTWARE, EVEN IF RENESAS OR ITS
20   *    AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
21   *
22   *    Renesas reserves the right, without notice, to make changes to this
23   *    software and to discontinue the availability of this software.
24   *    By using this software, you agree to the additional terms and
25   *    conditions found by accessing the following link:
26   *    http://www.renesas.com/disclaimer
27   *******************************************************************************
28   *    Copyright (C) 2011 Renesas Electronics Corporation. All rights reserved.
29   *************************** Technical reference data **************************
30   *    System Name : SH7268/SH7269 Sample Program
31   *    File Name   : main.c
32   *    Abstract    : Sample Program Main
33   *    Version     : 1.00.00
34   *    Device      : SH7268/SH7269
35   *    Tool-Chain  : High-performance Embedded Workshop (Ver.4.07.00).
36   *                : C/C++ compiler package for the SuperH RISC engine family
37   *                :                           (Ver.9.03Release02).
38   *    OS          : None
39   *    H/W Platform: R0K57269(CPU board)
40   *    Description :
41   *******************************************************************************
42   *    History     : Jul.06,2011 Ver.1.00.00
43   *******************************************************************************/
44   #include <stdio.h>
45   #include <string.h>
46   #include <machine.h>
47   #include "iodefine.h"
48   #include "r0k57269.h"
49
```

RENESAS

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

```
50      /* ==== prototype declaration ==== */
51      void main(void);
52      void io_init_cmt0(void);
53      void int_cmt_cmi0(void);
54
55      /* ==== Global variable ==== */
56      int g_led_onoff;         /* LED lighting/turning off */
57
58      /*****************************************************************************
59       * ID        :
60       * Outline     : main
61       *---------------------------------------------------------------------------
62       * Include    :
63       *---------------------------------------------------------------------------
64       * Declaration : void main(void);
65       *---------------------------------------------------------------------------
66       * Description :
67       *---------------------------------------------------------------------------
68       * Argument    : void
69       *---------------------------------------------------------------------------
70       * Return Value: void
71       *---------------------------------------------------------------------------
72       * Note       : None
73       *****************************************************************************/
74      void main(void)
75      {
76        puts("\nSH7269 CPU Board Sample Program. Ver.0.02.00");
77        puts("Copyright (C) 2010(2011) Renesas Electronics Corporation. All rights eserved.");
78        puts("\n");
79
80        /* ==== initial LED port ==== */
81        g_led_onoff = 1;
82        led_init();                        /* LED Initialization */
83
84        /* ==== start timer ==== */
85        io_init_cmt0();                    /* CMT Initialization */
86
87        /* ==== Setting of interrupt priority level ==== */
88        INTC.IPR12.BIT._CMT0 = 0x1;      /* CMI Priority level of interrupt = 1 */
89
90        while(1){
91             /* loop */
92        }
93      }
94
```

RENESAS

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

```
95      /*****************************************************************************
96       * ID          :
97       * Outline     : CMT0 setting
         *---------------------------------------------------------------------------
99       * Include     : #include "iodefine.h"
         *---------------------------------------------------------------------------
101      * Declaration : void io_init_cmt0(void) ;
         *---------------------------------------------------------------------------
103      * Description : CMT0 is set as a fixed cycle of about 500ms timer.
         *---------------------------------------------------------------------------
105      * Argument    : void
         *---------------------------------------------------------------------------
107      * Return Value: void
         *---------------------------------------------------------------------------
109      * Note        : None
110      *****************************************************************************/
111     void io_init_cmt0(void)
112     {
113
114         /* ---- STBCR7 setting ---- */
115         CPG.STBCR7.BIT.MSTP72 = 0;        /* Module standby clear */
116
117         /* ==== CMT0 setting ==== */
118         /* ---- CMSTR setting ---- */
119         CMT.CMSTR.BIT.STR0 = 0;            /* Count stop */
120         /* ---- CMCSR0 setting ---- */
121         CMT.CMCSR0.WORD = 0x0043;         /* Pclock/512 */
122
123         /* ---- CMCNT0 setting ---- */
124         CMT.CMCNT0.WORD = 0x0000;          /* Timer counter clear */
125
126         /* ---- CMCOR0 setting ---- */
127         CMT.CMCOR0.WORD = 0x7f08/5;        /* 500/5=100ms */
128
129       /* ---- CMSTR setting ---- */
130         CMT.CMSTR.BIT.STR0 = 0x1;          /* Count start */
131
132     }
133
```

4.2.4　　Application Program Listing "main.c"(4)

```
134    /****************************************************************************
135     * ID        :
136     * Outline    : CMI interrupt
137     *--------------------------------------------------------------------------
138     * Include    : #include "iodefine.h"
139     *--------------------------------------------------------------------------
140     * Declaration : void int_cmt_cmi0(void);
141     *--------------------------------------------------------------------------
142     * Description : The CMF flag is cleared, and the output of
143     *            : LED of each 0.5sec is reversed.
144     *--------------------------------------------------------------------------
145     * Argument   : void
146     *--------------------------------------------------------------------------
147     * Return Value: void
148     *--------------------------------------------------------------------------
149     * Note       : None
150     ****************************************************************************/
151    void int_cmt_cmi0(void)
152    {
153
154      /* ====CMF Clearness of flag ==== */
155      CMT.CMCSR0.BIT.CMF = 0;
156
157      /* ==== PORT Reversing output(LED blinking) ==== */
158      g_led_onoff ^= 1;
159      if(g_led_onoff == 0){
160        led_on(ID_LED1);
161        led_on(ID_LED2);
162      }
163      else{
164        led_off(ID_LED1);
165        led_off(ID_LED2);
166      }
167    }
168
169    /* End of File */
```

4.2.5 Application Program Listing "appinfo.c"(1)

```
 1      /***************************************************************************
 2      *   DISCLAIMER
 3      *
 4      *   This software is supplied by Renesas Electronics Corporation and is only
 5      *   intended for use with Renesas products. No other uses are authorized.
 6      *
 7      *   This software is owned by Renesas Electronics Corporation and is protected under
 8      *   all applicable laws, including copyright laws.
 9      *
10      *   THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
11      *   REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
12      *   INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
13      *   PARTICULAR PURPOSE AND NON-INFRINGEMENT.  ALL SUCH WARRANTIES ARE EXPRESSLY
14      *   DISCLAIMED.
15      *
16      *   TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
17      *   ELECTRONICS CORPORATION NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
18      *   FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
19      *   FOR ANY REASON RELATED TO THIS SOFTWARE, EVEN IF RENESAS OR ITS
20      *   AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
21      *
22      *   Renesas reserves the right, without notice, to make changes to this
23      *   software and to discontinue the availability of this software.
24      *   By using this software, you agree to the additional terms and
25      *   conditions found by accessing the following link:
26      *   http://www.renesas.com/disclaimer
27      ****************************************************************************
28      *   Copyright (C) 2011 Renesas Electronics Corporation. All rights reserved.
29      *************************** Technical reference data *************************
30      *   System Name : SH7268/SH7269 Sample Program
31      *   File Name   : appinfo.c
32      *   Abstract    : Generate the application program transfer information (appinfo).
33      *   Version     : 1.00.00
34      *   Device      : SH7268/SH7269
35      *   Tool-Chain  : High-performance Embedded Workshop (Ver.4.07.00).
36      *               : C/C++ compiler package for the SuperH RISC engine family
37      *               :                        (Ver.9.03 Release02).
38      *   OS          : None
39      *   H/W Platform: R0K57269(CPU board)
40      *   Description :
41      ****************************************************************************
42      *   History     : Jul.06,2011 Ver.1.00.00
43      ****************************************************************************/
```

RENESAS

4.2.6 Application Program Listing "appinfo.c"(2)

```
44      #include "appinfo.h"

45

46      #pragma section APPINFO

47

48      static APPINFO appinfo = {

49        __sectop("DAPPINFO"),   /* Start address in the start section of the application */

50                                /* program (program area, constant area, and initialized */

51                                /* data area). */

52

53        __secend("PCACHE")      /* End address in the end section of the application */

54                                /* program (program area, constant area, and initialized */

55                                /* data area) */

56      };

57

58      /* End of File */

59
```

4.2.7 Application Program Listing "appinfo.h"

```
1      /*******************************************************************************
2      *   DISCLAIMER
3      *
4      *   This software is supplied by Renesas Electronics Corporation and is only
5      *   intended for use with Renesas products. No other uses are authorized.
6      *
7      *   This software is owned by Renesas Electronics Corporation and is protected under
8      *   all applicable laws, including copyright laws.
9      *
10     *   THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
11     *   REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
12     *   INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
13     *   PARTICULAR PURPOSE AND NON-INFRINGEMENT.  ALL SUCH WARRANTIES ARE EXPRESSLY
14     *   DISCLAIMED.
15     *
16     *   TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
17     *   ELECTRONICS CORPORATION NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
18     *   FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
19     *   FOR ANY REASON RELATED TO THIS SOFTWARE, EVEN IF RENESAS OR ITS
20     *   AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
21     *
22     *   Renesas reserves the right, without notice, to make changes to this
23     *   software and to discontinue the availability of this software.
24     *   By using this software, you agree to the additional terms and
25     *   conditions found by accessing the following link:
26     *   http://www.renesas.com/disclaimer
27     *******************************************************************************
28     *   Copyright (C) 2011 Renesas Electronics Corporation. All rights reserved.
29     *************************** Technical reference data **************************
30     *   System Name : SH7268/SH7269 Sample Program
31     *   File Name   : appinfo.h
32     *   Abstract    : Header file of the application program transfer information (appinfo).
33     *   Version     : 1.00.00
34     *   Device      : SH7268/SH7269
35     *   Tool-Chain  : High-performance Embedded Workshop (Ver.4.07.00).
36     *               : C/C++ compiler package for the SuperH RISC engine family
37     *               :                           (Ver.9.03 Release02).
38     *   OS          : None
39     *   H/W Platform: R0K57269(CPU board)
40     *   Description :
41     *******************************************************************************
42     *   History     : Jul.06,2011 Ver.1.00.00
43     *******************************************************************************/
44     #ifndef __APPINFO_H__
45     #define __APPINFO_H__
46
47     typedef struct appinfo_t {
48       void *app_top;          /* Start address of the application program */
49       void *app_end;          /* End address of the application program */
50     } APPINFO;
51
52     #endif /* __APPINFO_H__ */
53     /* End of File */
```

RENESAS

4.3 Downloader

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

```
1     #/********************************************************************
2     #*    DISCLAIMER
3     #*
4     #*    This software is supplied by Renesas Electronics Corporation and is only
5     #*    intended for use with Renesas products. No other uses are authorized.
6     #*
7     #*    This software is owned by Renesas Electronics Corporation and is protected under
8     #*    all applicable laws, including copyright laws.
9     #*
10    #*    THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
11    #*    REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
12    #*    INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
13    #*    PARTICULAR PURPOSE AND NON-INFRINGEMENT.  ALL SUCH WARRANTIES ARE EXPRESSLY
14    #*    DISCLAIMED.
15    #*
16    #*    TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
17    #*    ELECTRONICS CORPORATION NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
18    #*    FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
19    #*    FOR ANY REASON RELATED TO THIS SOFTWARE, EVEN IF RENESAS OR ITS
20    #*    AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
21    #*
22    #*    Renesas reserves the right, without notice, to make changes to this
23    #*    software and to discontinue the availability of this software.
24    #*    By using this software, you agree to the additional terms and
25    #*    conditions found by accessing the following link:
26    #*    http://www.renesas.com/disclaimer
27    #*********************************************************************
28    #*    Copyright (C) 2011 Renesas Electronics Corporation. All rights reserved.
29    #************************** Technical reference data ************************
30    #*    System Name : SH7268/SH7269 Firm Update Sample Program
31    #*    File Name   : downloader.hdc
32    #*    Abstract    : ダウンローダ用バッチファイル
33    #*    Version     : 1.00.00
34    #*    Device      : SH7269/SH7269
35    #*    Tool-Chain  : High-performance Embedded Workshop (Ver.4.07.00).
36    #*                : C/C++ compiler package for the SuperH RISC engine family
37    #*                :                             (Ver.9.03Release02).
38    #*    OS          : None
39    #*    H/W Platform: R0K57269(CPU board)
40    #*    Description :
41    #*********************************************************************
42    #*    History     : Jul.06,2011 Ver.1.00.00
43    #*""FILE COMMENT END""*********************************************************/
44
45
46    tcl enable
47
48
```

RENESAS

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

```
49     #Macro downloader -Start
50     proc init_hardware {} {
51
52       # set CPG
53       MF H'FFFE0010 H'FFFE0011 H'1015 WORD
54
55       # set SYSCR5
56       MF H'FFFE0428 H'FFFE0428 H'0F BYTE
57     }
58
59
60     proc downloader {} {
61       # reset CPU
62       reset
63
64       # init_hardware call routines
65       init_hardware
66
67       #download all [Download modules] in High-performance Embedded Workshop
68       file_load_all
69
70       #permit user stack (to use software breakpoint)
71       sh2a_sbstk enable
72
73       # set software break point in _halt(refer to main.c)
74       set_disassembly_soft_break _halt set
75
76       # set software break point in _error(refer to main.c)
77       set_disassembly_soft_break _error set
78
79       # execute _downloader(refer to downloader.src). Wait till stops
80       go wait _downloader
81
82       # clear software break point set in _halt
83       set_disassembly_soft_break _halt clear
84
85       # clear the software break point set in _error
86       set_disassembly_soft_break _error clear
87
88     }
89
90     downloader
91     #Macro downloader -End
92
93
94
95     #Note: "tcl","reset","file_load","sh2a_sbstk","set_disassembly_soft_break","go" are、the
96     commands of #High-performance Embedded Workshop and E10A-USB emulator. For the details of
97     commands, see the Manual.。
98
99     # /* End of File */
```

RENESAS

4.3.3 Downloader Program Listing "dl_entry.src" (1)

```
1     ;/**************************************************************************
2     ;*   DISCLAIMER
3     ;*
4     ;*   This software is supplied by Renesas Electronics Corporation and is only
5     ;*   intended for use with Renesas products. No other uses are authorized.
6     ;*
7     ;*   This software is owned by Renesas Electronics Corporation and is protected under
8     ;*   all applicable laws, including copyright laws.
9     ;*
10    ;*   THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
11    ;*   REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
12    ;*   INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
13    ;*   PARTICULAR PURPOSE AND NON-INFRINGEMENT.  ALL SUCH WARRANTIES ARE EXPRESSLY
14    ;*   DISCLAIMED.
15    ;*
16    ;*   TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
17    ;*   ELECTRONICS CORPORATION NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
18    ;*   FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
19    ;*   FOR ANY REASON RELATED TO THIS SOFTWARE, EVEN IF RENESAS OR ITS
20    ;*   AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
21    ;*
22    ;*   Renesas reserves the right, without notice, to make changes to this
23    ;*   software and to discontinue the availability of this software.
24    ;*   By using this software, you agree to the additional terms and
25    ;*   conditions found by accessing the following link:
26    ;*   http://www.renesas.com/disclaimer
27    ;**************************************************************************
28    ;*   Copyright (C) 2010(2011) Renesas Electronics Corporation. All rights reserved.
29    ;************************** Technical reference data **************************
30    ;*   System Name : SH7268/SH7269 Firm Update Sample Program
31    ;*   File Name   : dl_entry.src
32    ;*   Abstract    : downloader start up
33    ;*   Version     : 0.03.00
34    ;*   Device      : SH7268/SH7269
35    ;*   Tool-Chain  : High-performance Embedded Workshop (Ver.4.07.00).
36    ;*               : C/C++ compiler package for the SuperH RISC engine family
37    ;*               :                             (Ver.9.03Release02).
38    ;*   OS          : None
39    ;*   H/W Platform: R0K57269(CPU board)
40    ;*   Description :
41    ;**************************************************************************
42    ;*   History     : Sep.06,2010 Ver.0.01.00
43    ;*               : Apr.27,2011 Ver.0.02.00 change comment
44    ;*               : May.08,2011 Ver.0.03.00 change comment
45    ;*""FILE COMMENT END""****************************************************/
```

4.3.4 Downloader Program Listing "dl_entry.src" (2)

```
46        .SECTION ENTRY,CODE,ALIGN = 4
47        .IMPORT _PowerON_Reset_PC
48
49     _downloader:
50       MOV.L STACK_POINTER,R15      ; setting stack pointer
51       MOV.L MAIN_PROGRAM,R0        ; get entry address of downloader
52       JMP @R0                      ; jump to entry address of downloader
53       NOP
54
55       .ALIGN 4
56     MAIN_PROGRAM:
57       .DATA.L _PowerON_Reset_PC    ; entry address of downloader
58
59     STACK_POINTER:
60       .DATA.L H'FFF84000           ; stack pointer of downloader(R15)
61
62       .pool
63       .end
64
65     ;/* End of File */
```

4.3.5 Downloader Program List "dl_main.c" (1)

```
1      /***************************************************************************
2      *   DISCLAIMER
3      *
4      *   This software is supplied by Renesas Electronics Corporation and is only
5      *   intended for use with Renesas products. No other uses are authorized.
6      *
7      *   This software is owned by Renesas Electronics Corporation and is protected under
8      *   all applicable laws, including copyright laws.
9      *
10     *   THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
11     *   REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
12     *   INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
13     *   PARTICULAR PURPOSE AND NON-INFRINGEMENT.  ALL SUCH WARRANTIES ARE EXPRESSLY
14     *   DISCLAIMED.
15     *
16     *   TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
17     *   ELECTRONICS CORPORATION NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
18     *   FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
19     *   FOR ANY REASON RELATED TO THIS SOFTWARE, EVEN IF RENESAS OR ITS
20     *   AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
21     *
22     *   Renesas reserves the right, without notice, to make changes to this
23     *   software and to discontinue the availability of this software.
24     *   By using this software, you agree to the additional terms and
25     *   conditions found by accessing the following link:
26     *   http://www.renesas.com/disclaimer
27     ***************************************************************************
28     *   Copyright (C) 2011 Renesas Electronics Corporation. All rights reserved.
29     *************************** Technical reference data ***************************
30     *   System Name : SH7268/SH7269 Firm Update Sample Program
31     *   File Name   : dl_main.c
32     *   Abstract    : downloader main
33     *   Version     : 1.00.00
34     *   Device      : SH7268/SH7269
35     *   Tool-Chain  : High-performance Embedded Workshop (Ver.4.07.00).
36     *               : C/C++ compiler package for the SuperH RISC engine family
37     *               :                          (Ver.9.03 Release02).
38     *   OS          : None
39     *   H/W Platform: R0K57269(CPU board)
40     *   Description :
41     ***************************************************************************
42     *   History     : Jul.06,2011 Ver.1.00.00
43     ***************************************************************************/
44     #include <stdio.h>
45     #include <string.h>
46     #include <machine.h>
47     #include "iodefine.h"
48     #include "serial_flash.h"
49
```

RENESAS

4.3.6 Downloader Program List "dl_main.c" (2)

```
50      /* ==== macro defined ==== */
51      #define SECTOR_SIZE        SF_SECTOR_SIZE     /* Sector size */
52      #define SECTOR_NUM         SF_NUM_OF_SECTOR   /* Total number of sectors in the device  */
53      #define DEVICE_SIZE        (SECTOR_SIZE * SECTOR_NUM)/* Device size  */
54
55      #if (SFLASH_DUAL == 0)
56      #define L_PROG_SIZE        8192               /* Loader program size        */
57      #else
58      #define L_PROG_SIZE        4096               /* Loader program size        */
59      #endif
60      #define L_PROG_SRC         0xFFF80000         /* Source address of the loader program */
61      #define L_PROG_DST         0x00000000         /* Destination address of the loader program */
62
63      #define APROG_TOP_SFLASH   0x00002000         /* Start address of the application program */
64
65      #define APROG_TOP_RAM      0x1C002000         /* Start address of the application program */
66                                                    /* When changing the start section of the */
67                                                    /* application program, change this definition */
68
69      #define APPINFO_TOP        APROG_TOP_RAM      /* Address the appinfo.app_top is located */
70      #define APPINFO_END      ( APROG_TOP_RAM + 4 ) /* Address the appinfo.app_end is located */
71
72      /* ==== prototype declaration ==== */
73      /*** User API ****/
74      void main(void);
75
76      static void halt(void);
77      static void error(void);
78      static void init_erase_flag(void);
79      static int Is_erased_sector(unsigned long sector_no);
80      static void set_erase_flag(unsigned long sector_no);
81      static int write_prog_data(unsigned char *program_data, unsigned long sflash_addr,
82                       unsigned long size);
83
84      /*** data ***/
85      static unsigned char sflash_erase_flag[SECTOR_NUM];/* 0: sector not erased,1: sector
        erased */
```

4.3.7 Downloader Program List "dl_main.c" (3)

```
 86
 87     /*****************************************************************************
 88      * ID          :
 89      * Outline     : Downloader main
 90      * Include     :
 91      * Declaration : void main(void);
 92      * Description : Writes the loader program and application program in serial
 93      *             : flash memory as the following procedures.
 94      *             : 1. Mask the interrupt while the downloader is operating.
 95      *             : 2. Initialize the RSPI0.
 96      *             : 3. Disable the software protection in serial flash memory.
 97      *             : 4. Write the loader program in serial flash memory.
 98      *             : 5. Write the application program in serial flash memory.
 99      * Argument    : void
100      * Return Value: void
101      *****************************************************************************/
102     void main(void)
103     {
104       unsigned long app_top_addr,app_end_addr,app_prog_size;
105
106
107       /* Initializes the erase flag */
108       init_erase_flag();
109
110       /* Initializes the SPIBSC */
111       sf_init_serial_flash_spibsc();
112
113       /* Disables the software protection in serial flash memory */
114       sf_protect_ctrl_spibsc(SF_REQ_UNPROTECT);
115
116       /* Writes the loader program */
117       if( write_prog_data( (unsigned char *)L_PROG_SRC, L_PROG_DST, L_PROG_SIZE) < 0 ){
118         error();
119       }
120
121       /* Retrieves the start address and end address from the application program
122         transfer information (appinfo) */
123       app_top_addr = *(volatile unsigned long *)APPINFO_TOP;
124       app_end_addr = *(volatile unsigned long *)APPINFO_END;
125
126       /* Calculates the size of the application program */
127       app_prog_size = app_end_addr - app_top_addr;
```

4.3.8   Downloader Program List "dl_main.c" (4)

```
128
129      /* Writes the application program */
130      if( write_prog_data((unsigned char *)APROG_TOP_RAM,APROG_TOP_SFLASH, app_prog_size)<0){
131        error();
132      }
133
134      /* Enables the software protection in serial flash memory */
135      sf_protect_ctrl_spibsc(SF_REQ_PROTECT);
136
137      /* Exits the downloader */
138      halt();
139    }
```

4.3.9 Downloader Program List "dl_main.c" (5)

```
140
141     /***************************************************************************
142      * ID          :
143      * Outline     : Write the program data
144      * Include     :
145      * Declaration : int write_prog_data(unsigned char *program_data,
146      *             :                     unsigned long sflash_addr, unsigned long size);
147      * Description : Writes the program data as the following procedures.
148      *             : 1. Erase the target sector when it is not erased.
149      *             : 2. Write the program data in serial flash memory.
150      *             : 3. Reads the data in serial flash memory and compare it with the
151      *             :    provided data.
152      * Argument    : unsigned char *program_data ; I : Start address of the program data
153      *             : unsigned long sflash_addr   ; I : Start address at the destination in
154      *             :                                     serial flash memory
155      *             : unsigned long size          ; I : Write size
156      * Return Value: Equal or bigger than 0: Success
157      *             : Less than 0: Error
158      ***************************************************************************/
159     int write_prog_data(unsigned char *program_data, unsigned long sflash_addr, unsigned long
        size)
160     {
161       unsigned long sector_no;
162       unsigned long saddr;
163       unsigned long sz;
164       unsigned char read_data[2];
165       unsigned char *w_p;
166       int wr_size;
167       int rd_size;
168       int bsz;
169
170       bsz = 1;
171
172       /* ==== Copies the value from the argument to the local variable ==== */
173       saddr = sflash_addr;
174       sz = size;
175       w_p = program_data;
176
177       /* ==== Writes data in serial flash memory ==== */
178       while( sz > 0){
179         if(sz > ((256 * bsz) - (saddr % (256 * bsz)))){
180             wr_size = (int)((256 * bsz) - (saddr % (256 * bsz)));
181         }
182         else{
183             wr_size = (int)sz;
184         }
185
186         sector_no = saddr / (SECTOR_SIZE * bsz);
```

RENESAS

4.3.10   Downloader Program List "dl_main.c" (6)

```
187
188        if( Is_erased_sector(sector_no) == 0 ){ /* When it is not erased */
189            sf_sector_erase_spibsc(sector_no);/* Erase */
190            set_erase_flag(sector_no);          /* When it is erased, set the erase flag */
191        }
192
193        sf_byte_program_spibsc(saddr, w_p, wr_size );
194                                              /* Writes data in units of */
195                                              /* single byte */
196      w_p += wr_size;
197      saddr += wr_size;
198      sz -= wr_size;
199      }
200
201      /* ==== Verifies data (serial flash memory is programmed successfully) ==== */
202      saddr = sflash_addr;
203      sz = size;
204      w_p = program_data;
205
206      rd_size = 1;
207      while( sz > 0){
208        sf_byte_read_spibsc(saddr,read_data, rd_size);
209                                        /* Reads the data written in */
210                                        /* serial flash memory */
211
212        if( w_p[0] != read_data[0] ){
213            return -1;                  /* Returns an error when the data */
214                                        /* unmatched */
215        }
216      w_p += rd_size;
217      saddr += rd_size;
218      sz -= rd_size;
219      }
220
221      return 0;
222    }
223
```

4.3.11 Downloader Program List "dl_main.c" (7)

```
224     /**********************************************************************************
225      * ID         :
226      * Outline    : Initialize the Erase Flag
227      * Include    :
228      * Declaration : static void init_erase_flag(void);
229      * Description : Initializes the table sflash_erase_flag[].
230      * Argument   : void
231      * Return Value: void
232      **********************************************************************************/
233     static void init_erase_flag(void)
234     {
235       int i;
236
237       for( i=0; i < SECTOR_NUM ;i++){
238         sflash_erase_flag[i] = 0;
239       }
240     }
241
242     /**********************************************************************************
243      * ID         :
244      * Outline    : Retrieve the Sector Erase Status
245      * Include    :
246      * Declaration : static int Is_erased_sector(unsigned long sector_no);
247      * Description : Returns the information (not erased or eraser) of the
248      *             : sector specified by the sector number.
249      * Argument   : unsigned long sector_no   ; I : Sector number
250      * Return Value: 1 : Sector in the specified address is already erased
251      *             : 0 : Sector in the specified address is not erased
252      **********************************************************************************/
253     static int Is_erased_sector(unsigned long sector_no)
254     {
255       return sflash_erase_flag[sector_no];
256     }
257
258     /**********************************************************************************
259      * ID         :
260      * Outline    : Set the Erase Flag
261      * Include    :
262      * Declaration : static void set_erase_flag(unsigned long sector_no);
263      * Description : Sets the erase flag to modify the information of the specified
264      *             : sector as erased.
265      * Argument   : unsigned long sector_no   ; I : Sector number
266      * Return Value: void
267      **********************************************************************************/
268     static void set_erase_flag(unsigned long sector_no)
269     {
270       sflash_erase_flag[sector_no] = 1;
271     }
272
```

RENESAS

4.3.12 Downloader Program List "dl_main.c" (8)

```
273     /*****************************************************************************
274      * ID          :
275      * Outline     : Program stops (successful).
276      * Include     :
277      * Declaration : static void halt(void);
278      * Description : When the downloader ends successfully, this function is called
279      *             : to stop the program.
280      * Argument    : void
281      * Return Value: void
282      *****************************************************************************/
283     static void halt(void)
284     {
285         while(1){
286             /* When the downloader ends successfully, this function stops the program. */
287         }
288     }
289
290     /*****************************************************************************
291      * ID          :
292      * Outline     : Program stops (error).
293      * Include     :
294      * Declaration : static void error(void);
295      * Description : When the downloader ends in error, this function is called
296      *             : to stop the program.
297      * Argument    : void
298      * Return Value: void
299      *****************************************************************************/
300     static void error(void)
301     {
302         while(1){
303         /* When the downloader ends in error, this function stops the program */
304         }
305     }
306
307     /* End of File */
308
```

5. Using the Downloader

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

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

5.1 Sample Program Configuration

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

Table 11   Sample Program Configuration

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

5.2   Writing Programs in the Serial Flash Memory

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

5.2.1   Registering the Download Module and the Batch File

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

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

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

Figure 15   [sh7269_spibsc_app] Workspace Directory Configuration

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

2. Changing the batch file
   The batch file setting registered in the project is changed in the **Set Batch File** dialog box. The following procedure will open the **Set Batch File** dialog box. On the View menu in the High-performance Embedded Workshop, click the **Command Line** to open the **Command Line** window. Open the **Set Batch File** dialog box from the **Batch File** pop-up menu on the **Command Line** window.
   For the procedure to register the batch file, refer to the High-performance Embedded Workshop User's Manual.
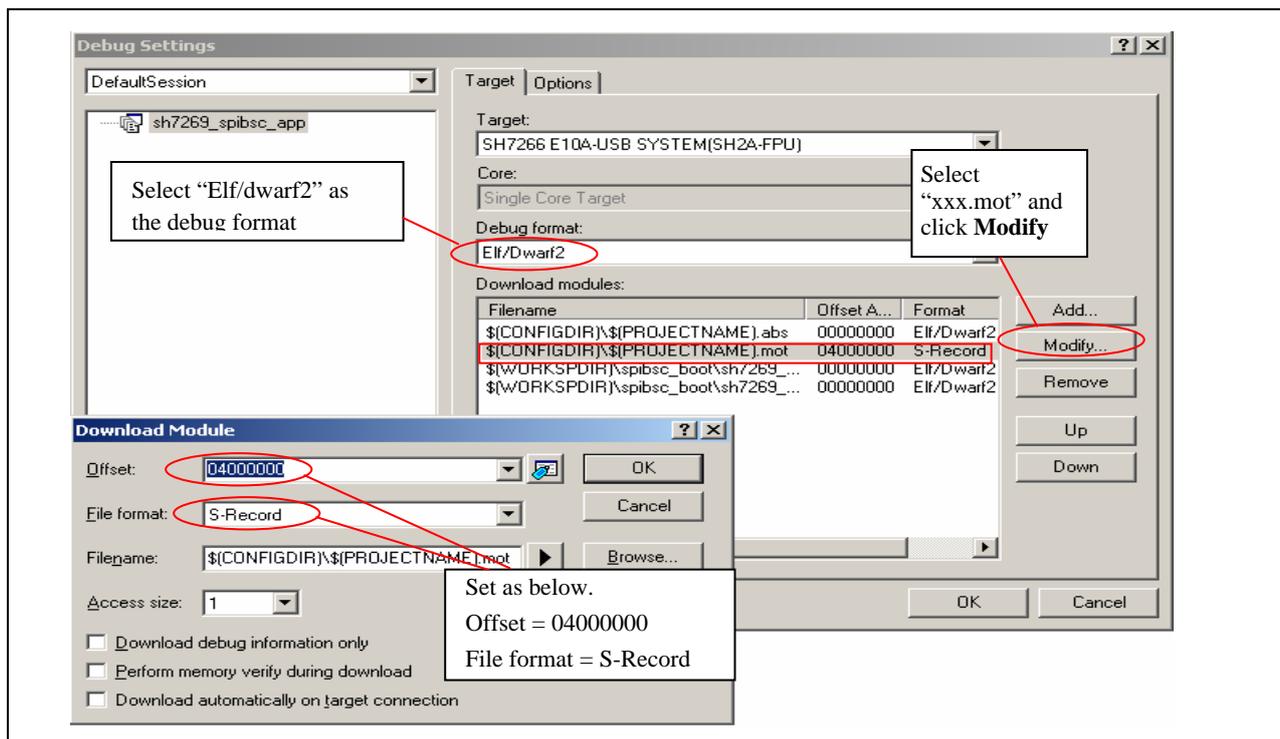
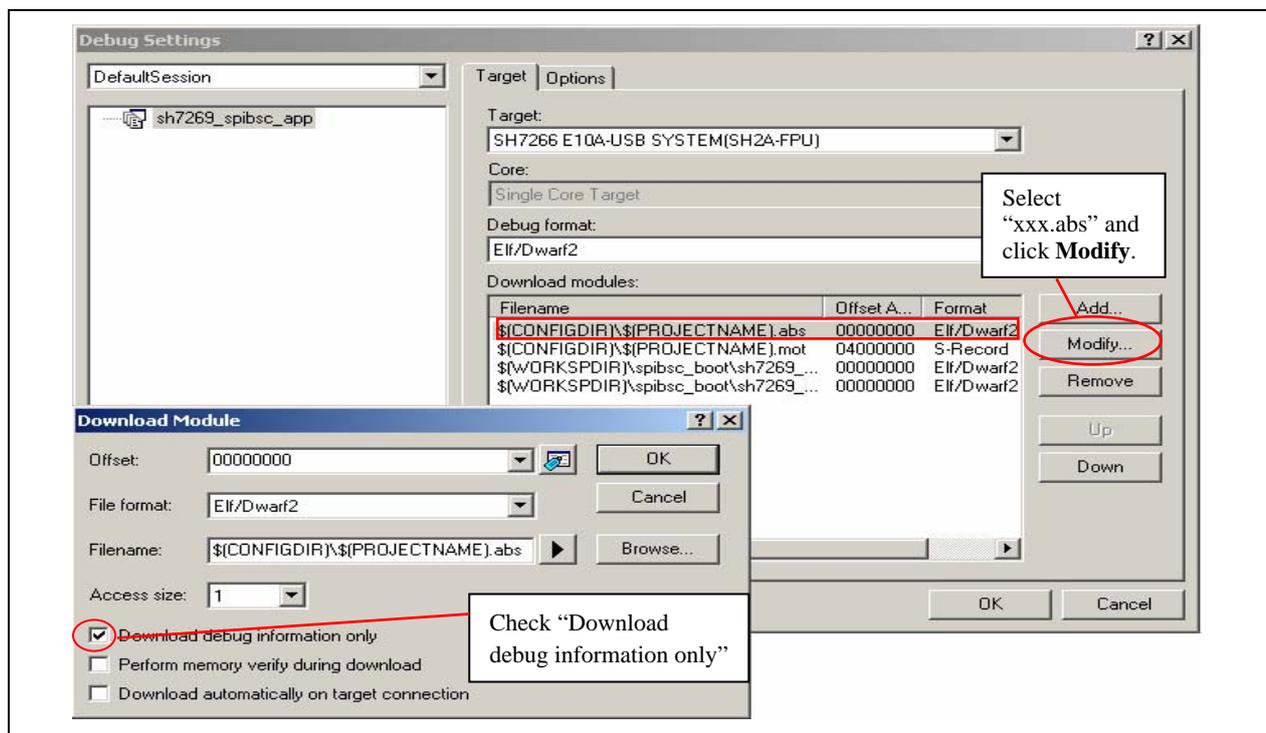**Figure 16   Example for Registering the Download Module as mot file**

Figure 17   Example for Registering the Download Module as abs file

5.2.2 Procedures to Writing Programs

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

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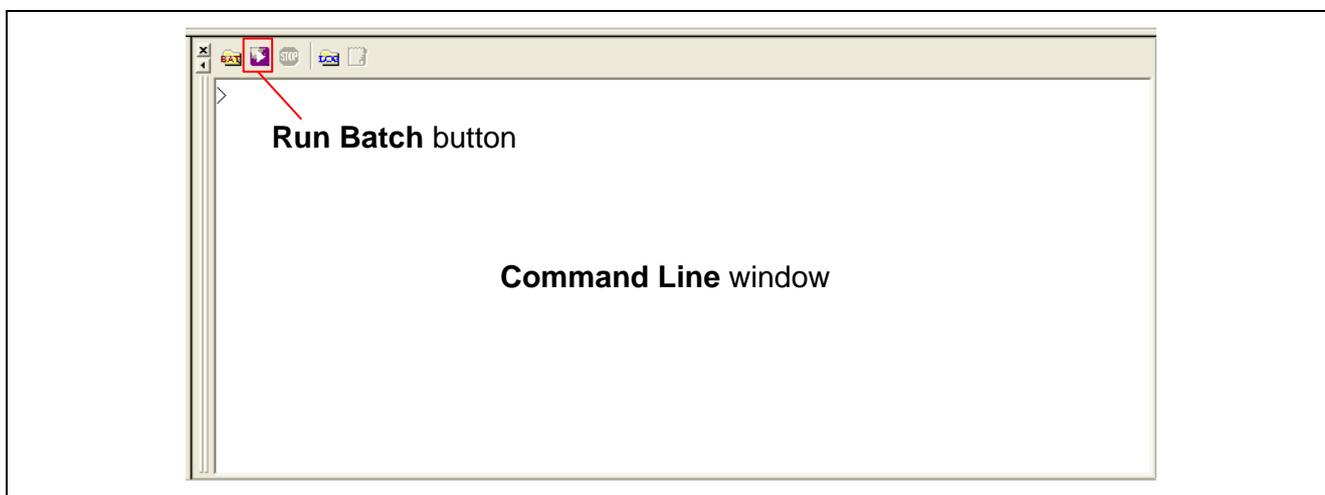
Figure 18   Command Line Window and Run Batch Button

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

8. When writing is completed successfully, the loader program and the application program can be executed after **Reset Go**.
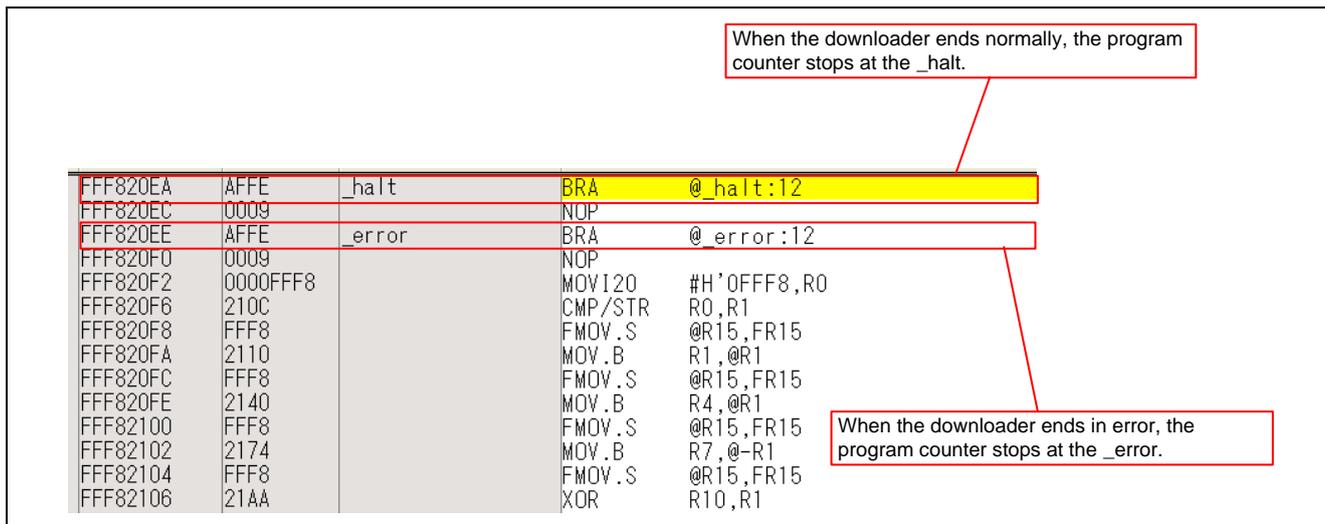
Figure 19   High-performance Embedded Workshop Window When the Downloader Ends

6. References

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

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

Website and Support

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

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

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

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Jul 11.11	—	First edition issued
1.01	Feb 16.12	—	Added sample code of SH726B

General Precautions in the Handling of MPU/MCU Products

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

1. Handling of Unused Pins

   Handle unused pins in accord with the directions given under Handling of Unused Pins in the manual.

   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

   The state of the product is undefined at the moment when power is supplied.

   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.

   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.

   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. 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.

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.

RENESAS

SALES OFFICES

Renesas Electronics Corporation

http://www.renesas.com

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

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

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

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

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

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

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

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

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

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

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

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