SH726A/SH726B Group

R01AN1178EJ0100
Rev.1.00
Jun. 25, 2012

E10A-USB Flash Memory Download Function

(Download to the Serial Flash Memory)

Abstract

E10A-USB emulator has the function to download a load module to the flash memory. This function requires a download program to access the flash memory (hereinafter called the "FMTOOL").

This document describes how to download a load module to the serial flash memory applying the FMTOOL.

Target Device

SH726A/SH726B Group (hereinafter called the "SH726B")

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

Contents

1. Specifications

Download the load module allocated in the SPI multi I/O bus space to the serial flash memory using the FMTOOL that supports the serial flash memory. The FMTOOL uses the SPI multi I/O bus controller and allows the serial flash memory corresponding to the multi I/O with its data bus width of 4 bit to be accessed.

Table 1.1 lists the peripheral functions and their applications. Figure 1.1 shows the download procedure using the FMTOOL.

Table 1.1 Peripheral Functions and Their Applications

Peripheral Function	Application
SPI multi I/O bus controller	Downloads to the serial flash memory
H-UDI	Connects the E10A-USB emulator

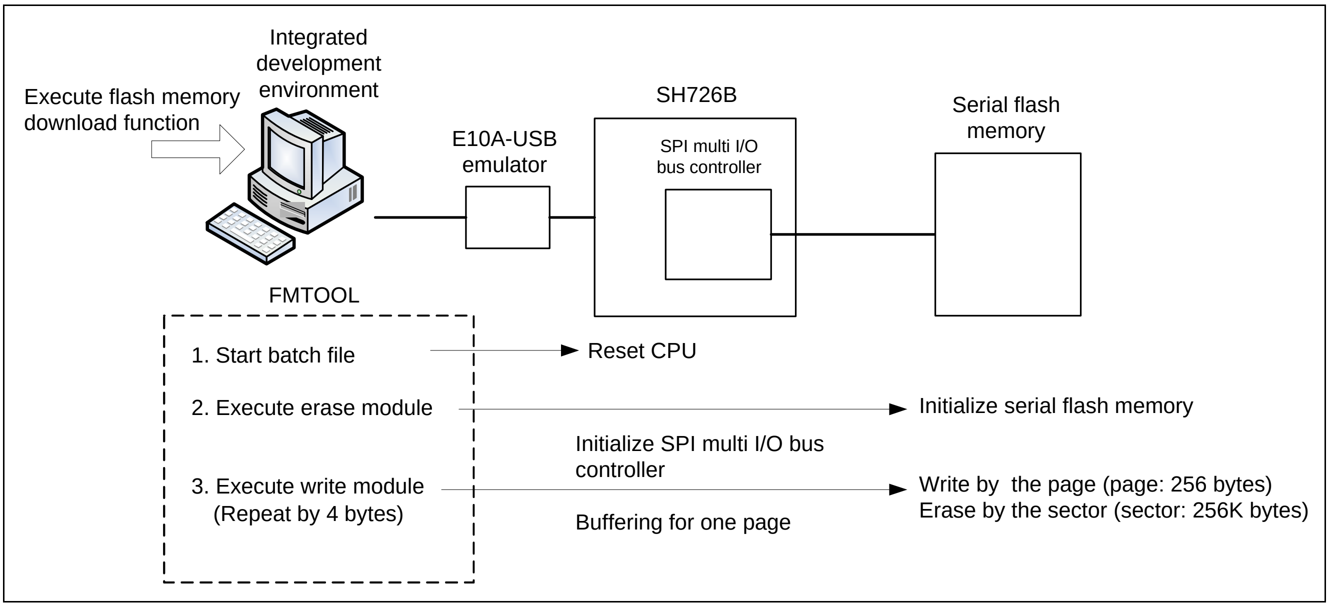

Integrated development environment
Execute flash memory download function
SH726B
Serial flash memory
E10A-USB emulator
SPI multi I/O bus controller
FMTOOL
1. Start batch file
Reset CPU
2. Execute erase module
Initialize serial flash memory
Initialize SPI multi I/O bus controller
3. Execute write module (Repeat by 4 bytes)
Write by the page (page: 256 bytes)
Erase by the sector (sector: 256K bytes)
Buffering for one page

Figure 1.1 Procedure of Download Using FMTOOL

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
MCU used	SH726B
Device used	Serial flash memory applicable to multi I/O bus manufacturer: Spansion. model: S25FL129P0XMFI01
Operating frequency	CPU clock (Iφ): 216MHz Bus clock (Bφ): 72MHz Peripheral clock 1 (Pφ): 36MHz
Operating voltage	Source power (I/O): 3.3V Source power (internal): 1.25V
Integrated development environment	Renesas Electronics High-performance Embedded Workshop Ver.4.07.00
C compiler	Renesas Electronics SuperH RISC engine Family C/C++ Compiler Package Ver.9.03 Release02
	Complier option -cpu=sh2afpu -fpu=single -include="$(WORKSPDIR)\inc" -object="$(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
Board used	R0K5726B0C000BR

3. Reference Application Note(s)

For additional information associated with this document, refer to the following application note(s).

- SH7268/SH7269 Group Boot From the Serial Flash Memory Using SPI Multi I/O Bus Controller (document No.: R01AN0663EJ)
- SH7268/SH7269 Group SPI Multi I/O Bus Controller Serial Flash Memory Connection Sample Program (document No.: R01AN0671EJ)
- Flash Memory Download Program for the E10A-USB Emulator Application Note (document No.: R01AN0957EJ)

4. Peripheral Functions

This chapter provides supplementary information on the SPI multi I/O bus controller. The basic information is described in the hardware manual.

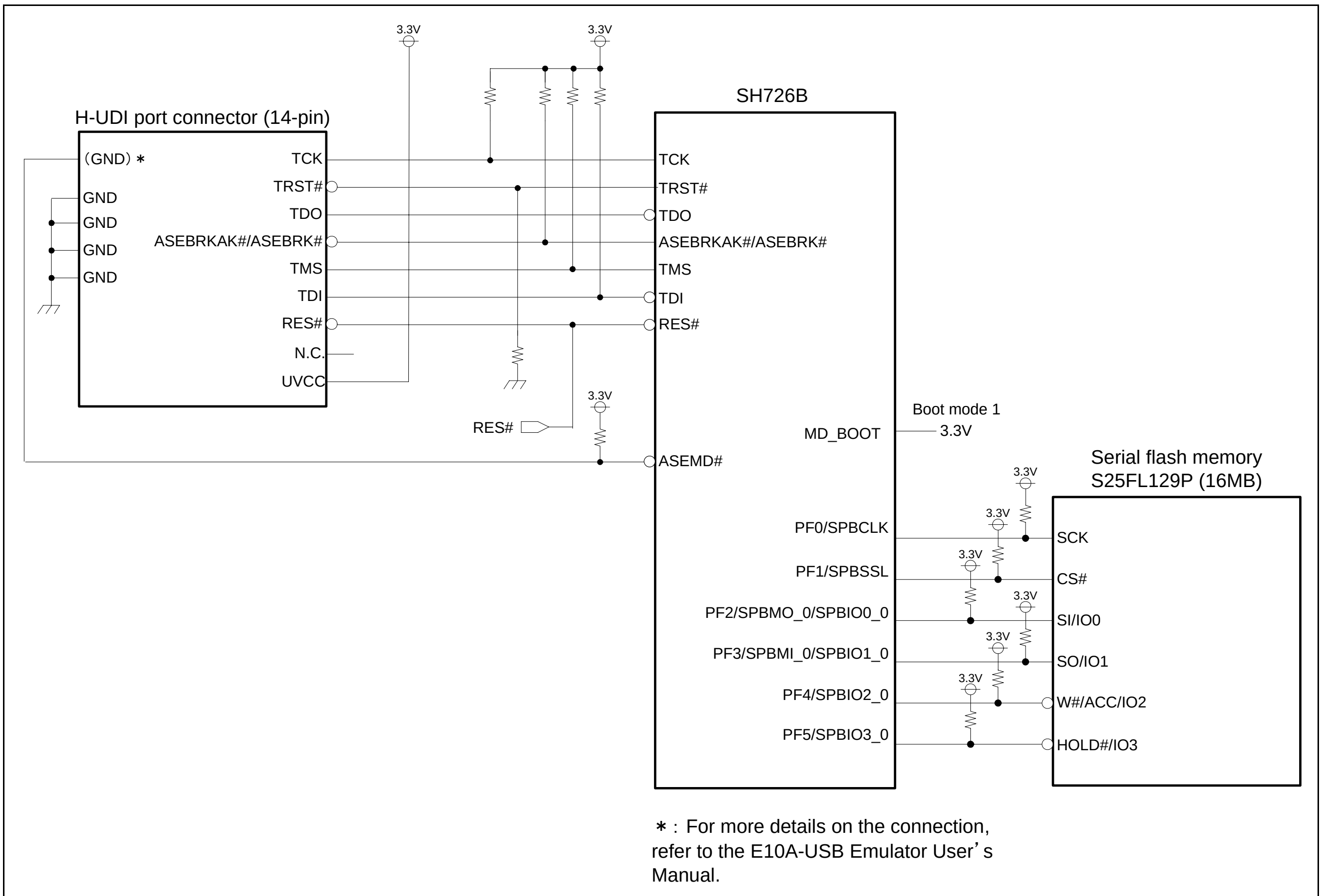
The SPI multi I/O bus controller has two modes; the SPI operation mode and the external address space read mode. The external address space read mode will be used when directly fetches the program written in the serial flash memory. The SPI operation mode will be used when erasing or writing the serial flash memory.

For details on the SPI operation mode setting procedure, refer to the application note, "SH7268/SH7269 Group SPI Multi I/O Bus Controller Serial Flash Memory Connection Sample Program (document No. R01AN0671EJ)".

5. Hardware

5.1 Hardware Configuration

Figure 5.1 shows the connection with the serial flash memory.



∗ : For more details on the connection, refer to the E10A-USB Emulator User's Manual.

Note: l "#" indicates a negative-true logic or an active low.

Figure 5.1 Connection Example

5.2 Pins Used

Table 5.1 shows the used pins and their functions.

Table 5.1 Used Pins and Functions

Pin name	Input/Output	Function
SPBCLK	Output	Clock output to the serial flash memory
SPBSSL	Output	Device selection signal output to the serial flash memory
SPBIO0_0	Input/Output	Data input/output to/from the serial flash memory (bit 0)
SPBIO1_0	Input/Output	Data input/output to/from the serial flash memory (bit 1)
SPBIO2_0	Input/Output	Data input/output to/from the serial flash memory (bit 2)
SPBIO3_0	Input/Output	Data input/output to/from the serial flash memory (bit 3)
MD_BOOT	Input	Boot mode selection
TCK	Input	Clock input from the E10A-USB emulator
TMS	Input	Mode selection from the E10A-USB emulator
TRST#	Input	Reset input from the E10A-USB emulator
TDI	Input	Data input from the E10A-USB emulator
TDO	Output	Data output to the E10A-USB emulator
ASEBRKAK#/ASEBRK#	Input/Output	Break request and response
RES#	Input	System reset signal
ASEMD#	Input	ASE mode selection

Note: "#" indicates a negative-true logic or an active low.

6. Software

6.1 Operation Overview

The FMTOOL consists of two programs; the erase module and the write module. The E10A-USB emulator writes program data in the flash memory using these programs. For details on the erase module and the write module, refer to the section "6.22 Download Function to the Flash Memory Area" in the Super H^{TM} Family E10A-USB Emulator User's Manual.

6.1.1 Batch File

Execute a reset command to initialize the SH726B using the batch file which has been started before downloading the load module. For details on the batch file and the reset command, refer to the manual listed in the integrated development environment.

6.1.2 Erase Module

Figure 6.1 shows the outline of the erase module in the FMTOOL. When downloading the load module, the FMTOOL is transmitted to the high-speed on-chip RAM on the SH726B. The erase module is executed only once after the transmission.

The erase module usually has the function for chip erase processing of the flash memory. Unlike this typical processing, the initialization of SPI multi I/O bus controller and the cancel protect setting in the flash memory are executed.

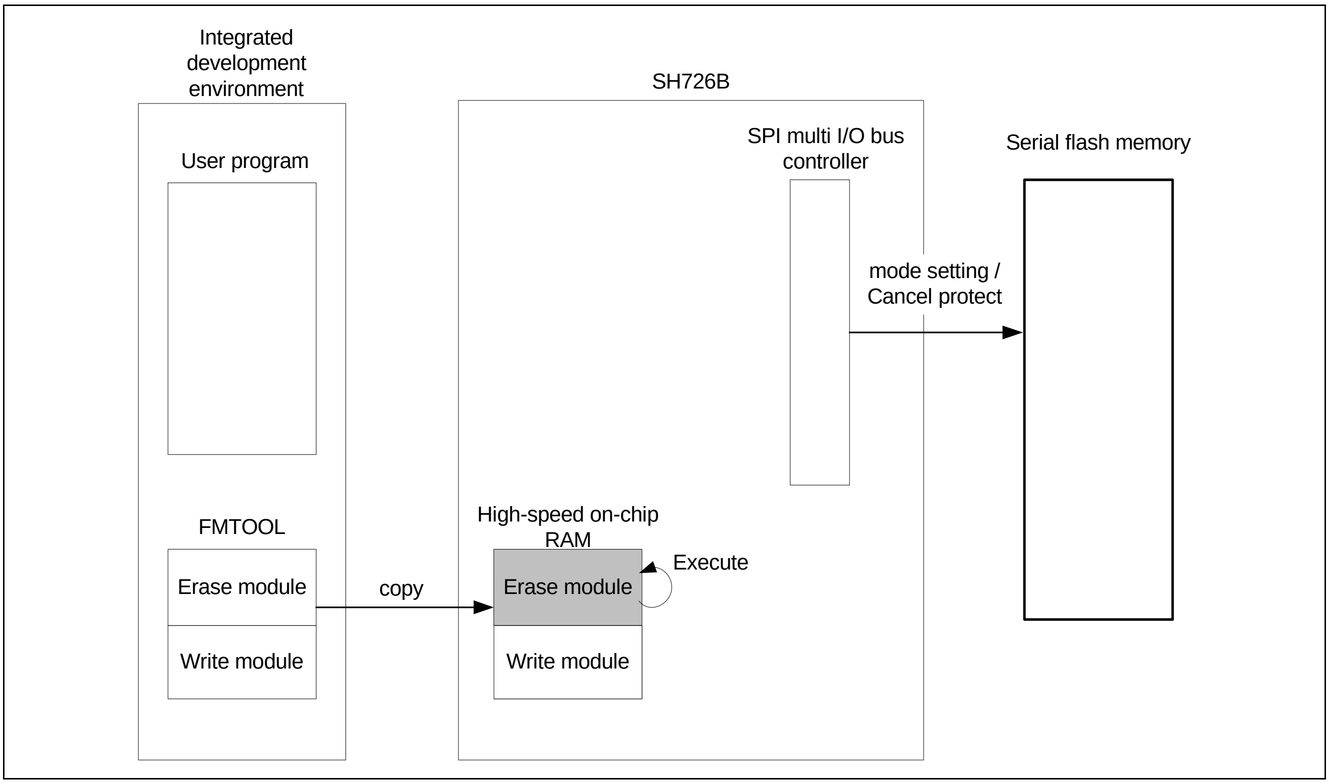


Figure 6.1 Erase Module Outline

6.1.3 Write Module

Figure 6.2 shows the outline of the write module in the FMTOOL. The write module is executed repeatedly in the high-speed on-chip RAM when downloading the load module. The write module receives the program data which are divided into the access size as an argument and writes the data to the serial flash memory after calculating the write destination address for the program data and buffering such data on a per-page basis. When the write destination address is in the undeleted sector, writes after erasing the sector.

The write destination address is calculated to make the start address in the SPI multi I/O bus space (address H'1800 0000) corresponded to the start address in the serial flash memory (address H'0000 0000).

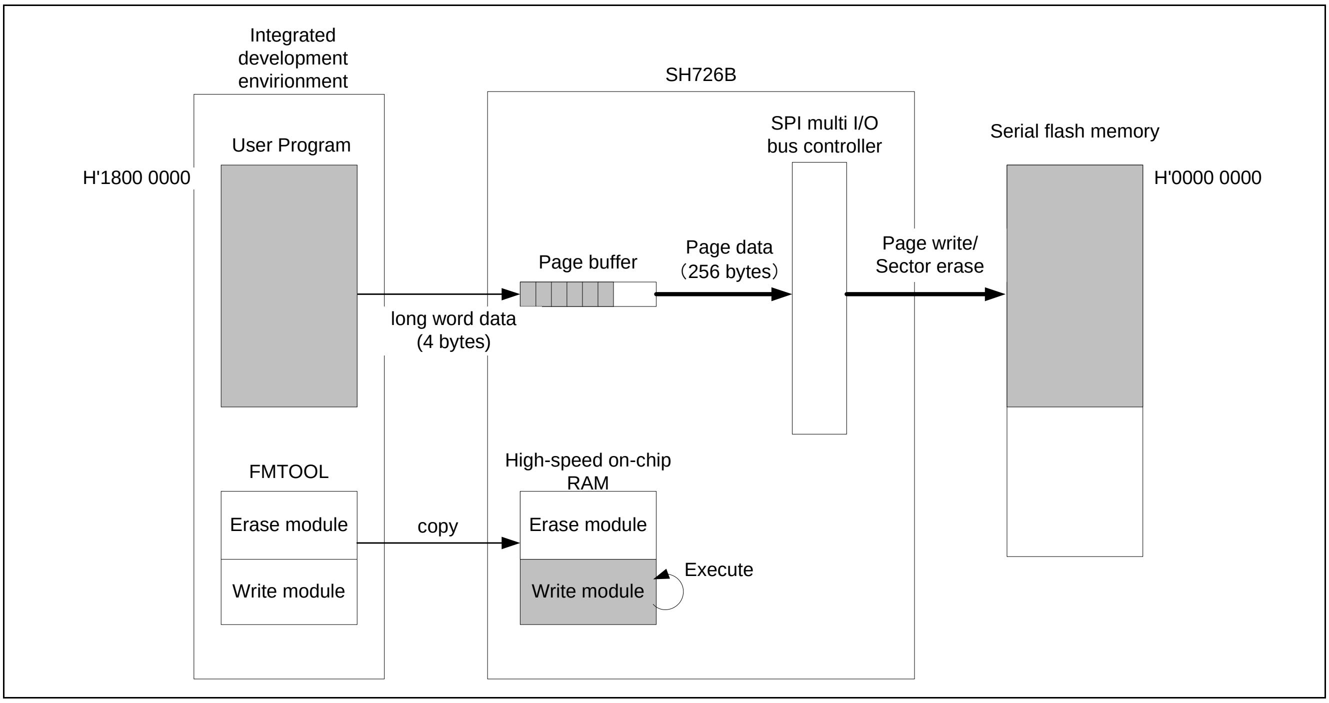


Figure 6.2 Write Module Outline

6.2 File Composition

Table 6.1 lists the file composition. The files generated by the integrated development environment should not be listed in this table.

Table 6.1 File Composition

File Name	Outline	Remarks
fmtool_entry.src	Entry module of FMTOOL	Entry of erase module and write module
fmtool_main.c	Main module of FMTOOL	
fmtool_cpg.c	Initialization for CPG	
fm_qserial_flash_spibsc.c	Serial flash memory processing	Multi I/O corresponding version
fm_io_spibsc.c	SPI multi I/O bus controller control	
qserial_flash_spibsc.h	I/F definition of fm_qserial_flash_spibsc.c	
io_spibsc.h	I/F definition of fm_io_spibsc.c	
serial_flash.h	Macro definition of the serial flash memory	
spibsc_cfg.h	Configuration file	
sh726b_spibsc_fmtool.hdc	Batch file	Registration in the integral development environment

6.3 Constants

Table 6.2 lists the constants used in the sample code.

Table 6.2 Constants Used in the Sample Code

Constant Name	Setting Value	Contents
SPI_BIT_WIDTH	4	Bit width selection for the serial flash memory
SPI_QOR_CMD	0	Does not use Quad Output Read Mode command (H’6B) Uses Quad I/O High Performance Read Mode command (H’EB)
SPI_QPP_CMD	1	Uses Quad page Program command (H’32)
SPI_QIOR_DIVIDE	1	Sets the division ratio of the SPBCLK to 1
SF_PAGE_SIZE	256	Page size (256 bytes)
PAGE_SIZE	SF_PAGE_SIZE	ditto
SF_SECTOR_SIZE	(256*1024)	Sector size (256K bytes)
SECTOR_SIZE	SF_SECTOR_SIZE	ditto
SF_REQ_PROTECT	0	Sets protect in the serial flash memory
SF_REQ_UNPROTECT	1	Cancels protect in the serial flash memory
SF_REQ_SERIALMODE	2	Specifies Serial mode in the serial flash memory
SF_REQ_QUADMODE	3	Specifies Quad mode in the serial flash memory
SR_Init	0x000000F0	Initial value of the status register
DEFAULT_VALUE	0xFFFFFFFF	Initial value of the management data used by the FMTOOL
SFLASH_ADDRESS_MASK	0xFC000000	Mask setting value to convert the SPI multi I/O bus space address to the serial flash memory address
TYPE_BYTE	0x4220	R5 parameter of write module (data access size : byte-size)
TYPE_WORD	0x5720	R5 parameter of write module (data access size: word-size)
TYPE_LONG	0x4C20	R5 parameter of write module (data access size: long-size)

6.4 Structure/Union List

Figure 6.3 shows the structure/union used in the sample code.

```
/* ==== Structure for the SPI multi I/O bus controller transfer control ==== */
typedef struct{
       /* ---- Setting value for the SPI mode enable setting register (SMENR）  ---- */
       uint32_t       cdb   :2;    /* command bit width */
       uint32_t       ocdb  :2;    /* optional command  bit width */
       uint32_t       adb   :2;    /* address bit width */
       uint32_t       opdb  :2;    /* optional data bit width */
       uint32_t       spidb :2;    /* transfer data bit width */
       uint32_t       cde   :1;    /* command enable */
       uint32_t       ocde  :1;    /* optional command enable */
       uint32_t       ade   :4;    /* address enable */
       uint32_t       opde  :4;    /* option data enable */
       uint32_t       spide :4;    /* transfer data enable */

       /* ---- Setting value for the SPI mode control register (SMCR）  ---- */
       uint32_t       sslkp :1;    /* retain the SPBSSL signal level */
       uint32_t       spire :1;    /* data read enable */
       uint32_t       spiwe :1;    /* data write enable */
       uint32_t             :5;

       /* ---- Setting value for the SPI mode command register (SMCMR) ---- */
       uint8_t        cmd;         /* command */
       uint8_t        ocmd;        /* optional command */

       /* ---- Setting value for the SPI mode address register (SMADR）  ---- */
       uint32_t       addr;

       /* ---- Setting value for the SPI mode optional setting register (SMOPR）  ---- */
       uint8_t        opd[4];      /* optional data 0~3 */

       /* ---- Setting value for the SPI mode read data register (SMRDR0,SMRDR1）  ---- */
       uint32_t       smrdr[2];

       /* ---- Setting value for the SPI mode write data register (SMWDR0,SMWDR1）  ---- */
       uint32_t       smwdr[2];

} st_spibsc_sm_t;
```

Figure 6.3 Structure/Union Used in the Sample Code

6.5 Variables

Table 6.3 lists the global variables. Table 6.4 lists the static variables

Table 6.3 Global Variables

Type	Variable Name	Contents	Function Used
st_spibsc_sm_t	SpibscSm	Setting data for the SPI multi I/O bus controller	sf_chip_erase_spibsc sf_sector_erase_spibsc sf_byte_program_spibsc sf_byte_read_spibsc read_status read_config write_enable write_status io_spibsc_transfer

Table 6.4 Static Variables

Type	Variable Name	Contents	Function Used
uint32_t	sflash_pre_erase_sctno	Management information of the erased sectors	fmtool_init, fmtool_write
uint32_t	sflash_appinfo_end	End address of the application program	fmtool_init
uint32_t	sflash_current_page	Start address in the buffering page	fmtool_init, fmtool_write
uint32_t	sflash_page_buffer[PAGE_SIZE / sizeof(int32_t)]	Page buffer	fmtool_write

6.6 Functions

Table 6.5 lists the functions.

Table 6.5 Functions

Function Name	Outline
_ERASE_ENTRY	Entry processing for erase module
_WRITE_ENTRY	Entry processing for write module
fmtool_init	Main processing for erase module (initialization)
fmtool_write	Main processing for write module (erase/write processing)
sf_bsz_get_spibsc	Serial flash memory operating function (detects the number of connected devices)
sf_bsz_set_spibsc	Serial flash memory operating function (sets the number of connected devices)
sf_allocate_cs6_spibsc	Serial flash memory operating function (sets the external address space read mode)
sf_init_serial_flash_spibsc	Serial flash memory operating function (initializes SPI multi I/O bus controller and sets mode for the serial flash memory)
sf_protect_ctrl_spibsc	Serial flash memory operating function (protect control)
sf_set_mode	Serial flash memory operating function (mode setting)
sf_chip_erase_spibsc	Serial flash memory operating function (chip erase processing)
sf_sector_erase_spibsc	Serial flash memory operating function (sector erase processing)
sf_byte_program_spibsc	Serial flash memory operating function (write processing)
sf_byte_read_spibsc	Serial flash memory operating function (read processing) *Read processing with SPI operation mode
io_set_cpg	Initial setting for the clock pulse generator

6.7 Function Specifications

The following tables list the sample code function specifications.

_ERASE_ENTRY

Outline	Entry processing for the erase module	
Header	None	
Declaration	_ERASE_ENTRY:	
Description	Allocates this function in the address H'FFF8 2000 in the entry section of the erase module. This module is activated by the E10A-USB flash memory download function. This module executes fmtool_init function after setting the stack pointer.	
Argument	R4 register	: Access size (byte: H'4220, word: H'5720, long: H'4C20)
Returned value	None	
Remarks	Described in the assembly language	

_WRITE_ENTRY

Outline	Entry processing for the write module	
Header	None	
Declaration	_WRITE_ENTRY:	
Description	Allocates this function in the address H'FFF8 2100 in the entry section of the write module. This module is activated by the E10A-USB flash memory download function. This module executes fmtool_write function after setting the stack pointer.	
Argument	R4 register	: The address where the write data are allocated
	R5 register	: Access size (byte: H'4220, word: H'5720, long: H'4C20)
	R6 register	: Write data
Returned value	R0 register is 0: normal end R0 register is 1: error end	
Remarks	Described in the assembly language	

fmtool_init

Outline	Main processing for the erase module (initialization)
Header	None
Declaration	void fmtool_init(void);
Description	Initializes the SPI multi I/O bus controller and the serial flash memory. This function is executed from the entry point of the FMTOOL (_ERASE_ENTRY).
Argument	None
Returned value	None
Remarks	

fmtool_write

Outline	Main processing for the write module (erase/write processing)	
Header	None	
Declaration	int32_t fmtool_write(uint32_t addr, int32_t access_size, uint32_t write_data, int32_t v_flag);	
Description	Executes erase and write processing for the serial flash memory. The serial flash memory is accessed by the sector for erasing and by the page for writing. This function is executed from the entry point of the FMTOOL (_WRITE_ENTRY).	
Argument	First argument: addr	: The address where write data are allocated
	Second argument: size	: Access size (byte: H'4220, word: H'5720, long: H'4C20)
	Third argument: write_data	: Write data
	Forth argument: v_flag	: Verify flag (0: not verify, 1: verify) * unused
Returned value	0: normal end -1: access size error -2: address error -3: system error	
Remarks	Only long word size is available for the access size.	

sf_bsz_get_spibsc

Outline	Serial flash memory operating function (detects the number of connected devices)	
Header	"spibsc_cfg.c", "qserial_flash_spibsc.h", "serial_flash.h", "io_spibsc.h"	
Declaration	int32_t sf_bsz_get_spibsc (void);	
Description	Returns the data bus width (the number of connected devices) to the SPI multi I/O bus controller.	
Argument	None	
Returned value	1: data bus width is 4 bits (device x 1) 2: data bus width is 8 bits (device x 2)	
Remarks		

sf_bsz_set_spibsc

Outline	Serial flash memory operating function (sets the number of connected devices)	
Header	"spibsc_cfg.c", "qserial_flash_spibsc.h", "serial_flash.h", "io_spibsc.h"	
Declaration	void sf_bsz_set_spibsc (int32_t bsz);	
Description	Sets the data bus width (the number of connected devices) to the SPI multi I/O bus controller.	
Argument	First argument: bsz	: data bus width (the number of devices connected)
Returned value	None	
Remarks		

sf_allocate_cs6_spibsc

Outline	Serial flash memory operating function (sets the external address space read mode)
Header	"spibsc_cfg.c", "qserial_flash_spibsc.h", "serial_flash.h", "io_spibsc.h"
Declaration	void sf_allocate_cs6_spibsc(void);
Description	Sets the external address space read mode to the SPI multi I/O bus controller.
Argument	None
Returned value	None
Remarks	

sf_init_serial_flash_spibsc

Outline	Serial flash memory operating function (initializes the SPI multi I/O bus controller and sets the serial flash memory mode)
Header	"spibsc_cfg.c", "qserial_flash_spibsc.h", "serial_flash.h", "io_spibsc.h"
Declaration	void sf_init_serial_flash_spibsc(void);
Description	Initializes the basic part of the SPI multi I/O bus controller. Sets the serial flash memory on Quad operation mode.
Argument	None
Returned value	None
Remarks	

sf_protect_ctrl_spibsc

Outline	Serial flash memory operating function (protect control)	
Header	"spibsc_cfg.c", "qserial_flash_spibsc.h", "serial_flash.h", "io_spibsc.h"	
Declaration	void sf_protect_ctrl_spibsc(enum sf_req req);	
Description	Sets/cancels protect for the serial flash memory.	
Argument	First argument: req	: protect request (SF_REQ_PROTECT: sets protect SF_REQ_UNPROTECT: cancels protect)
Returned value	None	
Remarks		

sf_set_mode

Outline	Serial flash memory operating function (mode setting)	
Header	"spibsc_cfg.c", "qserial_flash_spibsc.h", "serial_flash.h", "io_spibsc.h"	
Declaration	void sf_set_mode(enum sf_req_t req);	
Description	Sets mode for the serial flash memory.	
Argument	First argument: req	: protect request (SF_REQ_SERIALMODE : Dual/Serial mode SF_REQ_QUADMODE : Quad mode)
Returned value	None	
Remarks		

sf_chip_erase_spibsc

Outline	Serial flash memory operating function (chip erase processing)
Header	"spibsc_cfg.c", "qserial_flash_spibsc.h", "serial_flash.h", "io_spibsc.h"
Declaration	void sf_chip_erase_spibsc(void);
Description	Executes a chip erase for the serial flash memory. Write enable command is required before erasing or programming. Make sure that the serial flash memory is not in a busy state after erasing or programming.
Argument	None
Returned value	None
Remarks	

sf_sector_erase_spibsc

Outline	Serial flash memory operating function (sector erase processing)
Header	"spibsc_cfg.c", "qserial_flash_spibsc.h", "serial_flash.h", "io_spibsc.h"
Declaration	void sf_sector_erase_spibsc(int32_t sector_no);
Description	Executes a sector erase for the serial flash memory. The write enable command is required before erasing or programming. Make sure that the serial flash memory is not in a busy state after erasing or programming.
Argument	First argument: sector_no : sector number to be erased
Returned value	None
Remarks	

sf_byte_program_spibsc

Outline	Serial flash memory operating function (write processing)
Header	"spibsc_cfg.c", "qserial_flash_spibsc.h", "serial_flash.h", "io_spibsc.h"
Declaration	void sf_byte_program_spibsc(uint32_t addr, uint8_t * buf, int32_t size);
Description	Writes data specified by the argument in the serial flash memory. The write enable command is required before erasing or programming. Make sure that the serial flash memory is not in a busy state after erasing or programming. The maximum write data size is limited by the device.
Argument	First argument: addr : write address (the address in the serial flash memory) Second argument: buf : write data (start address in the buffer) Third argument: size : data byte count
Returned value	None
Remarks	

sf_byte_read_spibsc

Outline	Serial flash memory operating function (read processing)
Header	"spibsc_cfg.c", "qserial_flash_spibsc.h", "serial_flash.h", "io_spibsc.h"
Declaration	void sf_byte_read_spibsc(uint32_t addr, uint8_t * buf, int32_t size);
Description	Reads the specified number of bytes to the serial flash memory.
Argument	First argument: addr : read address (the address in the serial flash memory) Second argument: buf : start address in the read buffer Third argument: size : data byte count
Returned value	None
Remarks	Reads only by the 2 bytes when S-Flash x 2

io_set_cpg

Outline	Initial setting for the clock pulse generator
Header	None
Declaration	void io_set_cpg(void);
Description	Sets the system clock and allows clock supply to the peripheral module.
Argument	None
Returned value	None
Remarks	

6.8 Flowcharts

This section describes the procedure of major functions used in the sample code.

6.8.1 Erase Module

Figure 6.4 shows the flowchart of the erase module.

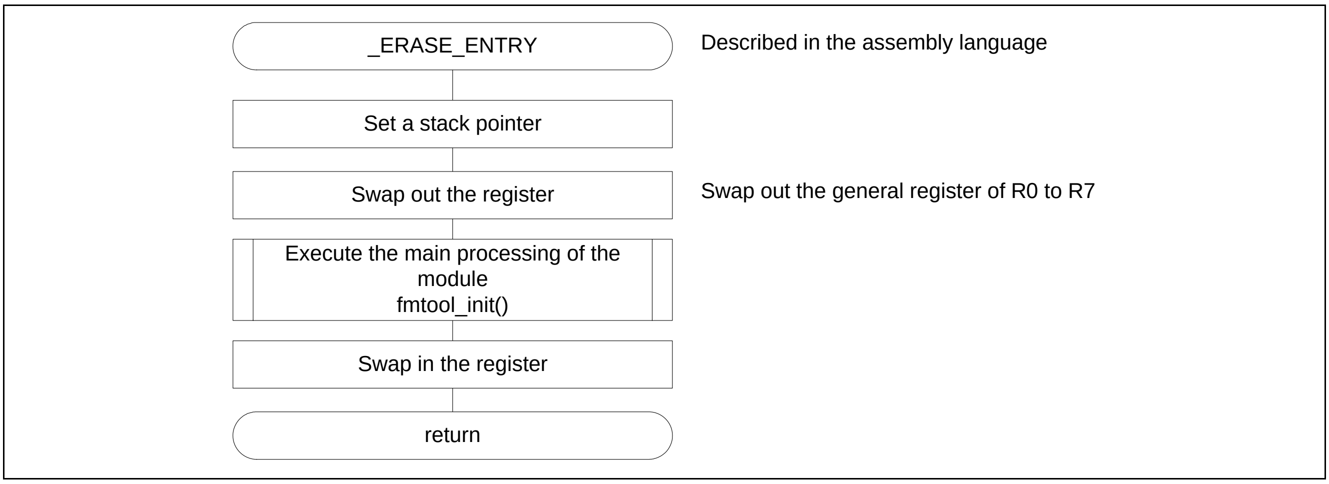

_ERASE_ENTRY
Described in the assembly language
Set a stack pointer
Swap out the register
Swap out the general register of R0 to R7
Execute the main processing of the module fmtool_init()
Swap in the register
return

Figure 6.4 Erase Module

6.8.2 Write Module

Figure 6.5 shows the flowchart of the write module.

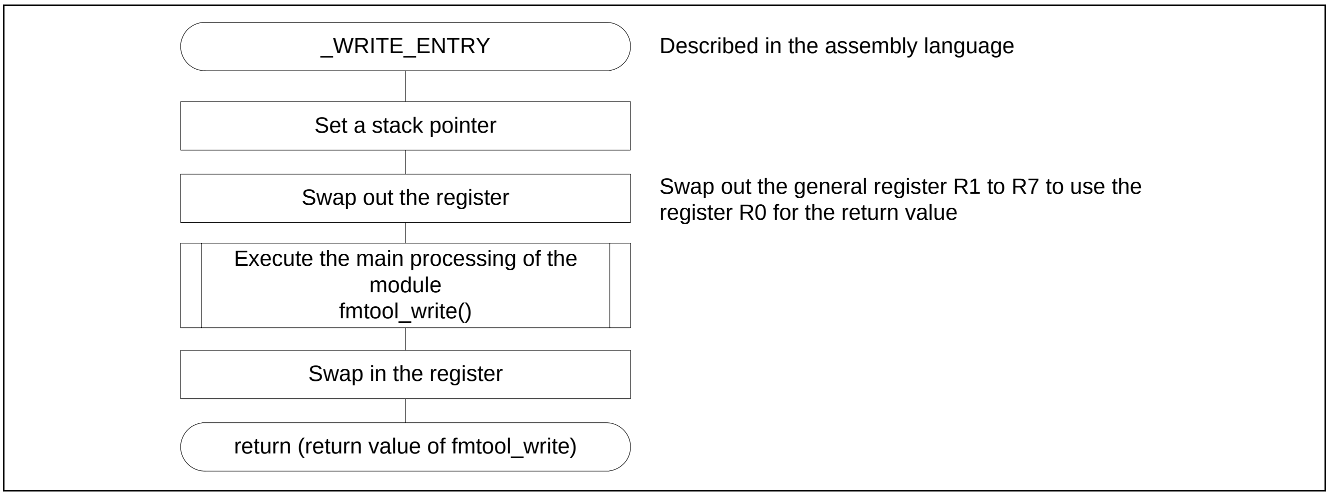

_WRITE_ENTRY
Described in the assembly language
Set a stack pointer
Swap out the register
Swap out the general register R1 to R7 to use the register R0 for the return value
Execute the main processing of the module fmtool_write()
Swap in the register
return (return value of fmtool_write)

Figure 6.5 Write Module

6.8.3 Initialization of the FMTOOL

Figure 6.6 shows the flowchart of the initialization of the FMTOOL.

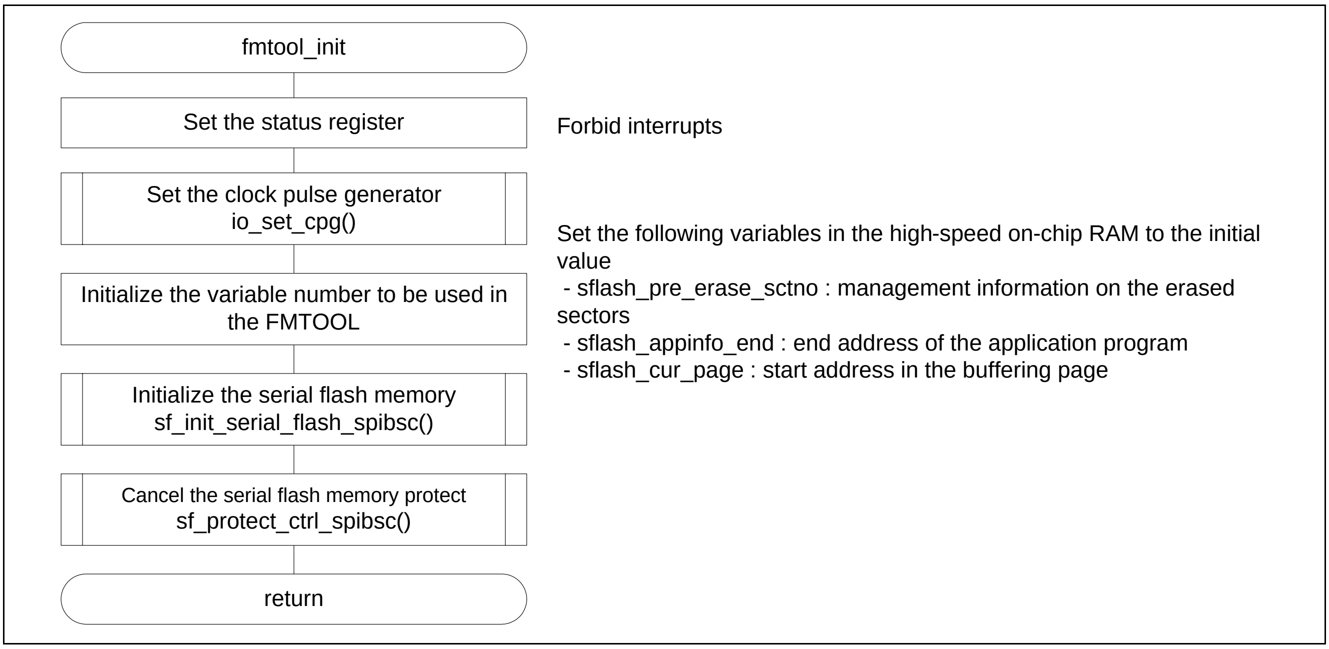

fmtool_init
Set the status register
Forbid interrupts
Set the clock pulse generator
io_set_cpg()
Initialize the variable number to be used in the FMTOOL
Set the following variables in the high-speed on-chip RAM to the initial value
- sflash_pre_erase_sctno : management information on the erased sectors
- sflash_appinfo_end : end address of the application program
- sflash_cur_page : start address in the buffering page
Initialize the serial flash memory
sf_init_serial_flash_spibsc()
Cancel the serial flash memory protect
sf_protect_ctrl_spibsc()
return

Figure 6.6 Initialization of FMTOOL

6.8.4 Write Processing for the Flash Memory

Figure 6.7 shows the flowchart of the write processing for the flash memory.

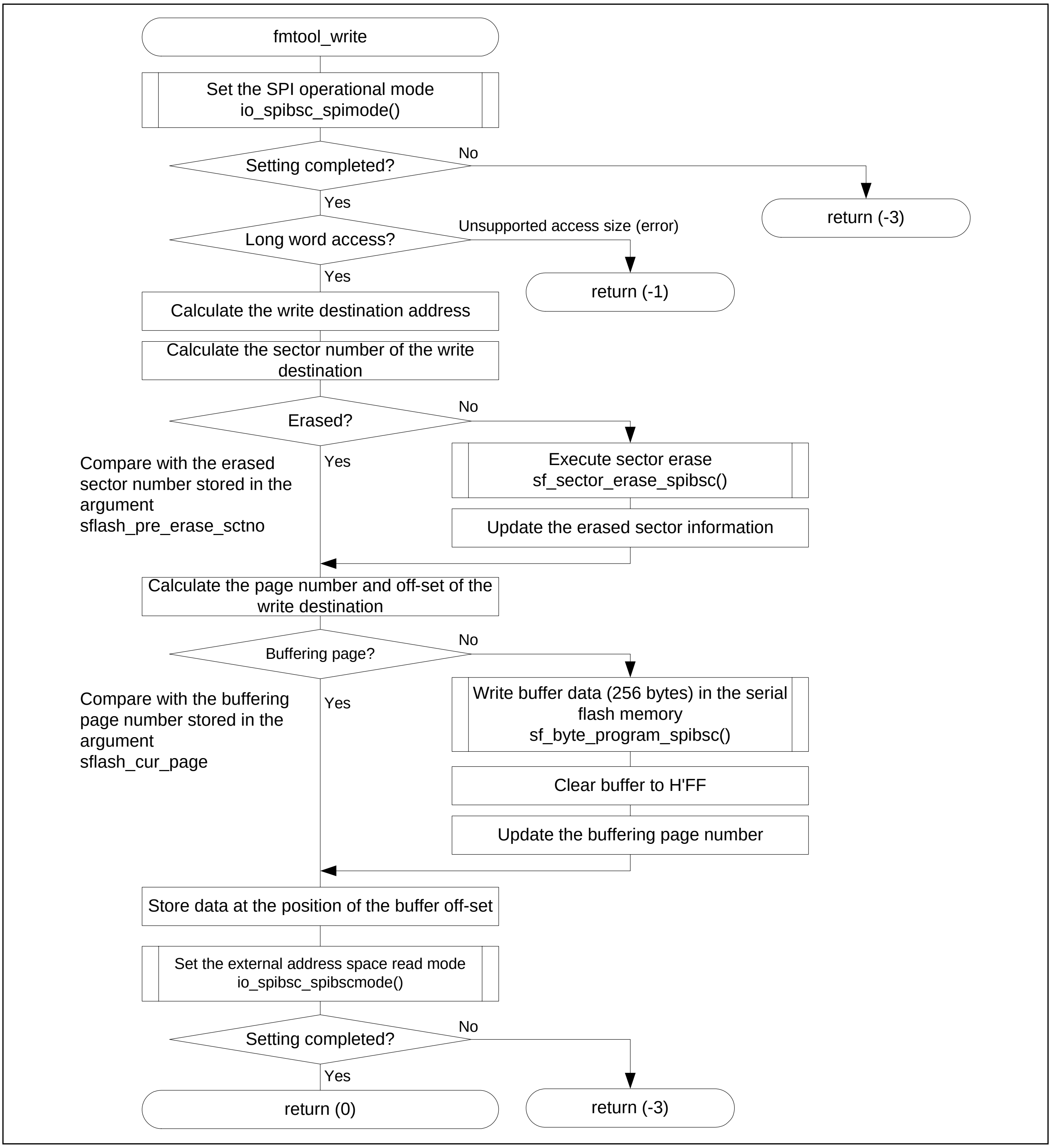

fmtool_write
Set the SPI operational mode
io_spibsc_spimode()
Setting completed?
No
return (-3)
Yes
Long word access?
Unsupported access size (error)
return (-1)
Yes
Calculate the write destination address
Calculate the sector number of the write destination
Erased?
No
Compare with the erased sector number stored in the argument sflash_pre_erase_sctno
Yes
Execute sector erase
sf_sector_erase_spibsc()
Update the erased sector information
Calculate the page number and off-set of the write destination
Buffering page?
No
Compare with the buffering page number stored in the argument sflash_cur_page
Yes
Write buffer data (256 bytes) in the serial flash memory
sf_byte_program_spibsc()
Clear buffer to H'FF
Update the buffering page number
Store data at the position of the buffer off-set
Set the external address space read mode
io_spibsc_spibscmode()
Setting completed?
No
Yes
return (0)
return (-3)

Figure 6.7 Write Processing for the Flash Memory

6.9 Basic Precautions

6.9.1 Adding Dummy Data to the Load Module

The FMTOOL buffers and writes the data by the page to improve its writing speed to the serial flash memory. Writing to the serial flash memory is performed at the time that the page address which differs from the one in the buffering page is assigned. Consequently, the data of the last page stays in the buffer and may not be written in the serial flash memory. Assigns dummy data in the last page of the load module to avoid leaving the valid data in the buffer.

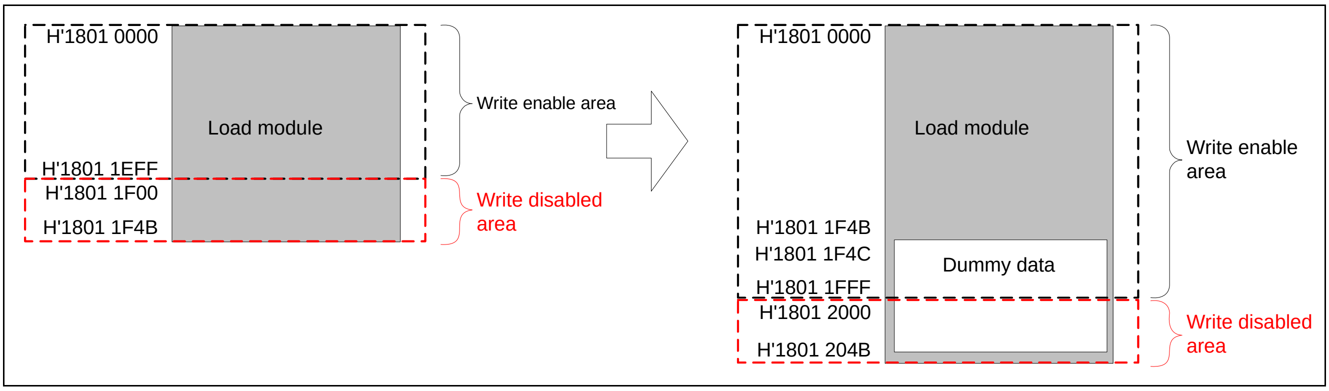
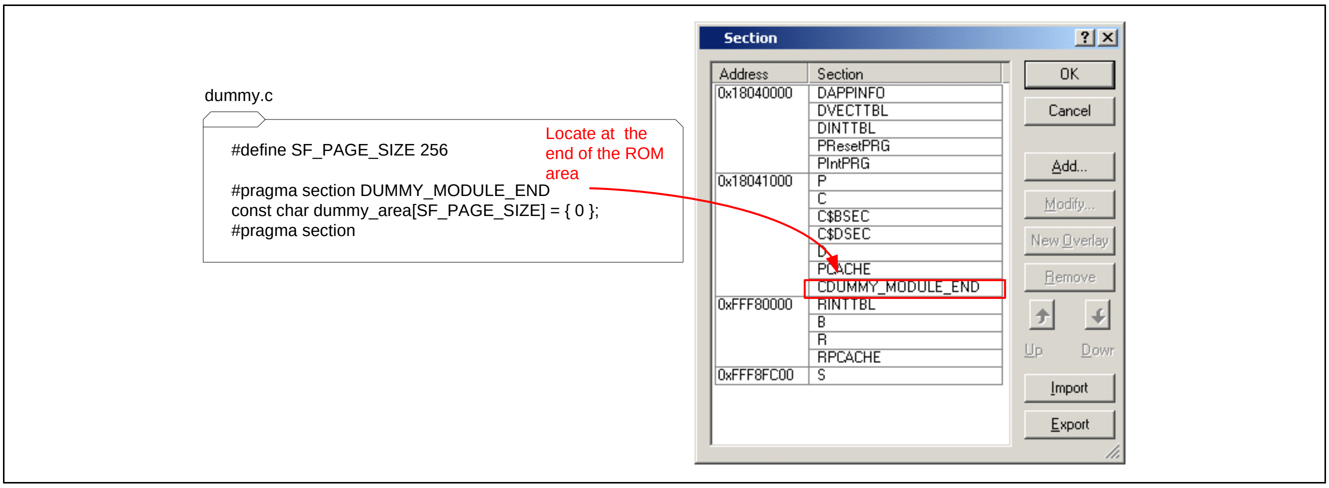


Figure 6.8 Write Disabled Area in Load Module

Figure 6.9 shows an example for adding dummy data to the section. Define the constant data of 256 bytes in the provided dummy section (CDUMMY_MODULE_END) and allocate it at the end of the ROM area.



dummy.c

```
#define SF_PAGE_SIZE 256

#pragma section DUMMY_MODULE_END
const char dummy_area[SF_PAGE_SIZE] = { 0 };
#pragma section
```

Locate at the end of the ROM area

Section

Address	Section
0x18040000	DAPPINFO
	DVECTTBL
	DINTTBL
	PResetPRG
	PIntPRG
0x18041000	P
	C
	C$BSEC
	C$DSEC
	D
	PCACHE
	CDUMMY_MODULE_END
0xFFF80000	RINTTBL
	B
	R
	RPCACHE
0xFFF8FC00	S

Figure 6.9 Example of Adding Dummy Data

6.9.2 Forbidding Sharing Sectors between the Load Modules

Figure 6.10 shows the operation under the assumption that two load module share one sector. Although it is possible to compose a user program downloaded by the FMTOOL of multiple load modules, sharing one sector between the load modules is not allowed. When downloading multiple data in one sector, the previously downloaded data accidentally will be erased.

The mentioned load module area includes the dummy data area described in the section 6.9.1.

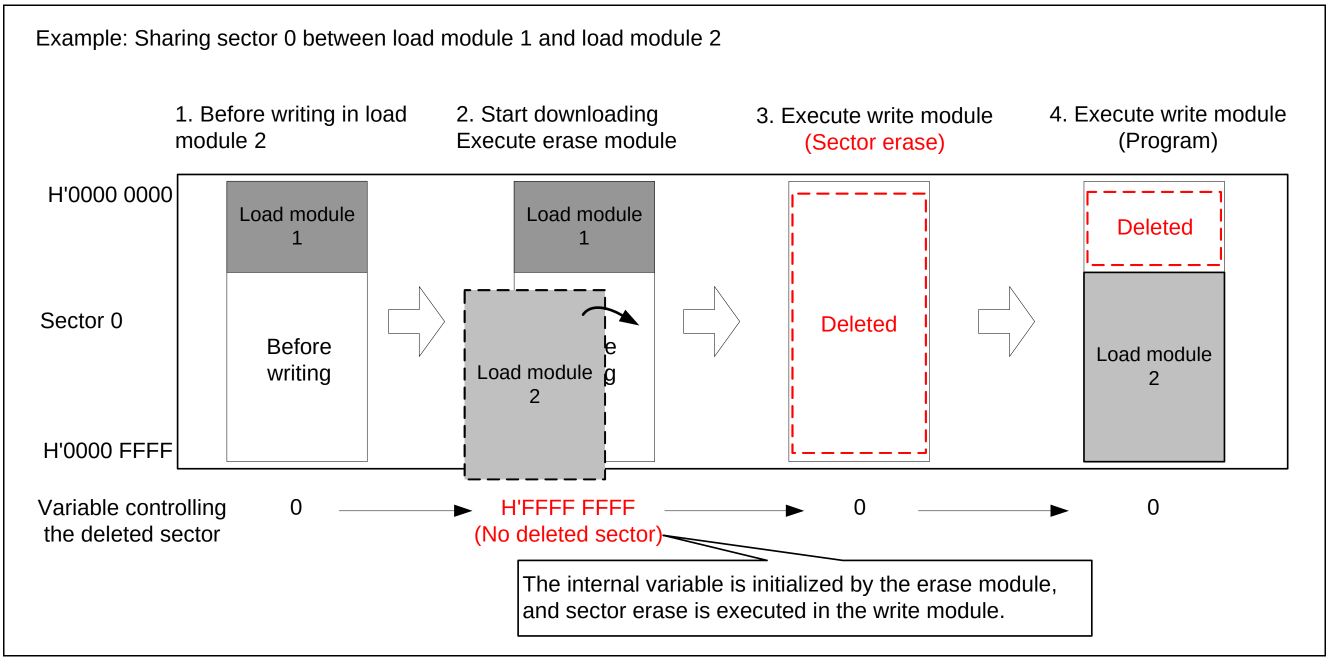

Example: Sharing sector 0 between load module 1 and load module 2
1. Before writing in load module 2
2. Start downloading Execute erase module
3. Execute write module (Sector erase)
4. Execute write module (Program)
H'0000 0000
Sector 0
H'0000 FFFF
Load module 1
Before writing
Load module 1
Load module 2
Deleted
Deleted
Load module 2
Variable controlling the deleted sector
0
H'FFFF FFFF (No deleted sector)
0
0
The internal variable is initialized by the erase module, and sector erase is executed in the write module.

Figure 6.10 Operation when Sharing A Sector between Load Modules

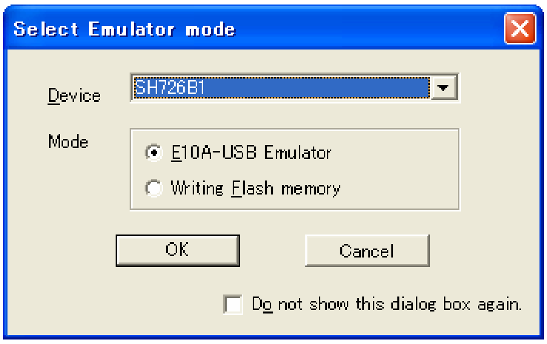
7. Application Example

7.1 Procedure of User Program Download

This section describes the procedure of downloading user programs to the serial flash memory using the created FMTOOL (sh726b_spibsc_fmtool.mot).

7.1.1 Prepare for the Download Environment

1. Connect user’s system with the E10A-USB emulator conned to PC.
2. Start the High-performance Embedded Workshop to open the work space for user programs.
3. The CPU select dialog box is open as shown in Figure 7.1
   Select the CPU in use from the drop-down listbox for Device. Click the OK button.



Select Emulator mode

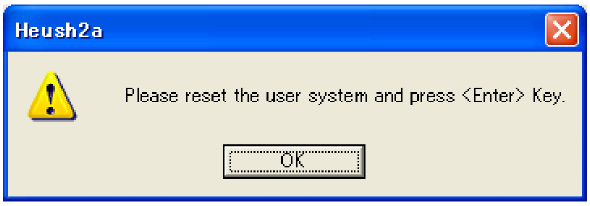
Device: SH726B1

Mode:
● E10A-USB Emulator
○ Writing Flash memory

Note: The shown window is an example agopting the SH726B1

Figure 7.1 CPU Select Dialog Box

4. The Connecting dialog box is displayed, and starts connecting the emulator.
   The reset signal request dialog box shown in Figure 7.2 is displayed.



Heush2a

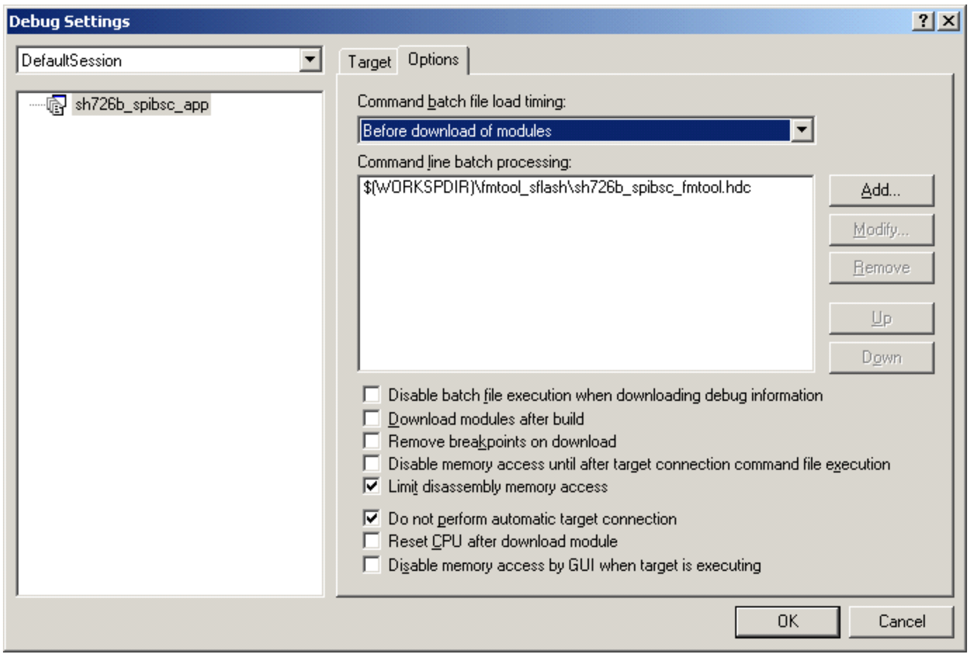
Please reset the user system and press <Enter> Key.

Figure 7.2 Reset Signal Request Dialog Box

5. Turn on the user’s system.
   Input the RESET signal from the user’s system, click the OK button.
   When "connected" is displayed on the Output Window in the High -performance Embedded Workshop, the E10A-USB emulator successfully started.

7.1.2 Registering a Batch File

1. Select in the menu; [Debug] → [Debug Settings]
2. The debug setting window shown in Figure 7.3 opens.
3. Select "Before download modules" in the pull-down menu of "Command batch file load timing".
4. Click the "Add" at "Command line batch processing" to add a batch file.
5. Click the OK button, and registration is completed.



DefaultSession

sh726b_spibsc_app

Command batch file load timing: Before download of modules

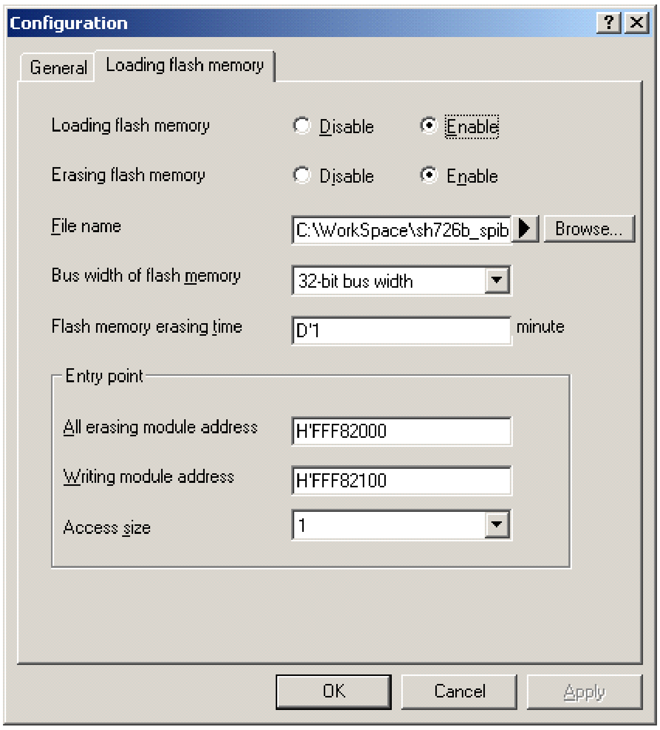
Command line batch processing: $(WORKSPDIR)\fmtool_sflash\sh726b_spibsc_fmtool.hdc

- ☐ Disable batch file execution when downloading debug information
- ☐ Download modules after build
- ☐ Remove breakpoints on download
- ☐ Disable memory access until after target connection command file execution
- ☑ Limit disassembly memory access
- ☑ Do not perform automatic target connection
- ☐ Reset CPU after download module
- ☐ Disable memory access by GUI when target is executing

Figure 7.3 Window for Debug Setting

7.1.3 Setting Configuration Dialog Box

Figure 7.4 shows the "Configuration" dialog box for setting to download a user program to the external flash memory using the E10A-USB emulator.



Loading flash memory ○ Disable ● Enable

Erasing flash memory ○ Disable ● Enable

File name: C:\WorkSpace\sh726b_spib

Bus width of flash memory: 32-bit bus width

Flash memory erasing time: D'1 minute

Entry point

All erasing module address: H'FFF82000

Writing module address: H'FFF82100

Access size: 1

Figure 7.4 Configuration Dialog Box (in the page of Loading flash memory)

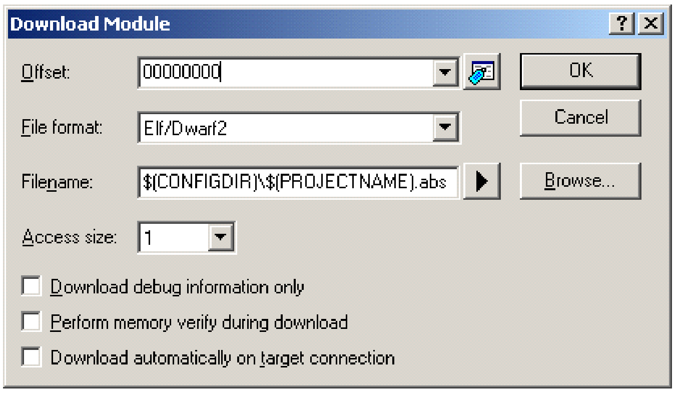
Table 7.1 lists the setting for each item. Finish setting and click the OK button, configuration is completed.

Table 7.1 Setting Value in the Configuration Dialog Box

Item	Setting Value
Loading flash memory	Enable
Erasing flash memory	Enable
File Name	\sh726b_spibsc_fmtool.mot
Bus width of flash memory	32-bit bus width
All erasing module address	Specify the start address of erase module (H'FFF8 2000)
Writing module address	Specify the start address of write module (H'FFF8 2100

7.1.4 Adding the Download Module

Open the debug setting window from the debug menu and click ″Add″. In the download module window shown in Figure 7.5, add the user program (which is to be loaded in the serial flash memory) to the download module.



Offset: 00000000

File format: Elf/Dwarf2

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

Access size: 1

☐ Download debug information only

☐ Perform memory verify during download

☐ Download automatically on target connection

Figure 7.5 Download Module Window

7.1.5 Downloading User Programs

Using the download function shown in Figure 7.6, download the user programs.

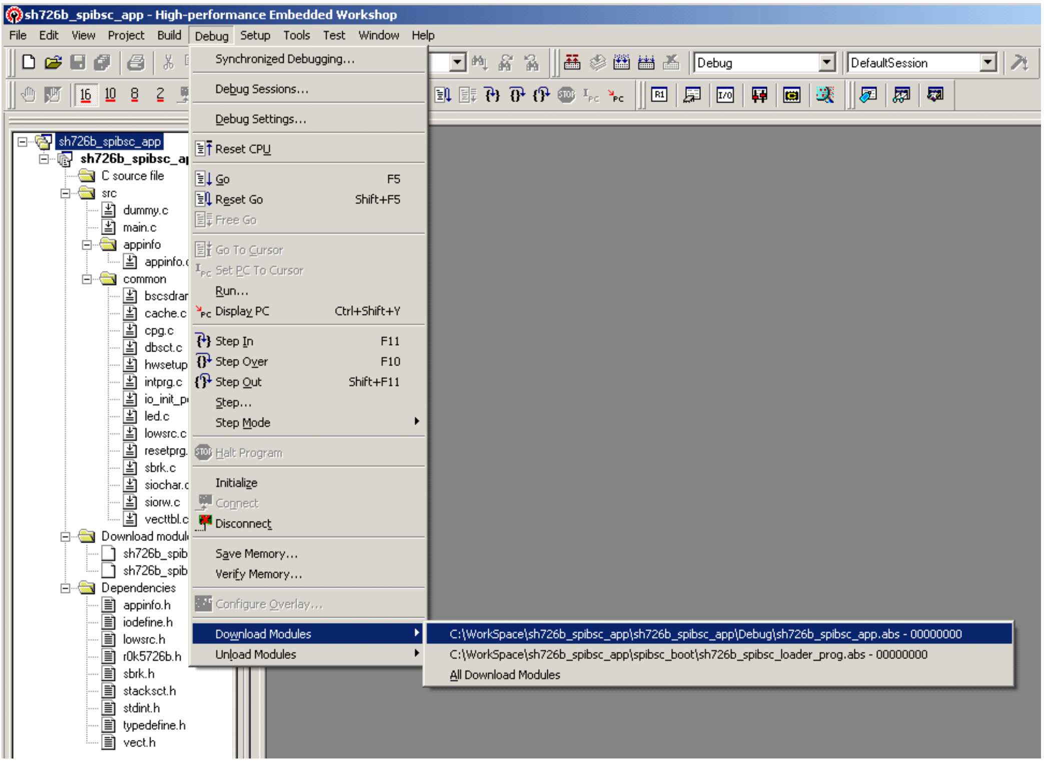


Figure 7.6 Downloading User Programs

7.2 Application to Serial Flash Boot

In this application note, the function to boot from the serial flash memory is called the "serial flash boot". For details on the serial flash boot, refer to "SH7268/SH7269 Group Boot From the Serial Flash Memory Using SPI Multi I/O Bus Controller (document No.:R01AN0663EJ)".

The changes when replacing the downloader, a flash write tool for the above application note (R01AN0663EJ), to the FMTOOL are described in this section.

7.2.1 Section Assignment

Figure 7.7 shows the section allocation when using the FMTOOL. Assign a loader program and application program with attention to the following points.

- Place sections in the SPI multi I/O bus space.
- Do not share one sector between different load modules. Example: placing the application program in the address of H'1801 0000. Not in H'1800 2000.)
- Map the loader program in the address of H'FFF8 0000 by using the optimizing linkage editor option (the section for mapping from ROM to RAM).

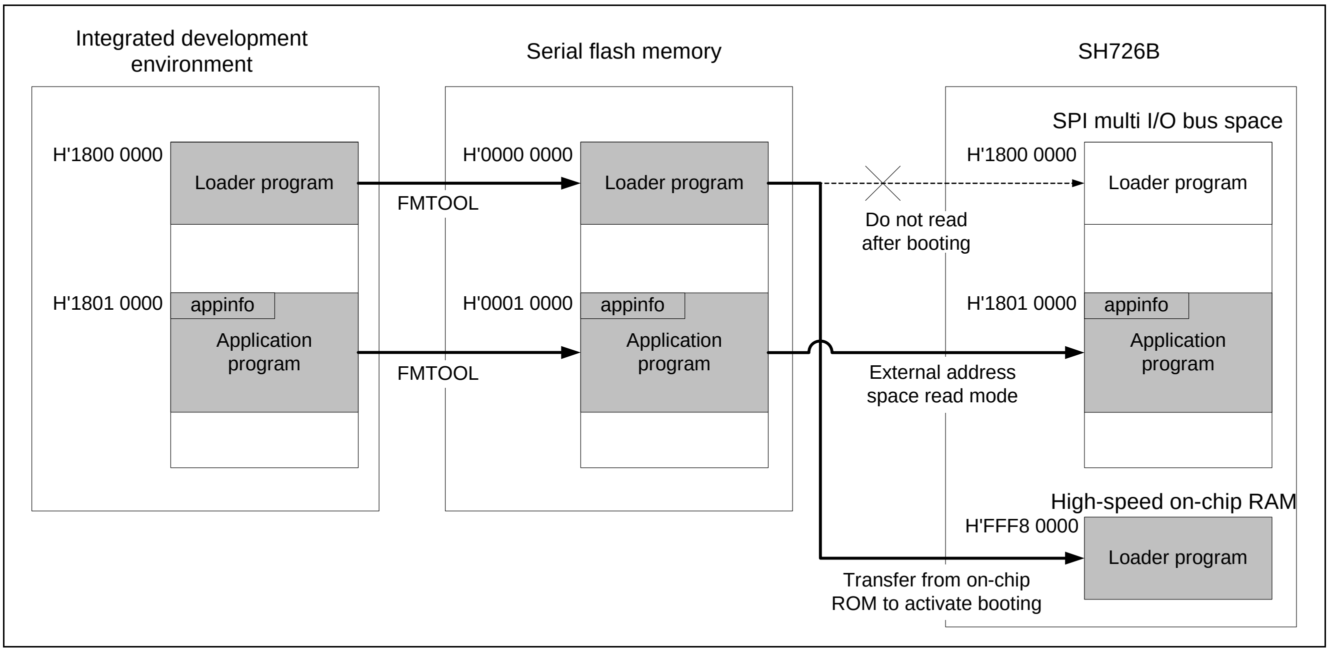


Figure 7.7 Section Allocation with FMTOOL

7.2.2 Adding Dummy Data

Make sure to add dummy data to the loader program and the application program as described in the section "6.9.1 Adding Dummy Data to the Load Module".

7.2.3 Downloading the Load Module

The operation procedure of the integrated development environment to download the load module is also changed. Download the procedure according to the section "7.1 Procedure of User Program Download".

7.3 Customizing FMTOOL

The sample code is dependent on the specification of the device in the serial flash memory. Customization of the program may be necessary when changing the device.

7.3.1 Device Specification Capable for Sample Code

Table 7.2 and Table 7.3 list the specification of the used device and the commands used in the sample code respectively.

Table 7.2 Specification of the Used Device

Item	Description
Manufacturer	Spansion Inc.
Model	S25FL129P0XMFI01
Capacity	16M bytes
Interface	SPI multi I/O bus (Single/Dual/Quad mode)
Access time	104MHz (Single mode), 80MHz (Dual/Quad mode)
Sector structure	Uniform
Sector size	256K bytes
Page size	256 bytes

Table 7.3 Commands Used in the Sample Code

Item	Description
Erase command	H'D8 (sector erase)
Program command	H'32 (Quad page programming)

7.3.2 Contents of Customization

Table 7.4 lists the necessary customizations and their contents.

Table 7.4 Necessary Customization and the Contents

Cases	Content
Quad mode is not available. (Operable in Single mode)	Alter the macro SPI_BIT_WIDTH setting value to 1
Sector size is improper. (not suitable for 256K-byte sector erase)	For the Uniform type sector structure, alter the setting value of macro SF_SECTOR_SIZE to the new sector size. Change the sector erase command used in sf_sector_erase_spibsc function to the command that supports the new sector size. For the Top or Bottom type structure, the algorithm to discriminate sector number in fmtool_write function should also be altered.
Procedure for device initialization is different.	Customization is required for the serial flash memory operation function and the SPI multi I/O bus controller control function. For details, refer to the sample code.
The command in Table 7.3 is unusable	
Electric characteristics are different.	

Note: The FMTOOL is flash memory specification dependent. Therefore the items in Table 7.4 do not cover all the cases. Check the data sheet and modify the FMTOOL according to the specification in it.

8. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

9. Reference Documents

Hardware Manual

SH726A/SH726B 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.

C Compiler Manual

SuperH RISC Engine Family C/C++ Compiler Package V.9.04
C Compiler User’s Manual Rev.1.01
The latest version can be downloaded from the Renesas Electronics website.

SuperH Family E10A-USB Emulator User’s Manual Rev.9.00
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	SH726A/SH726B Group Application Note E10A-USB Flash Memory Download Function (Download to Serial Flash Memory)

Rev.	**Date**	**Description**	
		Page	**Summary**
1.00	**Jun. 25, 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: +44-1628-585-100, Fax: +44-1628-585-900

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

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

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

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

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

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

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

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

Colophon 2.0