SH7734 Group

Example of Booting from Serial Flash Memory

R01AN0667EJ0100
Rev.1.00
Sep 24, 2012

Abstract

This application note presents a sample program for booting from serial flash memory connected to the SH7734 following a power-on reset.

Positioning of This Document

This application note is based on the sample program in *SH7734 Group: SH7734 Example of Initialization* (R01AN0665EJ) and presents a sample program for booting from serial flash memory. A description of the sample program for initial settings is omitted. Please refer to the application note *SH7734 Group: SH7734 Example of Initialization* (R01AN0665EJ).

Products

SH7734 Group

In order to use the sample program described in this application note for a microcontroller other than the above, make changes as appropriate to match the microcontroller to be used and perform careful evaluation.

Contents

1. Specifications 4
1.1 Functions Used 4
1.2 Serial Boot Terminology 4
1.3 Serial Boot Operation Overview 5
1.3.1 On-Chip ROM Program for Boot Startup Operation Overview 5
1.3.2 Loader Program Operation Overview 6
1.3.3 Application Program Operation Overview 6
1.3.4 Downloader Operation Overview 7

2. Operation Confirmation Conditions 8

3. Related Application Note 8

4. Peripheral Functions 9
4.1 Boot Mode 9

5. Hardware 10
5.1 Hardware Configuration Example 10
5.2 List of Pins Used 11

6. Software 12
6.1 Structure of Sample Code 12
6.2 Preface to Description of the Software of the Three Workspaces 12
6.2.1 Description Conventions 12
6.2.2 Terminology 13

7. Loader Program (RSPI) Software Description 14
7.1 Operation Overview 14
7.1.1 Memory Map 14
7.1.2 Allocation of Sections 15
7.1.3 SPI Mode and Bit Rate 15
7.2 File Structure 16
7.3 List of Constants 17
7.4 List of Functions 19
7.5 Function Specifications 20
7.6 Flowchart 23

8. Application Program Software Description 26
8.1 Operation Overview 26
8.1.1 Allocation of Sections 26
8.2 File Structure 28
8.3 Flowchart 29
8.4 Generating Application Program Transfer Information (appinfo) 30

9. Downloader Program Software Description 32
9.1 Detailed Specifications of Downloader Program 32
9.1.1 Operation Overview 32
9.1.2 Allocation of Sections 33
9.2 Structure 34
9.3 List of Constants 35
9.4 List of Functions 37
9.5 Function Specifications 38
9.6 Flowcharts 41
9.7 Serial Flash Memory Commands 44

10. Application Example 45
10.1 Writing (Programming) Programs to Serial Flash Memory 45
10.1.1 Using the Downloader 45
10.1.2 Downloader Automation (Command Batch File) 46
10.1.3 Registration of Download Modules and Batch File 47
10.1.4 Programming Procedure 48
10.2 Notes on Use of PC Break (Breakpoint) Function 50
10.3 Loader Program with Support for Quad-SPI (Reference) 51
10.3.1 Structure of Sample Code 51
10.3.2 Features of Loader Program with Support for Quad-SPI 51
10.3.3 Programming Loader Program with Support for Quad-SPI to Serial Flash Memory 53

11. Sample Code 54

12. Reference Documents 54

1. Specifications

When serial boot is selected as the boot mode, the SH7734 boots from the serial flash memory. This application note presents examples of a loader program and application program for use with serial boot mode. A description is also provided of the downloader used to program the loader program and application program to the serial flash memory.

Note: This application note describes the use of the Renesas serial peripheral interface (RSPI) to access the serial flash memory. However, a separate sample workspace that uses the Renesas quad-serial peripheral interface (Quad-SPI) to shorten the transfer time is provided for the loader program. For details, see the application example in section 10.3.

1.1 Functions Used

Table 1.1 Peripheral Functions Used and Their Applications

Peripheral Function	Application
Renesas serial peripheral interface (RSPI)	Used to access the serial flash memory (serial communication using single-SPI operation).
Renesas quad-serial peripheral interface (Quad-SPI)	Used to access the serial flash memory (serial communication using single-, dual-, or quad-SPI operation). Note: Supported by separate sample code provided for the loader program only.
Direct memory access controller for peripheral modules (HPB-DMAC)	Used for DMA transfers between peripheral modules on the HPB bus (RSPI or Quad-SPI in the case of this application note) and DDR2/DDR3-SDRAM.

1.2 Serial Boot Terminology

Table 1.2 lists terms used in this application note that relate to serial boot mode.

Table 1.2 Serial Boot Terminology

Term	Description
On-chip ROM program for boot startup	When starting up in serial boot mode, this program transfers the loader program stored at the start of the serial flash memory to the IL memory and then branches to the loader program. This program is stored beforehand in the CPU's on-chip ROM for boot startup; it is not created by the customer.
Loader program	This program transfers the application program from the serial flash memory to the DDR2/DDR3-SDRAM and then branches to the entry function of the main application program. The size of the loader program is fixed at 16 KB. It should be custom created for the system.
Application program	A program created by the customer specifically for the system.
Downloader	This program is used to program the loader program and application program to the serial flash memory. It should be custom created for the system.

1.3 Serial Boot Operation Overview

In serial boot mode, the operation sequence is on-chip ROM program for boot startup → loader program → application program. An overview of the operation of these programs is provided below.

1.3.1 On-Chip ROM Program for Boot Startup Operation Overview

In serial boot mode, after the SH7734 starts the on-chip ROM program for boot startup transfers the loader program from the serial flash memory connected to the RSPI to the IL memory. After the transfer finishes, operation branches to the start of the loader program. Figure 1.1 is an operation overview diagram of the on-chip ROM program for boot startup. This sequence of processing is performed automatically by the hardware.

Note: The on-chip ROM program for boot startup does not support the Quad-SPI.

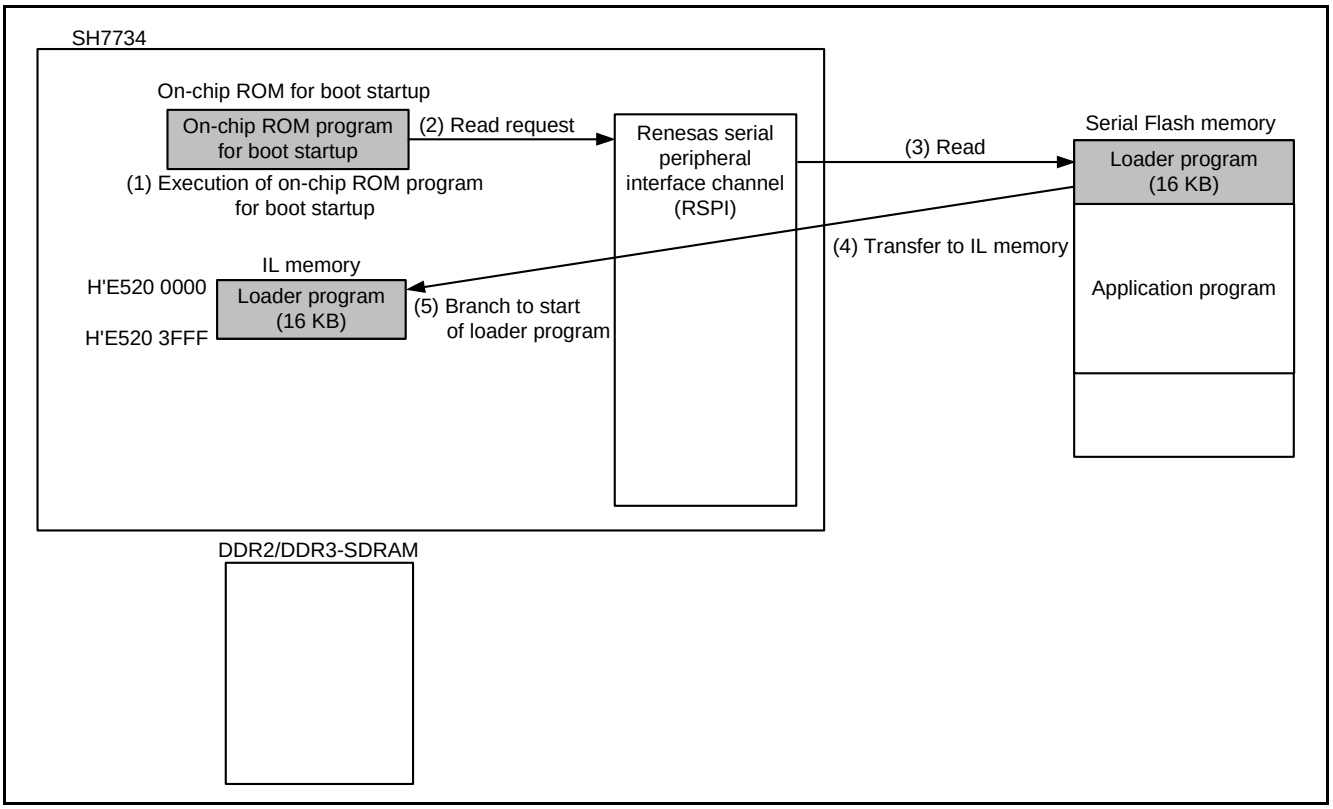

Figure 1.1 On-Chip ROM Program for Boot Startup Operation Overview Diagram

1.3.2 Loader Program Operation Overview

The loader program transfers the application program from the serial flash memory connected to the RSPI to the DDR2/DDR3-SDRAM. After the transfer finishes, operation branches to the entry function of the main application program. Figure 1.2 is an operation overview diagram of the loader program.

For details, see section 7, Loader Program (RSPI) Software Description.

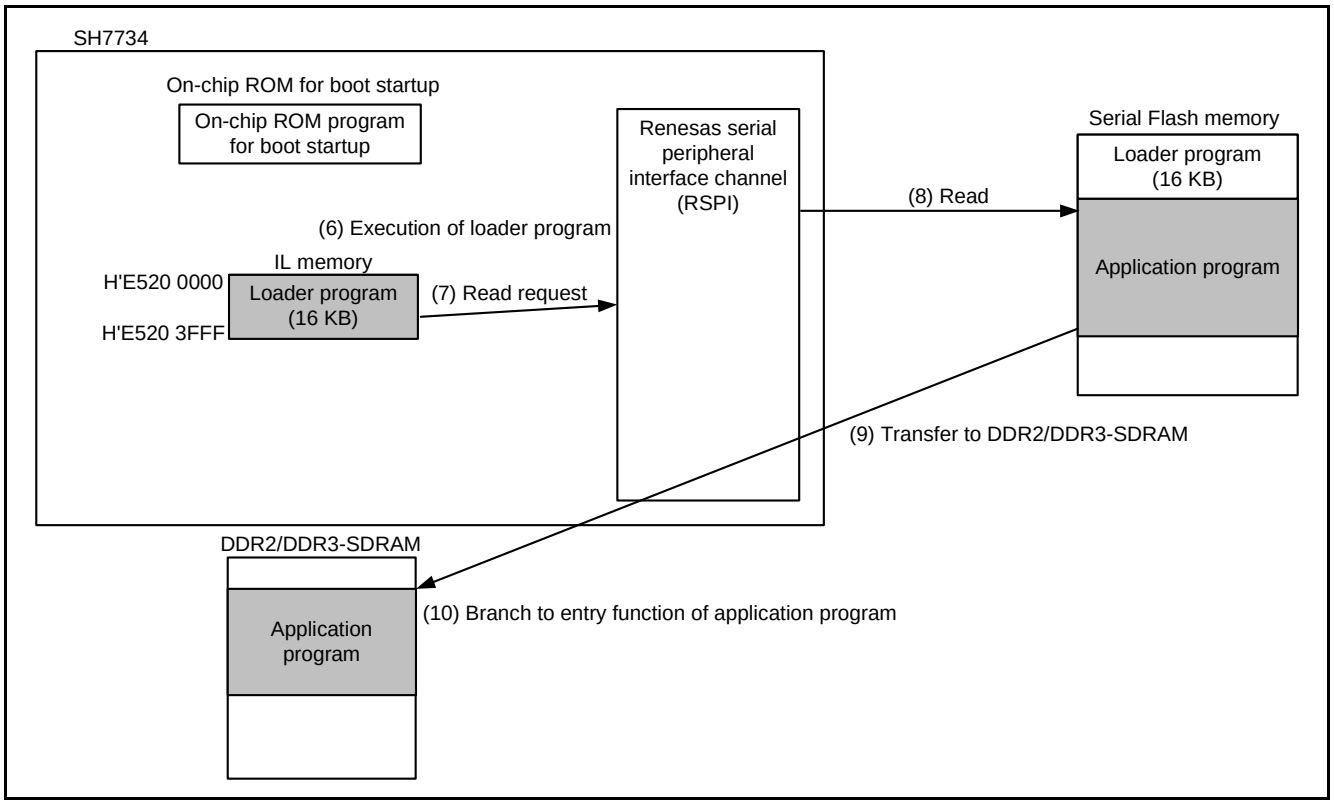

Figure 1.2 Loader Program Operation Overview Diagram

Note: Separate sample code that uses the Quad-SPI to shorten the transfer time is provided. For details, see the application example in section 10.3.

1.3.3 Application Program Operation Overview

This program is created by the customer specifically for the system. The sample application program presented in this application note uses sample code from *SH7734 Group: SH7734 Example of Initialization* (R01AN0665EJ) to output debugging information to a terminal program running on a PC.

For details, see section 8, Application Program Software Description.

1.3.4 Downloader Operation Overview

The downloader is a program that programs to the serial flash memory the loader program located in IL memory and the application program located in DDR2/DDR3-SDRAM. Before running the downloader, it is necessary to use a debugger to transfer the following from the development environment: the downloader to the OL memory, the loader program to the IL memory, and the application program to the DDR2-SDRAM. Figure 1.3 shows an overview of the operation of the downloader.

For details, see section 9, Downloader Program Software Description.

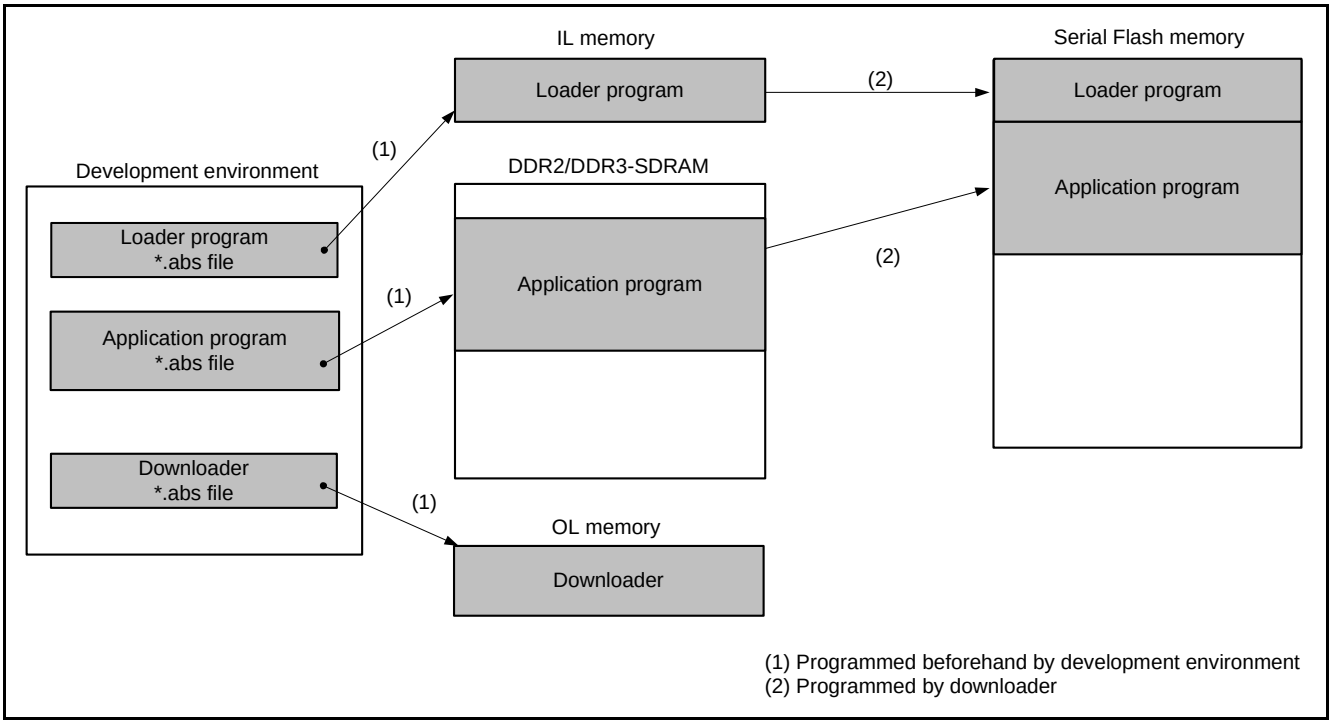

Figure 1.3 Downloader Operation Overview Diagram

2. Operation Confirmation Conditions

The sample code accompanying this application note has been run and confirmed under the conditions below.

Table 2.1 Operation Confirmation Conditions

Item	Contents
Microcontroller used	SH7734 (R8A77343)
Operating frequency	EXTAL input frequency: 33.3333 MHz CPU clock (clki): 400 MHz SHwy clock (clks): 200 MHz SHwy clock (clks1): 100 MHz DDR clock (MCK0/MCK0#/MCK1/MCK1#): 200 MHz Bus clock (clkb): 50 MHz Peripheral clock (clkp): 50 MHz
Operating voltage	IO supply power (3.3 V) Core supply power (1.25 V)
Integrated development environment	Renesas Electronics High-performance Embedded Workshop (Version 4.09.00.007)
C complier	Renesas Electronics C/C++ Compiler Package for SuperH Family (9.4.0.0)
	Compile option -cpu=sh4a -endian=little -include="$(PROJDIR)\inc" -object="$(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
Operating mode	Serial boot (MD19 = 0, MD14 = 0, MD18 = 0, MD17 = 1, MD16 = 1)
Sample code version	Ver. 1.00
Board used	Renesas Electronics SH7734 evaluation platform (R0P7734C00000RZ)
Address expansion mode	29-bit
Memory management unit (MMU)	Disabled
Device used	128 Mbit On-board QSPI flash ROM N25Q128A13TSF40F (Numonyx)

Note: Serial flash memory commands in serial boot mode
In serial boot mode, the serial flash memory High-Speed Read command (H'0B) is used to read program data from the serial flash memory. Make sure to use serial flash memory that supports use of the H'0B command in serial boot mode.

3. Related Application Note

The following application note is related to this application note. Refer to the two application notes in conjunction.

SH7734 Group: SH7734 Example of Initialization (R01AN0665EJ)

4. Peripheral Functions

A supplementary description of boot mode is provided below. For details of the peripheral functions referenced in this application note (RSPI and Quad-SPI), see *SH7734 User's Manual: Hardware* (R01UH0233EJ).

4.1 Boot Mode

When PRESET# is low level, external pins can be used to designate the boot mode. For the external pin setting used to designate the boot mode, see *SH7734 User's Manual: Hardware* (R01UH0233EJ).

When the mode setting is serial boot, after PRESET# goes high level the SH7734 copies the first 16 KB of data from the serial flash memory to the IL memory, then runs the program from the start address in IL memory.

5. Hardware

5.1 Hardware Configuration Example

Figure 5.1 shows a connection example in which the evaluation board used with the sample program is mounted with QSPI flash ROM. The RSPI and Quad-SPI employ multiplexed pins, and operation of the Quad-SPI is enabled by switching the pin settings. The items in figure 5.1 printed in bold italic text apply when the RSPI is selected. For details of the wiring connections to other peripheral circuits, etc., see the technical documentation of the SH7734 evaluation platform (R0P7734C00000RZ).

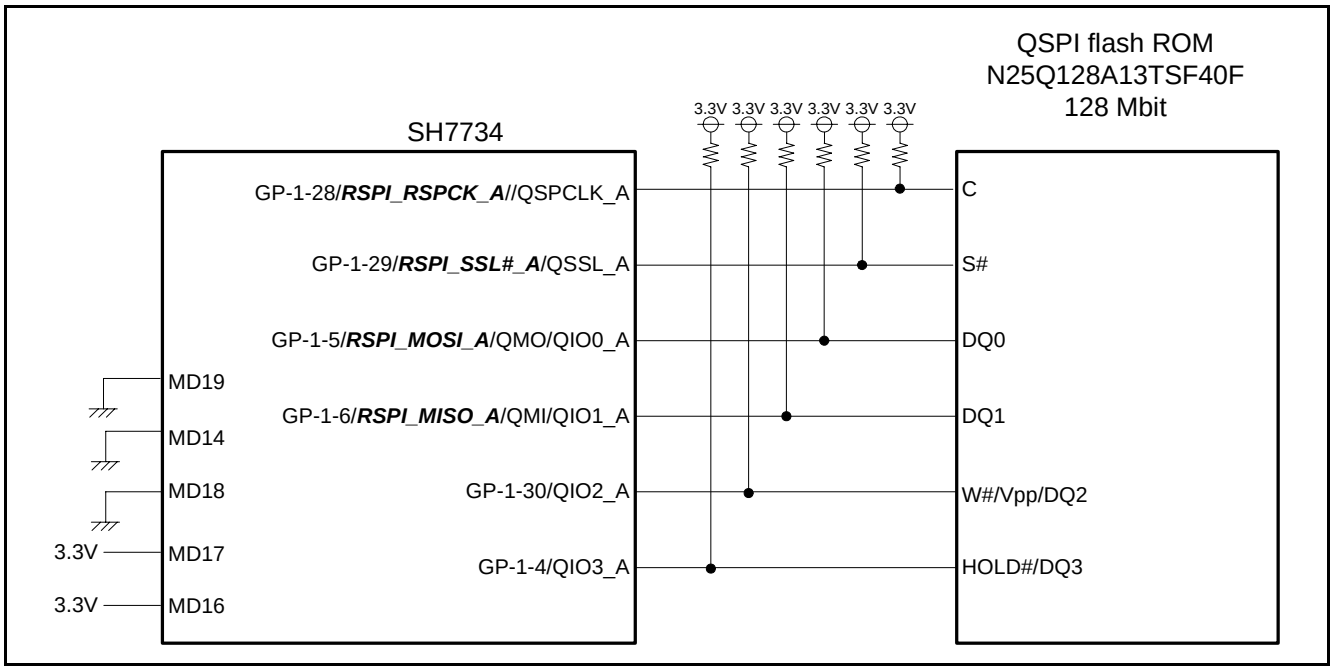

Figure 5.1 Connection Example

Note: The QSPI flash ROM (N25Q128A13TSF40F) mounted on the SH7734 evaluation platform (R0P7734C00000RZ) used with the sample program supports single-, dual-, and quad-SPI operation. DQ0 and DQ1 are used by either the RSPI or the Quad-SPI, depending on the data application, but DQ2 and DQ3 are used by the Quad-SPI only.

5.2 List of Pins Used

Table 5.1 lists the pins used by the RSPI and their functions, and table 5.2 lists the pins used by the Quad-SPI and their functions.

Table 5.1 Pins Used and Their Functions (RSPI)

Pin Name	I/O	Description
RSPI_SSL#_A	Output	Slave select*[1]
RSPI_RSPCK_A	Output	Clock output*[1]
RSPI_MOSI_A	Output	Master outgoing data*[1]
RSPI_MISO_A	Input	Master input data*[1]

Note: 1. This application note covers master operation only.

Table 5.2 Pins Used and Their Functions (Quad-SPI)

Pin Name	I/O	Description
QSSL#_A	Output	Slave select*[2]
QSPCLK_A	Output	Clock output*[2]
QMO/QIO0_A*[3]	I/O	Master outgoing data/data 0*[2]*[4]
QMI/QIO1_A*[3]	Input	Master input data/data 1*[2]*[4]
QIO2_A	Input	Data 2*[2]*[4]
QIO3_A	Input	Data 3*[2]*[4]

Notes: 2. Quad-SPI supports master operation only.
3. QMO/QMI in single-SPI mode, QIO0/QIO1 in dual- or quad-SPI mode.
4. This application note uses QMO for CMD, address, and dummy output; uses QIO0_A, QIO1_A, QIO2_A, and QIO3_A for data input; and covers read operation only.

6. Software

6.1 Structure of Sample Code

Separate workspaces are provided for the sample code of the loader program, application program, and downloader described in this application note. The code is divided among the three workspaces as indicated in table 6.1.

See the sections referenced in table 6.1 for descriptions of the software of each workspace.

Table 6.1 Structure of Sample Code

Workspace Name	Description	Applicable Section
sh7734_sflash_loader_prog (Loader program (RSPI))	This workspace project is used to build the loader program.	Section 7
sh7734_sflash_app (Application program)	This workspace project is used to build the application program. Also registered as part of this workspace project are the downloader built using the [sh7734_sflash_downloader] workspace, a batch file for launching the downloader, and the loader program built using the [sh7734_sflash_loader_prog] workspace. These are used to program the loader program and application program to the serial flash memory.	Section 8
sh7734_sflash_downloader (Downloader)	This workspace project is used to build the downloader.	Section 9

6.2 Preface to Description of the Software of the Three Workspaces

6.2.1 Description Conventions

(1) Items Omitted from Description

In the coverage of the topics listed below in sections 7, 8, and 9, descriptions are omitted of items that are generated automatically by the integrated development environment as well as of items covered in *SH7734 Group: SH7734 Example of Initialization* (R01AN0665EJ).

- File Structure
- List of Constants
- List of Structures/Unions
- List of Variables
- List of Functions
- Function Specifications

(2) References to DDR2/DDR3-SDRAM

The evaluation board used with the sample program is populated with DDR2-SDRAM, so the descriptions that follow refer to DDR2-SDRAM.

6.2.2 Terminology

(1) Terms Related to Application Program

As a preface to the description of the application program, related terms used in this application note are defined in table 6.2.

Table 6.2 Terms Related to Application Program

Term	Description
Application program transfer information (appinfo)	The loader program references the application program transfer information (appinfo) stored in serial flash memory and transfers the main application program to the DDR2-SDRAM. Table 7.6 lists the details of the application program transfer information (appinfo).
Main application program entry function address information	After transferring the application program, the loader program uses this information to branch to the address of the entry function of the main application program. It is stored in the serial flash memory.
Main application program	This is the main application program. In addition to the program area, it includes the data area sections (C, D, etc.) required by the program to operate.
Application program	This term refers collectively to the application program transfer information (appinfo), the entry function address information of the main application program, and the main application program.

7. Loader Program (RSPI) Software Description

7.1 Operation Overview

The loader program transfers the application program from the serial flash memory to the DDR2-SDRAM, then branches to the entry function of the main application program.

7.1.1 Memory Map

Figure 7.1 is a memory map of the loader program. See table 7.1 for more information on the sections that appear below.

1. The loader program uses the address range H'E520 0000 to H'E520 3FFF.
2. The RSTHandler section containing the start program of the loader program should be located at address H'E520 0000.
3. The other sections (P, C, B, etc.) required by the loader program and temporary sections required by exception handlers (PINTHandler, PIntPRG, VECTTBL, INTTBL, etc.) should be located at addresses that come after the RSTHandler section.
4. The sample program uses the address range H'E520 3C00 to H'E520 3FFF as the stack area of the loader program.

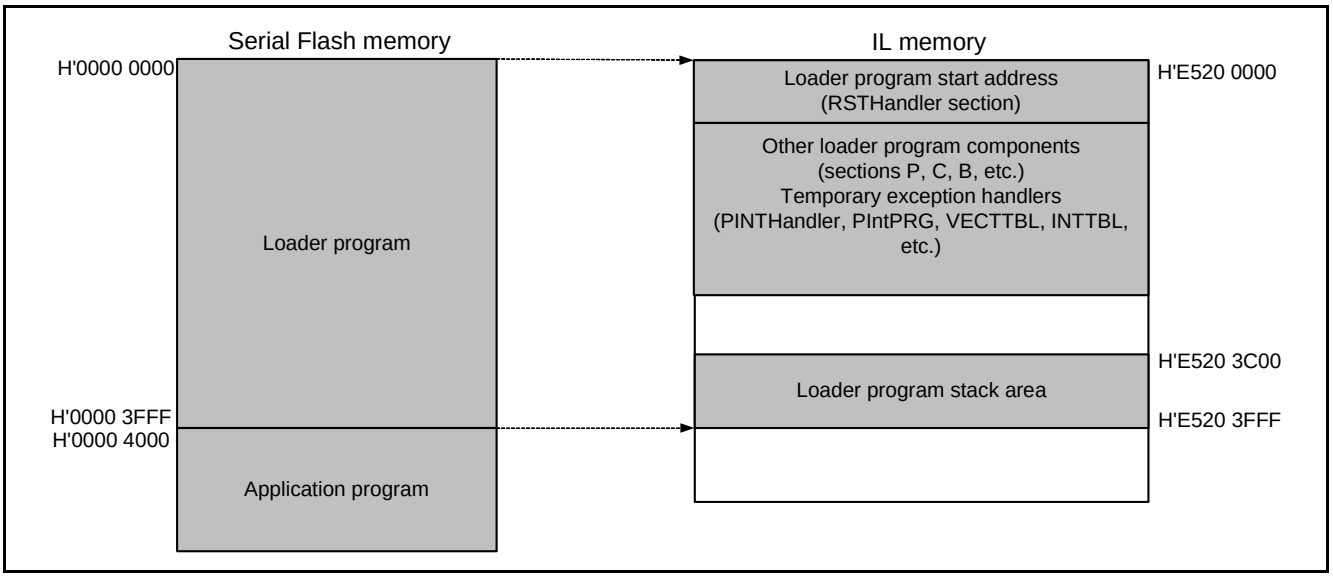

Figure 7.1 Loader Program (RSPI) Memory Map Example

7.1.2 Allocation of Sections

Table 7.1 shows the allocation of the sections of the loader program described in this application note. For details of the roles of the sections that appear below, see *SH7734 Group: SH7734 Example of Initialization* (R01AN0665EJ).

Table 7.1 Allocation of Loader Program Sections

Allocation Address	Section Name	Section Application
H'E520 0000 (IL memory)	RSTHandler	Reset handler (start program of loader program)
	PResetPRG	Reset program area
	P_DBSC3	DBSC3 initialization program area
	PnonCACHE	Cache manipulation program area
	P$PSEC	Section initialization program area
	PINTHandler	Exception/interrupt handler
	PIntPRG	Interrupt function
	P	Program area
	C	Constant area
	C$BSEC	Uninitialized data area address structure
	C$DSEC	Initialized data area address structure
	D	Initialized data (initial value)
	VECTTBL	Reset vector table
	INTTBL	Interrupt vector table
	B	Uninitialized data area
	R	Initialized data area
H'E520 3C00 (IL memory)	S	Stack area

7.1.3 SPI Mode and Bit Rate

Table 7.2 lists the SPI mode and bit rate specifications of the loader program (RSPI) described in this application note.

Table 7.2 SPI Mode and Bit Rate Specifications of Loader Program (RSPI)

Item	Description
SPI mode	SPI mode 3 (CPOL = 1, CPHA = 1)
Bit rate	12.5 MHz

7.2 File Structure

Table 7.3 lists the files composing the loader program (RSPI).

Table 7.3 Loader Program (RSPI) File Structure

File Name	Description	Remarks
sh7734_main_lp.c	Main processing routine of the loader program (RSPI)	
dbsct_lp.c	The dbsct.c source code file from *SH7734 Group: SH7734 Example of Initialization* (R01AN0665EJ) with changes to the section definitions used by the loader program	
r_rspi_lp.c	RSPI — serial flash memory control modules	
r_wdt_lp.c	Watchdog timer (WDT) control modules	The sample program activates a WDT to enable recovery should the system enter an unanticipated state while booting.
r_rspi_lp.h	Include headers for externally referencing RSPI — serial flash memory control modules	
r_wdt_lp.h	Include headers for externally referencing Watchdog timer (WDT) control modules	
vhandler_lp.src	The vhandler.src source code file from *SH7734 Group: SH7734 Example of Initialization* (R01AN0665EJ) with the following changes: Changes to processing of reset (RSTHandler) and TLB miss exception (VBR + H'400) handlers; additional code to branch to application program	The sample program does not include handlers for general exceptions (VBR + H'100) or TLB miss exceptions (VBR + H'400). The user should make any necessary changes to match the system characteristics.

Note: To distinguish the files of loader program from those of the application program, downloader program, etc., the loader program files have the format “XXX_lp.c”.

7.3 List of Constants

Table 7.4 lists the constants used in the loader program (RSPI) sample code.

Table 7.4 Constants Used in Loader Program (RSPI) Sample Code

Constant Name	Setting Value	Description
SF_PAGE_SIZE	256	Defined in r_rspi_lp.h Size of 1 page in serial flash memory (256 B)*[1]
SF_SECTOR_SIZE	H'10000	Defined in r_rspi_lp.h Size of 1 sector in serial flash memory (64 KB)*[1]
SFLASHCMD_CHIP_ERASE	H'C7	Defined in r_rspi_lp.c For Bulk Erase command*[1]
SFLASHCMD_SECTOR_ ERASE	H'D8	Defined in r_rspi_lp.c For Sector Erase command*[1]
SFLASHCMD_BYTE_ PROGRAM	H'02	Defined in r_rspi_lp.c For Page Program command*[1]
SFLASHCMD_BYTE_READ	H'0B	Defined in r_rspi_lp.c For Read Data Bytes at Higher Speed command*[1]
SFLASHCMD_BYTE_READ_ LOW	H'03	Defined in r_rspi_lp.c For Read Data Bytes command*[1]
SFLASHCMD_WRITE_ ENABLE	H'06	Defined in r_rspi_lp.c For Write Enable command*[1]
SFLASHCMD_WRITE_ DISABLE	H'04	Defined in r_rspi_lp.c For Write Disable command*[1]
SFLASHCMD_READ_ STATUS	H'05	Defined in r_rspi_lp.c For Read Status Register command*[1]
SFLASHCMD_WRITE_ STATUS	H'01	Defined in r_rspi_lp.c For Write Status Register command*[1]
D_SFLASH_MOD_SEL_SET	H'01000000	Defined in r_rspi_lp.c For setting MOD_SEL register For selecting group A RSPI
D_SFLASH_IPSR2_SET_ INIT	H'007E07E0	Defined in r_rspi_lp.c For setting IPSR2 register For initializing pins RSPI_RSPCK_A, RSPI_SSL#_A, RSPI_MOSI_A, and RSPI_MISO_A
D_SFLASH_IPSR2_SET_ SFLASH	H'00120120	Defined in r_rspi_lp.c For setting IPSR2 register For selecting pins RSPI_RSPCK_A, RSPI_SSL#_A, RSPI_MOSI_A, and RSPI_MISO_A
D_SFLASH_GPSR1_SET_ SFLASH	H'30000060	Defined in r_rspi_lp.c For setting GPSR1 register For switching between pins RSPI_RSPCK_A, RSPI_SSL#_A, RSPI_MOSI_A, and RSPI_MISO_A and GPIO pin
D_SFLASH_MSTPCR1_SET	H'00000800	Defined in r_rspi_lp.c For setting MSTPCR1 register For supplying HPB-DMAC clock
D_SFLASH_DMAC_DPTR27 _INIT	H'00000200	Defined in r_rspi_lp.c For setting DPTR27 register For making RSPI and HPB-DMAC interoperation settings

Constant Name	Setting Value	Description
D_SFLASH_DMAC_DCR27_INIT_LITTLE	H'06A02100	Defined in r_rspi_lp.c For setting DCR27 register Note: Little-endian support
D_SFLASH_DMAC_DCMDR27_START	H'00000001	Defined in r_rspi_lp.c For setting DCMDR27 register For HPB-DMAC activation start
D_SFLASH_DMAC_HSRSTR_SET_RESET	H'00000001	Defined in r_rspi_lp.c For setting HSRSTR register For resetting the HPB-DMAC module
SF_USE_DMAC	—	Defined in r_rspi_lp.c Used to distinguish between CPU transfers and HPB-DMAC transfers using #ifdef; defined when using HPB-DMAC transfers Note: This definition is enabled because the sample program uses interoperation between the RSPI and HPB-DMAC.
D_WDT_WDTBST_INIT	H'55000000	Defined in r_wdt_lp.c WDTBST register initial value
D_WDT_WDTBST	H'5500C350	Defined in r_wdt_lp.c WDTBST register setting value Set to 1 ms (clkp = 50 MHz)
D_WDT_WDTST_INIT	H'5A000000	Defined in r_wdt_lp.c WDTST register initial value
D_WDT_WDTST	H'5A0003E8	Defined in r_wdt_lp.c WDTST register setting value Watchdog timer setting value = WDTBST setting value (1 ms) × 1000 = 1 sec.
D_WDT_WDTCSR_START	H'A50000C0	Defined in r_wdt_lp.c WDTCSR setting value Watchdog timer start
D_WDT_WDTCSR_STOP	H'A5000000	Defined in r_wdt_lp.c WDTCSR setting value Watchdog timer stop
APROG_TOP_SFLASH	H'00004000	Defined in sh7734_main_lp.c Start address in serial flash memory where the application program is stored
APPINFO_TOP	APROG_TOP_SFLASH	Defined in sh7734_main_lp.c Address in serial flash memory where the application program start address is stored
APPINFO_END	APROG_TOP_SFLASH + 4	Defined in sh7734_main_lp.c Address in serial flash memory where the application program end address is stored

Note: 1. For details, see the datasheet of the serial flash memory used.

7.4 List of Functions

Table 7.5 lists the functions used in the loader program (RSPI) sample code.

Table 7.5 Functions Used in Loader Program (RSPI) Sample Code

Function Name	Description
main	Loader program (RSPI) main processing
get_appinfo	Application program transfer information (appinfo) acquisition processing
app_prog_transfer	Application program transfer processing
jmp_app_prog	Processing of branch to application program entry function
R_WDT_Start	Watchdog timer activation
R_WDT_RegisterInit1	Watchdog timer initialization sequence 1
R_WDT_RegisterInit2	Watchdog timer initialization sequence 2
R_RSPI_SFInitSerialFlash	RSPI initialization
R_RSPI_SFByteRead	RSPI serial flash memory read processing (1-byte units)
R_RSPI_SFByteReadLong	RSPI serial flash memory read processing (4-byte units)

7.5 Function Specifications

The specifications of the sample code functions are described below.

main	
Outline	Loader program (RSPI) main processing
Header	r_rspi_lp.h
Declaration	void main(void)
Description	• Transfers the application program from the serial flash memory to the DDR2-SDRAM, then branches to the entry function of the main application program. • The sample program illuminates LED4 during transfer of the application program for debugging purposes. • The sample program activates a watchdog timer in case the system enters an unanticipated error state while transfer of the application program is in progress.
Arguments	None
Return Value	None

get_appinfo	
Outline	Application program transfer information (appinfo) acquisition processing
Header	r_rspi_lp.h
Declaration	static void get_appinfo(uint32_t *app_top_addr, uint32_t *app_end_addr)
Description	Gets the transfer destination start address and transfer destination end address of the application program transfer information (appinfo) from the serial flash memory. See table 7.6 for details of the application program transfer information (appinfo).
Arguments	uint32_t *app_top_addr : Pointer to transfer destination start address storage location uint32_t *app_end_addr : Pointer to transfer destination end address storage location
Return Value	None
Notes	The application program described in this application note is transferred to the DDR2-SDRAM, so the transfer destination start address and transfer destination end address mentioned above are addresses in the DDR2-SDRAM.

app_prog_transfer	
Outline	Application program transfer processing
Header	r_rspi_lp.h
Declaration	static void app_prog_transfer(const uint32_t app_top_addr, const uint32_t app_end_addr)
Description	Transfers the application program.
Arguments	uint32_t app_top_addr : Transfer destination start address uint32_t app_end_addr : Transfer destination end address
Return Value	None
Notes	The application program described in this application note is transferred to the DDR2-SDRAM, so the transfer destination start address and transfer destination end address specified by arguments below are addresses in the DDR2-SDRAM.

jmp_app_prog	
Outline	Processing of branch to application program entry function
Header	r_rspi_lp.h
Declaration	void jmp_app_prog (uint32_t app_top_addr)
Description	Calculates the address of the entry function of the main application program from the transfer destination start address in the application program function information, then branches to the entry function.
Arguments	uint32_t app_top_addr : Transfer destination start address
Return Value	None

R_WDT_Start	
Outline	Watchdog timer activation
Header	r_wdt_lp.h
Declaration	void R_WDT_Start(void)
Description	• Starts the watchdog timer with a 1-second period. • The watchdog timer counter is reset by a TMU timer with a 10 ms period.
Arguments	None
Return Value	None

R_WDT_RegisterInit1	
Outline	Watchdog timer initialization sequence 1
Header	r_wdt_lp.h
Declaration	void R_WDT_RegisterInit1(void)
Description	Stops the watchdog timer.
Arguments	None
Return Value	None

R_WDT_RegisterInit2	
Outline	Watchdog timer initialization sequence 2
Header	r_wdt_lp.h
Declaration	void R_WDT_RegisterInit2(void)
Description	Initializes the watchdog timer counter.
Arguments	None
Return Value	None

R_RSPI_SFInitSerialFlash	
Outline	RSPI initialization
Header	r_rspi_lp.h
Declaration	void R_RSPI_SFInitSerialFlash(void)
Description	Supplies a clock to the RSPI, makes settings for the pins to be used, and initializes the RSPI.
Arguments	None
Return Value	None

R_RSPI_SFByteRead		
Outline	RSPI serial flash memory read processing (1-byte units)	
Header	r_rspi_lp.h	
Declaration	void R_RSPI_SFByteRead(const uint32_t addr, uint8_t *buf, const uint32_t size)	
Description	• Reads, in 1-byte units, data of the specified size from the serial flash memory address specified by the parameter. • Specifies 8 bits as the transfer data length (bits SPB0 to SPB3) in the SPCMD register.	
Arguments	uint32_t addr	: Serial flash memory address of read target
	uint8_t *buf	: Pointer for read data storage
	uint32_t size	: Read size (1-byte)
Return Value	None	

R_RSPI_SFByteReadLong		
Outline	RSPI serial flash memory read processing (4-byte units)	
Header	r_rspi_lp.h	
Declaration	void R_RSPI_SFByteReadLong(const uint32_t addr, uint32_t *buf, const uint32_t size)	
Description	• Reads, in 4-byte units, data of the specified size from the serial flash memory address specified by the parameter. • Specifies 32 bits as the transfer data length (bits SPB0 to SPB3) in the SPCMD register. • The size parameter specifies a size in 1-byte units, and the value should be a multiple of 4. Example: Specify 16 to read 16 bytes of data.	
Arguments	uint32_t addr	: Serial flash memory address of read target
	uint8_t *buf	: Pointer for read data storage
	uint32_t size	: Read size (1-byte)
Return Value	None	

7.6 Flowchart

Figure 7.2 is a flowchart of the processing performed by the loader program (RSPI).

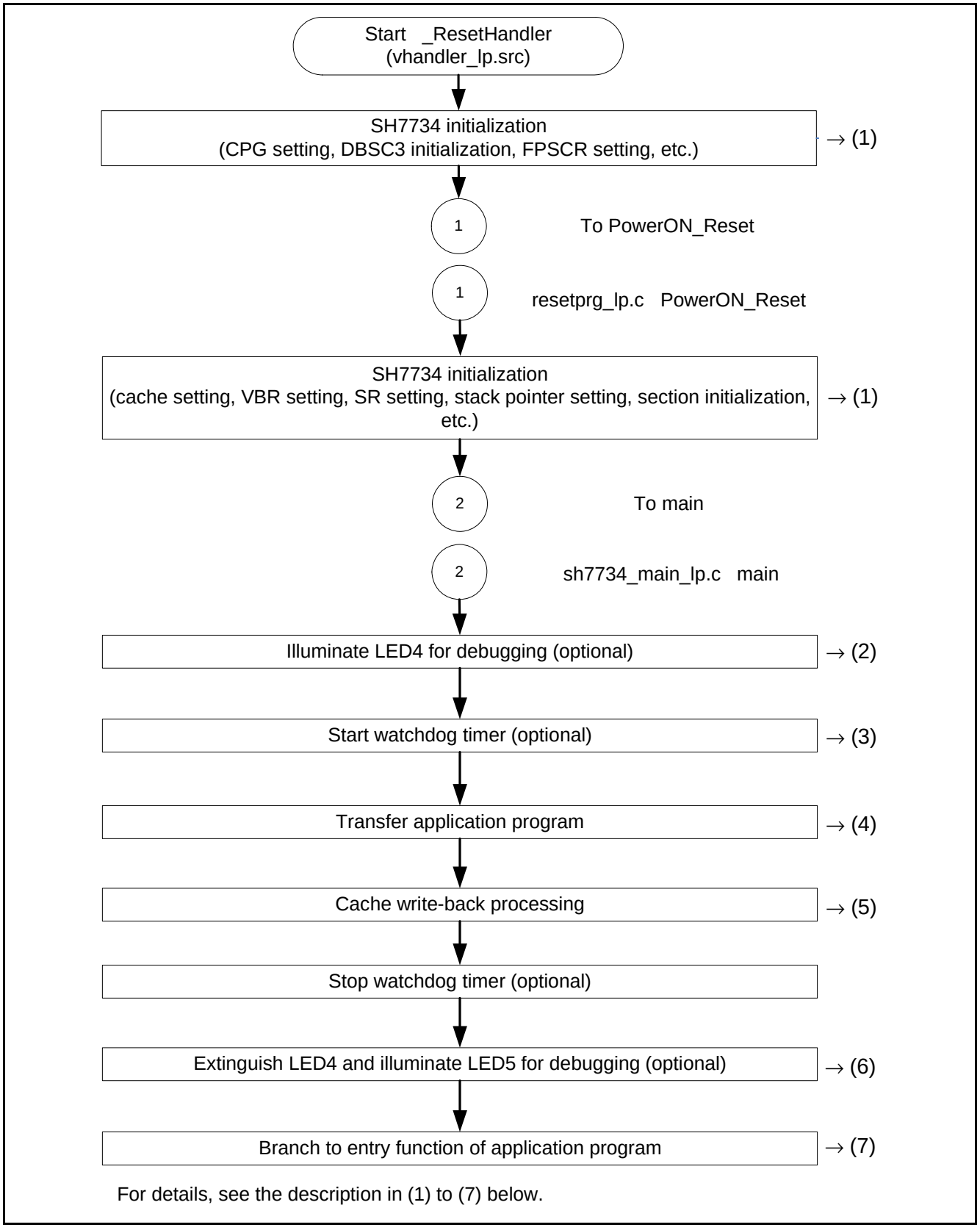

Figure 7.2 Flowchart of Processing Performed by Loader Program (RSPI)

(1) SH7734 Initialization Processing

The minimum processing required to operate the SH7734 is performed (CPG setting, DBSC3 initialization, cache setting, FPSCR setting, VBR setting, SR setting, stack pointer setting, section initialization, etc.).

The sample program reuses code from *SH7734 Group: SH7734 Example of Initialization* (R01AN0665EJ) for SH7734 initialization processing. Refer to that document for details.

(2) Illuminate LED4 (Optional)

The sample program illuminates LED4 during loading of the application program from serial flash memory for debugging purposes.

(3) Start Watchdog Timer (Optional)

The sample program activates a watchdog timer (1-second period) in case the system enters an unanticipated error state while transfer of the application program is in progress. At the same time, it activates a TMU timer with a 10 ms period to reset the watchdog timer counter. An interrupt is generated when TMU timeout occurs.

(4) Transfer Application Program

The loader program references the application program transfer information (appinfo) stored in the serial flash memory and then transfers the application program to the DDR2-SDRAM. As shown in table 7.6, the application program transfer information (appinfo) is allocated to the address range H'0000 4000 to H'0000 4007 in the serial flash memory. The address information of the entry function of the main application program is acquired from the address range H'0000 4008 to H'0000 400B in the serial flash memory, as shown in table 7.7. The entry function address information should be allocated to this area.

Table 7.6 Application Program Transfer Information (appinfo)

Item	Section	Serial Flash Memory Address	Description	Size
Transfer destination start address	DAPPINFO	H'0000 4000	H'0C000000	4
Transfer destination end address		H'0000 4004	H'0C000000 + application program size	4

Table 7.7 Main Application Program Entry Function Address Information

Item	Section	Serial Flash Memory Address	Description	Size
Main application program entry function address information	VECTTBL	H'0000 4008	Address of application program entry function (PowerON_Reset())	4

Figure 7.3 is a conceptual diagram of the transfer using the application program transfer information (appinfo). For details of the method for generating the application program transfer information (appinfo), see section 8.4.

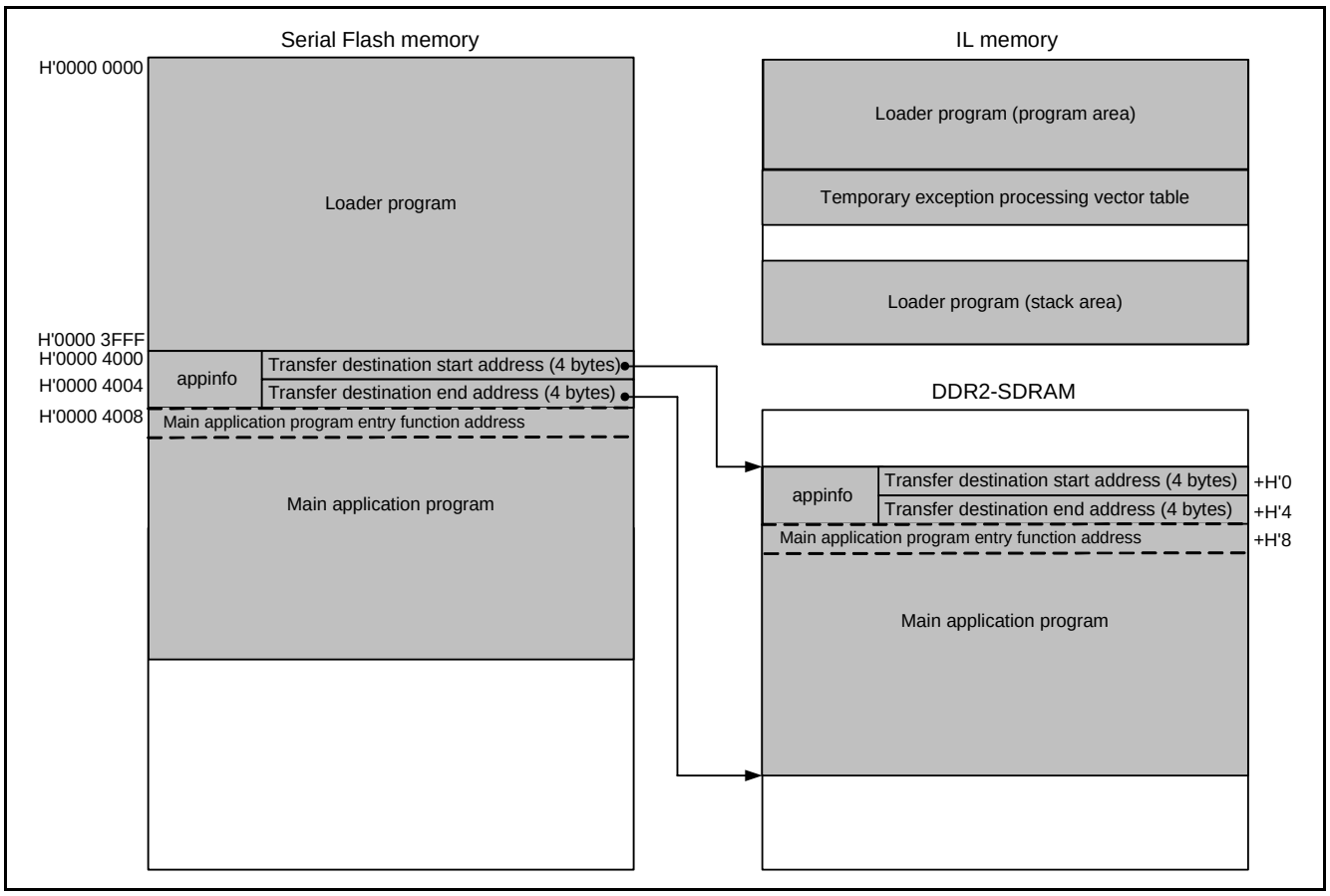

Figure 7.3 Conceptual Diagram of Application Program Transfer

(5) Cache Write-Back Processing

The loader program performs cache write-back processing to ensure coherency between the cache memory and the DDR2-SDRAM when transferring the application program to the DDR2-SDRAM by using the CPU. The sample program makes no special provisions of this type for transfers using the HPB-DMAC.

(6) Extinguish LED4 and Illuminate LED5 (Optional)

For debugging purposes, the sample program extinguishes LED4 and illuminates LED5 when the application program successfully finishes loading from the serial flash memory.

(7) Branch to Entry Function of Main Application Program

The loader program branches to the application program entry function (PowerON_Reset()) stored at the location indicated by the entry function address information of the main application program, as shown in table 7.7. For details of the method for setting the entry function address information of the main application program, see 8.1.1, Allocation of Sections.

8. Application Program Software Description

8.1 Operation Overview

The application program must be located in a memory area where it can be read by the loader program. The application program must also incorporate the application program transfer information (appinfo) and main application program entry function address information.

8.1.1 Allocation of Sections

The allocation of the sections of the application program is described below. For details of the roles of the sections that appear below, see *SH7734 Group: SH7734 Example of Initialization* (R01AN0665EJ). Table 8.1 shows the allocation of the sections described in this application note.

Table 8.1 Allocation of Application Program Sections

Allocation Address	Section Name	Section Application
H'0C00 0000 (DDR2-SDRAM)*1	DAPPINFO	Application program transfer information (appinfo)*3
H'0C00 0008 (DDR2-SDRAM)*1	VECTTBL	Main application program entry function address information*4 Reset vector table*7
	INTTBL*5	Interrupt vector table*7
	PResetPRG	Reset program
	PINTHandler*5	Exception/interrupt handler*6
	PIntPRG*5	Interrupt function*6
	P$PSEC	Section initialization program area
	P	Program area
	C	Constant area
	C$BSEC	Uninitialized data area address structure
	C$DSEC	Initialized data area address structure
	D*5	Initialized data (initial value)
	PnonCACHE*5	Cache manipulation program area*8
H'0F00 0000 (DDR2-SDRAM)*2	B	Uninitialized data area
	R*5	Initialized data area
H'0FFF F9F0 (DDR2-SDRAM)*2	S	Stack area
H'8FFF FDF0 (DDR2-SDRAM)*2	SP_S	Dedicated stack area of TLB miss handler
H'E501 0000 (OL memory)	INTTBL_OL*5	Interrupt mask table copy area*7
H'E520 0000 (IL memory)	PINTHandler_IL*5	Exception/interrupt handler copy area*6
	PIntPRG_IL*5	Interrupt function copy area*6
	PnonCACHE_IL*5	Cache manipulation program copy area*8

Notes: 1. The loader program included in the sample program uses the application program start address and end address information to transfer the application program from the serial flash memory to the DDR2-SDRAM. Therefore, the program area, constant area, and initialized data area of the application program must be located in physically contiguous areas. For details on specifying the start address and end address information, see section 8.4.
2. The uninitialized data area and stack area may be allocated to any address.
3. The sections are allocated such that the application program transfer information (appinfo) referenced by the loader program is located at a fixed address.
4. The sections are allocated such that the main application program entry function address information referenced by the loader program is located at a fixed address. The VECTTBL section is a section that starts with the definition of the main application program entry function (PowerON_Reset()), and that section is defined. The DAPPINFO and VECTTBL sections are allocated in order at the start of the application program.
5. The sample program settings for “mapping of sections from ROM to RAM” in the High-performance Embedded Workshop are shown in table 8.2. The initialization routine for sections run from the application program (InitSct()) copies the sections listed in table 8.2 from ROM* to RAM.
6. The IL memory is suitable for storing instructions. In the sample program, program-related sections used for interrupt handling are copied to and run from the IL memory to enable rapid startup of interrupt handling.
7. The OL memory is suitable for storing data. In the sample program, data-related sections used for interrupt handling are copied to and run from the OL memory to enable rapid startup of interrupt handling.
8. The cache manipulation program must be run from a space for which the cache is disabled. Therefore, the PnonCACHE section is copied to and run from the PnonCACHE_IL section in IL memory.

Table 8.2 Mapping of Sections from ROM to RAM

ROM*	RAM
D	R
INTTBL	INTTBL_OL
PIntPRG	PIntPRG_IL
PINTHandler	PINTHandler_IL
PnonCACHE	PnonCACHE_IL

Note: For cases described in *SH7734 Group: SH7734 Example of Initialization* (R01AN0665EJ) where data is mapped from the NOR flash (ROM) to the DDR2-SDRAM, IL memory, or OL memory (RAM), the sample program described in this application note uses the loader program to first transfer data from the serial flash to the DDR2-SDRAM and then to copy it to the IL memory. Thus, in the sample program described here, sections in the DDR2-SDRAM correspond to ROM*, and IL memory and OL memory correspond to RAM.

8.2 File Structure

Table 8.3 lists the files composing the application program.

Table 8.3 Application Program File Structure

File Name	Description	Remarks
sh7734_main.c	Main processing of the application program	Program code from *SH7734 Group: SH7734 Example of Initialization* (R01AN0665EJ), with changes to the serial log only
dbsct.c	Program code from *SH7734 Group: SH7734 Example of Initialization* (R01AN0665EJ), with changes to the section definitions used by the application program	
vhandler.src	Program code from *SH7734 Group: SH7734 Example of Initialization* (R01AN0665EJ), with reset handler processing deleted	This change was made because the loader program runs initialization processing after a reset.

8.3 Flowchart

The sample program reuses code from *SH7734 Group: SH7734 Example of Initialization* (R01AN0665EJ) to output debug information to a terminal program running on a PC.

Figure 8.1 is a flowchart of the application program.

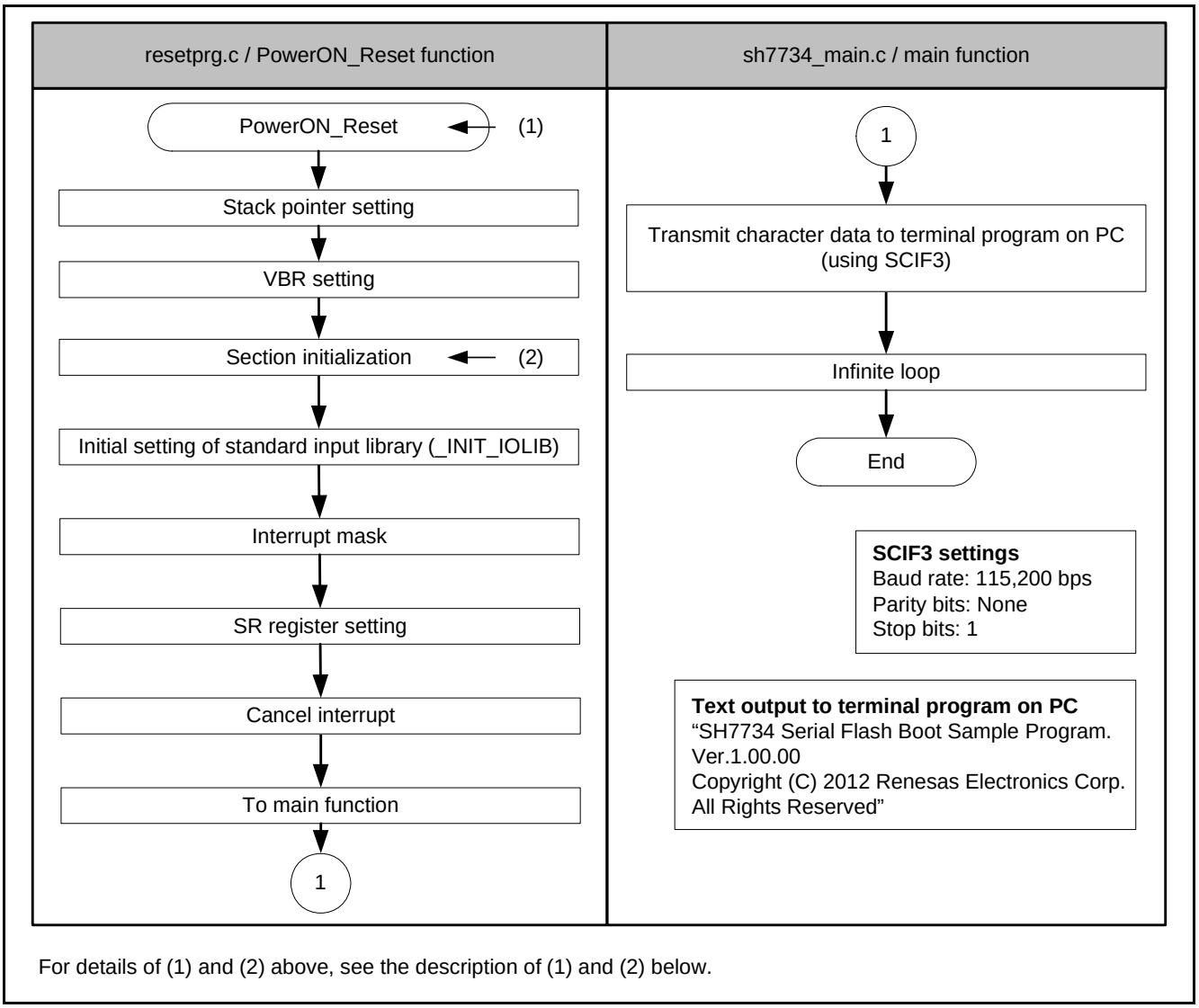

Figure 8.1 Application Program Flowchart

(1) Entry Function (PowerON_Reset) Setting

The address of the main application program entry function (PowerON_Reset) is set at the beginning of the reset vector table _RESET_Vectors. Table 8.4 lists the entry function address setting details.

Table 8.4 Entry Function Address Setting

Function	Description
File name	vecttbl.src
Allocated section name	VECTTBL
Table name	_RESET_Vectors
Setting function name	PowerON_Reset (entry function of main application program)

(2) Section Initialization

The sections are initialized by running the section initialization routine (InitSct function) defined in dbsct.c. For details, see *SH7734 Group: SH7734 Example of Initialization* (R01AN0665EJ).

The cache is already enabled when the loader program is run, so coherency between the cache memory and the DDR2-SDRAM may not be maintained after this section initialization routine (InitSct function) runs.

This does not present any particular problem for the sample program described in this application note, but cache write-back processing is performed nevertheless in consideration of programs for which failure to maintain coherency between the cache memory and the DDR2-SDRAM in the main program, etc., could be problematic.

8.4 Generating Application Program Transfer Information (appinfo)

Table 8.5 shows the structure used to generate the application program transfer information (appinfo). Sections address operators (__sectop and __secend) are used to acquire the start address and end address of the application program. This structure is located in the DAPPINFO section. Register the start address of the application program in app_top, and register the end address of the application program in app_end.

Table 8.5 Application Program Transfer Information (appinfo)

Item	Description		
File name	appinfo.c		
Structure name	appinfo		
Structure members	**Member name**	**Setting value**	**Description**
	void *app_top	__sectop("DAPPINFO")	Start address of application program
	void *app_end	__secend("PnonCACHE ")	End address of application program + 1
Allocation section name	DAPPINFO		

Note: Make sure that the total size of the loader program (16 KB) and application program does not exceed the capacity of the serial flash memory.

Figure 8.2 is a conceptual diagram of generation of the application program transfer information (appinfo).

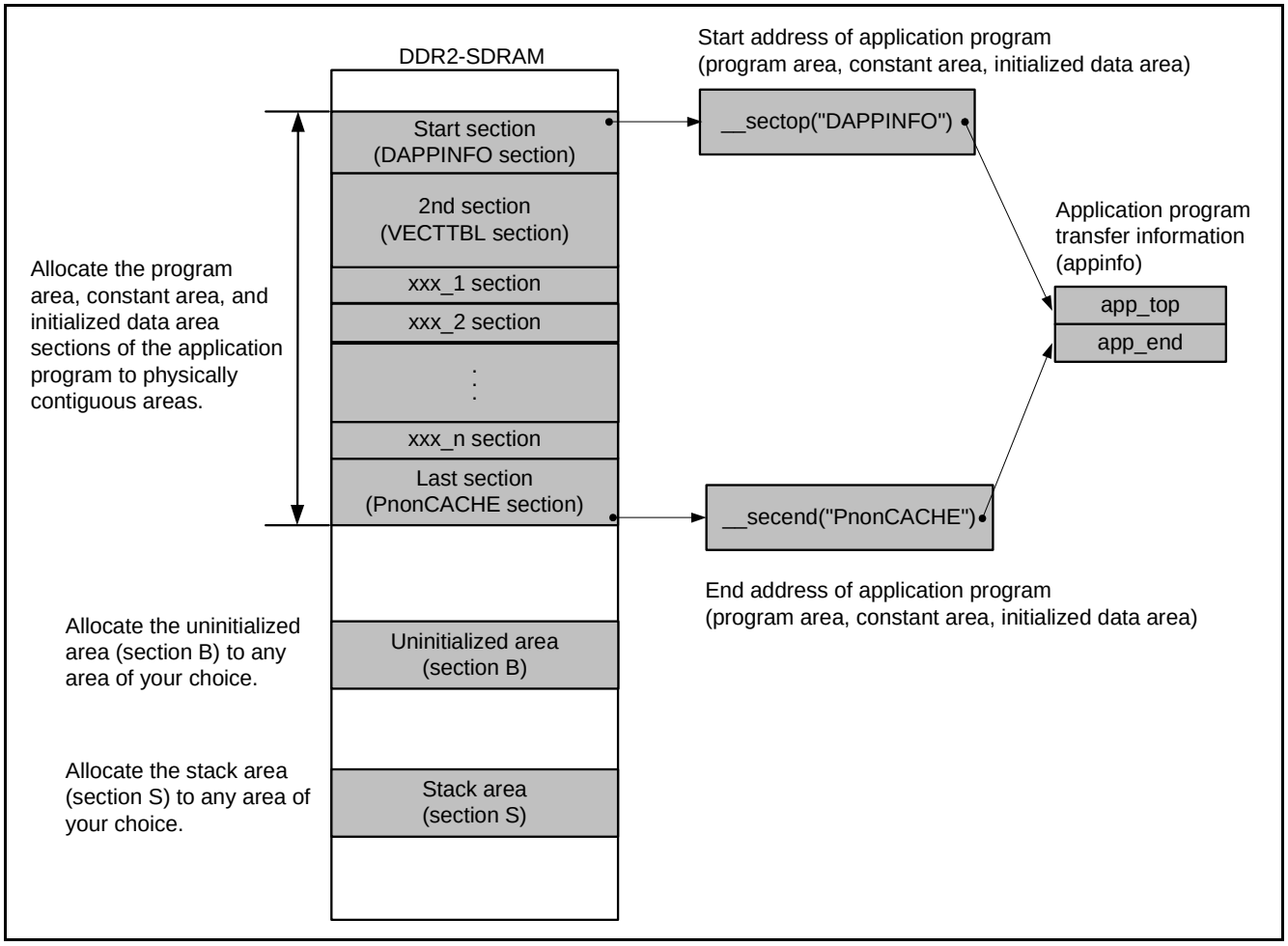

Figure 8.2 Conceptual Diagram of Generation of Application Program Transfer Information (appinfo)

9. Downloader Program Software Description

9.1 Detailed Specifications of Downloader Program

9.1.1 Operation Overview

Before running the downloader, the downloader must be transferred to the OL memory, the loader program to the IL memory, and the application program to the DDR2-SDRAM from the development environment used for debugging. Figure 9.1 is a conceptual diagram of this process.

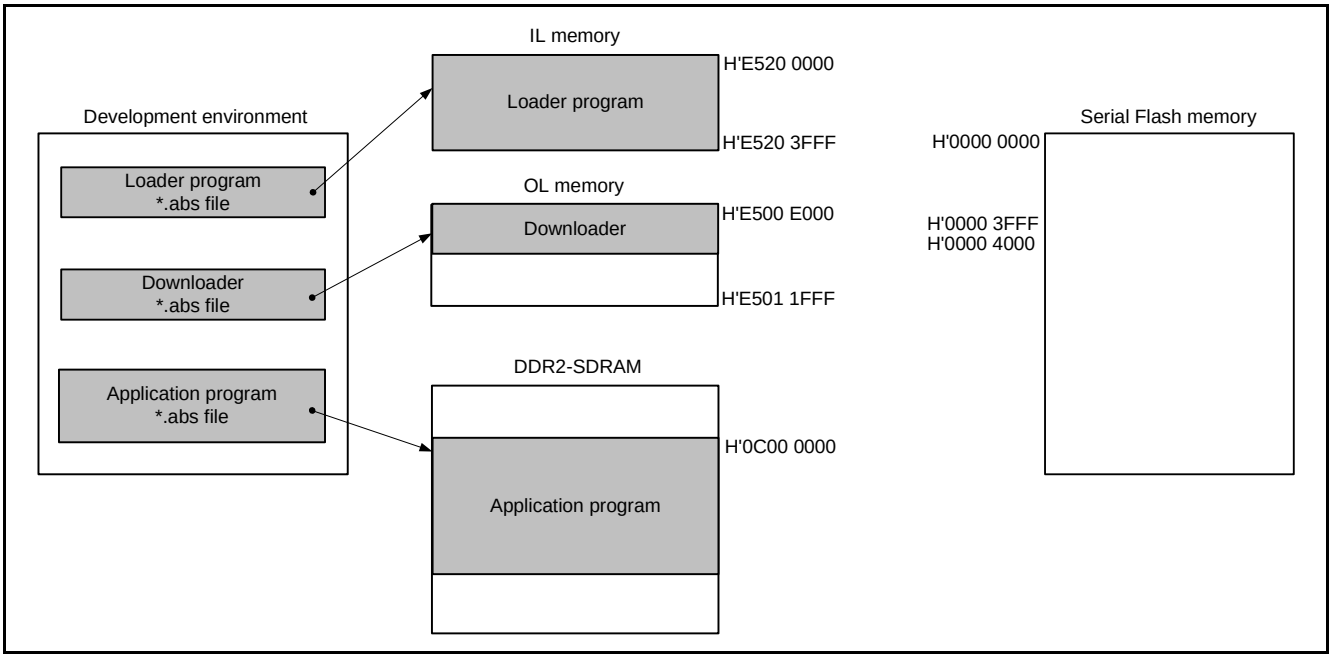

Figure 9.1 Conceptual Diagram of Downloader Operation (1)

Running the downloader programs the loader program and application program to the serial flash memory. The downloader programs the loader program to the address range H'0000 0000 to H'0000 3FFF in the serial flash memory and the application program to addresses starting at H'0000 4000. Figure 9.2 is a conceptual diagram of this process.

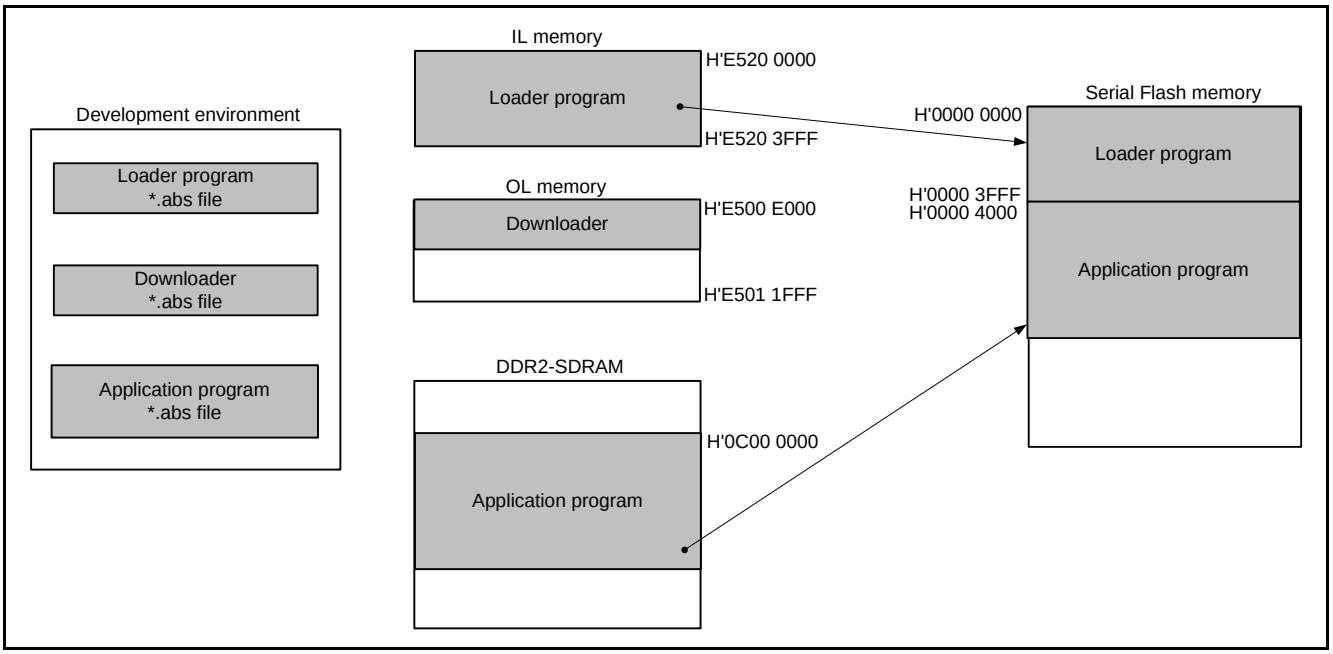

Figure 9.2 Conceptual Diagram of Downloader Operation (2)

Note: The sample program allocates the downloader to the OL memory, the loader program to the IL memory, and the application program to the DDR2-SDRAM to avoid contention among the sections of the loader program, application program, and downloader. There is no problem with allocating all program sections to the DDR2-SDRAM if care is taken to avoid contention.

9.1.2 Allocation of Sections

Table 9.1 shows the allocation of the sections of the downloader described in this application note. For details of the roles of the sections that appear below, see *SH7734 Group: SH7734 Example of Initialization* (R01AN0665EJ).

Table 9.1 Allocation of Downloader Sections

Allocation Address	Section Name	Section Application
H'E500 E000 (OL memory)	PResetPRG	Reset program area
	PnonCACHE	Cache manipulation program area
	P_DBSC3	DBSC3 initialization program area
	P$PSEC	Section initialization program area
	P	Program area
	C$BSEC	Uninitialized data area address structure
	C$DSEC	Initialized data area address structure
	C	Constant area
	D	Initialized data (Initial value)
	B	Uninitialized data area
	R	Initialized data area
H'E501 1C00 (OL memory)	S	Stack area

9.2 Structure

Table 9.2 lists the files composing the downloader.

Table 9.2 Downloader File Structure

File Name	Description	Remarks
sh7734_main_dl.c	Main processing routine of the downloader	
dbsct_dl.c	The dbsct.c source code file from *SH7734 Group: SH7734 Example of Initialization* (R01AN0665EJ) with changes to the section definitions used by the downloader	
r_rspi_dl.c	RSPI — serial flash memory control modules	Same as the corresponding loader program (RSPI) file listed in table 7.3.
resetprg_dl.c	Changes to processing after a reset for use by the downloader	Since the purpose of sample program is to write (program) data to the serial flash memory, the initialization processing after a reset has been kept to a bare minimum (no interrupt control, etc.).
r_rspi_dl.h	Include headers for externally referencing RSPI — serial flash memory control modules	Same as the corresponding loader program (RSPI) file listed in table 7.3.

Note: To distinguish the files of downloader program from those of the application program, loader program, etc., the downloader program files have the format “XXX_dl.c”.

9.3 List of Constants

Table 9.3 lists the constants used in the downloader sample code. Files identical to those described in section 7.3 for the loader program (RSPI) sample code are not listed.

Table 9.3 Constants Used in Downloader Sample Code

Constant Name	Setting Value	Description
SF_REQ_PROTECT	0	Defined in r_rspi_dl.h For serial flash memory software protect setting request
SF_REQ_UNPROTECT	1	Defined in r_rspi_dl.h For serial flash memory software protect cancel request
UNPROTECT_WR_STATUS	H'00	Defined in r_rspi_dl.c For serial flash memory software protect mode cancel request*[1]
PROTECT_WR_STATUS	H'5C	Defined in r_rspi_dl.c For serial flash memory software protect mode setting*[1]
SF_SECTOR_SIZE	H'10000	Defined in sh7734_main_dl.c Size of 1 sector in serial flash memory (64 KB)*[1]
SECTOR_NUM	256	Defined in sh7734_main_dl.c Total number of sectors in serial flash memory*[1]
DEVICE_SIZE	SECTOR_SIZE × SECTOR_NUM	Defined in sh7734_main_dl.c Total memory size of serial flash memory
L_PROG_SIZE	H'4000	Defined in sh7734_main_dl.c Size of loader program
L_PROG_SRC	H'E520 0000	Defined in sh7734_main_dl.c Data storage source start address (IL memory address) for programming the loader program Note: The sample program assumes that the loader program has been written beforehand to the IL memory.
L_PROG_DST	0	Defined in sh7734_main_dl.c Storage destination address (serial flash memory address) for programming the loader program Note: The SH7734 serial boot specification reads data beginning from the start address of the serial flash memory, so the loader program must be programmed to the start address of the serial flash memory.
APROG_TOP_RAM	H'AC00 0000	Defined in sh7734_main_dl.c Application program start address Note: The sample program stores the application program in the P2 area (caching disabled, MMU address translation disabled).
APROG_TOP_SFLASH	H'4000	Defined in sh7734_main_dl.c The start address in serial flash memory where the application program is stored
APPINFO_TOP	APROG_TOP_RAM	Defined in sh7734_main_dl.c The DDR2-SDRAM address where the application program start address is stored

Constant Name	Setting Value	Description
APPINFO_END	APROG_TOP_RAM + 4	Defined in sh7734_main_dl.c The DDR2-SDRAM address where the application program end address is stored

Note: 1. For details, see the datasheet of the serial flash memory used.

9.4 List of Functions

Table 9.4 lists the functions used in the downloader sample code. Functions identical to those described in section 7.4 for the loader program (RSPI) sample code are not listed.

Table 9.4 Functions Used in Downloader Sample Code

Function Name	Description
main	Downloader main processing
write_prog_data	Loader program/application program write processing
init_erase_flag	Erased sector management table initialization
is_erased_sector	Target sector erased conformation processing
set_erase_flag	Target sector erased setting processing
halt	Call at successful downloader program end
error	Call at downloader program end with error
R_RSPI_SFProtectCtrl	Serial flash memory software protect set/cancel
R_RSPI_SFSectorErase	Serial flash memory target sector erase processing
R_RSPI_SFByteProgram	RSPI serial flash memory write processing (1-byte units)

9.5 Function Specifications

The specifications of sample code functions are described below. Functions identical to those described in section 7.5 for the loader program (RSPI) sample code are not listed.

main	
Outline	Downloader main processing
Header	r_rspi_dl.h
Declaration	void main(void)
Description	• Programs the loader program and application program to the serial flash memory. • The sample program illuminates LED4 during downloader processing for debugging purposes. • The addresses where the loader program and application program are stored during programming are the same as the addresses to which they are allocated when run, so it is possible that even if the on-chip ROM program for boot startup or loader program cannot load correctly it may erroneously appear to be operating properly because the program stored during programming is running. To prevent this, when it programs the serial flash memory the sample program erases the area in the IL memory where the loader program was stored and the area in the DDR2-SDRAM where the application program was stored. (optional)
Arguments	None
Return Value	None

write_prog_data		
Outline	Loader program/application program write processing	
Header	r_rspi_dl.h	
Declaration	static int32_t write_prog_data(uint8_t *program_data, const uint32_t sflash_addr, const uint32_t size)	
Description	• Erases the write target sector before writing to (programming) the serial flash memory. • Writes (programs) the loader program or application program to the serial flash memory. • Performs verification processing after programming of the serial flash memory finishes.	
Arguments	uint8_t *program_data	: Pointer for storing write data
	const uint32_t sflash_addr	: Write target serial flash memory address
	const uint32_t size	: Write data size (1-byte units)
Return Value	0: Write successful Other than 0: Write failure	

init_erase_flag	
Outline	Erased sector management table initialization
Header	
Declaration	static void init_erase_flag(void)
Description	• Initializes the erased sector management table. • The erased sector management table is used to skip erase processing of sectors that have already been erased.
Arguments	None
Return Value	None

is_erased_sector	
Outline	Target sector erased conformation processing
Header	
Declaration	static volatile uint8_t is_erased_sector(const uint32_t sector_no)
Description	Checks whether or not the sector specified by the parameter has already been erased.
Arguments	uint32_t sector_no : Erasure confirmation target sector
Return Value	1: Erased 0: Not erased

set_erase_flag	
Outline	Target sector erased setting processing
Header	
Declaration	static void set_erase_flag(const uint32_t sector_no)
Description	Updates the erased sector management table to indicate that the sector specified by the parameter has already been erased.
Arguments	uint32_t sector_no : Erasure target sector
Return Value	None

halt	
Outline	Call at successful downloader program end
Header	
Declaration	static void halt(void)
Description	• Called when the downloader program finishes successfully. • Consists of only an infinite loop internally.
Arguments	None
Return Value	None

error	
Outline	Call at downloader program end with error
Header	
Declaration	static void error(void)
Description	• Called when the downloader program ends with an error. • Consists of only an infinite loop internally.
Arguments	None
Return Value	None

R_RSPI_SFProtectCtrl		
Outline	Serial flash memory software protect set/cancel	
Header	r_rspi_dl.h	
Declaration	void R_RSPI_SFProtectCtrl(const enum sf_req req)	
Description	Sets or cancels serial flash memory software protect according to the parameter.	
Arguments	enum sf_req req	UNPROTECT_WR_STATUS: Cancel protect PROTECT_WR_STATUS: Set protect
Return Value	None	

R_RSPI_SFSectorErase		
Outline	Serial flash memory target sector erase processing	
Header	r_rspi_dl.h	
Declaration	void R_RSPI_SFSectorErase(const uint32_t sector_no)	
Description	Erases the sector in the serial flash memory specified by the parameter.	
Arguments	uint32_t sector_no	: Erase target sector
Return Value	None	

R_RSPI_SFByteProgram		
Outline	R RSPI serial flash memory write processing (1-byte units)	
Header	r_rspi_dl.h	
Declaration	void R_RSPI_SFByteProgram(const uint32_t addr, const uint8_t *buf, const uint32_t size)	
Description	• Writes, in 1-byte units, data of the specified size to the serial flash memory address specified by the parameter. • Specifies 8 bits as the transfer data length (bits SPB0 to SPB3) in the SPCMD register.	
Arguments	uint32_t addr uint8_t *buf uint32_t size	: Serial flash memory address of write target : Pointer for write data storage : Write size (1-byte)
Return Value	None	

9.6 Flowcharts

Figure 9.3 is a flowchart of the main processing routine of the downloader. The downloader is run form the OL memory and programs data to the serial flash memory.

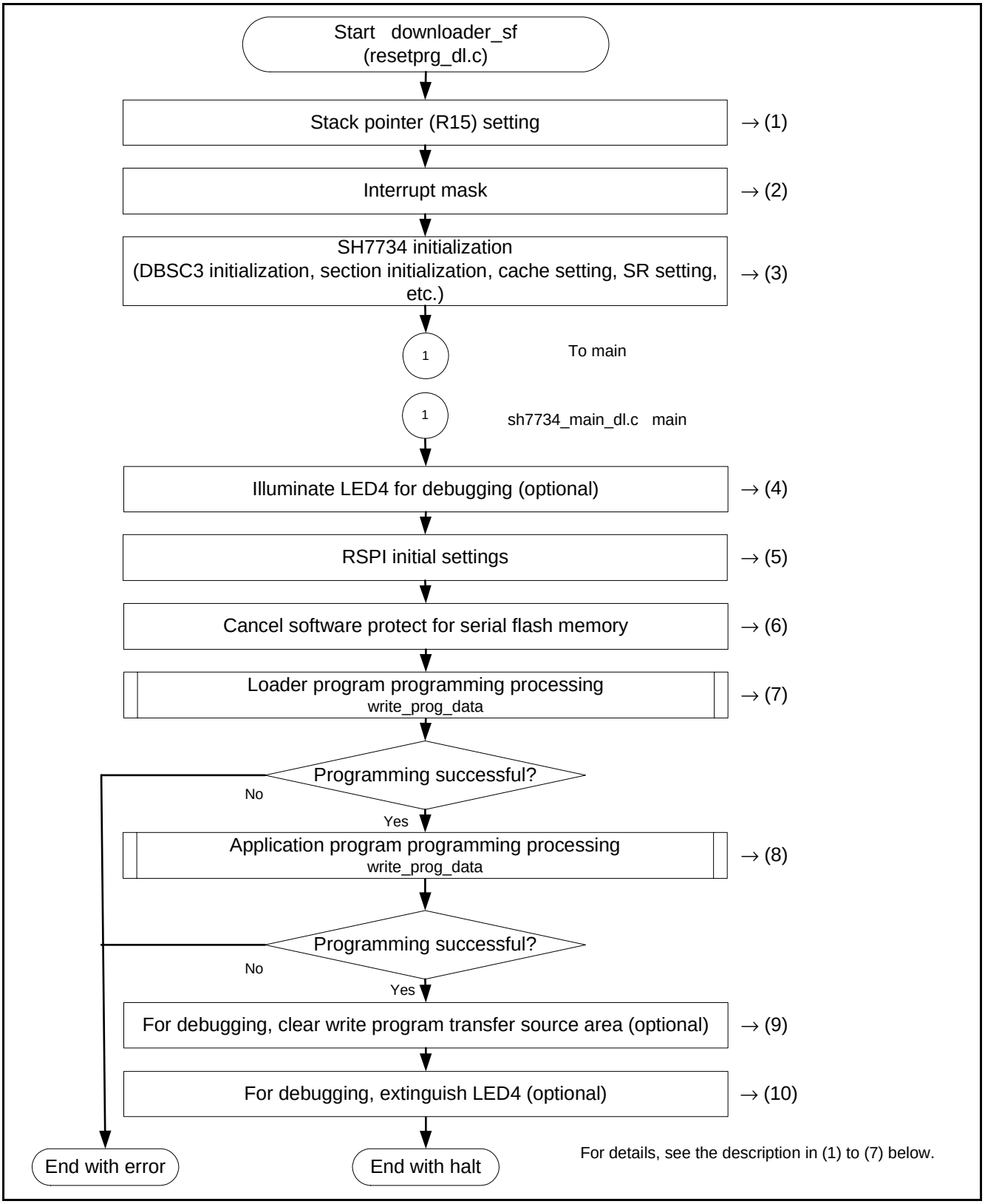

Figure 9.3 Flowchart of Downloader Processing

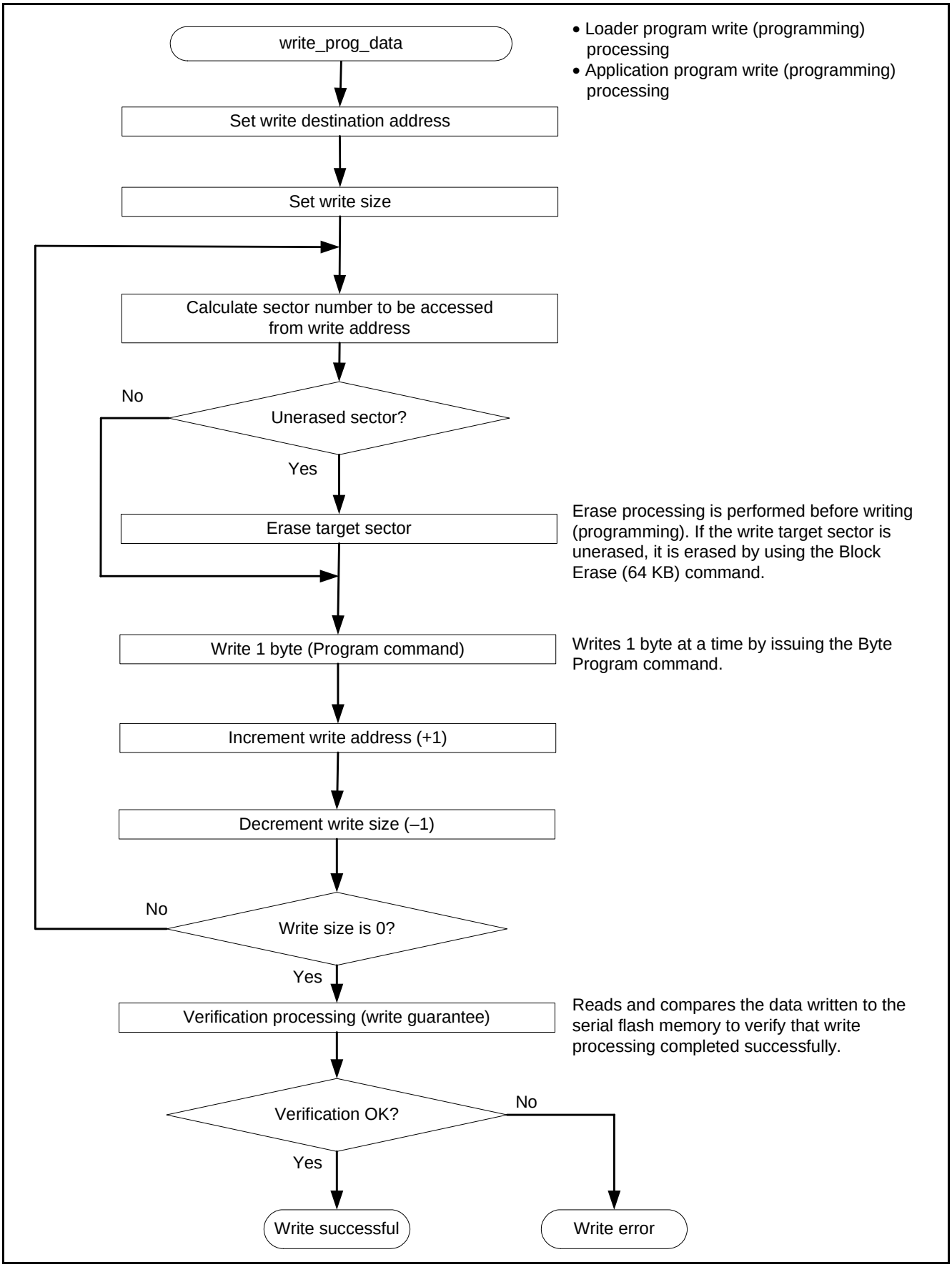

Figure 9.4 Flowchart of Processing to Program Serial Flash (write_prog_data())

(1) Stack Pointer Setting

The sample program sets the address H'E501 2000 in the stack pointer (R15). As shown in figure 9.5, this is specified by a #pragma entry.

```
#pragma entry downloader_sf (sp=0xE5012000)
```

Figure 9.5 Downloader Program Stack Pointer Setting

(2) Interrupt Mask

The downloader described in this application note does not include code for interrupt handling. The SR register is set to the initial value following a power-on reset (BL = 1).

(3) SH7734 Initialization Processing

The bare minimum of processing needed for the downloader to operate is performed (DBSC3 initialization. section initialization, cache setting, SR register setting, etc.).

The sample program presented in this application note reuses sample code from *SH7734 Group: SH7734 Example of Initialization* (R01AN0665EJ) to initialize the SH7734. Refer to that application note for details.

(4) LED4 Illumination (Optional)

For debugging purposes, the sample program illuminates LED4 during programming of the serial flash.

(5) RSPI Initial Settings

Initial settings are made to the RSPI to enable access to the serial flash memory.

(6) Software Protect Cancellation

The Write Status Register command is issued to the serial flash memory to cancel software protect.

(7) Loader Program Write Processing

The downloader programs to the address range H'0000 0000 to H'0000 3FFF in serial flash memory the loader program that is already allocated to the address range H'E520 0000 to H'E520 3FFF in IL memory. Table 9.5 summarizes the loader program write processing.

Table 9.5 Loader Program Write Processing

Item	Description
Loader program transfer source address (IL memory address)	H'E520 0000
Loader program transfer destination address (serial flash memory address)	H'0000 0000 (fixed)
Transfer size	H'4000 (fixed)
Write processing procedure	1. It is confirmed that the write destination address has been erased. 2. If it is an unerased area, erase processing is performed. 3. The Program command is issued to perform write processing. Data is written in 1-byte units.

(8) Application Program Write Processing

The downloader programs to an area starting at address H'0000 4000 in serial flash memory the application program that is already allocated to the DDR2-SDRAM. Table 9.6 summarizes the application program write processing.

Table 9.6 Application Program Write Processing

Item	Description
Application program transfer source address (DDR2-SDRAM)	appinfo allocation address in application program Defined by sh7734_main.c #define APROG_TOP_RAM in the downloader workspace.
Application program transfer destination address (serial flash memory)	H'0000 4000 (fixed)
Transfer size	Calculated from appinfo in application program (dependent on application program).
Write processing procedure	1. It is confirmed that the write destination address has been erased. 2. If it is an unerased area, erase processing is performed. 3. The Program command is issued to perform write processing. Data is written in 1-byte units.

(9) Clearing of Write Program Transfer Source Area (Optional)

To confirm that the loader program and application program load properly from the serial flash memory, when it programs the serial flash memory the sample program erases the data area in the IL memory where the loader program was stored and the data area in the DDR2-SDRAM where the application program was stored.

(10) Extinguishing LED4 (Optional)

For debugging purposes, the sample program extinguishes LED4 when programming of the serial flash finishes.

9.7 Serial Flash Memory Commands

Table 9.7 lists the serial flash memory commands used by the downloader. These commands are issued via the RSPI to manipulate the serial flash memory.

Table 9.7 Serial Flash Memory Commands Used by Downloader

Command Name	Opcode	Function
High-Speed Read	H'0B	Reads data.
Write Enable	H'06	Enables Program (write), Erase, and Write Status Register commands.
Write Disable	H'04	Disables Program (write), Erase, and Write Status Register commands, etc.
Read Status Register	H'05	Reads the status register.
Write Status Register	H'01	Writes to the status register (to cancel software protect).
Secter Erase (64 KB)	H'D8	Erases a sector (64 KB).
Page Program	H'02	Writes data (1 byte)

10. Application Example

10.1 Writing (Programming) Programs to Serial Flash Memory

The procedure for programming the loader program and application program to the serial flash memory using the [sh7734_sflash_app] workspace provided with this application note is described below.

10.1.1 Using the Downloader

The downloader provided with this application note is designed to be used in conjunction with High-performance Embedded Workshop and the E10A-USB emulator. In order to use it with another development environment, the program should be modified as necessary to accommodate the usage environment.

It is not possible to write (program) the programs to the serial flash memory simply by selecting Download from the [Debug] menu of High-performance Embedded Workshop and choosing the download modules. The procedure described below uses the downloader to write (program) the programs to the serial flash memory.

10.1.2 Downloader Automation (Command Batch File)

In order to program the loader program and application program to the serial flash memory, it is necessary to first transfer the loader program to the IL memory, the downloader to the OL memory, and the application program to the DDR2-SDRAM, and then to run the downloader. These processing steps can be performed manually, but the procedure described in this application note uses a High-performance Embedded Workshop command batch file to automate the process.

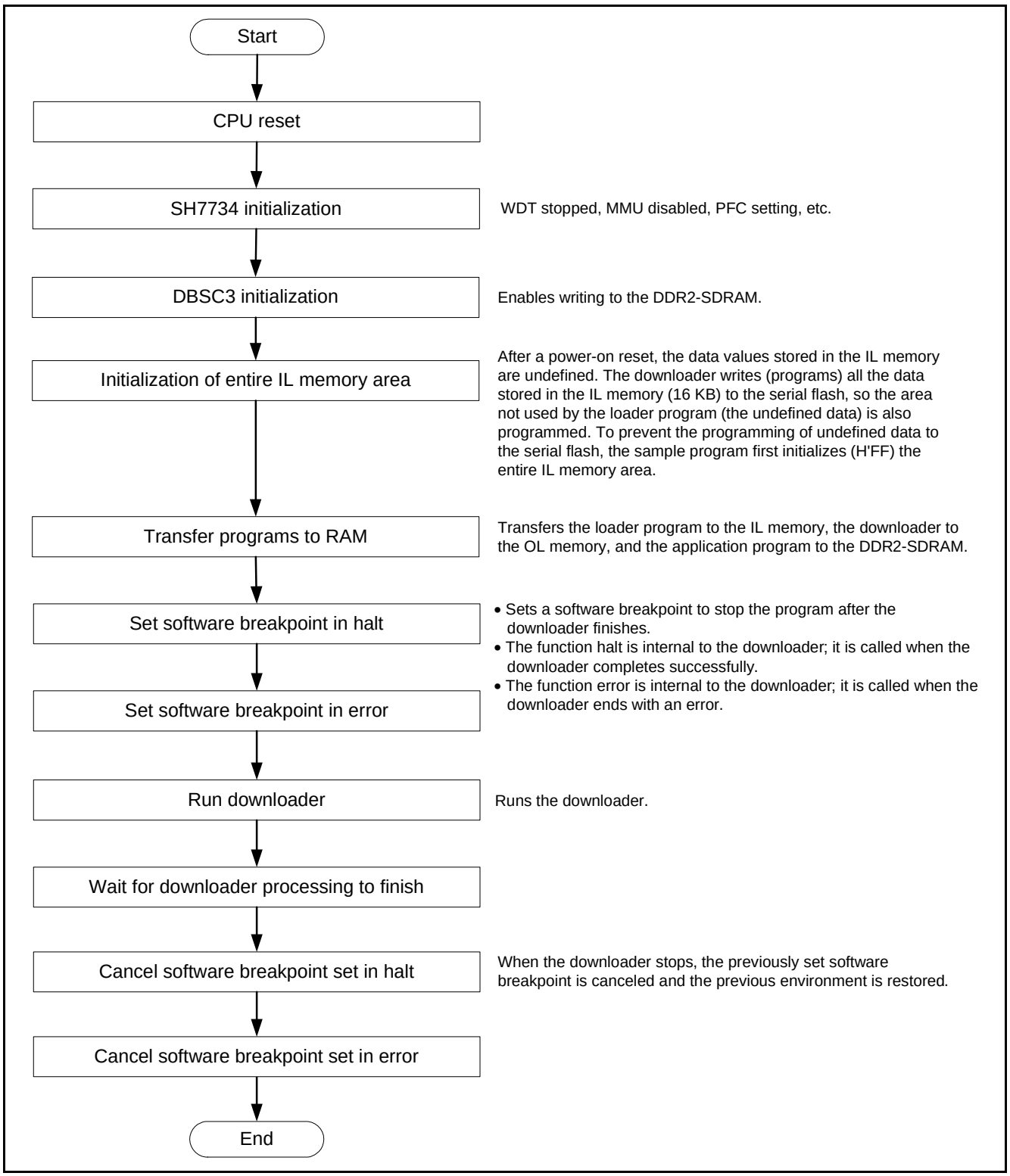

Figure 10.1 Flowchart of Command Batch File

10.1.3 Registration of Download Modules and Batch File

Figure 10.2 shows the directory structure of the [sh7734_sflash_app] workspace. In the [sh7734_sflash_app] workspace, download modules [1], [2], and [4] in figure 10.2 are registered in the project. In addition, batch file [3] in figure 10.2 is registered in the project.

```
\sh7734_sflash_app                  : Workspace directory
  |-sh7734_sflash_app               : Project directory
  |    |-debug                      :
  |       |-sh7734_sflash_app.abs   : Application program executable file ---------- [1]
  |                                 :
  |-inc                             : Directory for storing common include files:
  |-src                             : Directory for storing source files
  |-sflash_boot                     : Directory for storing downloader and loader program
    |-sh7734_sflash_downloader.abs  : Downloader executable file ---------------------- [2]
    |-downloader.hdc                : Batch file (for starting downloader) ------------ [3]
    |-sh7734_sflash_loader_prog.abs : Loader program executable file ------------------ [4]
```

Figure 10.2 Directory Structure of [sh7734_sflash_app] Workspace

(1) Changing the Download Modules

To change the download modules registered in the project, change the settings in the [Debug Settings] dialog box. To display the [Debug Settings] dialog box, select [Debug Settings] from the [Debug] menu of High-performance Embedded Workshop.

For details of the registration method, see the user's manual of High-performance Embedded Workshop.

(2) Changing the Batch File

To change the download batch file registered in the project, change the settings in the [Specify Batch File] dialog box. To display the [Specify Batch File] dialog box, do the following. First, select [Command Line] from the [Display] menu of High-performance Embedded Workshop. Then click the [Specify Batch File] button in the popup menu of the [Command Line] window to display the [Specify Batch File] dialog box.

For details of the registration method, see the user's manual of High-performance Embedded Workshop.

10.1.4 Programming Procedure

The procedure for programming the loader program and application program to the serial flash memory by using the [sh7734_sflash_app] workspace is as follows.

1. Copy the contents of the [sh7734_sflash_app] workspace directory to C:\WorkSpace.
2. Double click the file [sh7734_sflash_app].hws in the workspace directory. High-performance Embedded Workshop starts.
3. Select [Build All] from the [Build] menu of High-performance Embedded Workshop to build the program. This generates the application program.
4. Select [Connect] from the [Debug] menu of High-performance Embedded Workshop to establish a connection to the target.
5. After the connection is established, select [Command Line] from the [Display] menu of High-performance Embedded Workshop to display the [Command Line] window as shown in figure 10.3.
6. Click the [Run Batch File] button in the [Command Line] window to run the registered batch file [downloader.hdc].

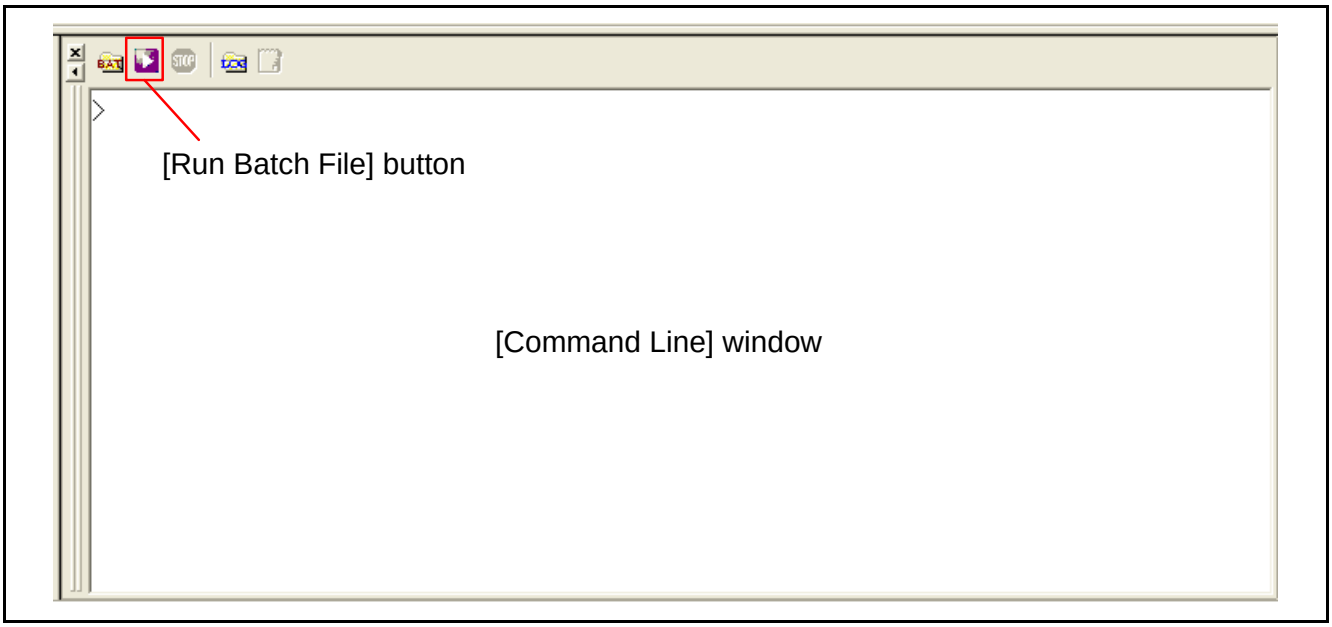

Figure 10.3 [Command Line] Window and [Run Batch File] Button

7. When the batch file [downloader.hdc] is run, all the download modules registered in the workspace (the loader program, application program, and downloader) are transferred to the RAM and the downloader is launched. As shown in figure 10.4, the program counter is stopped by halt when the downloader finishes successfully and it is stopped by error when the downloader ends with an error.
8. Once programming finishes successfully, the loader program and application program can be run after a reset.

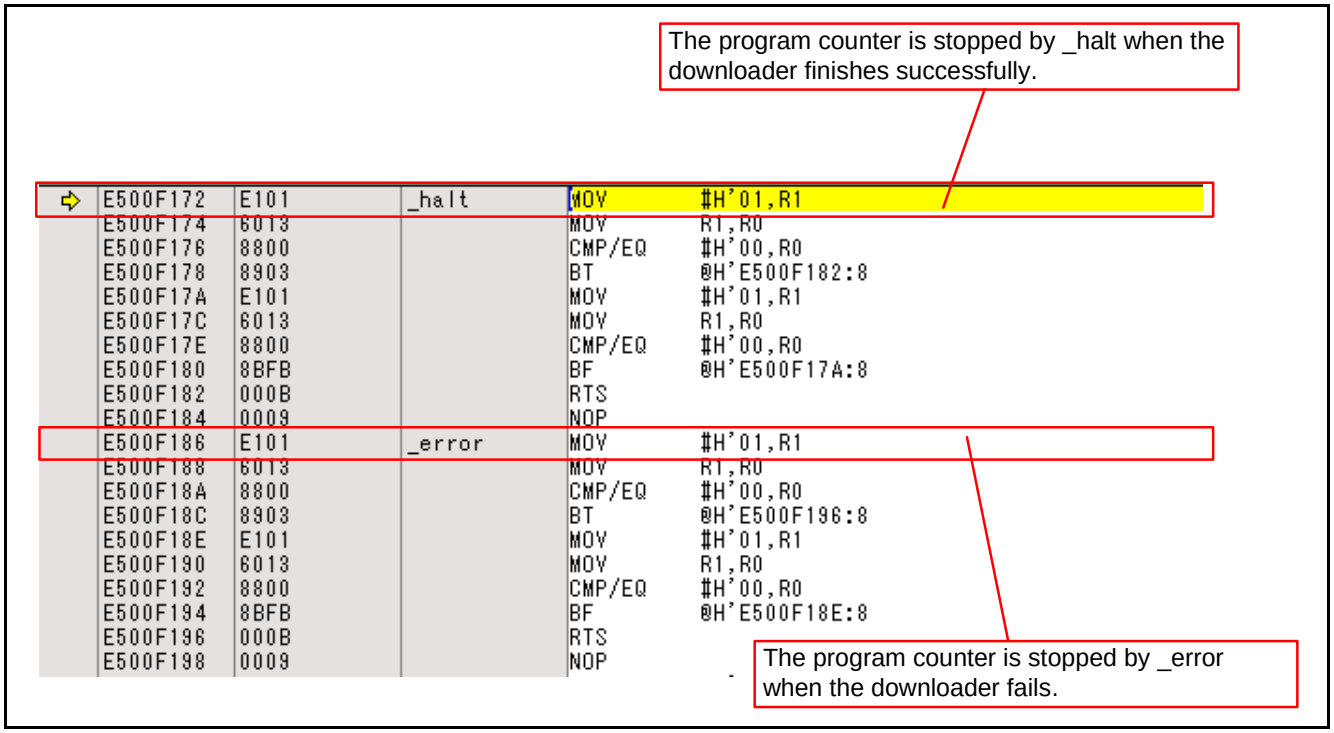

Figure 10.4 High-performance Embedded Workshop Screen when Downloader Ends

10.2 Notes on Use of PC Break (Breakpoint) Function

The following should be considered when using the PC break (breakpoint) function. The PC break function works by replacing the instruction at a specified address with a custom instruction. The PC break (breakpoint) function (custom instruction replacement) specified before launch of the loader program or application program overwrites the normal instructions during the boot process, making it impossible to boot the system. Therefore, a hardware break should be used to break operation of the loader program or application program. In order to use the PC break function, first run one of the programs and break its execution once by using a hardware break. After this the PC break function may be specified to insert a break.

10.3 Loader Program with Support for Quad-SPI (Reference)

A separate workspace is provided containing a version of the loader program sample code that uses the Renesas quad-serial peripheral interface (Quad-SPI) to shorten the transfer time.

10.3.1 Structure of Sample Code

Separate workspaces are provided for the sample code of the loader program with support for Quad-SPI, application program, and downloader described in this application note. The code is divided among the three workspaces as indicated in table 10.1.

Table 10.1 Structure of Sample Code

Workspace Name	Description
sh7734_qsflash_loader_prog (Loader program (Quad-SPI))	This workspace project is used to build a version of the loader program that supports the Quad-SPI.
sh7734_sflash_app (Application program)	Uses the same files as for the RSPI.
sh7734_sflash_downloader (Downloader)	Uses the same files as for the RSPI. Note: This sample program does not support programming the serial flash memory by using the Quad-SPI.

10.3.2 Features of Loader Program with Support for Quad-SPI

(1) Peripheral Functions Used

The Quad-SPI is used instead of the RSPI to access the serial flash memory.

(2) Loader Program Operation Overview

Aside from the fact that the Quad-SPI is used instead of the RSPI to access the serial flash memory, operation is identical to that described in 1.3.2, Loader Program Operation Overview.

(3) Transfer Efficiency

Transferring one byte of data takes eight clock cycles when using the RSPI, as shown in figure 10.5, but only two clock cycles when using the Quad-SPI (Quad-SPI mode). The Quad-SPI delivers four times the transfer efficiency, assuming the same bit rate.

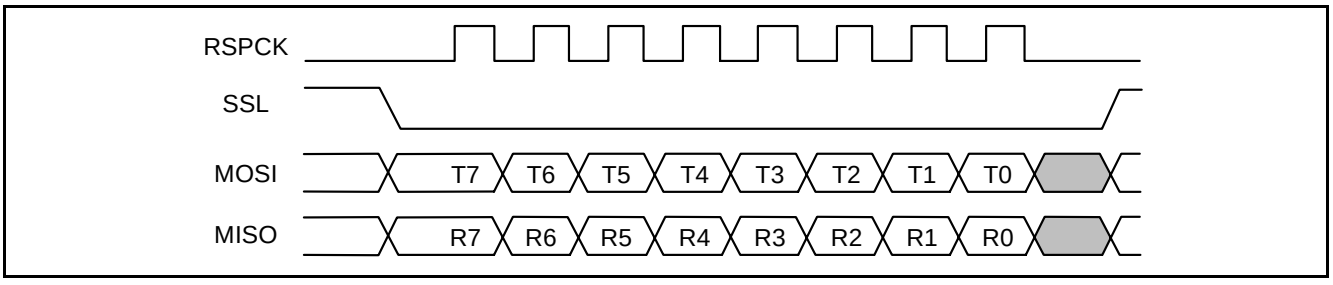

Figure 10.5 RSPI Transfer Format

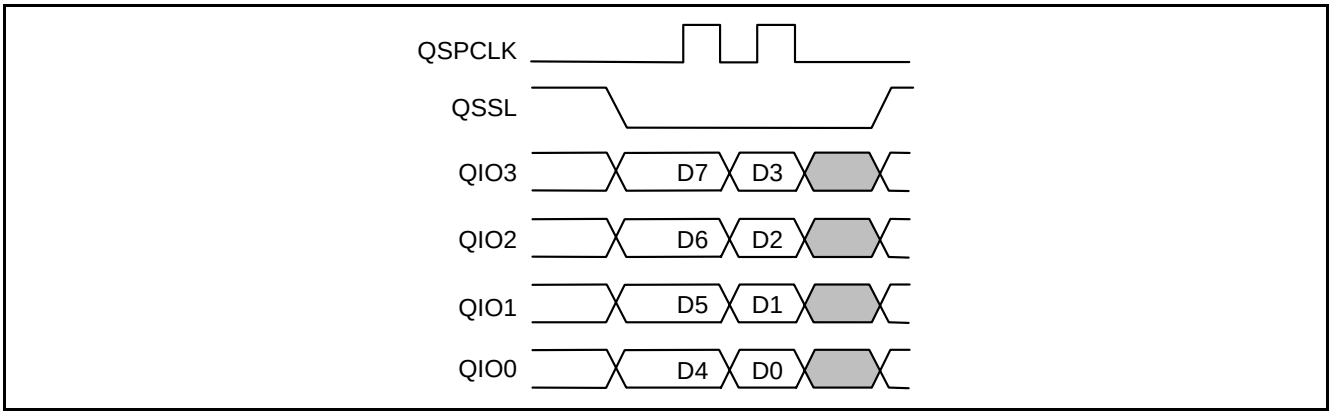

Figure 10.6 Quad-SPI (Quad-SPI Mode) Transfer Format

(4) Memory Map of Serial Flash and IL Memory

Identical to that for the RSPI. For details, see 7.1.1.

(5) SPI Mode and Bit Rate Specifications

Table 10.2 Loader Program (Quad-SPI) SPI Mode and Bit Rate Specifications

Item	Description
SPI mode	SPI mode 3 (CPOL = 1, CPHA = 1)
Bit rate	12.5 MHz

(6) File Structure

Only the points of difference from the RSPI version of the loader program are noted below.

Table 10.3 Files Used by Loader Program (Quad-SPI)

File Name	Description	Remarks
sh7734_main_lp.c	Main processing routine of the loader program (Quad-SPI)	
r_dmac_lp.c	Quad-SPI — HPB-DMAC control modules	Implemented in r_rspi_lp.c in the RSPI version of the loader program, but a separate file is provided for the Quad-SPI version.
r_rqspi_lp.c	Quad-SPI control modules	
r_rqspi_qserial_flash_lp.c	Quad-SPI — serial flash memory control modules	Implemented in r_rspi_lp.c in the RSPI version of the loader program, but a separate file is provided containing the user interface portion of the Quad-SPI version related to serial flash memory control.
r_rqspi_lp.h	Include headers for externally referencing Quad-SPI control modules	
r_rqspi_qserial_flash_lp.h	Include headers for externally referencing Quad-SPI — serial flash memory control modules	

(7) Constants and List of Functions

These differ from the RSPI version. For details refer to the sample code in the sh7734_qsflash_loader_prog workspace.

(8) Flowchart

Other then use of the Quad-SPI instead of the RSPI to transfer the application program, operation is identical to that shown in the flowchart in section 7.6.

10.3.3 Programming Loader Program with Support for Quad-SPI to Serial Flash Memory

Change the Session setting in High-performance Embedded Workshop from DefaultSession to qsflashSession, and store the .abs file generated using the sh7734_qsflash_loader_prog workspace in location [1].

Other than that, the procedure for programming the serial flash memory is basically the same as that described in section 10.1.

```
\sh7734_sflash_app                    : Workspace directory
  |-sh7734_sflash_app                 : Project directory
  |    |-debug                        :
  |        |-sh7734_sflash_app.abs     : Application program executable file ---------- Identical to that for the RSPI.
  |
  |-inc                               : Directory for storing common include files:
  |-src                               : Directory for storing source files
  |-sflash_boot                       : Directory for storing downloader and loader program
    |-sh7734_sflash_downloader.abs    : Downloader executable file ----------------------- Identical to that for the RSPI.
    |-downloader.hdc                  : Batch file (for starting downloader) ------------- Identical to that for the RSPI.
    |-qsflash                         : Folder containing loader program executable file
      |-sh7734_sflash_loader_prog.abs : Loader program executable file ------------------- [1]
```

Figure 10.7 Directory Structure of [sh7734_sflash_app] Workspace with Loader Program with Support for Quad-SPI

11. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

12. Reference Documents

User's Manual: Hardware
SH7734 Group User's Manual: Hardware Rev.1.00
The latest version can be downloaded from the Renesas Electronics website.

Technical Update/Technical News
The latest information can be downloaded from the Renesas Electronics website.

User's Manual: Development Tools
SuperH C/C++ Compiler Package V.9.04 User's Manual 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/contact/

REVISION HISTORY	SH7734 Group Application Note Example of Booting from Serial Flash Memory

Rev.	Date	Description	
		Page	Summary
1.00	Sep. 24, 2012	—	First edition issued

General Precautions in the Handling of MPU/MCU Products

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

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

Notice

1. 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.
2. 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.
3. 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.
4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
   "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 etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.
   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). 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 for which it is not intended. 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 which the product is not intended by Renesas Electronics.
6. 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.
7. 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 systems manufactured by you.
8. 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.
9. 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. You should not use Renesas Electronics products or 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. When exporting the Renesas Electronics 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.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
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: +44-1628-651-700, Fax: +44-1628-651-804

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.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

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

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

Colophon 2.2