RZ/T1 Group

Example of Downloading to Serial Flash Memory by Using “Semihosting” of ARM® Development Studio 5 (DS-5™)

R01AN2951EJ0120
Rev.1.20
Sep. 15, 2017

Summary

This application note presents a method for downloading programs to the serial flash memory allocated to the external address space (SPI multi-I/O bus space) of an RZ/T1 group microcontroller.

Note that the method of downloading described in this application note utilizes the “semihosting” (file operation) functionality of ARM® Development Studio 5 (DS-5™, hereafter abbreviated as DS-5). You will need to obtain DS-5 separately. For details of the semihosting functionality of DS-5, refer to the documentation*1 provided by ARM®.

Note 1. Refer to “ARM® Compiler toolchain Developing Software for ARM® Processors, Semihosting” for details.

Applicable Devices

RZ/T1 Group

When applying the program covered in this application note to another microcontroller, modify the program according to the specifications for the target microcontroller and conduct an extensive evaluation of the modified program.

Table of Contents

1. Specifications

The serial flash memory is a type of nonvolatile memory typically used to store program codes and data. Writing to the serial flash memory requires an appropriate algorithm for the serial flash memory in use. This application note presents such an algorithm as a C-language program that runs in the tightly-coupled memory (specifically, the ATCM) of an RZ/A1H group microcontroller. It also describes how to use the semihosting functionality of DS-5 to refer to binary files for SPIBSC initial settings and for applications*1 which are stored on the hard disk of the host computer on which DS-5 is running, and to write them to the serial flash memory.

Table 1.1 lists the peripheral modules used and their applications.

Note 1. See Table 5.1 for details of binary files for SPIBSC initial settings and for applications.

Table 1.1 Peripheral Modules and Their Applications

Peripheral Module	Application
SPI multi-I/O bus controller (SPIBSC)	• This is used to generate signals for use in access to the serial flash memory connected to the external address space (SPI multi-I/O bus space).
ARM® Development Studio 5 (DS-5™) “semihosting” functionality	• Semihosting is used to have code running on the target (the program running on the board) handle transfer to and from the I/O functions of the host computer on which the debugger is running. • This is used to refer to the terminal output from the target to the application console of DS-5 and to the handling of application binary files stored on the hard disk of the host computer.

2. Conditions for Checking Operations

Operation of the sample program covered in this application note has been confirmed under the conditions below.

Table 2.1 Conditions for Checking Operations

Item	Description
MCU used	RZ/T1 Group
Operating frequency	CPUCLK = 450 MHz, PCLKA = 150 MHz
Operating voltage	3.3 V
Integrated development environment	DS-5 Version 5.25.0 from ARM®
Operating mode	SPI boot mode (serial flash memory)
Board used	RZ/T1 evaluation board (RTK7910022C00000BR)
Devices used (functions to be used on the board)	Serial flash memory allocated to the SPI multi-I/O bus space (1- or 4-bit bus width) • Manufacturer: Macronix International Co., Ltd. • Product type number: MX25L51245GMI-10G

3. Related Application Notes

The application notes related to the descriptions in this application note are listed below. Also consult the following documents along with this application note.

- RZ/T1 Group Initial Settings (R01AN2554EJ)
- RZ/T1 Group Initial Settings of the Microcomputers Incorporating the R-IN Engine (R01AN2989EJ)

4. Description of Hardware

4.1 List of Pins

Table 4.1 lists the pins used and their functions.

Table 4.1 Pins Used and Their Functions

Pin Name	I/O	Description
SPBCLK_0	Output	Clock output
SPBSSL_0	Output	Slave select
SPBMO0_0/SPBIO00_0	I/O	Master send data: data 0
SPBMI0_0/SPBIO10_0	I/O	Master input data: data 1
SPBIO20_0	I/O	Data 2
SPBIO30_0	I/O	Data 3
MD2, MD1, MD0	Input	Selection of boot mode (set to SPI boot mode) MD2: “L” MD1: “L” MD0: “L”
TCK	Input	Clock input from the ARM® emulator
TMS	Input	Mode selection from the ARM® emulator
TRST#	Input	Reset input from the ARM® emulator
TDI	Input	Data input from the ARM® emulator
TDO	Output	Data output to the ARM® emulator
RES#	Input	System reset signal

Note: Symbol # represents a negative logic (or active low).

4.2 Reference Circuit

Figure 4.1 is a connection example.

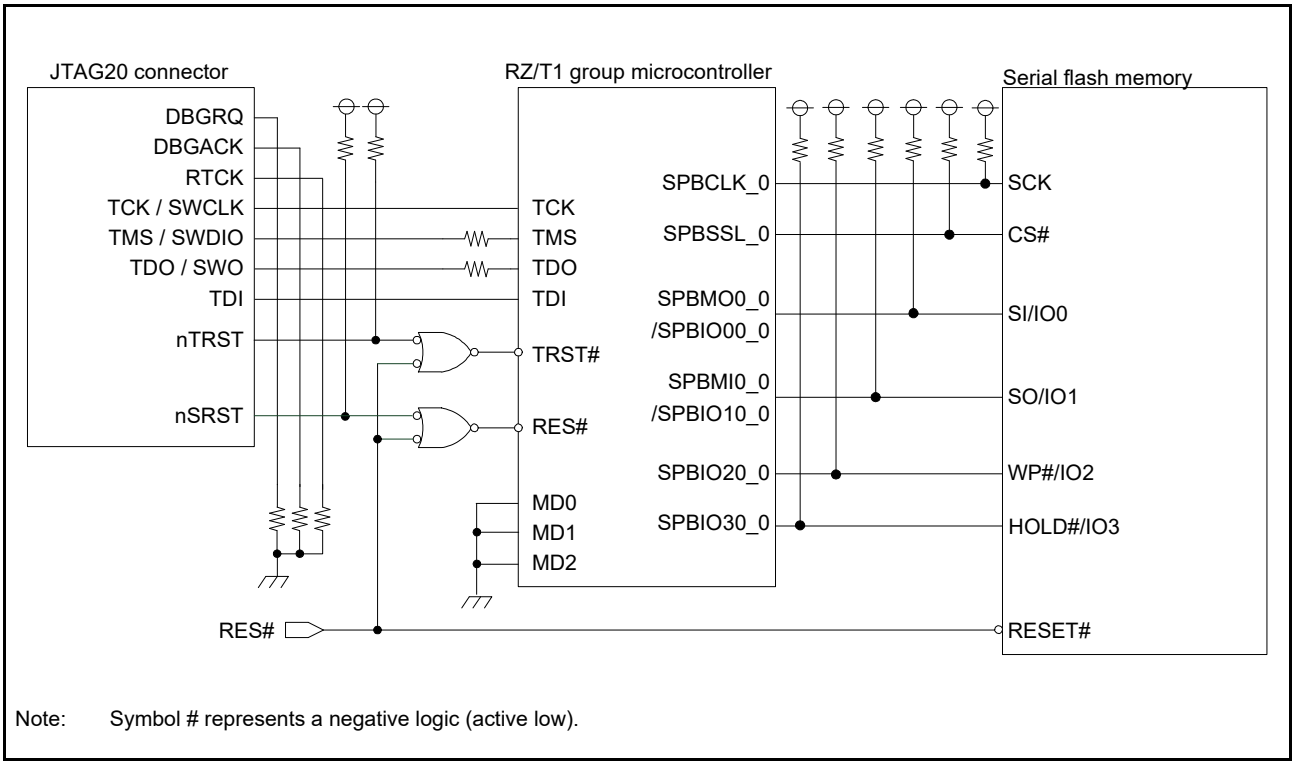


Figure 4.1 Connection Example

5. Outline of Downloading to the Serial Flash Memory

This section gives an outline of downloading to the serial flash memory.

5.1 Terms Related to Downloading to the Serial Flash Memory

Table 5.1 lists the terms related to downloading to the serial flash memory that are used in this application note.

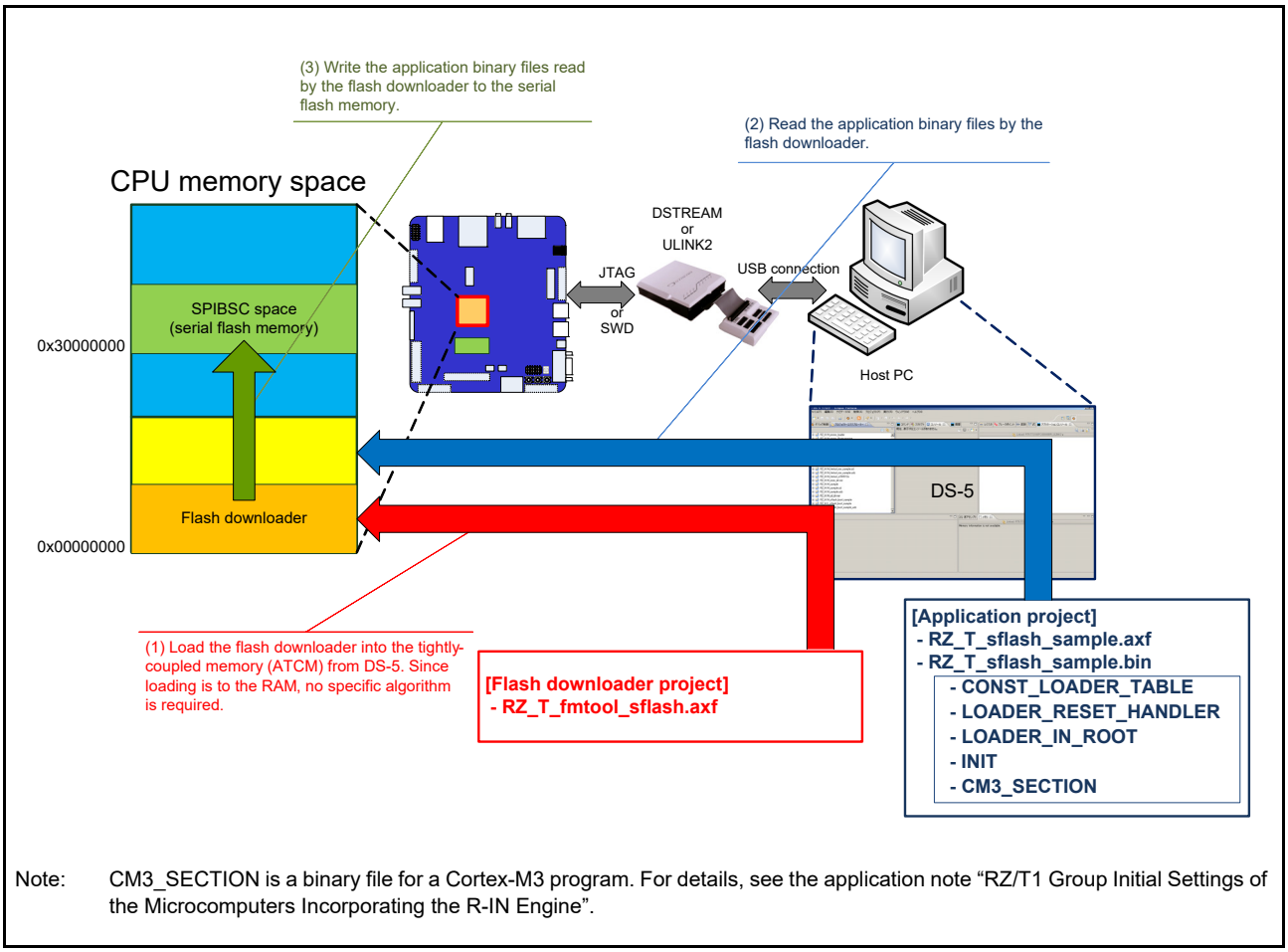
Table 5.1 Terms Related to Downloading to the Serial Flash Memory

Term	Description
Application program	The application program is a program which is created by the customer to suit the system.
Flash downloader	The flash downloader is a program for writing the SPI multi-I/O bus controller initial settings program and application program to the serial flash memory. Customers should use this application note for reference and create flash downloader programs to match their systems.
Semihosting	Semihosting is a mechanism where I/O request code running on an ARM® CPU uses the I/O functions of DS-5 through transfer to and from a debugger. Running standard C language functions such as printf, scanf, etc. on the ARM® CPU allows I/O processing on the screen and keyboard of the host PC through the I/O functions of DS-5 rather than through the I/O functions in the ARM® CPU on the target system. For details, see the document provided by ARM®.
Application project	This project is used to generate an application program executable file (axf file) in DS-5. The application program includes parameter information for the loader to be referred to by the RZ/T1 group microcontroller and the loader program itself.
Flash downloader project	This project is used to generate a flash downloader executable file (axf file) in DS-5. The application program includes parameter information for the loader to be referred to by the RZ/T1 group microcontroller and the loader program itself.
Application binary file	The application binary file is a data file containing the application program to be written to the serial flash memory. A binary file generator tool (fromelf.exe)*1 is used to generate this file from the application program executable file (axf file) that is generated when the application project is built in DS-5.

Note 1. The binary file generator tool is included in DS-5. For details, see “ARM® DS-5™ DS-5 Getting Started Guide, ARM DS-5 Product Overview” provided by ARM®.

5.2 Schematic View of Flash Downloader Operation

Figure 5.1 is a schematic view of the operation of the flash downloader. The flash downloader runs in the tightly-coupled memory (ATCM) of the RZ/T1 group microcontroller; it uses semihosting to refer to the application binary files stored on the hard disk of the host computer on which DS-5 is running and to write them to the serial flash memory.



Note: CM3_SECTION is a binary file for a Cortex-M3 program. For details, see the application note “RZ/T1 Group Initial Settings of the Microcomputers Incorporating the R-IN Engine”.

Figure 5.1 Schematic View of Flash Downloader Operation

5.3 Developing a Flash Downloader

Figure 5.2 shows the flow of developing a flash downloader. The flash downloader is developed as a DS-5 project. This project is called the flash downloader project. The flash downloader handles processing for reading the application binary files by means of semihosting, CPU initialization, and programming to suit the given serial flash memory. The sample program covered in this application note handles programming of the serial flash memory on the RZ/T1 evaluation board as a serial flash memory interface function. For details of the serial flash memory interface functions, see Section 7, Flash Memory Interface Functions.

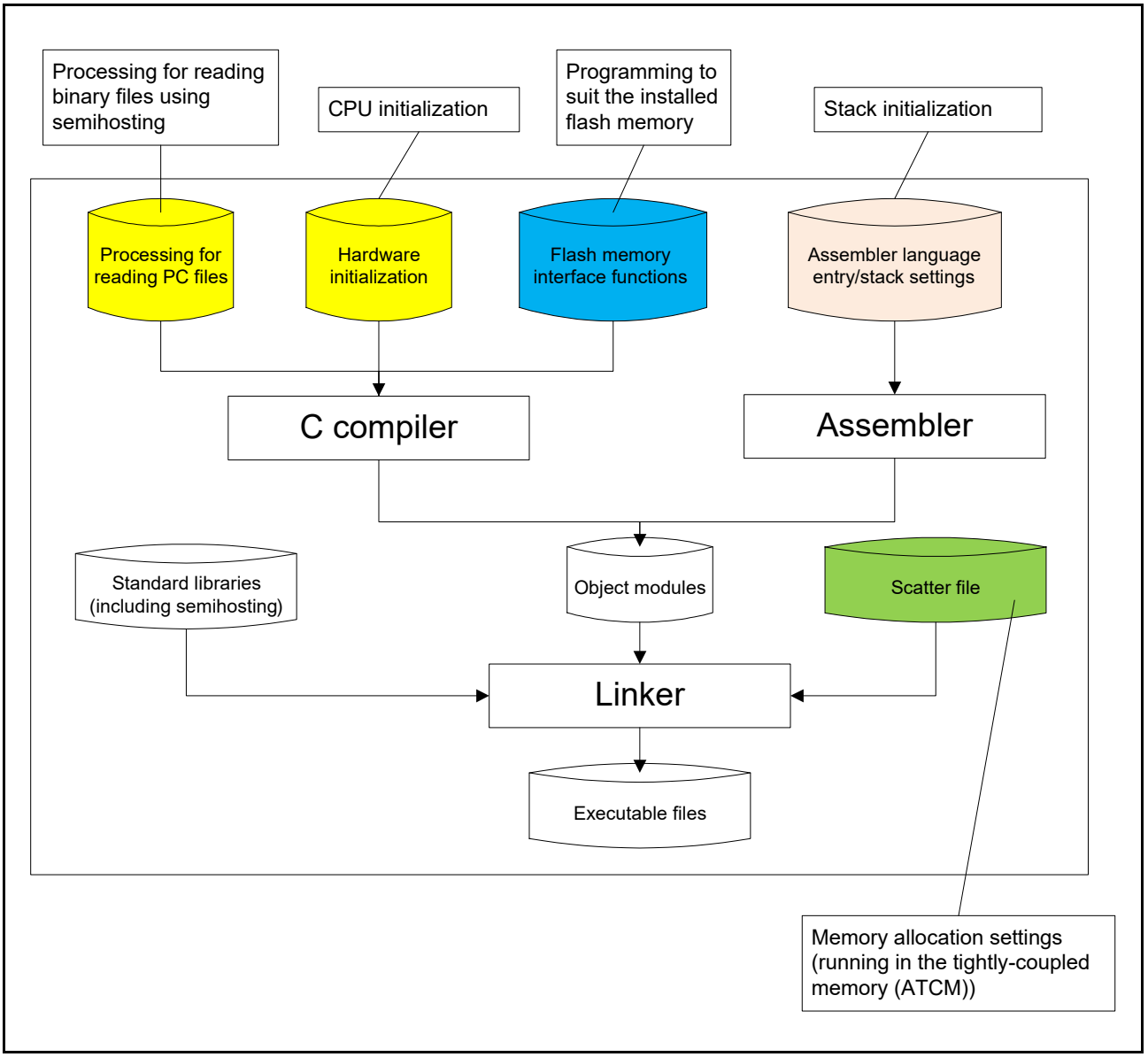


Figure 5.2 Flash Downloader Development Flow

5.3.1 Memory Map

Since the flash downloader runs in the tightly-coupled memory (ATCM) of the RZ/T1 group microcontroller, a scatter file*1 is used to allocate it to the tightly-coupled memory (ATCM). Figure 5.3 shows the memory allocation of the flash downloader.

Note 1. A scatter file is text in which memory layout and allocation of codes and data are described. For details, see “ARM® Compiler toolchain Using the Linker, Image structure and generation”.

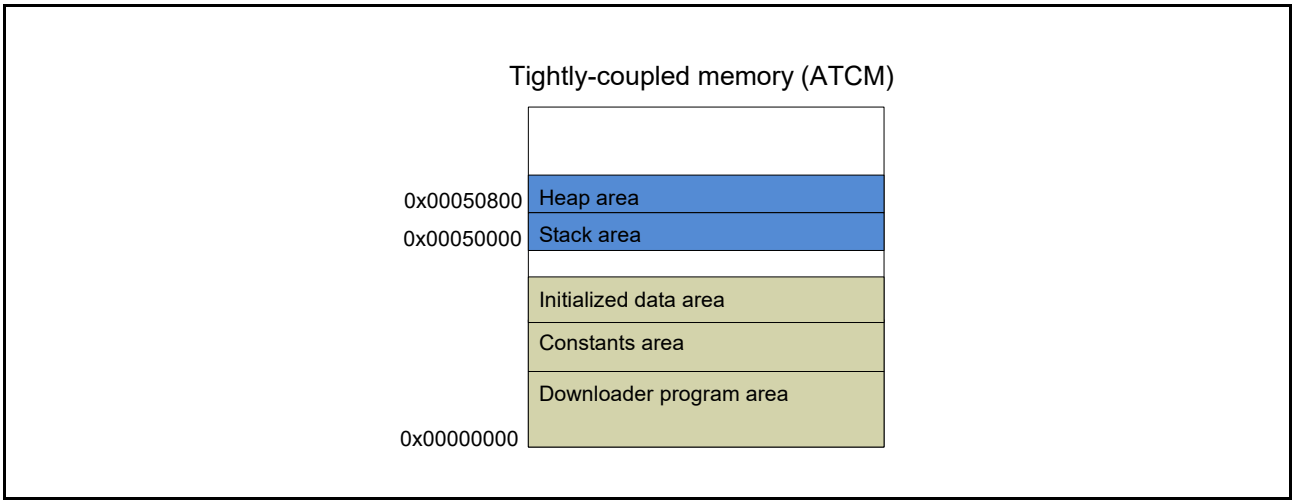


Figure 5.3 Memory Allocation of the Flash Downloader

1. The flash downloader is allocated to the tightly-coupled memory (ATCM) area of the RZ/T1 group microcontroller. Address 0x00000000 is set as the entry point of the flash downloader.
2. The stack area, heap area, etc. used by the flash downloader are allocated to the tightly-coupled memory (ATCM) area.
3. An exception handler vector table need not be implemented for the flash downloader since semihosting provides this functionality.

5.4 Customizing the Examples of Downloading to the Serial Flash Memory

This section describes the procedure for customizing the examples of downloading to the serial flash memory presented in this application note.

You can customize the items listed in Table 5.2. Customize them to suit the specifications of your system.

Table 5.2 Customizable Items

Item	Description
Customization to suit the application project to be downloaded	The names of the application binary files and the write start addresses can be customized to suit the application project to be downloaded to the serial flash memory. For details of the customization procedure, see Section 10.1, Changing the Binary File Names and Destination Addresses for Writing.
Customization of the serial flash memory interface functions	The flash memory interface functions can be customized to suit the serial flash memory to be programmed. For details of the customization procedure, see Section 10.2, Changing the Sample Program to Suit the Given Flash Memory.

6. Example of Downloading to the RZ/T1 Evaluation Board (RTK7910022C00000BR)

This section presents the procedure for downloading the application program (RZ_T_sflash_sample) to the serial flash memory on the RZ/T1 evaluation board (RTK7910022C00000BR) by using DS-5 and the ARM® emulator according to the method of downloading presented in this application note.

Figure 6.1 shows an outline of the downloading procedure.

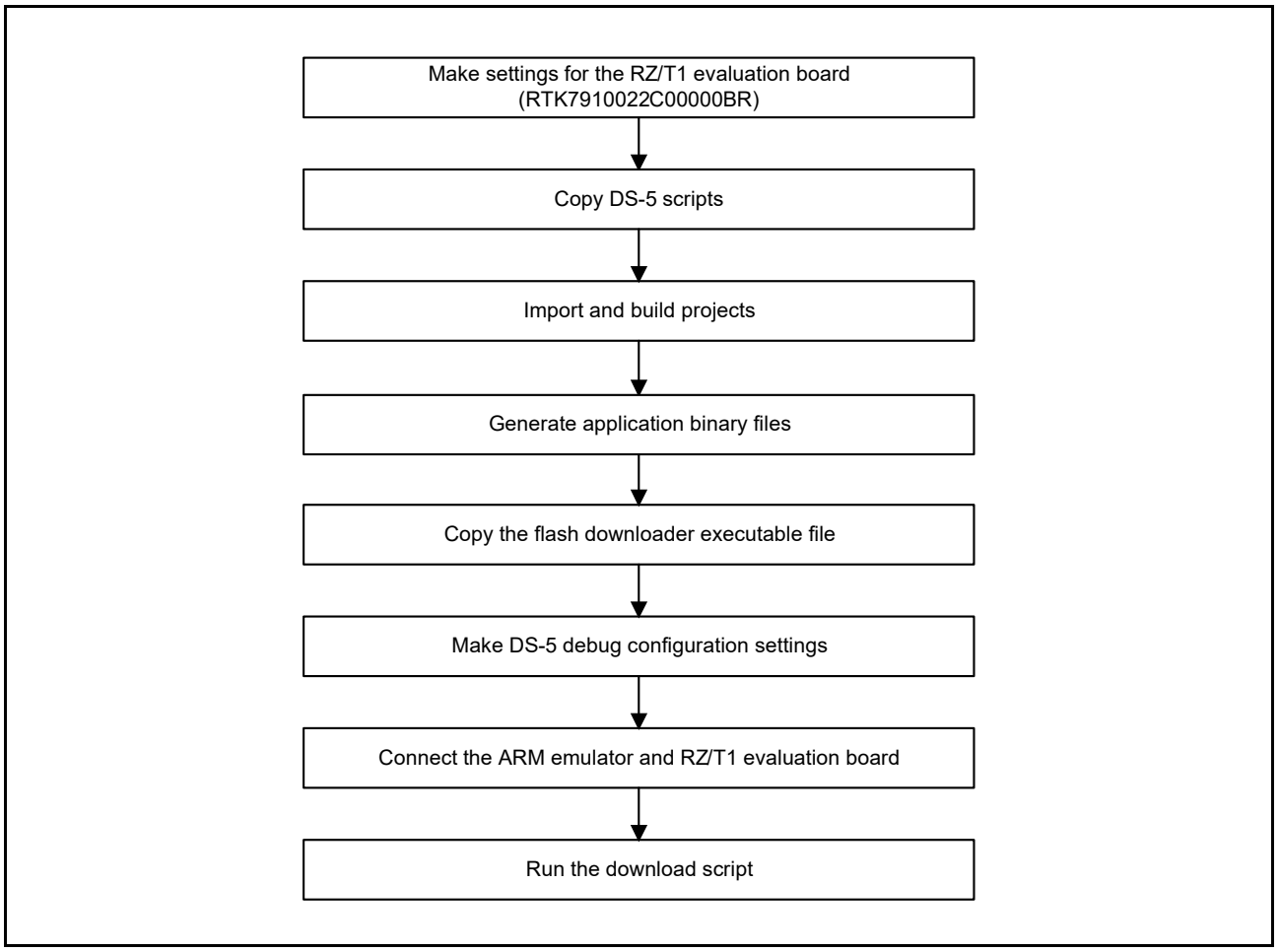


Figure 6.1 Outline of Downloading Procedure

6.1 Settings for the RZ/T1 Evaluation Board (RTK7910022C00000BR)

Table 6.1 lists the settings for the RZ/T1 evaluation board (RTK7910022C00000BR) to run the sample program in this application note.

Make settings for the RZ/T1 evaluation board (RTK7910022C00000BR) as indicated in Table 6.1.

Table 6.1 Settings for the RZ/T1 Evaluation Board (RTK7910022C00000BR)

SW	Setting	Description
SW4-1	ON	MD0 = low level
SW4-2	ON	MD1 = low level
SW4-3	ON	MD2 = low level
SW4-4	ON	BSCANP = low level
SW4-5	ON	OSCTH = low level
SW4-6	OFF	PU7 = high level

6.2 Copying DS-5 Scripts

Create a directory [script_sflash] under the application project (RZ_T_sflash_sample) directory and copy the DS-5 scripts listed in Table 6.2 into it.

Note: For details of the DS-5 workspace directory, see “Using the ARM® DS-5™ Debugger” provided by ARM®.

Table 6.2 List of DS-5 Script Files

Script Name	Description
init_RZ-T.ds	This is the RZ/T1 evaluation board initialization script. This DS-5 script is for executing processing, such as enabling writing to the tightly-coupled memory (ATCM) of the RZ/T1 group microcontroller, when DS-5 is connected to the RZ/T1 evaluation board.
RZ_T_sflash_sample.ds	This is the application downloading script. This DS-5 script contains commands for the sequence of operations for writing the application program to the serial flash memory allocated to the external address space (SPI multi-I/O bus space) of the RZ/T1 group microcontroller.
init_RZ-T2.ds	This is the RZ/T1 evaluation board initialization script to be executed from the application downloading script. It is identical to init_RZ-T.ds, except that it does not make settings for the DS-5 memory area.

6.3 Importing and Building Projects

Import the projects listed in Table 6.3 to the DS-5 workspace directory. Then, build the projects to generate executable files.

[Procedure]

1. Select [All Programs] - [ARM DS-5 v5.21.1] - [Eclipse for DS-5] from the DS-5 start menu.
2. Select [File (F)] - [Import (I)], then open the [Import - Select] window.
3. Select [General] - [Existing Projects into Workspace], then click [Next].
4. Display the projects by clicking [Reference] in the [Import - Import Projects] window, then select the projects to be imported. In the option, check [Copy Projects into Workspace (C)], then click [End].
5. Select the projects imported by the project explorer in order, then select [Project (P) - Build Projects (B)] to build the projects.

Table 6.3 List of Projects

Project	Description	Executable File
RZ_T_fmtool_sflash	This project is used to build the flash downloader. We refer to it as the flash downloader project.	RZ_T_fmtool_sflash.axf
RZ_T_sflash_sample	This project is used to build the application program. We refer to it as the application project.	RZ_T_sflash_sample.axf

6.4 Generating Application Binary Files

Run the command*1 in Figure 6.3 from [DS-5 Command Prompt] of DS-5 to generate application binary files (RZ_T_sflash_sample.bin). Table 6.4 lists the binary files generated when this command is run.

In the project included in this application note, this processing is handled by a batch file (¥RZ_T_sflash_sample¥Debug¥after_build.bat) when building the project.

[Procedure]

1. Select [All Programs] - [ARM DS-5 v5.21.1] - [DS-5 Command Prompt] from the DS-5 Command Prompt start menu.
2. Type [select_toolchain] and press enter. Select a toolchain to use and press enter (see Figure 6.2).
3. Set a path to the [fmtool] folder created in Section 6.3, Importing and Building Projects, and then run the command*1 listed in Figure 6.3

Note 1. For details of the command, see “ARM® DS-5™ DS-5 Getting Started Guide, ARM DS-5 Product Overview” provided by ARM®.

```
You can change the compiler toolchain for this environment at any time by
running the 'select_toolchain' command. A default for all future environments
can be set  with the 'select_default_toolchain' command.


C:¥Program Files¥DS-5 v5.21.1¥bin>select_toolchain
Select a toolchain to use in the current environment

 1 - ARM Compiler 5 (DS-5 built-in)
 2 - GCC 4.x [arm-linux-gnueabihf] (DS-5 built-in)
Enter a number or <return> for no toolchain: 1

Environment configured for ARM Compiler 5 (DS-5 built-in)

C:¥Program Files¥DS-5 v5.21.1¥bin>
```

Figure 6.2 Configuring a Toolchain

```
fromelf --bin --output=RZ_T_sflash_sample.bin RZ_T_sflash_sample.axf
```

Figure 6.3 Application Binary File Generation Command

Table 6.4 List of Application Binary Files

Directory	Binary File	Description
RZ_T_sflash_sample.bin	CONST_LOADER_TABLE	Application (1) (loader parameter information) binary file
	LOADER_RESET_HANDLER	Application (2) (loader program) binary file
	LOADER_IN_ROOT	Application (3) (loader program) binary file
	INIT	Application (4) (user program) binary file
	CM3_SECTION*1	Application (5) (user program) binary file (Cortex-M3 program)

Note 1. CM3_SECTION is a binary file for a Cortex-M3 program. For details, refer to the application note “RZ/T1 Group Initial Settings of the Microcomputers Incorporating the R-IN Engine”.

6.5 Copying the Flash Downloader Executable File

Create a directory [fmtool] directly under the application project (RZ_T_sflash_sample) directory that was imported in Section 6.3, Importing and Building Projects, and copy the flash downloader project executable file (RZ_T_fmtool_sflash.axf) into it.

In the project included in this application note, this processing is handled by a batch file (l¥RZ_T_fmtool_sflash¥Debug¥after_build.bat) when building the project.

6.6 DS-5 Debug Configuration Settings

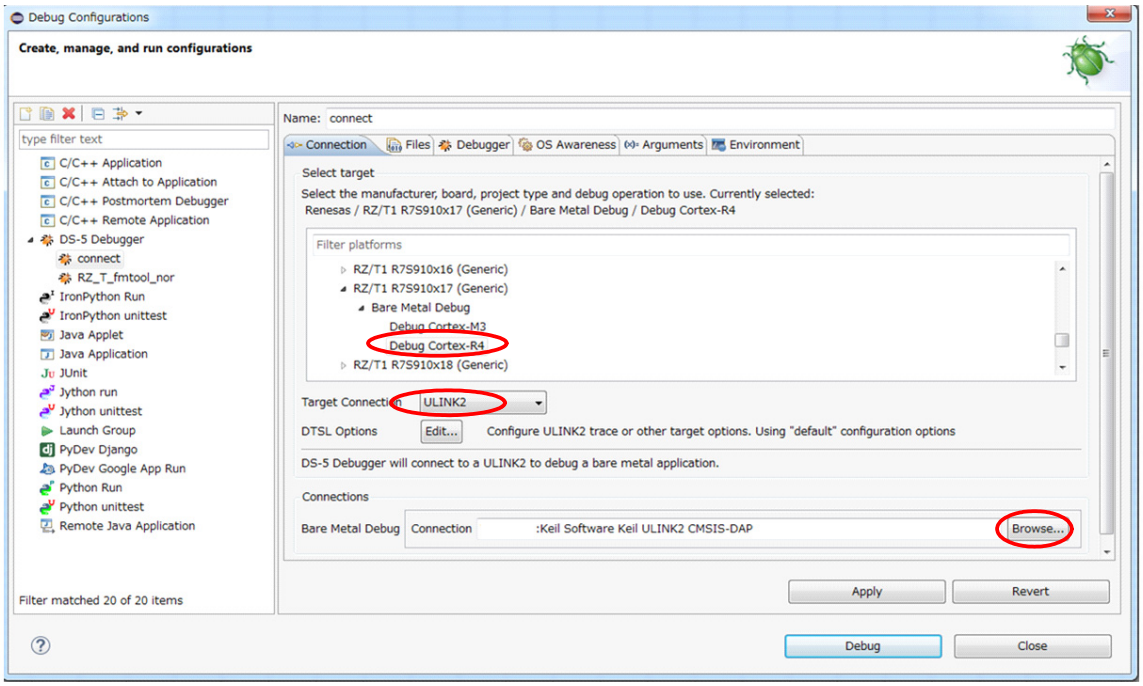
Follow the procedure below to make settings for a DS-5 debug configuration. The DS-5 debug configuration settings specify that the RZ/T1 evaluation board initialization script (init_RZ-T.ds) is run when DS-5 is connected to the RZ/T1 evaluation board*1. For details of processing by the RZ/T1 evaluation board initialization script (init_RZ-T.ds), see Section 9.2, RZ/T1 Evaluation Board Initialization Script.

[Procedure]

1. In DS-5, select [Run (R)] - [Debug Configurations (B)] to display the [Debug Configurations] window.
2. In the [Connection] tab of the DS-5 [Debug Configurations] window, select the target. As the target, select [Renesas] / [RZ/T1 R7S910x17(Generic)] / [Bare Metal Debug] / [Debug of Cortex-R4]*2.
3. In the [Connection] tab of the DS-5 [Debug Configurations] window, select target connection and connection browser. In [Target Connection], select the debugger to connect, and then press the [Browse] button in [Bare Metal Debug] to select the connected debugger in [Connection Browser].
4. In the [Debugger] tab of the DS-5 [Debug Configurations] window, check the box for [Connect only] under Run control.
5. In the [Debugger] tab of the DS-5 [Debug Configurations] window, check the box for [Run target initialization debugger script (.ds/.py)] under Run control, and set a path to the RZ/T1 evaluation board initialization script (init_RZ-T.ds).

Note 1. The above procedure assumes that the RZ/T1 evaluation board has been registered with the DS-5 platform. If the RZ/T1 evaluation board has not been registered with the DS-5 platform, use the DS-5 debugger hardware configuration tools to register it.

Note 2. The name of the target to select may differ according to the version of the DS-5 you are using.



Note: This screen shot is for when a debugger to connect to the target is ULINK2™.

Figure 6.4 Selecting a Debugger in DS-5

6.7 Connecting the RZ/T1 Evaluation Board with an ARM® Emulator

Follow the procedure below to connect the RZ/T1 evaluation board with an ARM® emulator.

[Procedure]

1. In the [Debug Control] tab of DS-5, select the connection under the name specified in step 2 of Section 6.6, DS-5 Debug Configuration Settings. Then right click to select [Connect to Target].
2. Connection starts in step 1. After the connection is established, the RZ/T1 evaluation board initialization script (init_RZ-T.ds) registered in step 4 of Section 6.6, DS-5 Debug Configuration Settings, is run.

6.8 Running the Download Script

Follow the procedure below to run the download script (RZ_T_sflash_sample.ds).

[Procedure]

1. In the [Scripts] tab of DS-5, register the download script (RZ_T_sflash_sample.ds).
2. Double-click the download script (RZ_T_sflash_sample.ds) registered in step 1 to run the script.
3. When the download script runs, the flash downloader is launched and starts writing to the flash memory. Figure 6.5 shows the message displayed in [Application Console].
4. When downloading is complete, the symbol information of the flash downloader is discarded, the RZ/T1 evaluation board initialization script (init_RZ-T2.ds) is run from the download script (RZ_T_sflash_sample.ds), and the symbol information of the application program (RZ_T_sflash_sample) is loaded.

```
RZ/T1 CPU Board S-Flash Programming Sample. Ver.1.00
Copyright (C) 2015 Renesas Electronics Corporation. All rights reserved.

Initializing Flash...
Start to load Binary Data to Flash Memory.
loop=1, file=CONST_LOADER_TABLE, flash address=0x30000000.
Calculating Data Size...
Data Size is 76
Programing Flash...
Calcurating Checksum of Loader Parameter.
Verifying Flash...
loop=1, Flash Programming Success!!
loop=2, file=LOADER_RESET_HANDLER, flash address=0x30000200.
Calculating Data Size...
Data Size is 11812
Programing Flash...
Verifying Flash...
loop=2, Flash Programming Success!!
loop=3, file=LOADER_IN_ROOT, flash address=0x30006200.
Calculating Data Size...
Data Size is 236
Programing Flash...
Verifying Flash...
loop=3, Flash Programming Success!!
loop=4, file=INIT, flash address=0x30010000.
Calculating Data Size...
Data Size is 2592
Programing Flash...
Verifying Flash...
loop=4, Flash Programming Success!!
loop=5, Could not open file. Exiting.
Flash Programming Complete
```

Note: Processing by loop = 5 is dedicated to writing a Cortex-M3 program. When writing a Cortex-R4F program, the required writing of the program is completed by loop = 4.

Figure 6.5 Messages Output to the Application Console

7. Flash Memory Interface Functions

This section describes the flash memory interface functions.

7.1 Fixed-Width Integers

Table 7.1 lists fixed-width integers used in the sample program.

Table 7.1 Fixed-Width Integers Used in the Sample Program

Symbol	Description
char8_t	8-bit signed integer
int16_t	16-bit signed integer
int32_t	32-bit signed integer
uint8_t	8-bit unsigned integer
uint16_t	16-bit unsigned integer
uint32_t	32-bit unsigned integer

7.2 Structures and Unions

Table 7.2 to Table 7.9 list the structures used in the sample program.

Table 7.2 Structure of SPIBSC External Address Read Settings (st_spibsc_cfg_t) (1)

Member	Description
uint8_t udef_cmd	Read command • Sets the read command output to the serial flash memory when converting read operations for the SPI multi-I/O bus space to SPI communications. • The value of this member is set in the CMD[7:0] bits of the data read command setting register (DRCMR).
uint8_t udef_cmd_width	Read command bit width • Sets the bit width for issuing read commands. • Available settings: SPIBSC_1BIT: 1-bit width SPIBSC_4BIT: 4-bit width • The value of this member is set in the CDB[1:0] bits of the data read enable register (DRENR).
uint8_t udef_opd3 uint8_t udef_opd2 uint8_t udef_opd1 uint8_t udef_opd0	Optional data • Set the optional data output to the serial flash memory when converting read operations for the SPI multi-I/O bus space to SPI communications. • The values of these members are set in the OPD3[7:0], OPD2[7:0], OPD1[7:0], and OPD0[7:0] bits of the data read option setting register (DROPR).
uint8_t udef_opd_enable	Optional data enable • Selects whether the optional data is to be issued. • Available settings: SPIBSC_OUTPUT_DISABLE: No data is output SPIBSC_OUTPUT_OPD_3: OPD3 is output SPIBSC_OUTPUT_OPD_32: OPD3 and OPD2 are output SPIBSC_OUTPUT_OPD_321: OPD3, OPD2, and OPD1 are output SPIBSC_OUTPUT_OPD_3210: OPD3, OPD2, OPD1, and OPD0 are output. • The value of this member is set in the OPDE[3:0] bits of the data read enable register (DRENR).
uint8_t udef_opd_width	Optional data bit width • Sets the bit width for issuing the optional data. • Available settings: SPIBSC_1BIT: 1-bit width SPIBSC_4BIT: 4-bit width • The value of this member is set in the OPDB[1:0] bits of the data read enable register (DRENR).

Table 7.3 Structure of SPIBSC External Address Read Settings (st_spibsc_cfg_t) (2)

Member	Description
uint8_t udef_dmycyc_num	Number of dummy cycles • Sets the number of dummy cycles output to the serial flash memory when converting read operations for the SPI multi-I/O bus space to SPI communications. • Available settings: SPIBSC_DUMMY_1CYC: 1 cycle SPIBSC_DUMMY_2CYC: 2 cycles SPIBSC_DUMMY_3CYC: 3 cycles SPIBSC_DUMMY_4CYC: 4 cycles SPIBSC_DUMMY_5CYC: 5 cycles SPIBSC_DUMMY_6CYC: 6 cycles SPIBSC_DUMMY_7CYC: 7 cycles SPIBSC_DUMMY_8CYC: 8 cycles • The value of this member is set in the DMCYC[2:0] bits of the data read dummy cycle setting register (DRDMCR).
uint8_t udef_dmycyc_enable	Dummy cycle enable • Selects whether dummy cycles are to be inserted. • Available settings: SPIBSC_DUMMY_CYC_DISABLE: Dummy cycles are not inserted SPIBSC_DUMMY_CYC_ENABLE: Dummy cycles are inserted • The value of this member is set in the DME bit of the data read enable register (DRENR).
uint8_t udef_dmycyc_width	Dummy cycle bit width • Sets the bit width for issuing dummy cycles. • Available settings: SPIBSC_1BIT: 1-bit width SPIBSC_4BIT: 4-bit width • The value of this member is set in the DMDB[1:0] bits of the data read dummy cycle setting register (DRDMCR).
uint8_t udef_data_width	Data read bit width • Sets the bit width for reading data from the serial flash memory when converting read operations for the SPI multi-I/O bus space to SPI communications. • Available settings: SPIBSC_1BIT: 1-bit width SPIBSC_4BIT: 4-bit width • The value of this member is set in the DRDB[1:0] bits of the data read enable register (DRENR).

Table 7.4 Structure of SPIBSC External Address Read Settings (st_spibsc_cfg_t) (3)

Member	Description
uint8_t udef_spbr	Bit rate • Sets the bit rate of the serial clock (SPBCLK) output to the serial flash memory when converting read operations for the SPI multi-I/O bus space to SPI communications. • Available settings: Make settings to match the bit-rate division setting (udef_brdv). • The value of this member is set in the SPBR[7:0] bits of the bit rate setting register (SPBCR).
uint8_t udef_brdv	Bit-rate division setting • Sets the bit rate of the serial clock (SPBCLK) output to the serial flash memory when converting read operations for the SPI multi-I/O bus space to SPI communications. • Available settings: Make settings to match the bit-rate division setting (udef_brdv). • The value of this member is set in the BRDV[1:0] bits of the bit-rate setting register (SPBCR).
uint8_t udef_addr_width	Address bit width • Sets the width in bits of the address line or lines for output to the serial flash memory when converting read operations for the SPI multi-I/O bus space to SPI communications. • Available settings: SPIBSC_1BIT: 1-bit width SPIBSC_4BIT: 4-bit width • The value of this member is set in the ADB[1:0] bits of the data read enable register (DRENR).
uint8_t udef_addr_mode	Address enable • Sets the address for output to the serial flash memory when converting read operations for the SPI multi-I/O bus space to SPI communications. • Available settings: SPIBSC_OUTPUT_ADDR_24: 24-bit address output SPIBSC_OUTPUT_ADDR_32: 32-bit address output • The value of this member is set in the ADE[3:0] bits of the data read enable register (DRENR).

Table 7.5 Structure of SPIBSC SPI Mode Settings (st_spibsc_spimd_reg_t) (1)

Member	Description
uint32_t cdb	Command bit width • Sets the command bit width in SPI operation mode. • Available settings: SPIBSC_1BIT: 1-bit width SPIBSC_4BIT: 4-bit width • The value of this member is set in the CDB[1:0] bits of the SPI mode enable register (SMENR).
uint32_t ocdb	Optional command bit width • Specifies the optional command bit width in SPI operation mode. • Available settings: SPIBSC_1BIT: 1-bit width SPIBSC_4BIT: 4-bit width • The value of this member is set in the OCDB[1:0] bits of the SPI mode enable register (SMENR).
uint32_t adb	Address bit width • Specifies the width in bits of the address line or lines in SPI operation mode. • Available settings: SPIBSC_1BIT: 1-bit width SPIBSC_4BIT: 4-bit width • The value of this member is set in the ADB[1:0] bits of the SPI mode enable register (SMENR).
uint32_t opdb	Optional data bit width • Specifies the optional data bit width in SPI operation mode. • Available settings: SPIBSC_1BIT: 1-bit width SPIBSC_4BIT: 4-bit width • The value of this member is set in the OPDB[1:0] bits of the SPI mode enable register (SMENR).
uint32_t spidb	Transfer data bit width • Specifies the transfer data bit width in SPI operation mode. • Available settings: SPIBSC_1BIT: 1-bit width SPIBSC_4BIT: 4-bit width • The value of this member is set in the SPIDB[1:0] bits of the SPI mode enable register (SMENR).
uint32_t cde	Sets whether commands are to be output in SPI operation mode. • Available settings: SPIBSC_OUTPUT_DISABLE: Output is disabled. SPIBSC_OUTPUT_ENABLE: Output is enabled. • The value of this member is set in the CDE bit of the SPI mode enable register (SMENR).

Table 7.6 Structure of SPIBSC SPI Mode Settings (st_spibsc_spimd_reg_t) (2)

Member	Description
uint32_t ocde	Optional command enable • Sets whether the optional command is to be output in SPI operation mode. • Available settings: SPIBSC_OUTPUT_DISABLE: Output is disabled. SPIBSC_OUTPUT_ENABLE: Output is enabled. • The value of this member is set in the OCDE bit of the SPI mode enable register (SMENR).
uint32_t ade	Address enable • Sets whether the address is to be output in SPI operation mode. • Available settings: SPIBSC_OUTPUT_DISABLE: Output is disabled. SPIBSC_OUTPUT_ADDR_24: ADR[23:0] is output. SPIBSC_OUTPUT_ADDR_32: ADR[31:0] is output. • The value of this member is set in the ADE[3:0] bits of the SPI mode enable register (SMENR).
uint32_t opde	Optional data enable • Sets whether the optional data is to be issued in SPI operation mode. • Available settings: SPIBSC_OUTPUT_DISABLE: Output is disabled. SPIBSC_OUTPUT_OPD_3: OPD3 is output. SPIBSC_OUTPUT_OPD_32: OPD3 and OPD2 are output. SPIBSC_OUTPUT_OPD_321: OPD3, OPD2, and OPD1 are output. SPIBSC_OUTPUT_OPD_3210: OPD3, OPD2, OPD1, and OPD0 are output. • The value of this member is set in the OPDE[3:0] bits of the SPI mode enable setting register (SMENR).
uint32_t spide	Transfer data enable • Sets whether data transfer is to proceed in SPI operation mode. • Available settings: SPIBSC_OUTPUT_DISABLE: Output is disabled. SPIBSC_OUTPUT_SPID_8: 8- (or 16-) bit transfer SPIBSC_OUTPUT_SPID_16: 16- (or 32-) bit transfer SPIBSC_OUTPUT_SPID_32: 32- (or 64-) bit transfer • The value of this member is set in the SPIDE[3:0] bits of the SPI mode enable register (SMENR).
uint32_t sslkp	SPBSSL signal level retention • Sets the state of the SPBSSL signal after the end of transfer in SPI operation mode. • Available settings: SPIBSC_SPISSL_NEGATE: The signal is negated at the end of transfer. SPIBSC_SPISSL_KEEP: The level of the SPBSSL signal is retained from the end of transfer to the start of next access. • The value of this member is set in the SSLKP bit of the SPI mode control register (SMCR).

Table 7.7 Structure of SPIBSC SPI Mode Settings (st_spibsc_spimd_reg_t) (3)

Member	Description
uint32_t spire	Data read enable • Enables or disables reading of data in SPI operation mode. • Available settings: SPIBSC_SPIDATA_DISABLE: Reading of data is disabled. SPIBSC_SPIDATA_ENABLE: Reading of data is enabled. • The value of this member is set in the SPIRE bit of the SPI mode control register (SMCR).
uint32_t spiwe	Data write enable • Enables or disables writing of data in SPI operation mode. • Available settings: SPIBSC_SPIDATA_DISABLE: Writing of data is disabled. SPIBSC_SPIDATA_ENABLE: Writing of data is enabled. • The value of this member is set in the SPIWE bit of the SPI mode control register (SMCR).
uint32_t dme	Dummy cycle enable • Sets whether dummy cycles are to be inserted in SPI operation mode. • Available settings: SPIBSC_DUMMY_CYC_DISABLE: Dummy cycles are not inserted. SPIBSC_DUMMY_CYC_ENABLE: Dummy cycles are inserted. • The value of this member is set in the DME bit of the SPI mode enable register (SMENR).
uint32_t adder	Address DDR enable • Selects SDR or DDR transfer of the address for output in SPI operation mode. • Available setting: SPIBSC_SDR_TRANS SDR transfer • The value of this member is set in the ADDRE bit of the SPI mode DDR enable register (SMDRENR).
uint32_t opdre	Optional data DDR enable • Selects SDR or DDR transfer of the optional data for output in SPI operation mode. • Available settings: SPIBSC_SDR_TRANS: SDR transfer • The value of this member is set in the OPDRE bit of the SPI mode DDR enable register (SMDRENR).
uint32_t spidre	Transfer data DDR enable • Selects SDR or DDR transfer of the data for transfer in SPI operation mode. • Available setting: SPIBSC_SDR_TRANS: SDR transfer • The value of this member is set in the SPIDRE bit of the SPI mode DDR enable register (SMDRENR).

Table 7.8 Structure of SPIBSC SPI Mode Settings (st_spibsc_spimd_reg_t) (4)

Member	Description
uint8_t dmdb	Dummy cycle bit width • Sets the bit width of dummy cycles in SPI operation mode. • Available settings: SPIBSC_1BIT: 1-bit width SPIBSC_4BIT: 4-bit width • The value of this member is set in the DMDB[1:0] bits of the SPI mode dummy cycle setting register (SMDMCR).
uint8_t dmcyc	Number of dummy cycles • Sets the number of dummy cycles in SPI operation mode. • Available settings: SPIBSC_DUMMY_1CYC: 1 cycle SPIBSC_DUMMY_2CYC: 2 cycles SPIBSC_DUMMY_3CYC: 3 cycles SPIBSC_DUMMY_4CYC: 4 cycles SPIBSC_DUMMY_5CYC: 5 cycles SPIBSC_DUMMY_6CYC: 6 cycles SPIBSC_DUMMY_7CYC: 7 cycles SPIBSC_DUMMY_8CYC: 8 cycles • The value of this member is set in the DMCYC[2:0] bits of the SPI mode dummy cycle setting register (SMDMCR).
uint8_t cmd	Command • Sets the command for output in SPI operation mode. • The value of this member is set in the CMD[7:0] bits of the SPI mode command setting register (SMCMR).
uint8_t ocmd	Optional command • Sets the optional command for output in SPI operation mode. • The value of this member is set in the OCMD[7:0] bits of the SPI mode command setting register (SMCMR).
uint32_t addr	Address • Sets the address for output in SPI operation mode. • The value of this member is set in the ADR[31:0] bits of the SPI mode address setting register (SMADR).
uint8_t opd[4]	Optional data • Sets the optional data for output in SPI operation mode. • The value of this member is set in the OPDn[7:0] bits of the SPI mode option setting register (SMOPR) as follows. OPD3[7:0] ← opd[0] OPD2[7:0] ← opd[1] OPD1[7:0] ← opd[2] OPD0[7:0] ← opd[3]

Table 7.9 Structure of SPIBSC SPI Mode Settings (st_spibsc_spimd_reg_t) (5)

Member	Description
uint32_t smrdr[2]	Read data storage buffer • Holds the data read in SPI operation mode (the value of the SPI mode read data register n (SMRDRn)) as follows. SMRDR0 → smrdr[0] SMRDR1 → smrdr[1]
uint32_t smwdr[2]	Write data storage buffer • Holds the data for writing in SPI operation mode (the value of the SPI mode write data register n (SMWDRn)) as follows. SMWDR0 ← swrdr[0] SMWDR1 ← swrdr[1]

7.3 Constants

Table 7.10 to Table 7.13 list the constants used in the sample program.

Table 7.10 Constants Used in the Sample Program (1)

Constant	Setting	Description
SF_REQ_PROTECT	(0)	Setting for protection of the serial flash memory
SF_REQ_UNPROTECT	(1)	Releasing the serial flash memory from protection
SFLASHCMD_SECTOR_ERASE	(0xD8)	Erase 256 KB (3-byte address) command
SFLASHCMD_BYTE_PROGRAM	(0x02)	Page programming (3-byte address) command
SFLASHCMD_FAST_READ	(0x0B)	Read fast (3-byte address) command
SFLASHCMD_QUAD_IO_READ	(0xEB)	Quad I/O read (3-byte address) command
SFLASHCMD_WRITE_ENABLE	(0x06)	Write enable command
SFLASHCMD_READ_STATUS	(0x05)	Read status register-1 command
SFLASHCMD_READ_CONFIG	(0x15)	Read configuration register-1 command
SFLASHCMD_WRITE_STATUS	(0x01)	Write register (status-1, configuration-1) command
SFLASHCMD_SECTOR_ERASE_4B	(0xDC)	Erase 256 KB (4-byte address) command
SFLASHCMD_BYTE_PROGRAM_4B	(0x12)	Page programming (4-byte address) command
SFLASHCMD_FAST_READ_4B	(0x0C)	Read fast (4-byte address) command
SFLASHCMD_QUAD_IO_READ_4B	(0xEC)	Quad I/O read (4-byte address) command

Table 7.11 Constants Used in the Sample Program (2)

Constant	Setting	Description
STREG_SRWD_BIT	(0x80)	Status register/SRWD bit mask value
STREG_QUAD_BIT	(0x40)	Status register/QUAD bit mask value
STREG_BPROTECT_BIT	(0x3C)	Status register/block protection bit mask value
STREG_WEL_BIT	(0x02)	Status register/write enable latch bit mask value
STREG_WIP_BIT	(0x01)	Status register/write in progress bit mask value
CFREG_LC_BIT	(0xC0)	Configuration register/latency code bit mask value
CFREG_4BYTE_BIT	(0x20)	Configuration register/4-byte bit mask value

Table 7.12 Constants Used in the Sample Program (3)

Constant	Setting	Description
SF_PAGE_SIZE	(256)	Flash memory page size (amount of data to be written)
SF_SECTOR_SIZE	(64 * 1024)	Sector size
SF_NUM_OF_SECTOR	(1024)	Number of sectors

Table 7.13 Constants Used in the Sample Program (4)

Constant	Setting	Description
SPIBSC_CMNCR_BSZ_SINGLE	(0)	One serial flash memory is connected to the SPIBSC data bus.
SPIBSC_1BIT	(0)	Sets the bit width for issuing read commands to 1 bit.
SPIBSC_4BIT	(2)	Sets the bit width for issuing read commands to 4 bits.
SPIBSC_OUTPUT_DISABLE	(0)	Specifies that no command is output when a read command is issued.
SPIBSC_OUTPUT_ENABLE	(1)	Specifies that a command is output when a read command is issued.
SPIBSC_OUTPUT_ADDR_24	(0x07)	Outputs 24-bit addresses.
SPIBSC_OUTPUT_ADDR_32	(0x0f)	Outputs 32-bit addresses.
SPIBSC_OUTPUT_OPD_3	(0x08)	Outputs the optional data OPD3 when a read command is issued.
SPIBSC_OUTPUT_OPD_32	(0x0c)	Outputs the optional data OPD3 and OPD2 when a read command is issued.
SPIBSC_OUTPUT_OPD_321	(0x0e)	Outputs the optional data OPD3, OPD2, and OPD1 when a read command is issued.
SPIBSC_OUTPUT_OPD_3210	(0x0f)	Outputs the optional data OPD3, OPD2, OPD1, and OPD0 when a read command is issued.
SPIBSC_OUTPUT_SPID_8	(0x08)	Enables 8- (or 16-) bit transfer in SPI operation mode.
SPIBSC_OUTPUT_SPID_16	(0x0c)	Enables 16- (or 32-) bit transfer in SPI operation mode.
SPIBSC_OUTPUT_SPID_32	(0x0f)	Enables 32- (or 64-) bit transfer in SPI operation mode.
SPIBSC_SPISSL_NEGATE	(0)	Sets the state of SPBSSL signal after the end of transfer to the negated state in SPI operation mode.
SPIBSC_SPISSL_KEEP	(1)	Specifies that the level of the SPBSSL signal is retained from the end of transfer to the start of next access in SPI operation mode.
SPIBSC_SPIDATA_DISABLE	(0)	Disables reading of data in SPI operation mode.
SPIBSC_SPIDATA_ENABLE	(1)	Enables reading of data in SPI operation mode.
SPIBSC_DUMMY_CYC_DISABLE	(0)	Disables insertion of dummy cycles.
SPIBSC_DUMMY_CYC_ENABLE	(1)	Enables insertion of dummy cycles.
SPIBSC_DUMMY_1CYC	(0)	Sets the number of dummy cycles for output to the serial flash memory when converting read operations for the SPI multi-I/O bus space to SPI communications to 1.
SPIBSC_DUMMY_2CYC	(1)	Sets the number of dummy cycles for output to the serial flash memory when converting read operations for the SPI multi-I/O bus space to SPI communications to 2.

7.4 Variables

Table 7.14 and Table 7.15 list the static variable and global variables, respectively.

Table 7.14 Static Variables

Type	Variable Name	Description
static uint8_t	g_erased_flag[];	Sector erasure flag Assigns a byte containing a flag to one sector of the serial flash memory. The sector erasure flag is set to 0 (indicating the non-erased state) when running the initialization entry function. The flag is set to 1 (indicating the erased state) following sector erasure.

Table 7.15 Global Variables

Type	Variable Name	Description
st_spibsc_cfg_t	g_spibsc_cfg	SPIBSC external address read settings storage variable • Stores the SPIBSC external address read settings.
st_spibsc_spimd_reg_t	g_spibsc_spimd_reg	SPIBSC SPI mode operation settings storage variable • Stores the SPIBSC settings when the SPIBSC is used in SPI mode. In the sample program, these settings are also used as arguments when running serial flash control functions within the API functions and user-defined functions.

7.5 Flash Memory Interface Functions

Table 7.16 lists the serial flash memory interface functions. Implement processing to suit the serial flash memory to be programmed in these functions.

Table 7.16 List of Flash Memory Interface Functions

Function	Description
flash_init	Initialization interface function This function sets the peripheral modules for use in access to the serial flash memory connected to the external bus (SPI multi-I/O bus space) of the RZ/T1. It initializes the flash memory interface functions.
flash_write	Write interface function This function handles writing to the serial flash memory connected to the external bus (SPI multi-I/O bus space) of the RZ/T1. If sector erasure is not executed for the specified address after the execution of the initialization interface function, this function also handles processing to erase the sector.
flash_write_entry	Serial flash memory write mode entry function
flash_veify_entry	Serial flash memory read mode entry function

Table 7.17 Flash Memory Interface Internal Functions

Function	Description
R_SFLASH_Exmode_Setting	SPIBSC initial settings function Makes initial settings required for controlling the serial flash memory and for using the SPIBSC in external address read mode. This function also makes register settings in the flash memory to suit the initial settings. After the initial settings, it places the SPIBSC in external address read mode.
R_SFLASH_WaitTend	SPIBSC data transfer end wait function Waits for the completion of data transfer from the SPIBSC.
R_SFLASH_Exmode	SPIBSC external address mode setting function Places the SPIBSC in external address read mode.
R_SFLASH_Set_Config	SPIBSC external address read settings function Makes initial settings required for using the SPIBSC in external address read mode.
R_SFLASH_SpibscStop	SPIBSC stop function Stops the SPIBSC.
R_SFLASH_Spimode	SPIBSC SPI mode setting function Places the SPIBSC in SPI mode.
R_SFLASH_Exmode_Init	SPIBSC external address mode initial settings function Makes initial settings for using the SPIBSC in external address read mode. After the initial settings, it places the SPIBSC in external address read mode.
R_SFLASH_Spimode_Init	SPIBSC SPI mode initial settings function Makes initial settings required for using the SPIBSC in SPI mode. After the initial settings, it places the SPIBSC in SPI mode.
R_SFLASH_EraseSector	Serial flash erasure function Uses SPI mode of the SPIBSC to erase the serial flash memory.
R_SFLASH_ByteProgram	Serial flash programming function Uses SPI mode of the SPIBSC to write data to the serial flash memory.
R_SFLASH_Spibsc_Transfer	Serial flash control function Issues commands to the serial flash memory according to arguments.
R_SFLASH_Ctrl_Protect	Serial flash memory protection releasing function Makes settings for the registers in the serial flash memory to release protection according to the function's arguments.

Table 7.18 Flash Memory Interface User-Defined Functions

Function	Description
Userdef_SPIBSC_Set_Config	SPIBSC external address read settings function Determines the SPIBSC external address read mode settings to suit the serial flash memory in use. The sample program makes initial settings required for using the SPIBSC in external address read mode on the basis of the settings made by this function. The sample program makes SPIBSC initial settings for use of a Macronix serial flash memory (product type name: MX25L51245G).
Userdef_SFLASH_Set_Mode	Serial flash memory internal register settings function Makes settings for the registers in the serial flash memory required when using the SPIBSC in external address read mode, to suit the serial flash memory in use. The sample program makes initial settings for the registers in the Macronix serial flash memory (product type name: MX25L51245G).
Userdef_SFLASH_Write_Enable	Serial flash memory write enable function Makes settings for the registers in the serial flash memory to enable writing, to suit the serial flash memory in use. The sample program makes settings for the registers in the Macronix serial flash memory (product type name: MX25L51245G).
Userdef_SFLASH_Busy_Wait	Serial flash memory ready wait function Reads the registers in the serial flash memory and waits for the serial flash memory to enter the ready state, over a period that suits the serial flash memory in use. The sample program waits for the Macronix serial flash memory (product type name: MX25L51245G) to enter the ready state.
Userdef_SFLASH_Ctrl_Protect	Serial flash memory protection releasing function Makes settings for the registers in the serial flash memory to release it from protection, to suit the flash memory in use. The sample program releases the Macronix serial flash memory (product type name: MX25L51245G) from protection.

7.6 Details of the Flash Memory Interface Functions

The following tables list the details of the flash memory interface functions.

flash_init	
Synopsis	Initialization interface function
Declaration	int32_t flash_init(void);
Description	This function sets the peripheral modules for use in access to the serial flash memory connected to the external bus (SPI multi-I/O bus space) of the RZ/T1. It initializes the flash interface functions. It sets the sector erasure flag (fmtool_pre_erase_sctno) to 0 (indicating the non-erased state).
Arguments	None
Return value	0: Initialization has succeeded (always set to 0 in the sample program). -1: Initialization has failed.

flash_write		
Synopsis	Write interface function	
Declaration	int32_t flash_write(uint32_t *fm_adrs, uint32_t *data, int32_t size);	
Description	This function handles writing to the serial flash memory connected to the external bus (SPI multi-I/O bus space) of the RZ/T1. It writes the amount specified by the argument “size” of data specified by the argument “data” to the address specified by the argument fm_adrs. If the sector including the address specified by the argument fm_adrs was not erased following a call of the initialization entry function, this function handles processing to erase the sector. Note that erasure or non-erasure of the sector is determined by the value of the sector erasure flag (fmtool_pre_erase_sctno). If the sector has been erased, the value of the sector erasure flag (fmtool_pre_erase_sctno) is set to 1 (indicating the erased state).	
Arguments	uint32_t *fm_adrs	: Write start address
	uint32_t *data	: Write data storage address
	int32_t size	: Amount of data to be written (always in 512 bytes)
Return value	0: Writing has succeeded. -1: Writing has failed. -2: Verification after writing has failed.	

flash_write_entry	
Synopsis	Serial flash memory write mode entry function
Declaration	int32_t flash_write_entry(void);
Description	This function places the SPIBSC in SPI mode. The SPIBSC SPI mode setting function (R_SFLASH_Spimode) is executed from within this function.
Arguments	None
Return value	0: Success -1: Failure

flash_veify_entry	
Synopsis	Serial flash memory read mode entry function
Declaration	int32_t flash_veify_entry(void);
Description	This function places the SPIBSC in external address read mode. The SPIBSC external address mode setting function (R_SFLASH_Exmode) is executed from within this function.
Arguments	None
Return value	0: Success -1: Failure

R_SFLASH_Exmode_Setting		
Synopsis	SPIBSC initialization setting function	
Declaration	int32_t R_SFLASH_Exmode_Setting (st_spibsc_cfg_t *spibsccfg);	
Description	Makes initial settings required for controlling the serial flash memory and for using the SPIBSC in external address read mode. This function also makes register settings in the flash memory to suit the initial settings. After the initial settings, it places the SPIBSC in external address read mode. The SPIBSC external address mode initial settings function (R_SFLASH_Exmode_Init) is executed from within this function.	
Arguments	st_spibsc_cfg_t *spibsccfg	SPIBSC external address read settings For details of the settings, see Table 7.2 to Table 7.4.
Return value	0: Normal termination -1: Error	

R_SFLASH_Set_Config		
Synopsis	SPIBSC external address read settings function	
Declaration	void R_SFLASH_Set_Config(st_spibsc_cfg_t *spibsccfg);	
Description	Determines the settings for using the SPIBSC in external address read mode to suit the serial flash memory in use. The user-defined function (SPIBSC external address read settings function: Userdef_SPIBSC_Set_Config) is executed from within this function.	
Arguments	st_spibsc_cfg_t *spibsccfg	SPIBSC external address read settings For details of the settings, see Table 7.2 to Table 7.4.
Return value	0: Normal termination -1: Error	

R_SFLASH_SpibscStop	
Synopsis	SPIBSC stop function
Declaration	int32_t R_SFLASH_SpibscStop(void);
Description	This function stops the SPIBSC.
Arguments	None
Return value	None

R_SFLASH_WaitTend	
Synopsis	SPIBSC data transfer end wait function
Declaration	void R_SFLASH_WaitTend(void);
Description	Waits for the completion of data transfer from the SPIBSC.
Arguments	None
Return value	None

R_SFLASH_Exmode	
Synopsis	SPIBSC external address mode setting function
Declaration	int32_t R_SFLASH_Exmode(void);
Description	This function places the SPIBSC in external address read mode.
Arguments	None
Return value	0: Setting has succeeded.

R_SFLASH_Spimode	
Synopsis	SPIBSC SPI mode setting function
Declaration	int32_t R_SFLASH_Spimode(void);
Description	This function places the SPIBSC in SPI mode.
Arguments	None
Return value	0: Setting has succeeded.

R_SFLASH_Exmode_Init		
Synopsis	SPIBSC external address mode initial settings function	
Declaration	int32_t R_SFLASH_Exmode_Init(st_spibsc_cfg_t *spibsccfg)	
Description	This function makes initial settings required for using the SPIBSC in external address read mode. After the initial settings, it places the SPIBSC in external address read mode.	
Arguments	st_spibsc_cfg_t *spibsccfg	SPIBSC external address read settings For details of the settings, see Table 7.2 to Table 7.4.
Return value	0: Normal termination -1: Error	

R_SFLASH_Spimode_Init		
Synopsis	SPIBSC SPI mode initial settings function	
Declaration	int32_t R_SFLASH_Spimode_Init(uint32_t data_width, uint32_t addr_mode, uint32_t spbr, uint32_t brdv);	
Description	This function makes initial settings required for using the SPIBSC in SPI mode. After the initial settings, it places the SPIBSC in SPI mode.	
Arguments	uint32_t data_width	Data read bit width Bit width for reading data from the serial flash memory when converting read operations for the SPI multi-I/O bus space to SPI communications. SPIBSC_1BIT: 1-bit width SPIBSC_4BIT: 4-bit width
	uint32_t addr_mode	Address mode setting Sets the address for output to the serial flash memory when converting read operations for the SPI multi-I/O bus space to SPI communications. SPIBSC_OUTPUT_ADDR_24: 24-bit address output SPIBSC_OUTPUT_ADDR_32: 32-bit address output
	uint32_t spbr	Bit rate Sets the bit rate of the serial clock (SPBCLK) to match the bit-rate division setting (brdv).
	uint32_t brdv	Bit-rate division setting Sets the bit rate of the serial clock (SPBCLK) to match the bit-rate division setting (spbr).
Return value	0: Setting has succeeded. -1: Setting has failed.	

R_SFLASH_EraseSector		
Synopsis	Serial flash memory erase function	
Declaration	int32_t R_SFLASH_EraseSector(uint32_t addr, uint32_t data_width, uint32_t addr_mode);	
Description	This function erases the serial flash memory using SPI mode of the SPIBSC.	
Arguments	uint32_t addr	Address to be erased in serial flash memory
	uint32_t data_width	Data read bit width Bit width for reading data from the serial flash memory when converting read operations for the SPI multi-I/O bus space to SPI communications. SPIBSC_1BIT: 1-bit width SPIBSC_4BIT: 4-bit width
	uint32_t addr_mode	Address mode setting Sets the address for output to the serial flash memory when converting read operations for the SPI multi-I/O bus space to SPI communications. SPIBSC_OUTPUT_ADDR_24: 24-bit address output SPIBSC_OUTPUT_ADDR_32: 32-bit address output
Return value	0: Setting has succeeded. -1: Setting has failed.	

R_SFLASH_ByteProgram		
Synopsis	Serial flash memory write function	
Declaration	int32_t R_SFLASH_ByteProgram(uint32_t addr, uint8_t *buf, int32_t size, uint32_t data_width, uint32_t addr_mode);	
Description	This function writes data to the serial flash memory using the SPI mode of the SPIBSC.	
Arguments	uint32_t addr	Address to be written to in serial flash memory
	uint8_t *buf	Write data storage buffer
	int32_t size	Amount of data to be written (in bytes)
	uint32_t data_width	Data read bit width Bit width for reading data from the serial flash memory when converting read operations for the SPI multi-I/O bus space to SPI communications. SPIBSC_1BIT: 1-bit width SPIBSC_4BIT: 4-bit width
	uint32_t addr_mode	Address mode setting Sets the address for output to the serial flash memory when converting read operations for the SPI multi-I/O bus space to SPI communications. SPIBSC_OUTPUT_ADDR_24: 24-bit address output SPIBSC_OUTPUT_ADDR_32: 32-bit address output
Return value	0: Setting has succeeded. -1: Setting has failed.	

R_SFLASH_Spibsc_Transfer		
Synopsis	Serial flash memory control function	
Declaration	int32_t R_SFLASH_Spibsc_Transfer(st_spibsc_spimd_reg_t *regset);	
Description	This function accesses the serial flash memory using SPI mode of the SPIBSC.	
Arguments	st_spibsc_spimd_reg_t * regset	SPIBSC SPI mode setting For details of the settings, see Table 7.2 to Table 7.4.
Return value	0: Setting has succeeded. -1: Setting has failed.	

R_SFLASH_Ctrl_Protect		
Synopsis	Serial flash memory protection releasing function	
Declaration	int32_t R_SFLASH_Ctrl_Protect(uint32_t req, uint32_t data_width);	
Description	This function makes settings for the registers in the serial flash memory to release it from protection.	
Arguments	uint32_t_t req	Register settings information SF_REQ_PROTECT: Sets protection. SF_REQ_UNPROTECT: Releases protection.
	uint32_t data_width	Data read bit width Bit width for reading data from the serial flash memory when converting read operations for the SPI multi-I/O bus space to SPI communications. SPIBSC_1BIT: 1-bit width SPIBSC_4BIT: 4-bit width
Return value	None	

Userdef_SPIBSC_Set_Config		
Synopsis	SPIBSC external address read settings function	
Declaration	void Userdef_SPIBSC_Set_Config(st_spibsc_cfg_t *spibsccfg);	
Description	Determines the SPIBSC external address read mode settings to suit the serial flash memory in use. This function makes initial settings required for using the SPIBSC in the area specified by the argument spibsccfg in external address read mode. For details of the settings as argument spibsccfg, see Table 7.2 to Table 7.4.	
Arguments	st_spibsc_cfg_t *spibsccfg	SPIBSC external address read settings For details of the settings, see Table 7.2 to Table 7.4.
Return value	None	
Note	The sample program makes SPIBSC initial settings for use of a Macronix serial flash memory (product type name: MX25L51245G).	

Userdef_SFLASH_Set_Mode		
Synopsis	Serial flash memory internal register settings function	
Declaration	int32_t Userdef_SFLASH_Set_Mode(uint32_t data_width, uint32_t addr_mode);	
Description	Within this function, implement processing for required setting of the registers in the serial flash memory for use with the SPIBSC in external address read mode, to suit the serial flash memory to be used.	
Arguments	uint32_t data_width	Data read bit width Bit width for reading data from the serial flash memory when converting read operations for the SPI multi-I/O bus space to SPI communications. SPIBSC_1BIT: 1-bit width SPIBSC_4BIT: 4-bit width
	uint32_t addr_mode	Address mode setting Sets the address for output to the serial flash memory when converting read operations for the SPI multi-I/O bus space to SPI communications. SPIBSC_OUTPUT_ADDR_24: 24-bit address output SPIBSC_OUTPUT_ADDR_32: 32-bit address output
Return value	0: Setting has succeeded. -1: Setting has failed.	
Note	The sample program makes SPIBSC initial settings for use of a Macronix serial flash memory (product type name: MX25L51245G).	

Userdef_SFLASH_Write_Enable	
Synopsis	Serial flash memory write enable function
Declaration	int32_t Userdef_SFLASH_Write_Enable(void);
Description	Within this function, implement processing for setting of the registers in the serial flash memory to enable writing, to suit the serial flash memory to be used.
Arguments	None
Return value	0: Setting has succeeded. -1: Setting has failed.
Note	The sample program makes SPIBSC initial settings for use of a Macronix serial flash memory (product type name: MX25L51245G).

Userdef_SFLASH_Busy_Wait		
Synopsis	Serial flash memory ready wait function	
Declaration	int32_t Userdef_SFLASH_Busy_Wait(uint32_t data_width);	
Description	Within this function, implement processing for reading the registers in the serial flash memory for the transition of the serial flash memory to the ready state, to suit the serial flash memory to be used.	
Arguments	uint32_t data_width	Data read bit width Bit width for reading data from the serial flash memory when converting read operations for the SPI multi-I/O bus space to SPI communications. SPIBSC_1BIT: 1-bit width SPIBSC_4BIT: 4-bit width
Return value	None	
Note	The sample program makes SPIBSC initial settings for use of a Macronix serial flash memory (product type name: MX25L51245G).	

Userdef_SFLASH_Ctrl_Protect		
Synopsis	Serial flash memory protection releasing function	
Declaration	int32_t Userdef_SFLASH_Ctrl_Protect(uint32_t req, uint32_t data_width);	
Description	Within this function, implement processing for setting the registers in the serial flash memory to release it from protection, to suit the serial flash memory to be used.	
Arguments	uint32_t req	Register settings information SF_REQ_PROTECT: Sets protection. SF_REQ_UNPROTECT: Releases protection.
	uint32_t data_width	Data read bit width Bit width for reading data from the serial flash memory when converting read operations for the SPI multi-I/O bus space to SPI communications. SPIBSC_1BIT: 1-bit width SPIBSC_4BIT: 4-bit width
Return value	None	
Note	The sample program makes SPIBSC initial settings for use of a Macronix serial flash memory (product type name: MX25L51245G).	

7.7 Flowcharts of the Flash Memory Interface Functions

7.7.1 Flow of the Initialization Interface Function

Figure 7.1 shows the flow of the initialization interface function.

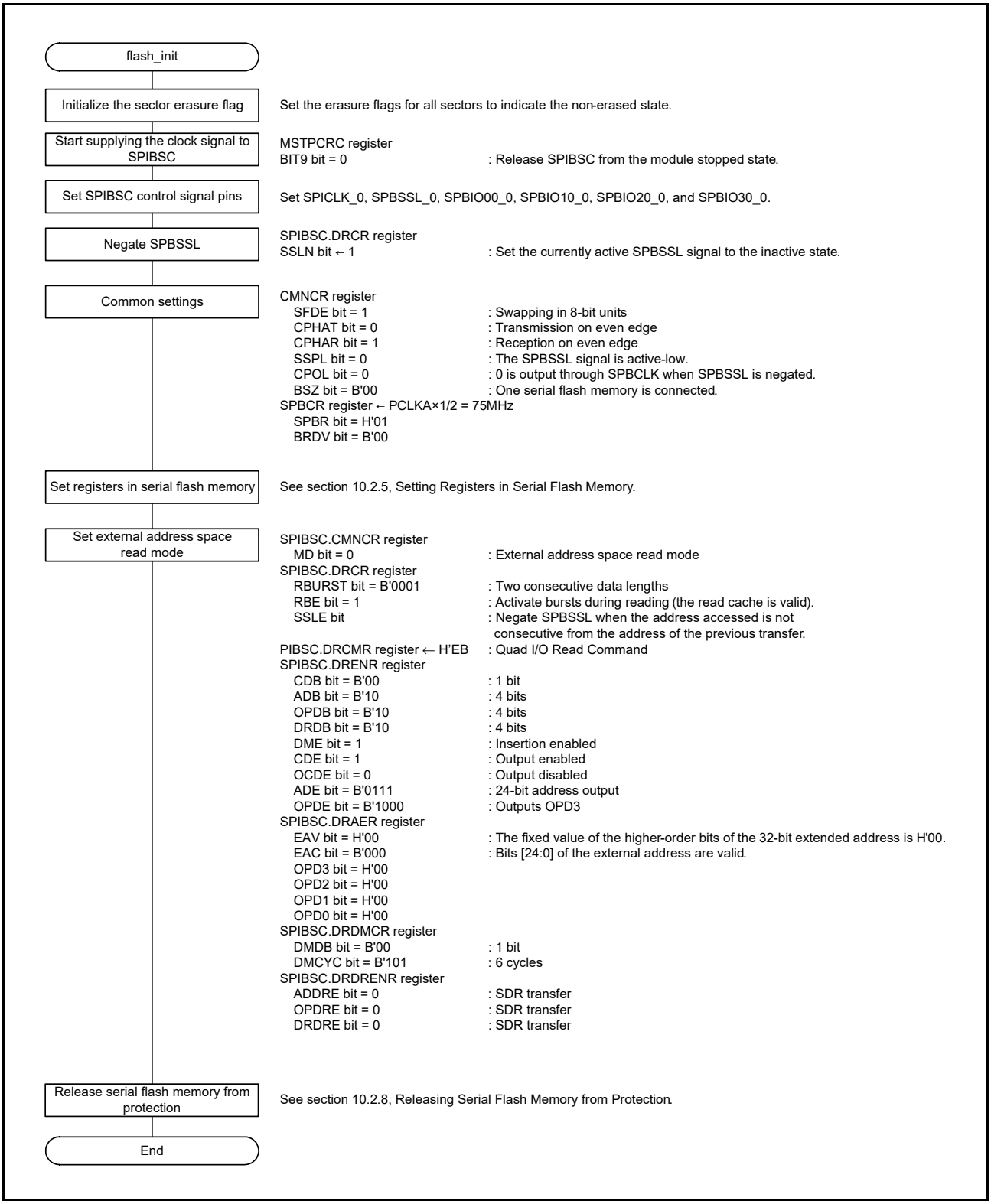


Figure 7.1 Flow of the Initialization Interface Function

7.7.2 Serial Flash Memory Write Mode Entry Function

Figure 7.2 shows the flow of the serial flash memory write mode entry function.

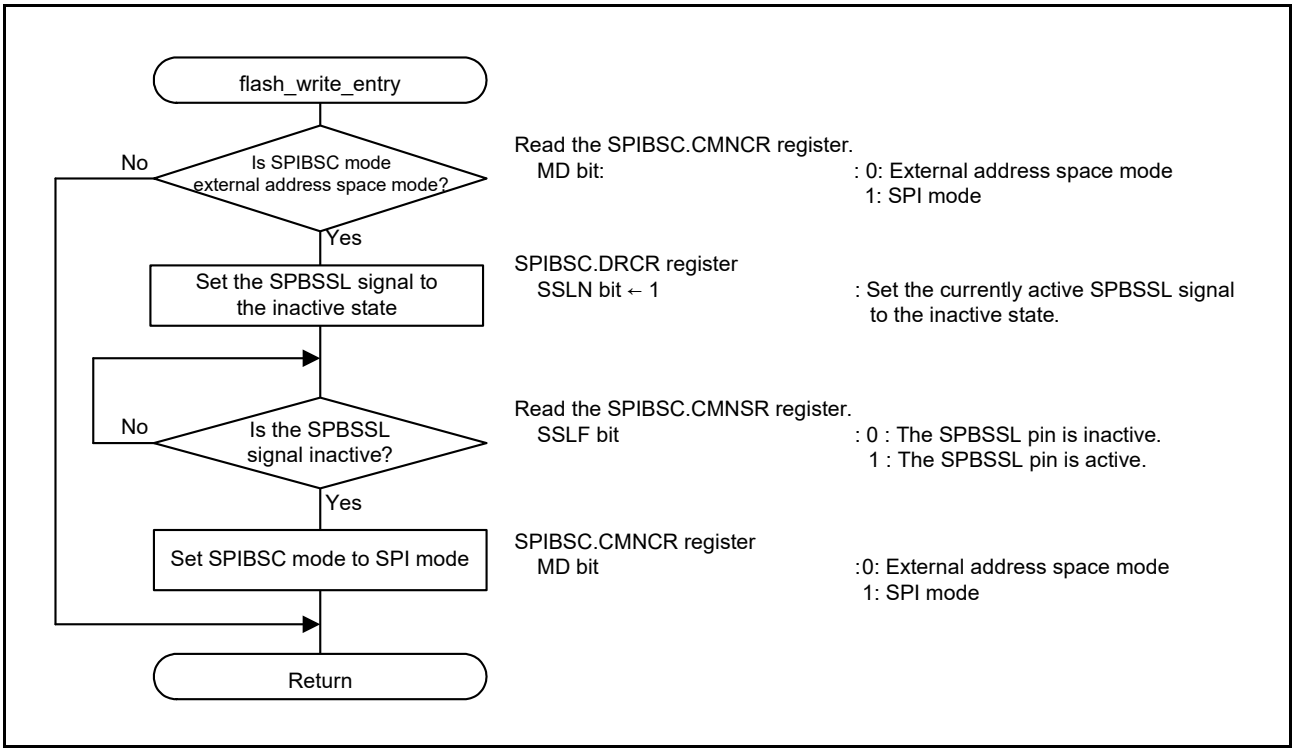


Figure 7.2 Flow of the Serial Flash Memory Write Mode Entry Function

7.7.3 Serial Flash Memory Read Mode Entry Function

Figure 7.3 shows the flow of the serial flash memory read mode entry function.

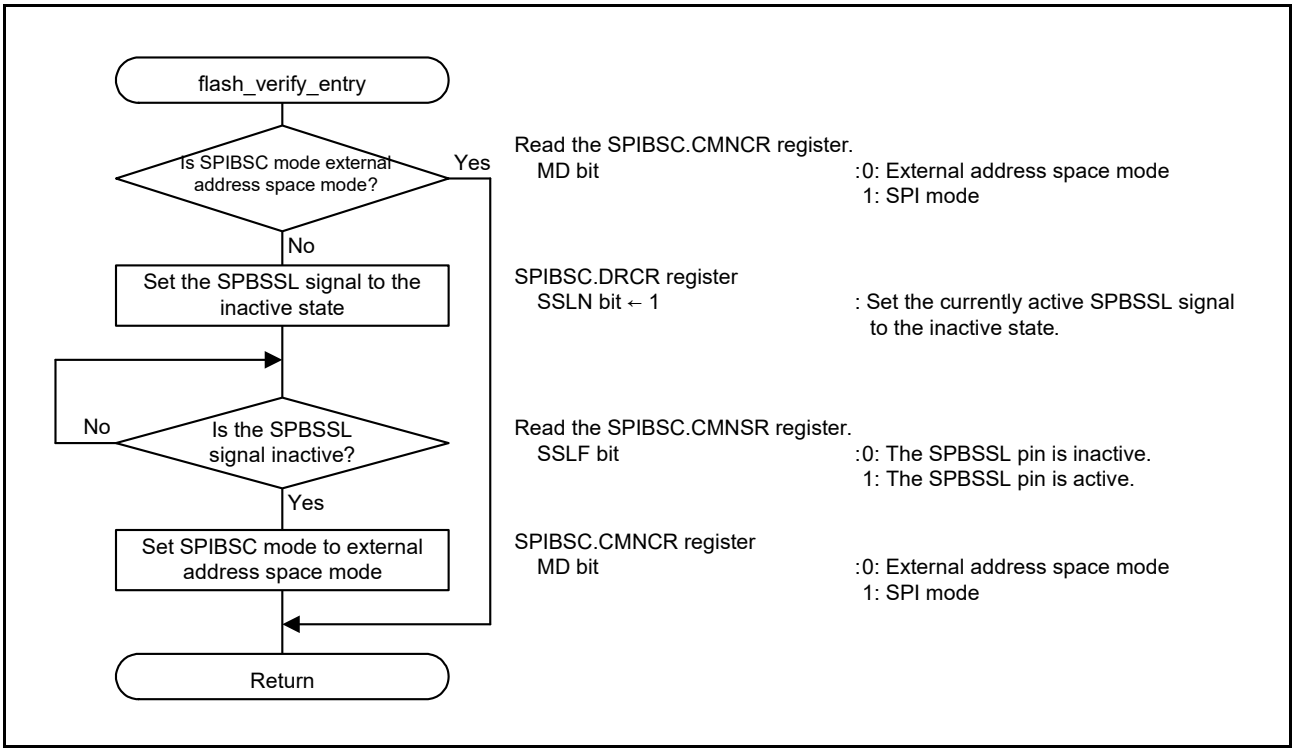


Figure 7.3 Flow of the Serial Flash Memory Read Mode Entry Function

7.7.4 Flow of the Write Interface Function

Figure 7.4 shows the flow of the serial flash memory write function.

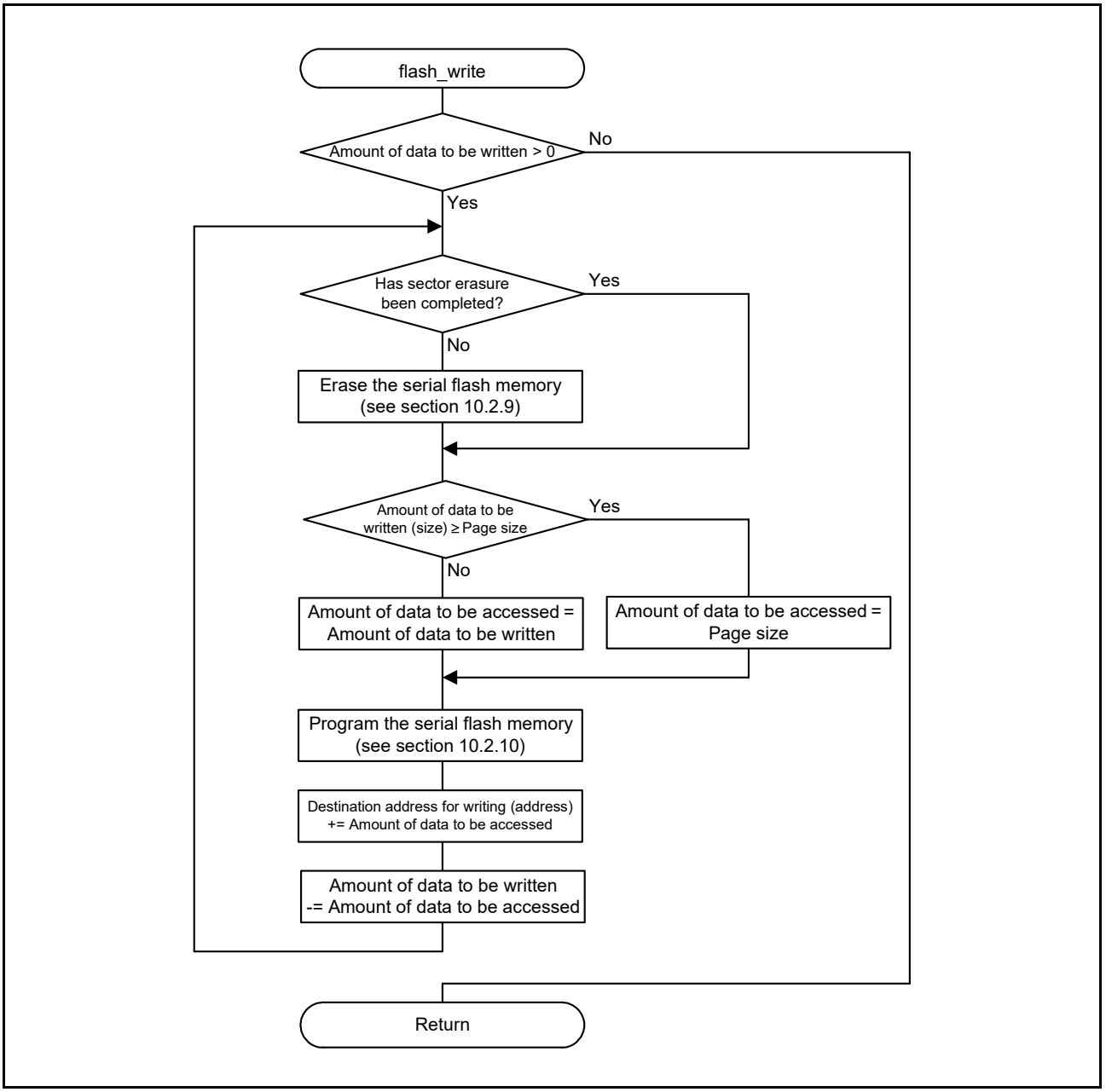


Figure 7.4 Flow of the Serial Flash Memory Write Function

8. Operation of the Flash Downloader

This section describes operation of the flash downloader.

8.1 Memory Allocation of the Application Program

Figure 8.1 shows the memory allocation of the application program which is written by the flash downloader presented in this application note.

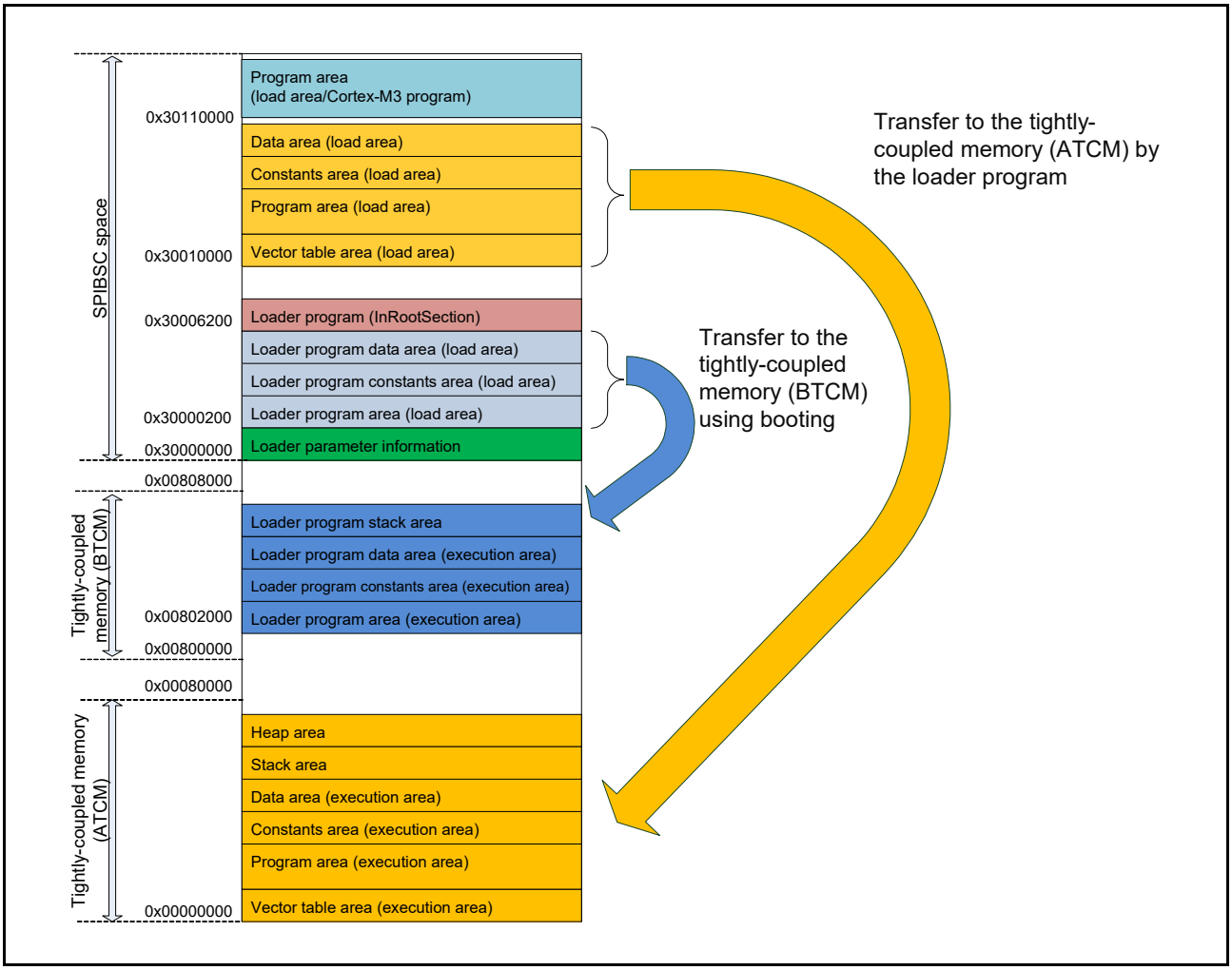


Figure 8.1 Example Memory Allocation of the Application Program

1. The application program has four areas: the areas of the loader parameter information for reference by the RZ/T1 group microcontroller and use by the loader program, the loader program (InRootSection) area, and the application program.
2. Binary data of the four areas are generated as three application binary files*1 by the binary file generator tool from the executable file (axf file) that was generated from the application project.

Note 1. See Table 9.3 for the applications binary files to be generated.

8.2 Flow of Flash Downloader Processing

Figure 8.2 to Figure 8.4 show the flow of processing by the flash downloader.

The flash downloader is loaded by DS-5 to the tightly-coupled memory (ATCM) area of the RZ/T1 group microcontroller and run from the entry point at address 0x00000000. After the flash downloader initializes the stack pointers, it runs __main(), which is the entry function to the main function. The __main() function is supplied as a standard library function of the ARM® compiler; running this function enables the semihosting functionality*1. The flash download main function (flash_main) is run from $Sub$$main(), which is run from the __main function. After flash_main runs, the prg_complete function is run to determine if downloading by the application downloading script (described below) has completed, and processing enters an infinite loop.

Note 1. For details, refer to “ARM® Compiler toolchain Developing Software for ARM® Processors, Embedded Software Development”.

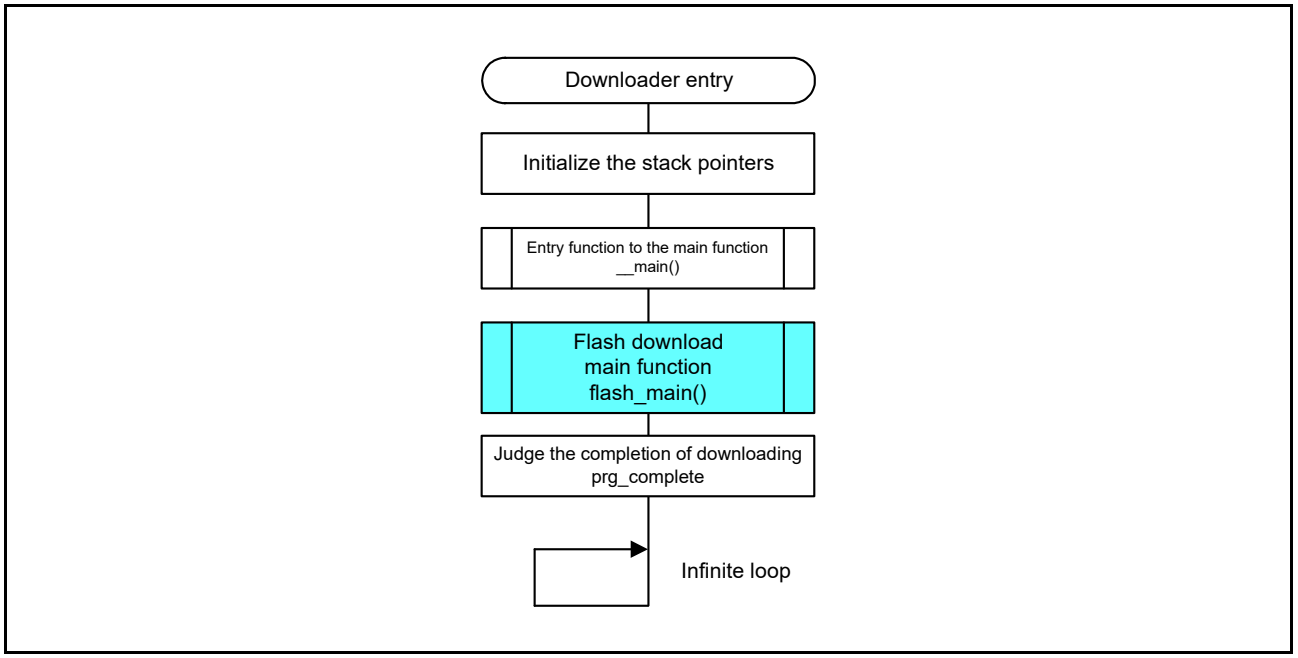


Figure 8.2 Flash Downloader Processing Flow (1/3)

The flash_main function judges whether to perform downloading based on semihosting terminal input. When “Y” is entered via semihosting terminal input, programming of the flash memory starts. When “N” is entered via semihosting terminal input, downloading is judged to have been completed, and the flash_main function ends. Figure 8.3 shows the flow of processing by the flash_main function.

Programming of the flash memory proceeds by calling the RZ_T1_FlashProgram_Sub function. This uses semihosting to read data from an application binary file, and write the data to the serial flash memory. Figure 8.4 shows the flow of processing by the RZ_T1_FlashProgram_Sub function.

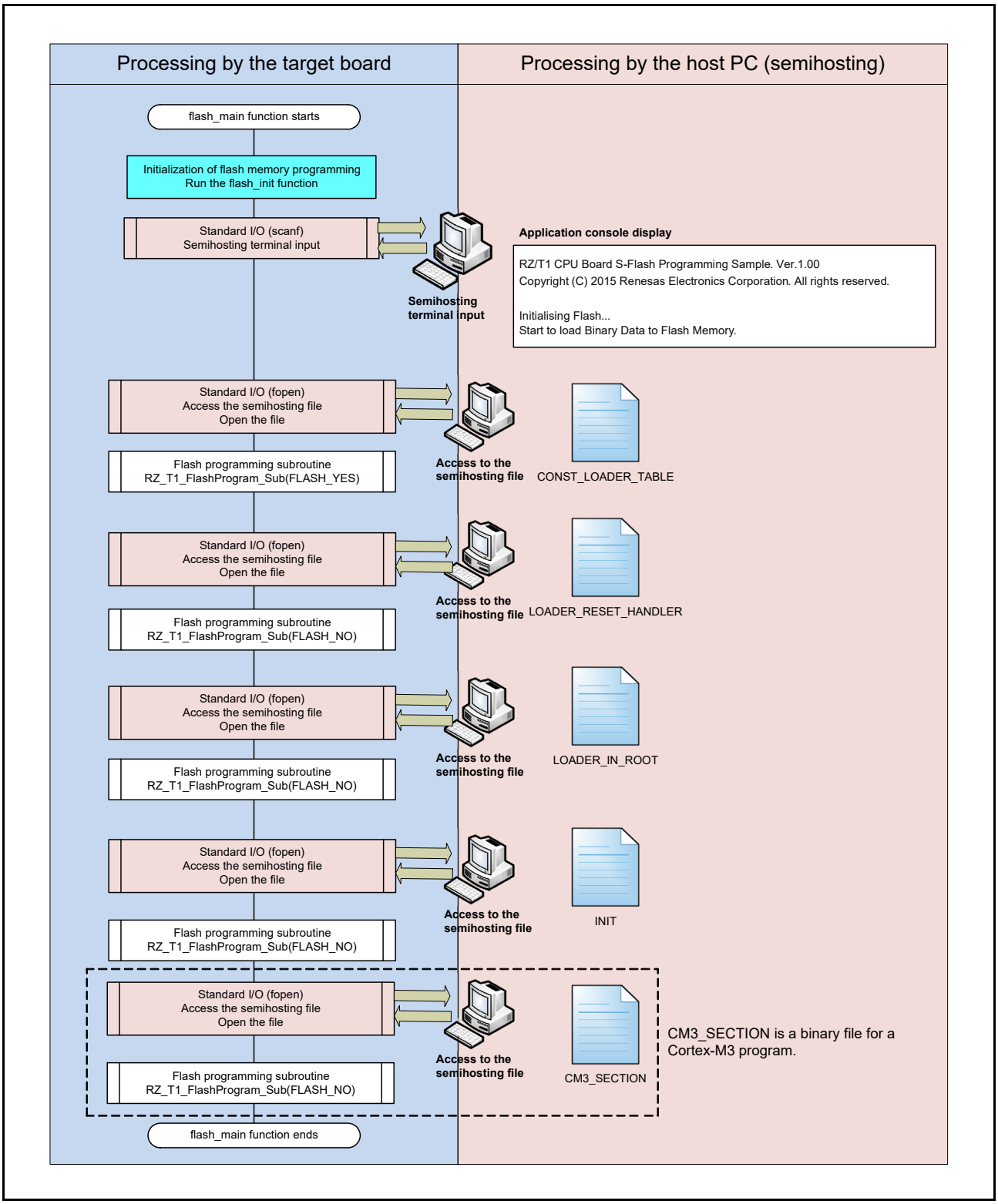


Figure 8.3 Flash Downloader Processing Flow (2/3)

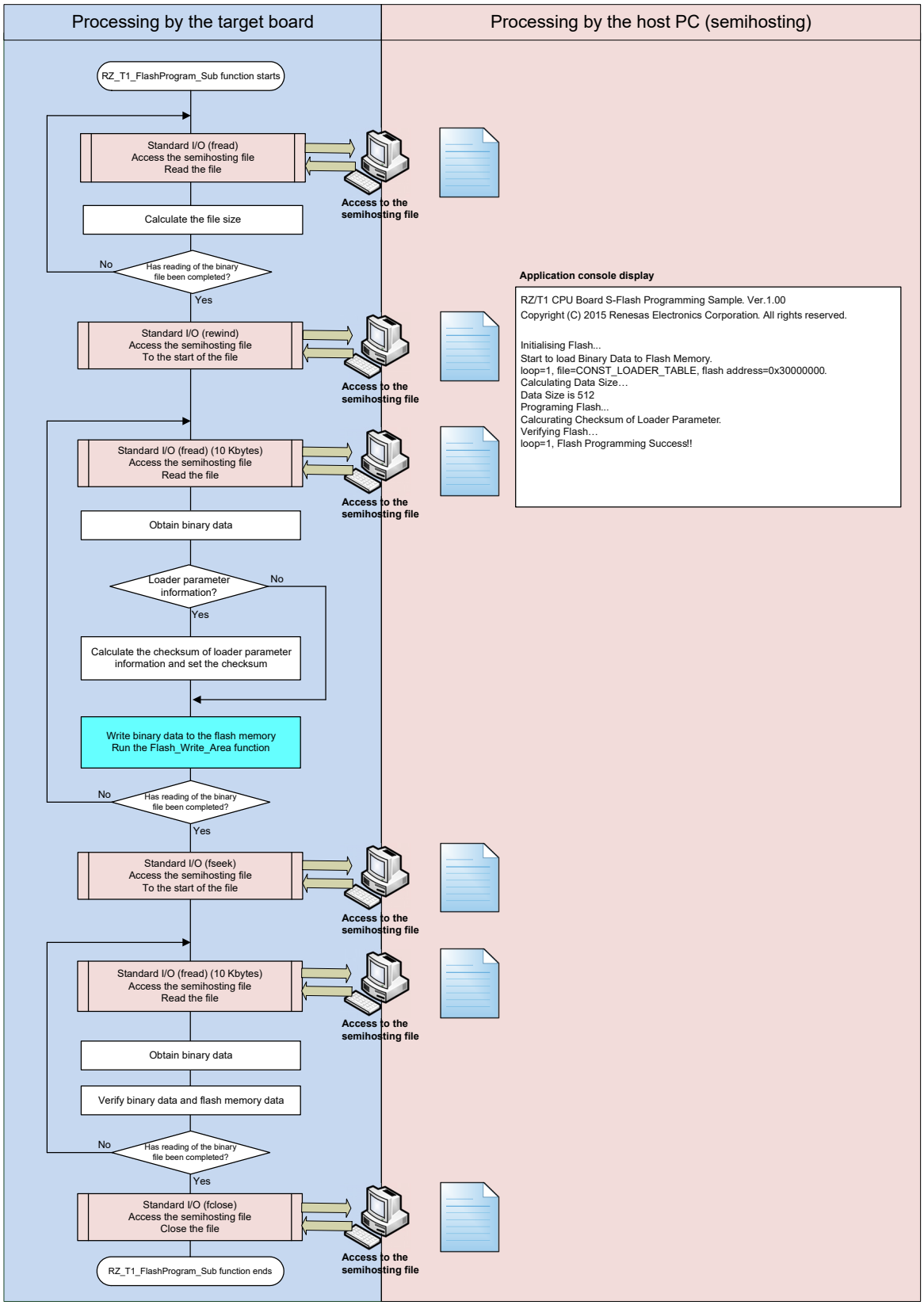


Figure 8.4 Flash Downloader Processing Flow (3/3)

8.2.1 Calculating the Checksum of the Loader Parameter Information

The flash downloader is capable of calculating the checksum of the loader parameter information for reference by the RZ/T1 group microcontroller in booting and writing the result to the flash memory.

If FLASH_YES is specified for the argument check_sum_flag of the RZ_T1_FlashProgram_Sub function for execution, the binary file specified by the argument srcfile is taken as the binary file of loader parameter information and the checksum for the 72 bytes (H'48 bytes) from the start of the binary file is calculated. If the value up to the 72nd byte (byte H'48) from the start of the binary file is H'17320508, the calculated checksum is written to the flash memory as the checksum of the loader parameters. If the value up to the 72nd byte (byte H'48) from the start of the binary file is not H'17320508, the calculated checksum is compared with that for the given data, and if the values do not match, subsequent processing does not proceed and processing is abnormally terminated.

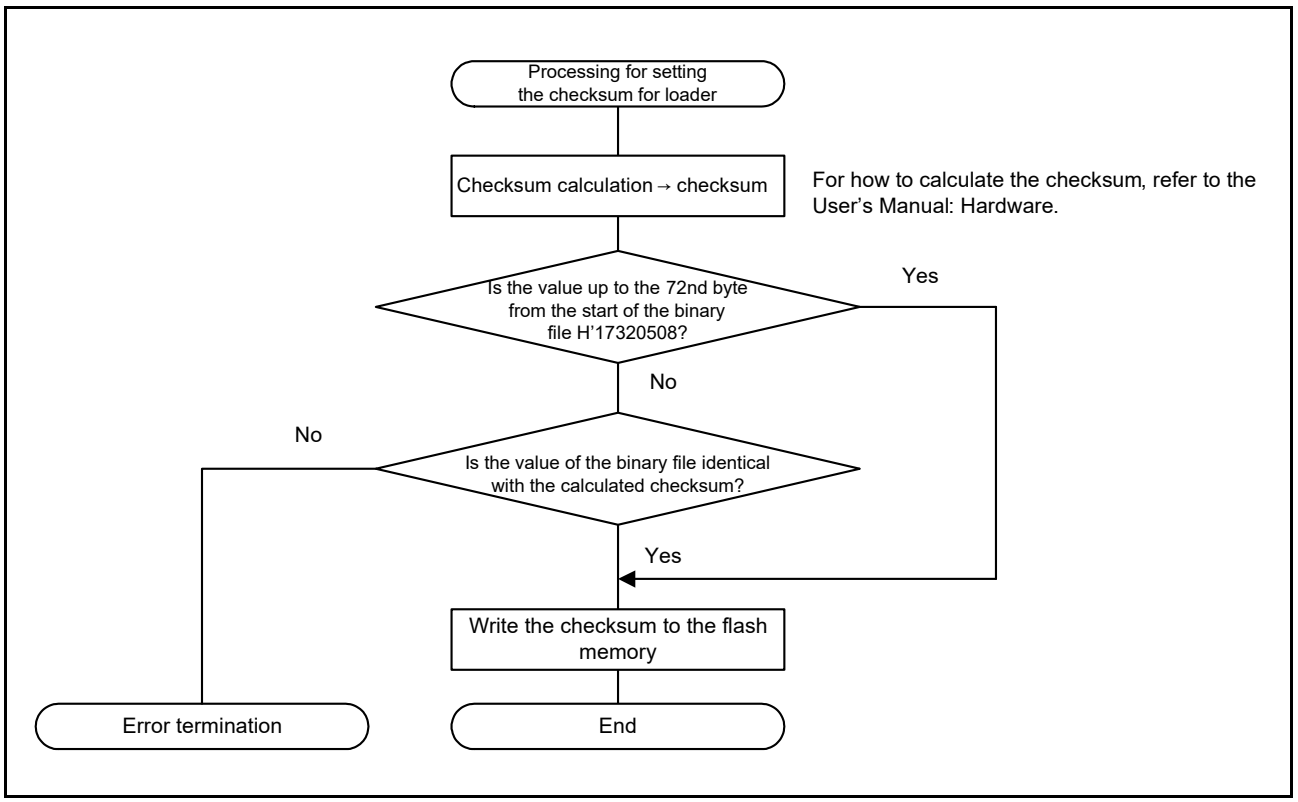


Figure 8.5 Flow of Processing for Setting the Checksum of the Loader Parameter Information

9. Configuration of the Flash Downloader

9.1 Configuration of Projects

The flash downloader comprises the DS-5 projects and DS-5 scripts listed respectively in Table 9.1 and Table 9.2. Table 9.3 lists the application binary files generated from the application project. Follow the flow described in Section 8.2, Flow of Flash Downloader Processing, to write these application binary files to the flash memory.

Table 9.1 List of Projects

Project	Description
RZ_T_fmtool_sflash	This project is used to build the flash downloader. We refer to it as the flash downloader project.
RZ_T_sflash_sample	This project is used to build the application program. We refer to it as the application project. The binary generator tool (fromelf.exe) is used to generate an application binary file from the executable file (axf file) generated by building the project.

Table 9.2 List of Script Files

Script	Description
init_RZ-T.ds	This is the RZ/T1 evaluation board initialization script. This DS-5 script is for executing processing, such as enabling writing to the tightly-coupled memory (ATCM) of the RZ/T1 group microcontroller, when DS-5 is connected to the RZ/T1 evaluation board.
RZ_T_sflash_sample.ds	This is the application downloading script This DS-5 script contains commands for the sequence of operations for writing the application program to the serial flash memory allocated to the external address space (SPI multi-I/O bus space) of the RZ/T1 group microcontroller.
init_RZ-T2.ds	This is the RZ/T1 evaluation board initialization script to be executed from the application downloading script. It is identical to init_RZ-T.ds, except that it does not make settings for the DS-5 memory area.

Table 9.3 List of Application Binary Files

Binary File	Write Start Address*1	Description
CONST_LOADER_TABLE	H'30000000	Application (1) (loader parameter information) binary file
LOADER_RESET_HANDLER	H'30000200	Application (2) (loader program) binary file
LOADER_IN_ROOT	H'30006200	Application (3) (loader program) binary file
INIT	H'30010000	Application (4) (user program) binary file
CM3_SECTION*2	H'30110000	Application (5) (user program) binary file (Cortex-M3 program)

Note 1. In the case of the memory allocation of the application program shown in Figure 8.1.
Note 2. CM3_SECTION is a binary file for a Cortex-M3 program. For details, refer to the application note “RZ/T1 Group Initial Settings of the Microcomputers Incorporating the R-IN Engine”.

9.2 RZ/T1 Evaluation Board Initialization Script

Figure 9.1 shows the details of processing by the RZ/T1 evaluation board initialization script.

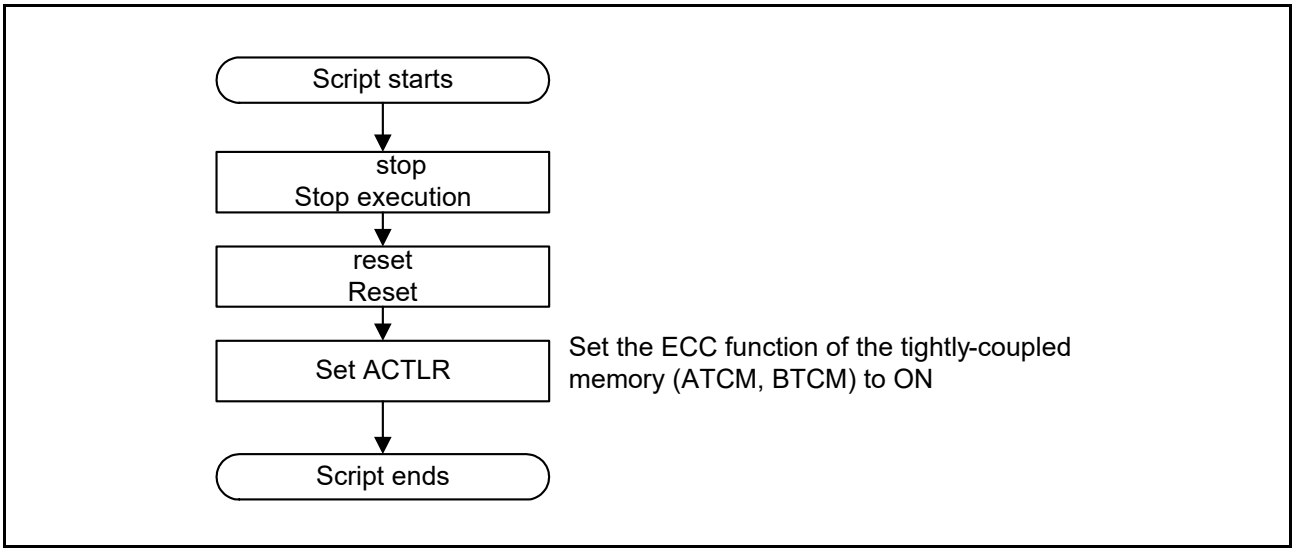


Figure 9.1 Details of Processing by the RZ/T1 Evaluation Board Initialization Script

9.3 Application Downloading Script

Figure 9.2 shows the details of processing by the application downloading script for writing the application program RZ_T_sflash_sample to the serial flash memory allocated to the external address space (SPI multi-I/O bus space) of the RZ/T1 group microcontroller. By running this script from DS-5, the application program RZ_T_sflash_sample is written to the serial flash memory allocated to the external address space (SPI multi-I/O bus space) of the RZ/T1 group microcontroller and the symbol information of the application program RZ_T1_sflash_sample is loaded into DS-5.

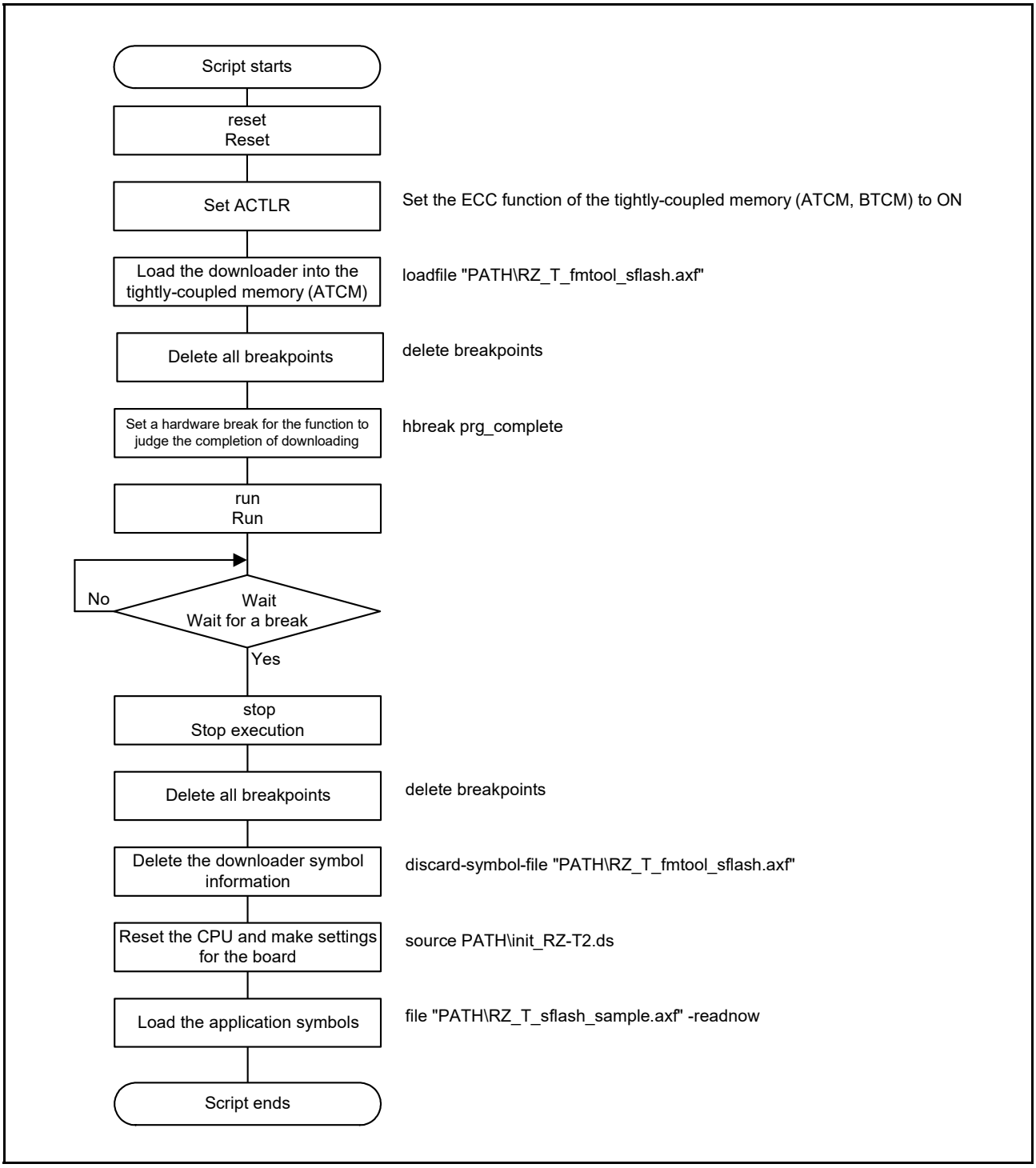


Figure 9.2 Details of Processing by the Application Downloading Script

10. Application Examples

This section describes how the customer can change the binary files for writing to the flash memory and how to customize the sample program to suit the flash memory used by the customer, as an example of the practical application of the sample program.

10.1 Changing the Binary File Names and Destination Addresses for Writing

This section describes how to change the binary file names for writing to the flash memory and destination addresses in flash memory for writing according to the flow described in Section 8.2, Flow of Flash Downloader Processing.

10.1.1 Changing the Binary File Names for Writing to the Flash Memory

The binary file names for writing to the flash memory are in the RZ_T1_FlashProgrammer function in the Flash_Programming.c file. The names of binary files to be written to the flash memory can be changed by changing the names of the binary files in the RZ_T1_FlashProgrammer function.

The current directory when DS-5 semihosting is executed is set by default to the DS-5 workspace directory*1. This allows development of the flash downloader by using relative paths in consideration of it running on another host PC.

Figure 10.1 and Figure 10.2 show examples of implementation.

Note 1. For information on the current directory, refer to “ARM® Compiler toolchain Developing Software for ARM® Processors, Semihosting” provided by ARM®.

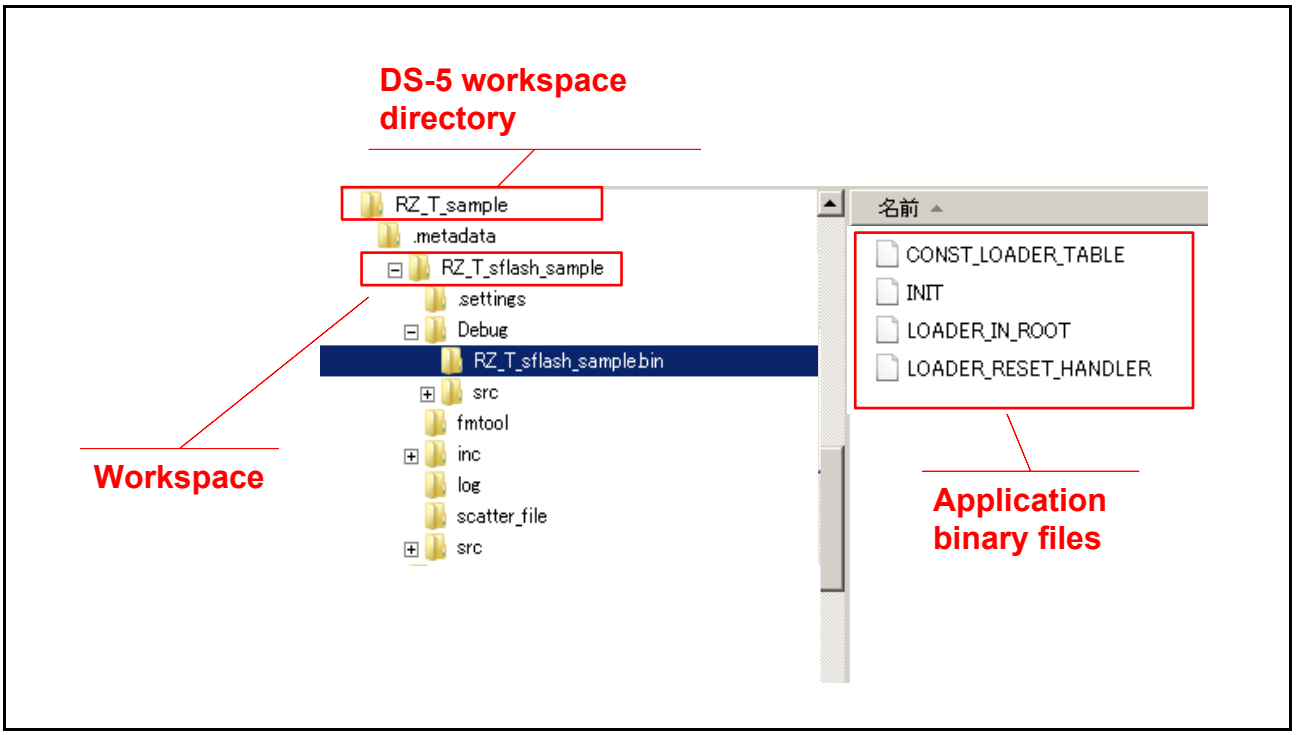


Figure 10.1 Structure of the Directory in the Implementation Example

Before change

```
srcfile = fopen(".¥¥RZ_T_sflash_sample¥¥Debug¥¥RZ_T_sflash_sample.bin¥¥INIT", "r");
if( srcfile == 0 )
{
    printf("loop=%d, Could not open file. Exiting.¥n", loop);
    return;
}
address = (uint32_t *)0x30010000;
printf("loop=%d, file=INIT, flash address=0x%08x.¥n", loop, (uint32_t)address);

ret = RZ_T1_FlashProgram_Sub( srcfile, address, FLASH_NO );
fclose( srcfile );
if( ret != 0 )
{
    printf("loop=%d, Flash Programming Error!!¥n", loop);
    return;
}
```

After change

```
srcfile = fopen(".¥¥RZ_T_sflash_sample¥¥Debug¥¥RZ_T_sflash_sample.bin¥¥INIT2", "r");
if( srcfile == 0 )
{
    printf("loop=%d, Could not open file. Exiting.¥n", loop);
    return;
}
address = (uint32_t *)0x30040000;
printf("loop=%d, file=INIT2, flash address=0x%08x.¥n", loop, (uint32_t)address);

ret = RZ_T1_FlashProgram_Sub( srcfile, address, FLASH_NO );
fclose( srcfile );
if( ret != 0 )
{
    printf("loop=%d, Flash Programming Error!!¥n", loop);
    return;
}
```

Figure 10.2 Implementation Example

10.1.2 Changing the Destination Addresses for Writing to the Flash Memory

As with file pathname input, the destination addresses for writing to the flash memory are in the RZ_T1_FlashProgrammer function in the Flash_Programming.c file. The destination addresses can be changed by changing the addresses for writing in the RZ_T1_FlashProgrammer function.

If you use a “scatter file” to set up the image layout, an application binary file is generated for each load module (LOAD_MODULE). The destination addresses for writing each generated application binary file to the flash memory will depend on the image layout which has been set in the application project.

Figure 10.3 shows an example image layout (scatter file) for the application program RZ_T_sflash_sample with the memory allocation shown in Figure 8.1. Table 10.1 lists the destination addresses where writing to flash memory is to start for the various application binary files to be generated.

For the implementation examples, see Figure 10.1 to Figure 10.3.

```
LOAD_MODULE1 0x30000000    0x00000200
{
     CONST_LOADER_TABLE    0x30000000    FIXED
    {
        * (CONST_LOADER_TABLE)
    }{
}
LOAD_MODULE2 0x30000200    0x00006000
{
    LOADER_RESET_HANDLER    0x00802000    FIXED
    {
        * (LOADER_RESET_HANDLER, +FIRST)
        * (USER_PROG_JUMP)
    }
    Omitted below
}
LOAD_MODULE3 0x30006200    (0x30020000 - 0x30006200)
{
    LOADER_IN_ROOT   0x40006200    FIXED
    {
        * (InRoot$$Sections)
    }
}
LOAD_MODULE4 0x30010000    (0x30110000 - 0x30010000)
{
    INIT             0x00000000    FIXED
    {
        * (VECTOR_TABLE, +FIRST)
        * (RESET_HANDLER)
        * (IRQ_FIQ_HANDLER)
    }
    Omitted below
}
LOAD_MODULE5 0x40120000    (0x34000000 - 0x30110000)
{
    CM3_SECTION      0x30120000    FIXED
    {
        cm3.o(sdram)
    }
}
```

Figure 10.3 Example Image Layout (Scatter File)

Table 10.1 Destination Addresses where Writing to Flash Memory is to Start for the Various Application Binary Files to be Generated

Binary File	Write Start Address	Description
CONST_LOADER_TABLE	H'30000000	Application (1) (loader parameter information) binary file
LOADER_RESET_HANDLER	H'30000200	Application (2) (loader program) binary file
LOADER_IN_ROOT	H'30006200	Application (3) (loader program) binary file
INIT	H'30010000	Application (4) (user program) binary file
CM3_SECTION*1	H'30110000	Application (5) (user program) binary file

Note 1. CM3_SECTION is a binary file for a Cortex-M3 program. For details, refer to the application note “RZ/T1 Group Initial Settings of the Microcomputers Incorporating the R-IN Engine”.

10.2 Changing the Sample Program to Suit the Given Flash Memory

This section describes how to change the sample program to suit the flash memory used by the customer, as an example of the practical application of the sample program.

10.2.1 Conditions for the Sample Program

The sample program makes settings optimized for use of a Macronix serial flash memory (product type name: MX25L51245G) under the conditions listed in Table 10.2.

How to change the sample program when these conditions are to be changed is described below.

Table 10.2 Conditions for the Sample Program

Condition	Settings	Remark
Serial flash memory	Serial flash memory from Macronix (product type name: MX25L51245G)	—
Data bus width	4 bits	Bit width for reading data
	1 bit	Bit width for writing data
Number of address bytes	4 bytes	Number of bytes to be issued when specifying addresses

10.2.2 Changing the Sample Program when Not Changing the Serial Flash Memory

Table 10.3 lists how to change the sample program when the serial flash memory in use is not to be changed.

Table 10.3 How to Change the Sample Program when the Serial Flash Memory is not to be Changed

Condition	Changes	How to Make Changes
Data bus width for reading data	1 bit	Define (1) in the macro definition (SPIBSC_BUS_WITDH)*1.
	4 bits	Define (4) in the macro definition (SPIBSC_BUS_WITDH).
Number of address bytes	3 bytes	Define (SPIBSC_OUTPUT_ADDR_24) in the macro definition (SPIBSC_OUTPUT_ADDR)*2.
	4 bytes	Define (SPIBSC_OUTPUT_ADDR_32) in the macro definition (SPIBSC_OUTPUT_ADDR).

Note 1. The macro definition (SPIBSC_BUS_WITDH) is defined in the spibsc_ioset_userdef.c file.
Note 2. The macro definition (SPIBSC_OUTPUT_ADDR) is defined in the spibsc_ioset_userdef.c file.

10.2.3 Changing the Sample Program when Changing the Serial Flash Memory

When changing the serial flash memory, the sample program must be changed to suit the specifications of the flash memory in use.

Table 10.4 lists the points to be changed in the sample program.

Table 10.4 Points to be Changed in the Sample Program

Points to be Changed	Description
Read command waveform	In external address space read mode, change the signal output to the serial flash memory when converting read operations for the SPI multi-I/O bus space to SPI communications to match the read command of the serial flash memory you are using.
Register settings in the serial flash memory	Make settings for the registers in the serial flash memory required when using the SPIBSC in external address read mode, to suit the serial flash memory in use.
Enabling writing to the serial flash memory	Set the registers in the serial flash memory to enable writing, to suit the serial flash memory in use.*1
Waiting for the serial flash memory to be ready	Read the registers in the serial flash memory and wait for the serial flash memory to enter the ready state, to suit the flash memory in use.
Releasing the serial flash memory from protection	Make settings for the registers in the serial flash memory to release it from protection, to suit the flash memory in use.*2
Erasure of the serial flash memory	Erase the sector of the serial flash memory to suit the flash memory in use.
Programming of the serial flash memory	Program the serial flash memory to suit the flash memory in use.

Note 1. Enabling of writing may be required to set the registers in the serial flash memory depending on the serial flash memory in use
Note 2. Release from protection may be required to set the registers in the serial flash memory depending on the serial flash memory in use.

10.2.4 Changing the Read Command Waveforms

In external address space read mode, change the signal output to the serial flash memory when converting read operations for the SPI multi-I/O bus space to SPI communications to match the read command of the serial flash memory you are using.

The signal output to the serial flash memory in external address space read mode is changed by a setting in an SPIBSC control register.

In the sample program, the values set in the SPIBSC control register can be changed by a global variable (SPIBSC external address read setting storage function: Userdef_SPIBSC_Set_Config). A user-defined function (SPIBSC external address read settings function: Userdef_SPIBSC_Set_Config) makes settings for spibsc_cfg.

Figure 10.4 shows the relationship between the SPIBSC control register settings and the waveforms output to the serial flash memory while the SPIBSC is reading from an external address, and Table 10.5 lists the settings of the SPIBSC control register in the sample program.

Refer to these example settings to make settings for spibsc_cfg to match the read command of the serial flash memory in use.

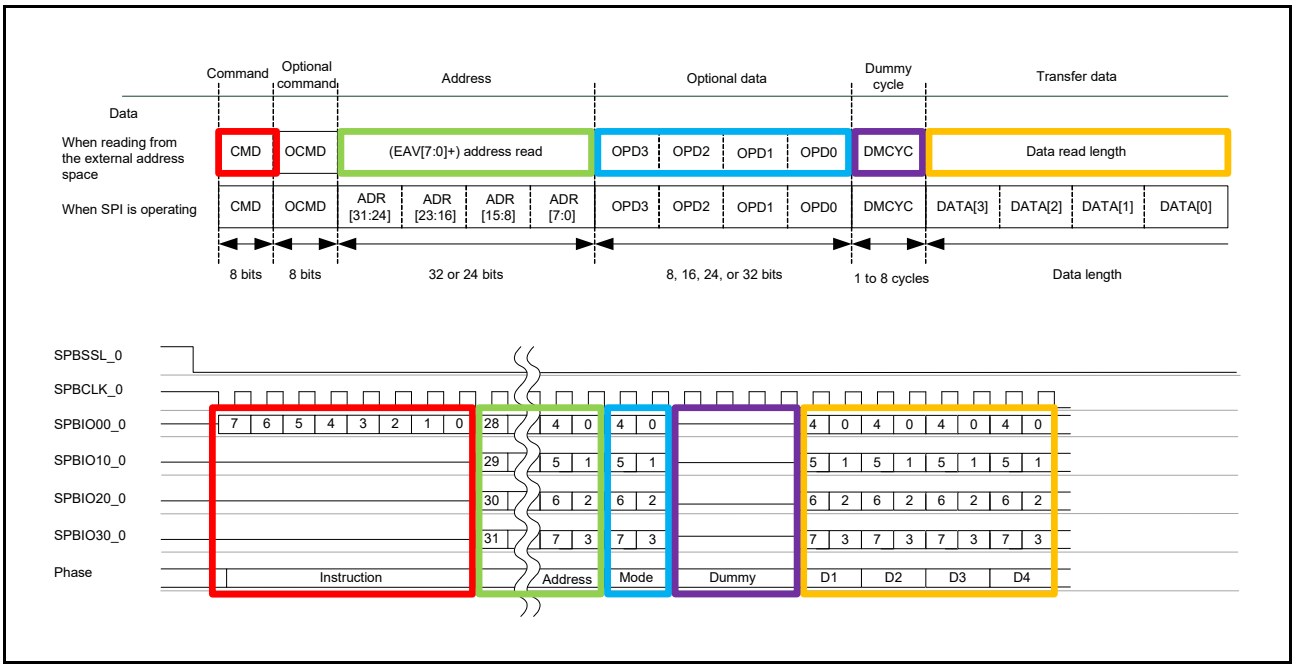


Figure 10.4 Relationship between the SPIBSC Control Register Settings and the Waveforms Output to the Serial Flash Memory while the SPIBSC is Reading from an External Address

Table 10.5 SPIBSC Control Register Settings in the Sample Program

SPIBSC Registers		Setting	Remark
DRCMR	CMD[7:0]	H’EB	Quad I/O Read Command
	OCMD[7:0]	H’00	—
DROPR	OPD3[7:0]	H’00	—
	OPD2[7:0]	H’00	—
	OPD1[7:0]	H’00	—
	OPD0[7:0]	H’00	—
DRENR	CDB[1:0]	B’00	Command bit width: 1-bit width
	OCDB[3:0]	B’0000	—
	ADB[1:0]	B’10	Address bit width: 4-bit width
	OPDB[1:0]	B’10	Optional data bit width: 4-bit width
	DRDB[1:0]	B’10	Data read bit width: 4-bit width
	DME	B’1	Dummy cycles: Inserted
	CDE	B’1	Commands: Issued
	OCDE	B’0	Optional command: Not issued
	ADE[3:0]	B’0111	Address enable: 24-bit address output
	OPDE[3:0]	B’1000	Optional data: OPD3 is output
DRDMCR	DMDB[1:0]	B’00	Bit width of dummy cycles: 1-bit width
	DMCYC[2:0]	B’101	Number of dummy cycles: 6 cycles
SPBCR	SPBR[7:0]	H’01	Bit rate: PCLKA/2
	BRDV[1:0]	B’00	

10.2.5 Setting Registers in the Serial Flash Memory

The registers in the serial flash memory must be set when reading from the serial flash memory in Section 10.2.4, Changing the Read Command Waveforms.

In the sample program, the user-defined function (serial flash memory internal register settings function: Userdef_SFLASH_Set_Mode) handles the processing to set the QUAD bit in the status register of the Macronix serial flash memory (product type name: MX25L51245G) to 1 (= quad), the DC1 (dummy cycle 1) bit in the configuration register to 1, and the DC0 (dummy cycle 0) bit to 0.

When setting the registers in the serial flash memory, SPI mode of the SPIBSC is used. To set the registers in the Macronix serial flash memory (product type name: MX25L51245G), the write enable latch (WEL) bit must be set to 1 by issuing the write enable command (WREN). This must be done before the status register and configuration register can be set. In the sample program, the user-defined function (serial flash memory write enable function: Userdef_SFLASH_Write_Enable) handles the processing to issue the write enable command (WREN).

Figure 10.5 shows the flow of setting the registers in the serial flash memory in the sample program.

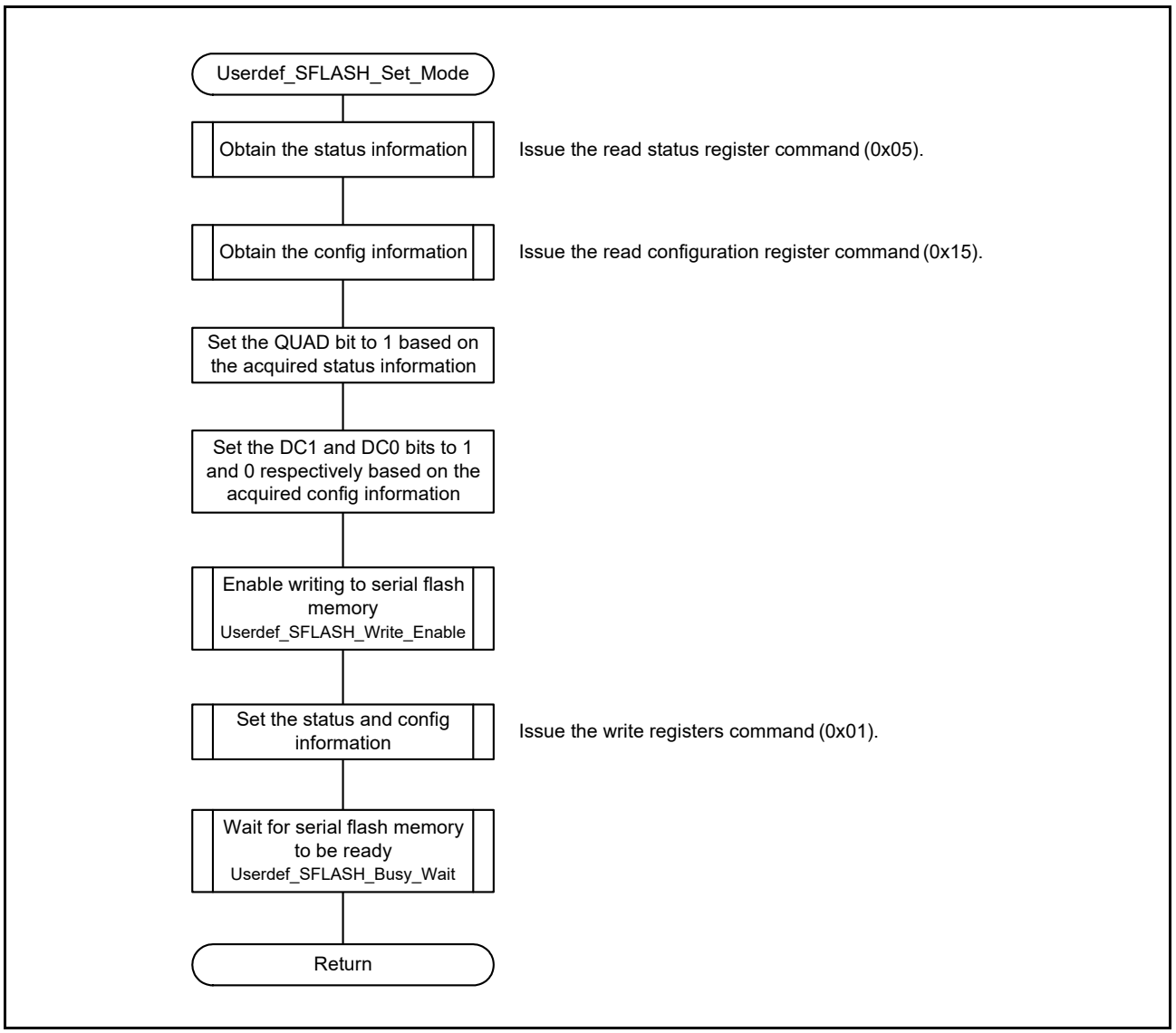


Figure 10.5 Flow of Setting the Registers in the Serial Flash Memory

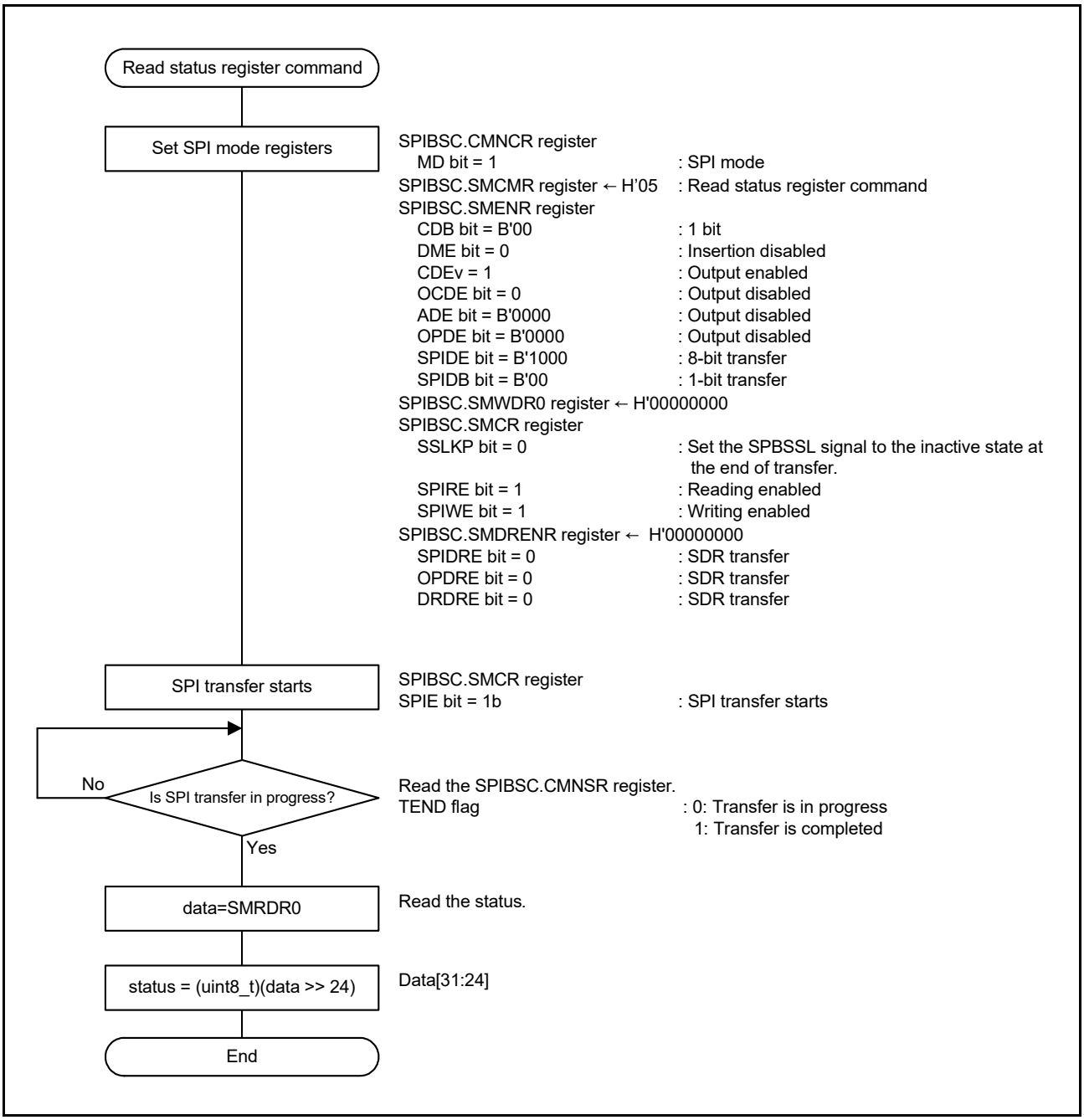


Figure 10.6 Read Status Register Command Flow

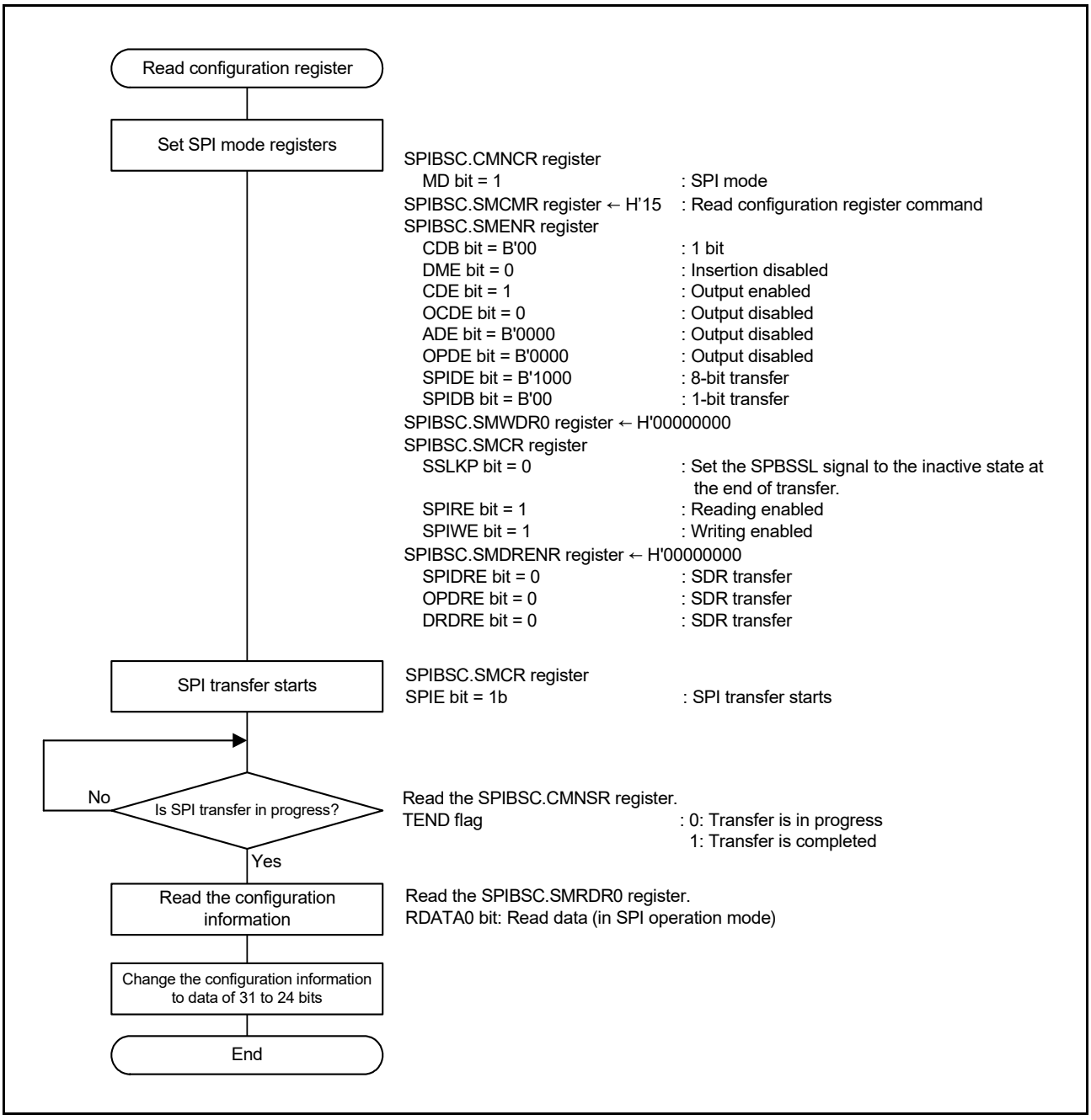


Figure 10.7 Read Configuration Register Command Flow

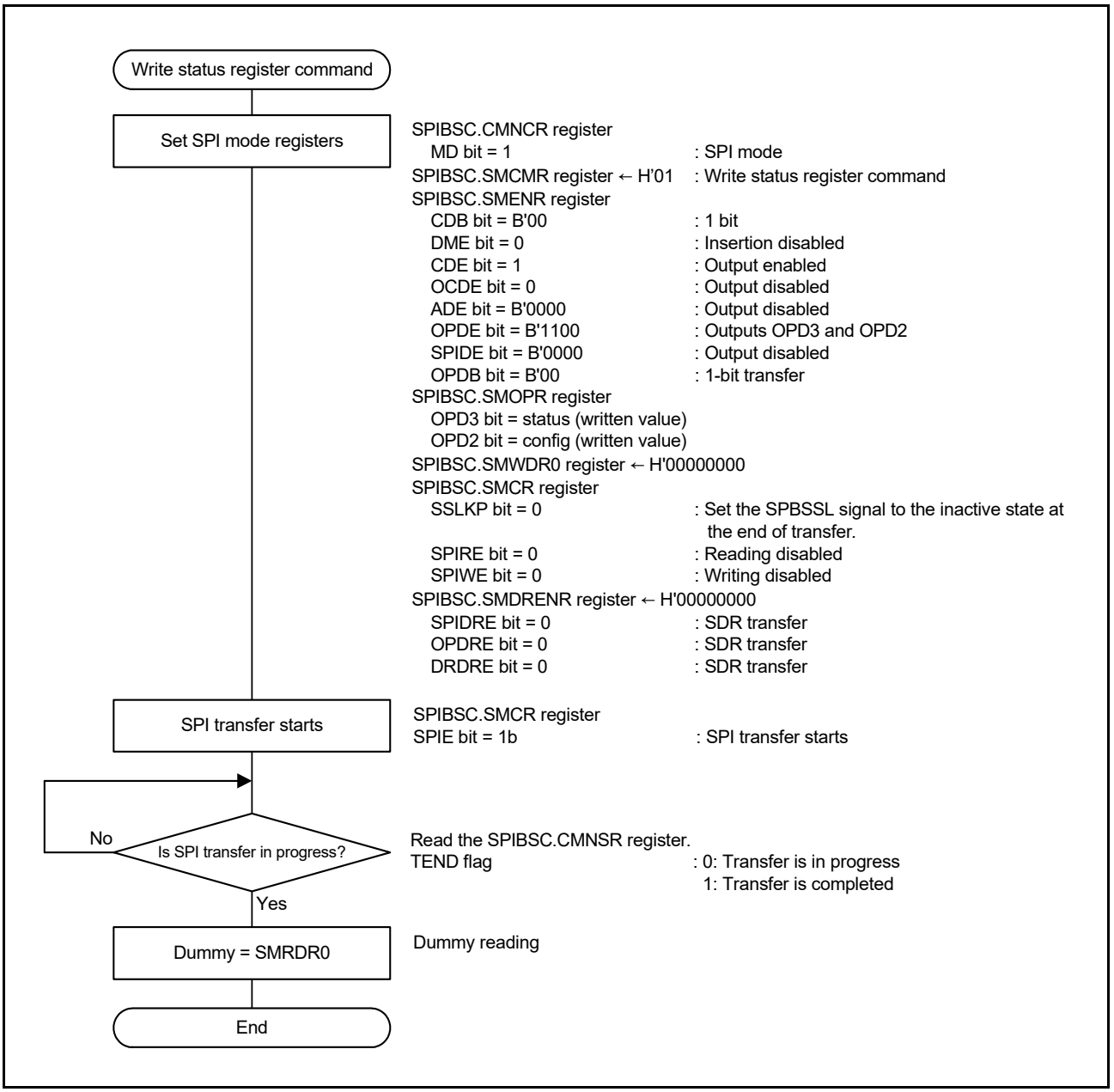


Figure 10.8 Write Status Register Command Flow

10.2.6 Enabling Writing to the Serial Flash Memory

Write the code to enabling writing, which is required for setting the registers in the serial flash memory, as described in Section 10.2.5, Setting Registers in the Serial Flash Memory, in a user-defined function (serial flash memory write enable function: Userdef_SFLASH_Write_Enable).

Figure 10.9 shows the flow of enabling writing to the serial flash memory in the sample program.

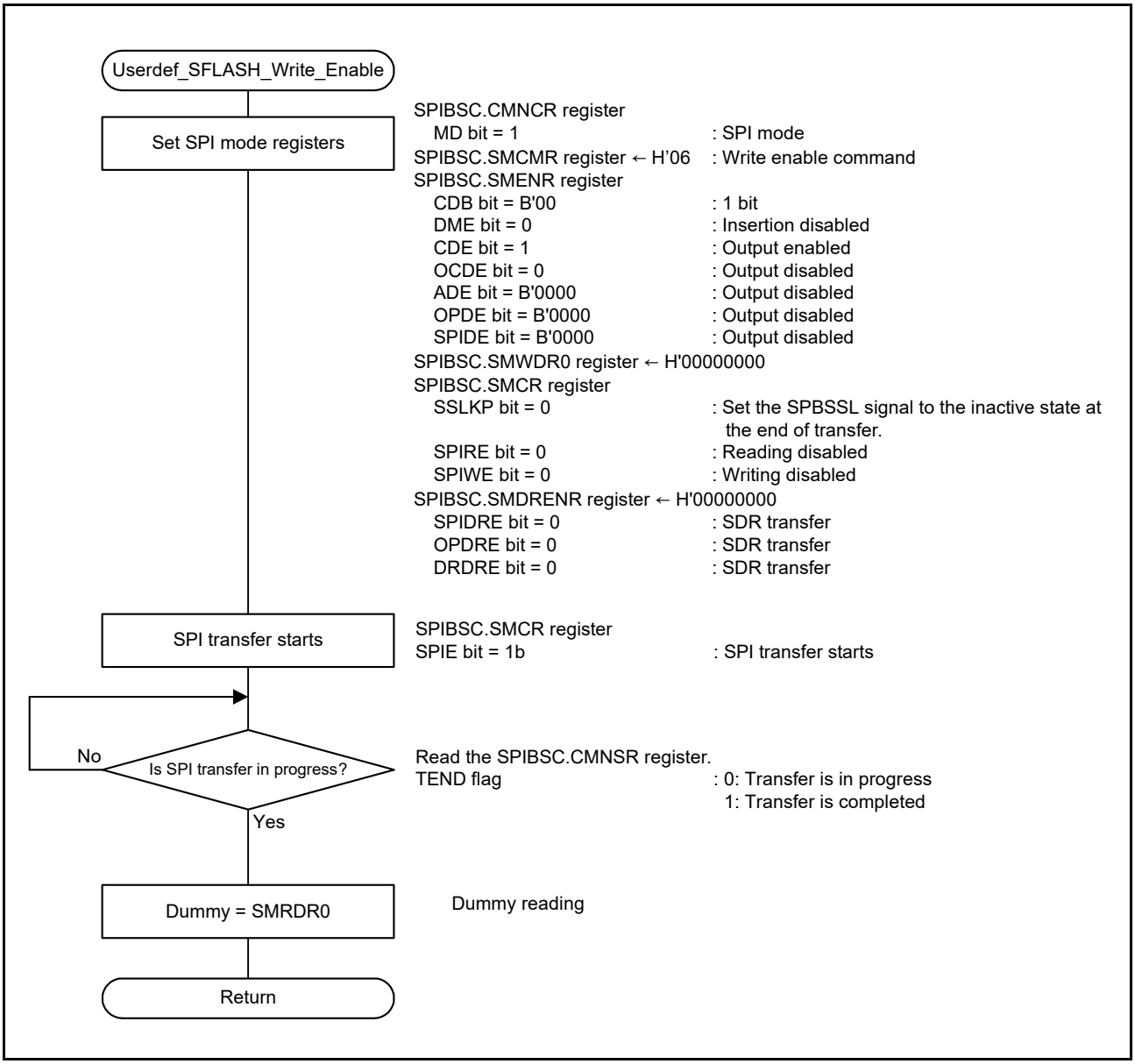


Figure 10.9 Flow of Enabling Writing to the Serial Flash Memory

10.2.7 Waiting for the Serial Flash Memory to be Ready

Issuing a programming command (page programming) or erase command (sector erase) to the serial flash memory leads to the serial flash memory being placed in the busy state. Write the code to wait for the transition from the busy state to the ready state in a user-defined function (serial flash memory ready wait function: Userdef_SFLASH_Busy_Wait).

With the Macronix serial flash memory (product type name: MX25L51245G), you can check for a transition to the ready state by reading a register in the serial flash memory.

Figure 10.10 shows the flow of waiting for the serial flash memory to be ready in the sample program.

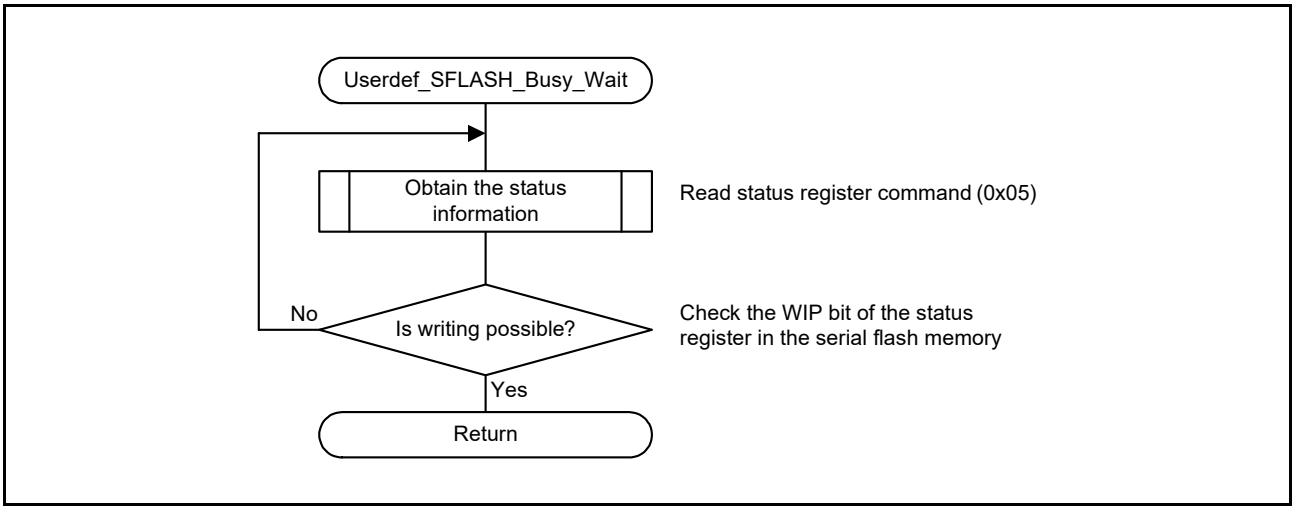


Figure 10.10 Flow of Waiting for the Serial Flash Memory to be Ready

10.2.8 Releasing the Serial Flash Memory from Protection

When changing the data in the serial flash memory in accord with the specifications of the serial flash memory, it must be released from protection by writing to registers in the serial flash memory.

With the Macronix serial flash memory (product type name: MX25L51245G), programming or erasure of the serial flash memory cannot proceed if it is in the protected state. To release it from protection, the block protection (BP0, BP1, BP2, and BP3) bits in the status register must be set to 0.

Figure 10.11 shows the flow of releasing the serial flash memory from protection in the sample program.

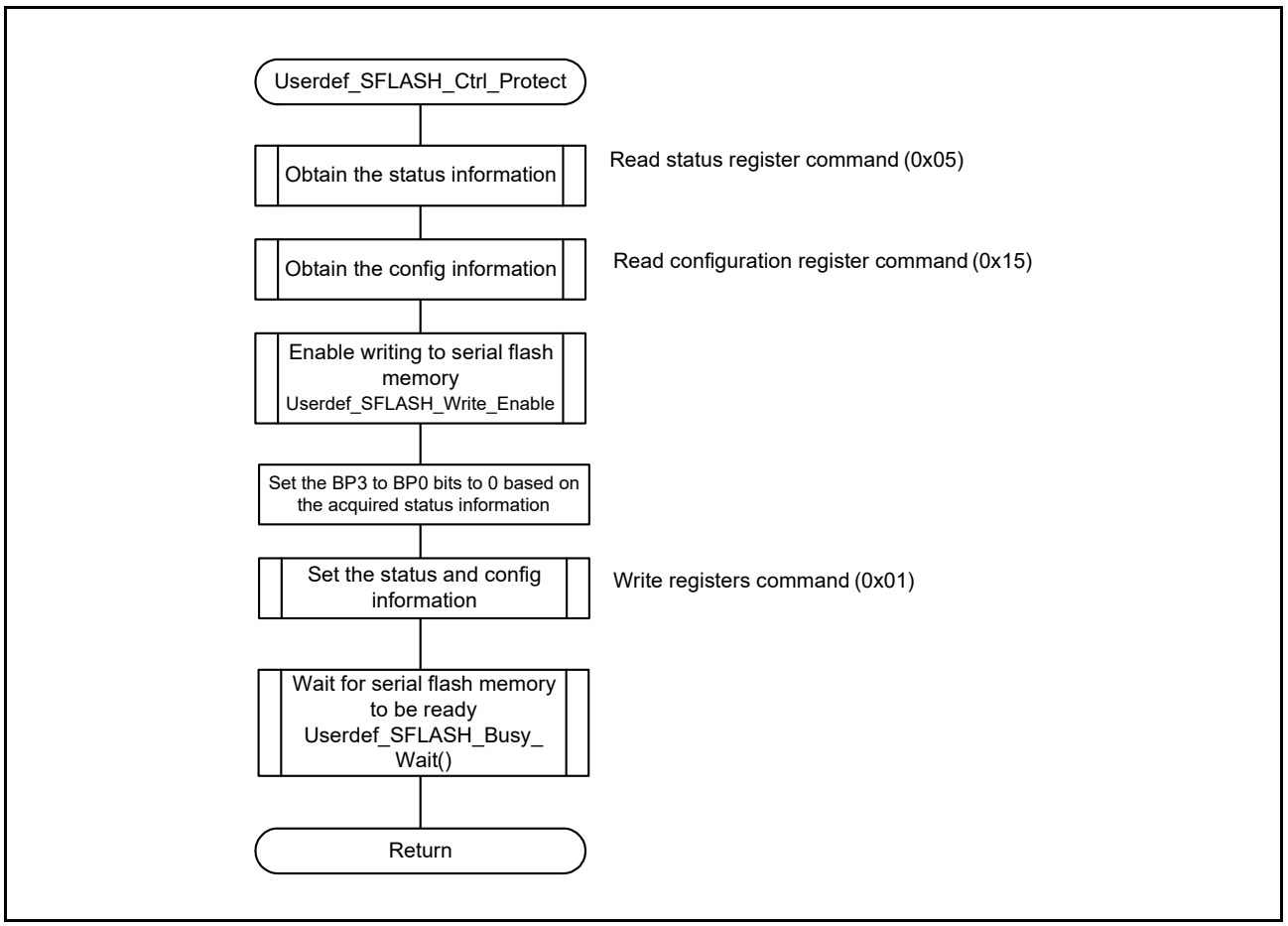


Figure 10.11 Flow of Releasing the Serial Flash Memory from Protection

10.2.9 Erasing the Serial Flash Memory

The sample program uses the erase command (sector erase) to erase the sector of the serial flash memory.

When the erase command has to be changed in accord with the specifications of the serial flash memory, change the flash erasure function (R_SFLASH_EraseSector).

Figure 10.12 shows the flow of erasing the serial flash memory in the sample program.

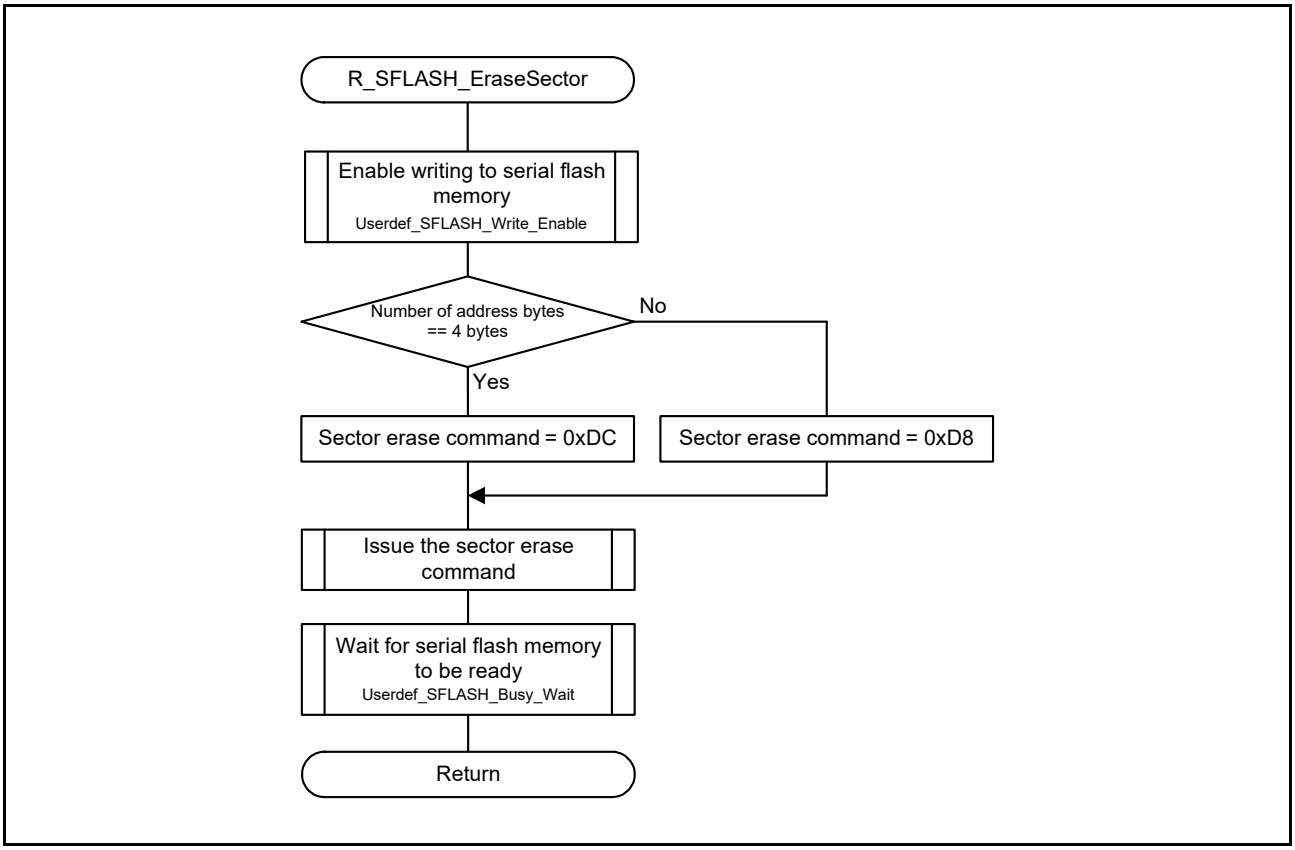


Figure 10.12 Flow of Erasing the Serial Flash Memory

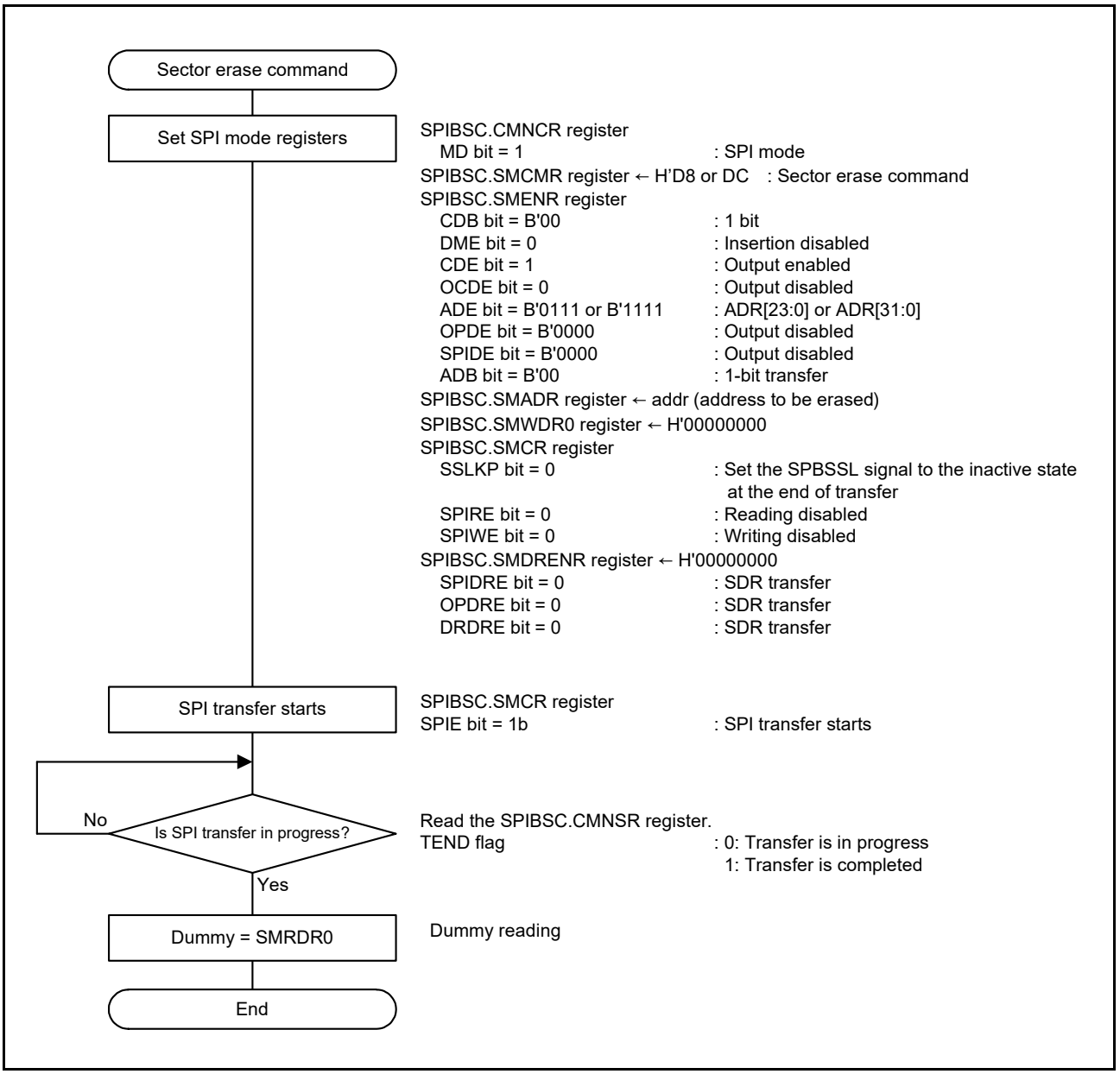


Figure 10.13 Sector Erase Command Flow

10.2.10 Programming the Serial Flash Memory

The sample program uses the programming command (page programming) for programming of the serial flash memory.

When the programming command has to be changed in accord with the specifications of the serial flash memory, change the serial flash programming function (R_SFLASH_ByteProgram).

Figure 10.14 shows the flow of programming the serial flash memory in the sample program.

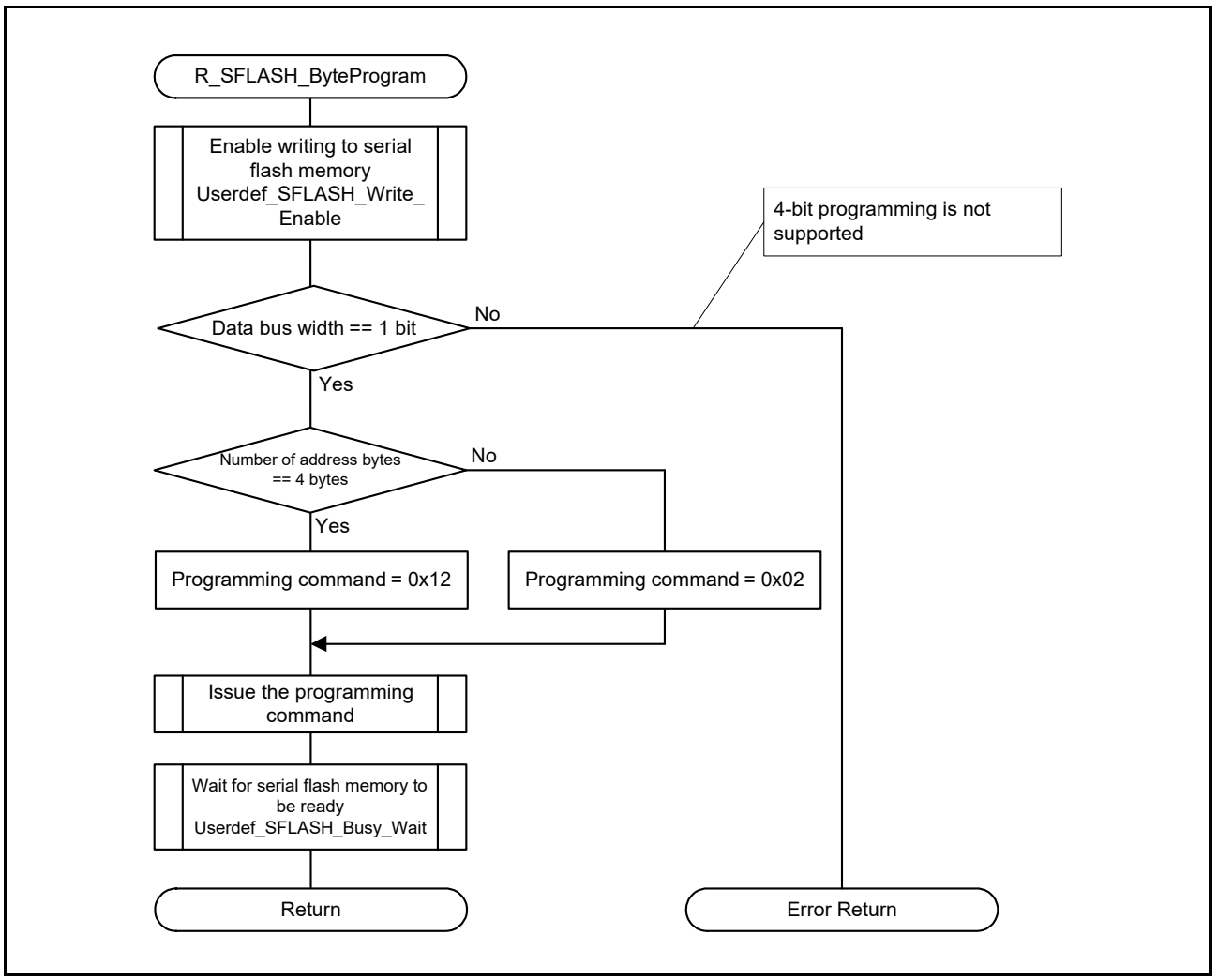


Figure 10.14 Flow of Programming the Serial Flash Memory

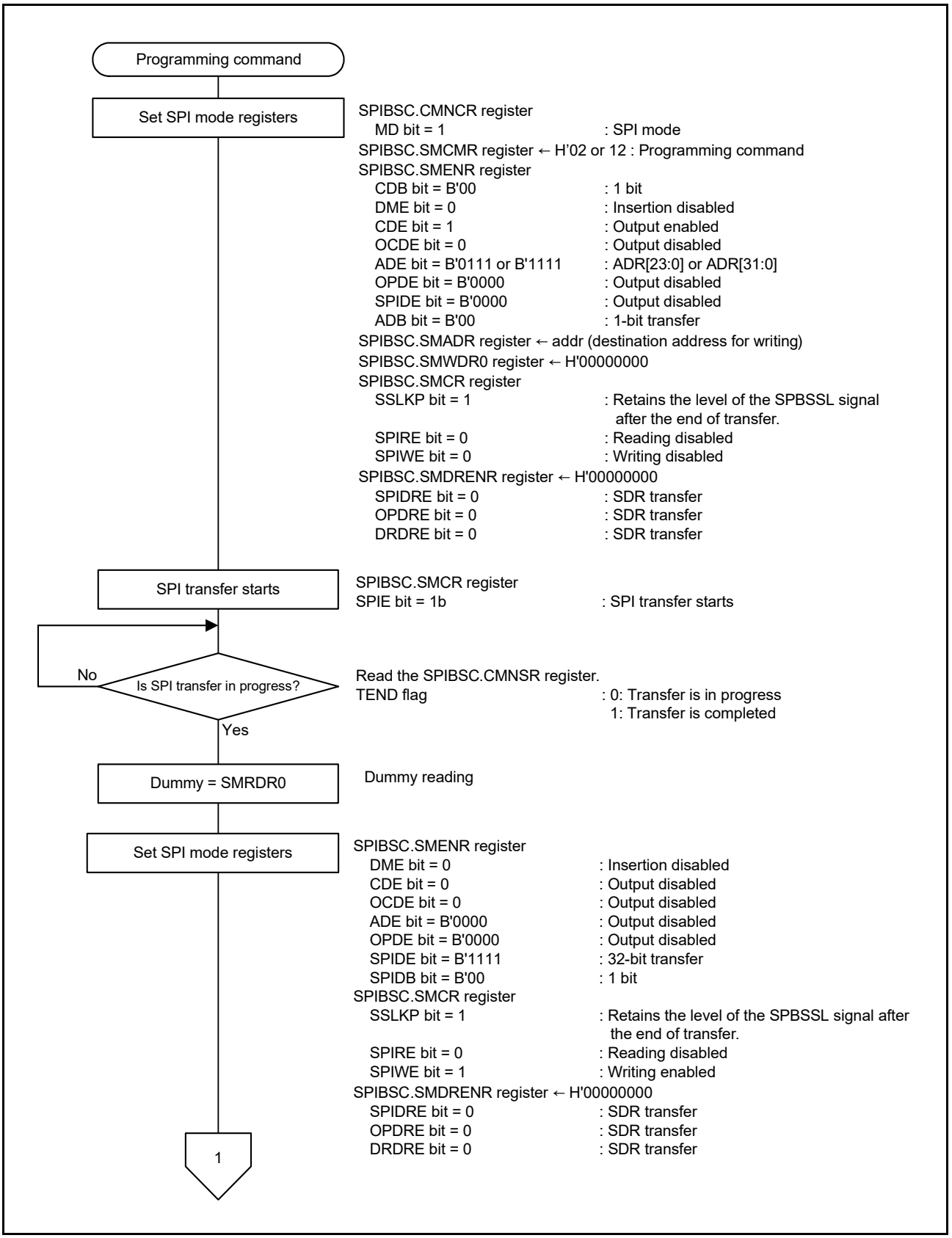


Figure 10.15 Programming Command Flow (1)

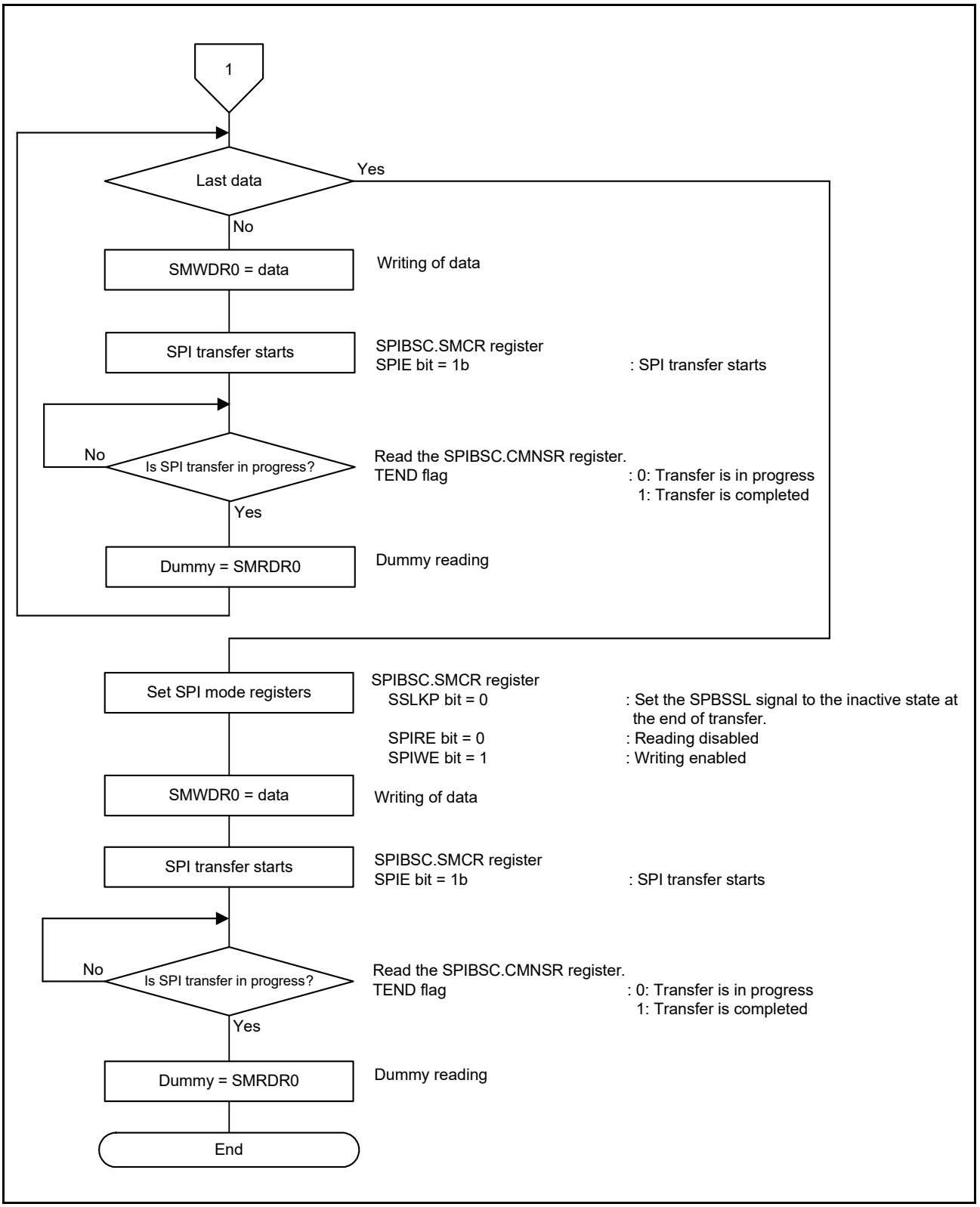


Figure 10.16 Programming Command Flow (2)

10.3 Customizing the Sample Program for Initial Settings of the Microcomputers Incorporating the R-IN Engine (Cortex-M3)

In the sample program for initial settings of devices with a built-in R-IN Engine, CM3_SECTION is generated when an application binary file is generated. In downloading, a binary file for the Cortex-M3 is also downloaded as processing for devices with a built-in R-IN Engine (loop = 5).

For details of initial settings of devices with a built-in R-IN Engine, refer to the application note “RZ/T1 Group Initial Settings of the Microcomputers Incorporating the R-IN Engine”.

Table 10.6 Application Binary Files

Directory	Binary File	Description
RZ_T_sflash_sample.bin	CONST_LOADER_TABLE	Application (1) (loader parameter information) binary file
	LOADER_RESET_HANDLER	Application (2) (loader program) binary file
	LOADER_IN_ROOT	Application (3) (loader program) binary file
	INIT	Application (4) (user program) binary file
	CM3_SECTION	Application (5) (user program) binary file (Cortex-M3 program)

```
RZ/T1 CPU Board S-Flash Programming Sample. Ver.1.00
Copyright (C) 2015 Renesas Electronics Corporation. All rights reserved.

Initializing Flash...
Start to load Binary Data to Flash Memory.
loop=1, file=CONST_LOADER_TABLE, flash address=0x30000000.
Calculating Data Size...
Data Size is 76
Programing Flash...
Calcurating Checksum of Loader Parameter.
Verifying Flash...
loop=1, Flash Programming Success!!
loop=2, file=LOADER_RESET_HANDLER, flash address=0x30000200.
Calculating Data Size...
Data Size is 11812
Programing Flash...
Verifying Flash...
loop=2, Flash Programming Success!!
loop=3, file=LOADER_IN_ROOT, flash address=0x30006200.
Calculating Data Size...
Data Size is 236
Programing Flash...
Verifying Flash...
loop=3, Flash Programming Success!!
loop=4, file=INIT, flash address=0x30010000.
Calculating Data Size...
Data Size is 2592
Programing Flash...
Verifying Flash...
loop=4, Flash Programming Success!!
loop=5, file=CM3_SECTION, flash address=0x30110000.
Calculating Data Size...
Data Size is 1296
Programing Flash...
Verifying Flash...
loop=5, Flash Programming Success!!

 finish
Flash Programming Complete
```

Figure 10.17 Messages Output to the Application Console

11. Sample Program

The sample program is available from the Renesas Electronics website.

12. Documents for Reference

- User's Manual: Hardware
  RZ/T1 Group User's Manual: Hardware
  (Download the latest version from the Renesas Electronics website.)

  RZ/T1 Evaluation Board RTK7910022C00000BR User's Manual
  (Download the latest version from the Renesas Electronics website.)

  ARM Architecture Reference Manual ARMv7-A and ARMv7-R edition Issue C
  (Download the latest version from the ARM® website.)

  ARM Generic Interrupt Controller Architecture Specification Architecture version 1.0
  (Download the latest version from the ARM® website.)

- Technical Update and Technical News
  (Download the latest version from the Renesas Electronics website.)

- User's Manual: Development Environment
  For the ARM software development tools (ARM Compiler toolchain, ARM DS-5, etc.), visit the ARM® website.
  (Download the latest version from the ARM® website.)

Website and Support

Renesas Electronics website

http://www.renesas.com/

Inquiries

http://www.renesas.com/inquiry

Revision History	Application Note: Example of Downloading to Serial Flash Memory by Using "Semihosting" of ARM® Development Studio 5 (DS-5™)

Rev.	Date	Description	
		Page	Summary
1.00	Nov. 24, 2015	—	First Edition issued
1.10	Sep. 16, 2016	All	Application Note "RZ/T1 Group Dual Core Control" changed to Application Note "RZ/T1 Group Initial Settings of the Microcomputers Incorporating the R-IN Engine"
1.20	Sep. 15, 2017	2. Conditions for Checking Operations	
		5	Table 2.1 Conditions for Checking Operations DS-5 version from ARM®, modified

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 document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Handling of Unused Pins

   Handle unused pins in accordance 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 a product with a different part number, confirm that the change will not lead to problems.

   — The characteristics of an MPU or MCU in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

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 or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other disputes involving 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, including but not limited to, the product data, drawing, chart, program, algorithm, application examples.
3. 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 shall not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics products.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended 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 (traffic lights); large-scale communication equipment; key financial terminal systems; safety control 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 (space and undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability 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. When using the Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat radiation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions or failure or accident 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 Renesas Electronics 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 ensure to implement safety measures to guard them against the possibility of bodily injury, injury or damage caused by fire, and social damage in the event of failure or malfunction of Renesas Electronics products, 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 by your own responsibility as warranty for your products/system. Because the evaluation of microcomputer software alone is very difficult and not practical, 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 investigate applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive carefully and sufficiently and use Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall 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 shall not use Renesas Electronics products or technologies for (1) any purpose relating to the development, design, manufacture, use, stockpiling, etc., of weapons of mass destruction, such as nuclear weapons, chemical weapons, or biological weapons, or missiles (including unmanned aerial vehicles (UAVs)) for delivering such weapons, (2) any purpose relating to the development, design, manufacture, or use of conventional weapons, or (3) any other purpose of disturbing international peace and security, and you shall not sell, export, lease, transfer, or release Renesas Electronics products or technologies to any third party whether directly or indirectly with knowledge or reason to know that the third party or any other party will engage in the activities described above. When exporting, selling, transferring, etc., Renesas Electronics products or technologies, you shall comply with any applicable export control laws and regulations promulgated and administered by the governments of the countries asserting jurisdiction over the parties or transactions.
10. Please acknowledge and agree that you shall bear all the losses and damages which are incurred from the misuse or violation of the terms and conditions described in this document, including this notice, and hold Renesas Electronics harmless, if such misuse or violation results from your resale or making Renesas Electronics products available any third party.
11. This document shall not be reprinted, 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.

(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.

(Rev.3.0-1 November 2016)

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.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

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

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, 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 1207, 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 India Pvt. Ltd.
No.777C, 100 Feet Road, HAL II Stage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 6.0