User's Manual

16

RL78 Family

Flash Self-Programming Library Type 01

Additional Document for User's Manual
Bank Programming for the RL78/I1C (512 KB)

16-Bit Single-Chip Microcontrollers

Target devices:

RL78/I1C (512KB)

Renesas Electronics
www.renesas.com

Rev.1.00 Feb 2021

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 claims 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, drawings, charts, programs, algorithms, and 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 be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer 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, copying or reverse engineering.
6. 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; 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.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or 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 damage (space system; 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 any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using 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 dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. 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. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a 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. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using 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.
11. 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 comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.
(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan

www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit: www.renesas.com/contact/.

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit 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. Precaution against Electrostatic Discharge (ESD)
   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on
   The state of the product is undefined at the time 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 time 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 time 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 time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state
   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. 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 the 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.
5. Clock signals
   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is 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. Additionally, 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.
6. Voltage application waveform at input pin
   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses
   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products
   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of 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.

How to Use This Manual

1. Purpose and Target Readers

This supplementary document for the user's manual is designed to provide the user with an understanding of how to implement bank programming, a facility specific to the RL78/I1C (512 KB) by using the flash functions for Flash Self-Programming Library Type 01. It is intended for users designing application systems.

The usage of the flash functions is described in the RL78 Family Flash Self-Programming Library Type 01 User's Manual. For the facilities of the device, also refer to the RL78/I1C (512 KB) User's Manual: Hardware.

The newest versions of the listed documents are available on the Renesas Electronics Web site.

Document Title	Document No.
RL78 Family Flash Self-Programming Library Type 01 User's Manual	*1
RL78/I1C (512 KB) User's Manual: Hardware	R01UH0889EJ

Note 1: Refer to the user's manual for your region.

Target devices: R5F10NML and R5F10NPL, 512-KB products of the RL78/I1C group

2. Notation of Numbers and Symbols

Note:	Footnote for item marked with **Note** in the text	
Caution:	Information requiring particular attention	
Numerical representations:	Binary	×××× or ××××B
	Decimal	××××
	Hexadecimal	××××H or '0x'××××

3. List of Abbreviations and Acronyms

Abbreviation	Full Form
FSL	Flash Self-programming Library
SCIM	Status check internal mode
SCUM	Status check user mode

Table of Contents

1. Overview

1.1 Overview

Combining the flash functions of FSL Type 01 and the functions specific to the RL78/I1C (512 KB) listed in Table 1-1 makes the RL78/I1C (512 KB) capable of swapping between two 256-KB banks during execution of the user program after updating of the program in the code flash memory. This is generally referred to as bank programming in this document.

Target devices: R5F10NML and R5F10NPL, 512-KB products of the RL78/I1C group

Functions listed in Table 1-1 are only available in the above target devices.

Table 1-1 Functions Specific to the RL78/I1C (512 KB)

Function Name	Required Operation
Switching bank modes	Manipulating registers in the device.
Bank programming	Using flash functions of FSL Type 01.
Bank swapping	Using flash functions of FSL Type 01.

The usage of the flash functions of FSL Type 01 is described in the user’s manual for FSL Type 01. For the functions of the devices, refer to the RL78/I1C (512 KB) User’s Manual: Hardware.

2. Bank Programming Function

Switching bank modes, bank programming, and bank swapping which are specific to the RL78/I1C (512 KB) are used for the bank programming function. This chapter gives explanations of each of the functions and the procedures for execution with the use of the flash functions of FSL Type 01.

2.1 Use Cases for the RL78/I1C (512 KB)

The use cases for the RL78/I1C (512 KB) in terms of the bank programming function are used and non-used. The method for rewriting the code flash memory or the available functions will differ in some ways according to the bank programming use case.

Figure 2-1 is a schematic view of these use cases of the RL78/I1C (512 KB).

- When the bank programming function is to be used
  The code flash memory is used as two 256-KB bank areas.
  Switching bank modes, bank programming, and bank swapping are available.
- When the bank programming function is not to be used
  The code flash memory is used as a 512-KB area.
  Switching bank modes, bank programming, and bank swapping are not available.
  In this case, reference to this document is not required since the code flash memory can be rewritten with conventional self-programming.

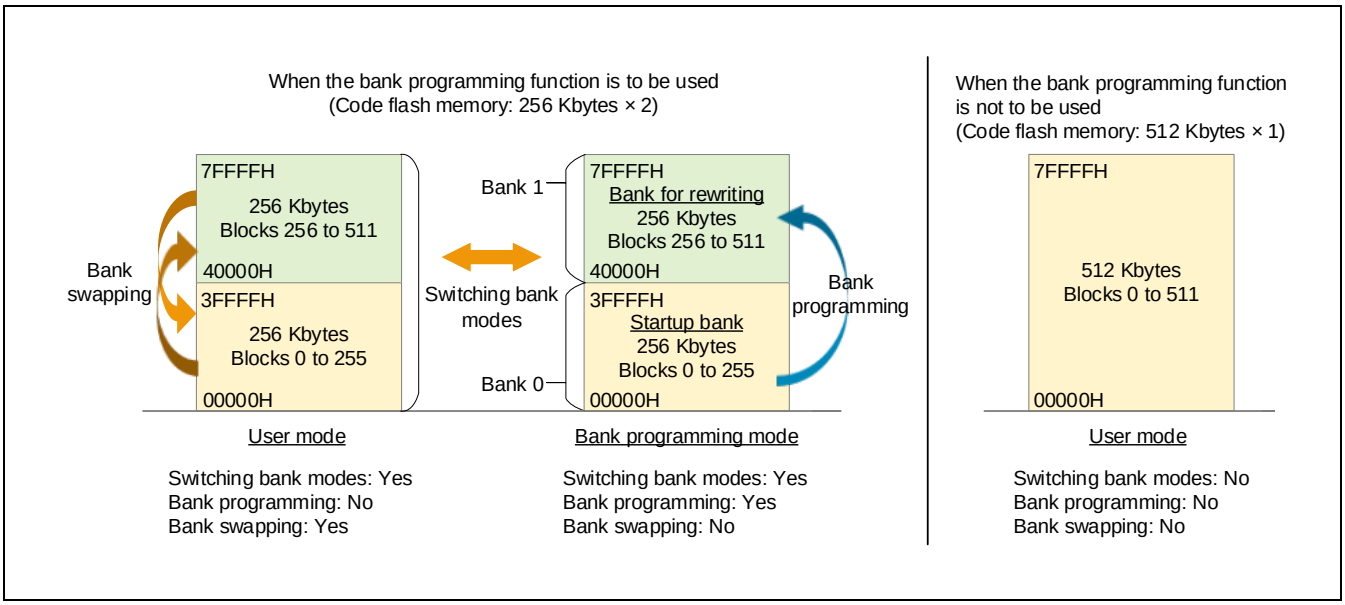

Figure 2-1 Bank Use Cases of the RL78/I1C (512 KB)

Caution: 1. Switching bank modes, bank programming, and bank swapping are only available for the RL78/I1C (512 KB).
2. Boot swapping is not available in the RL78/I1C (512 KB).

2.2 Switching Bank Modes

The RL78/I1C (512 KB) has a function for switching between user and bank programming modes. The user needs to operate the flash operating mode select register (FLMODE) of the device to switch the mode. For the method of switching to bank mode programming, refer to the user's manual for the device. After the reset is released, the RL78/I1C (512 KB) is always activated in user mode.

Figure 2-2 is a schematic view of switching bank modes.

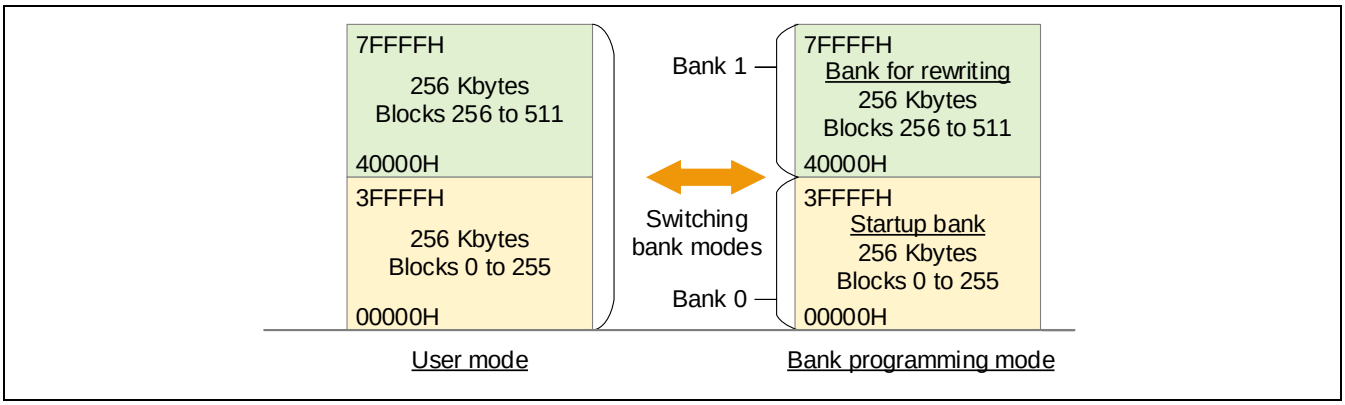

Figure 2-2 Schematic View of Switching Bank Modes

Caution: If you are not using bank programming, do not switch to bank programming mode from user mode.

2.3 Bank Programming

Bank programming enables rewriting of the bank for rewriting while the user program is running from the startup bank (ROM) of the code flash memory by using flash functions of FSL Type 01. Bank programming can only be executed by using the function for switching bank modes to switch to bank programming mode. Figure 2-3 is a schematic view of bank programming.

Allocate the same program which includes processing for communications, rewriting, and that to be executed during bank programming to fixed areas of the startup bank and the bank for rewriting. Bank programming provides processing for rewriting a fixed area of the startup bank to rewrite the program area of the bank for rewriting to the new program.

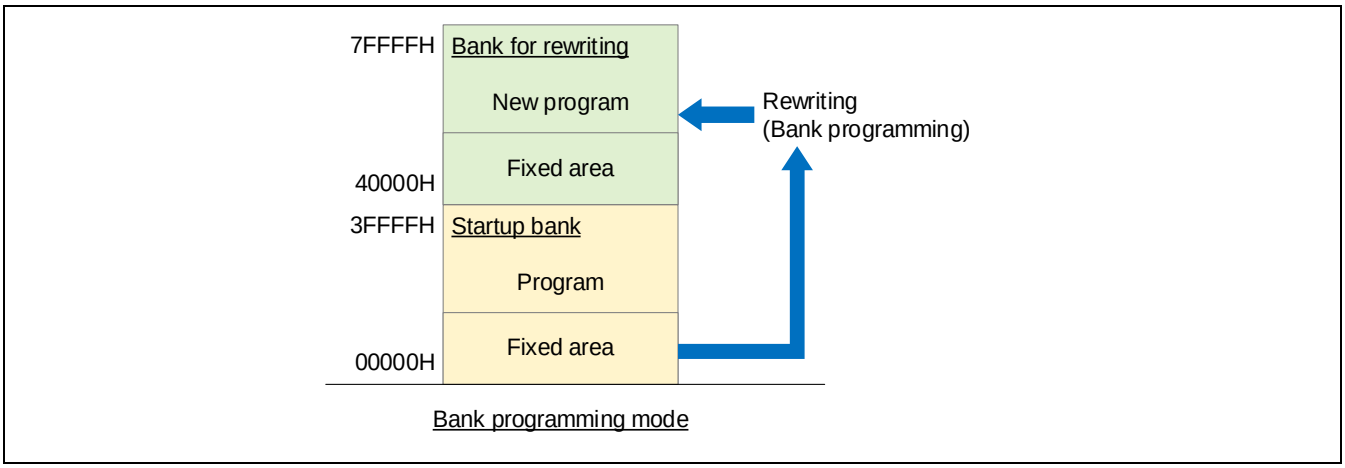

Figure 2-3 Schematic View of Bank Programming

Caution: 1. Bank programming is not available in user mode.
2. After you have used bank programming, switch back to user mode.

2.4 Bank Swapping

Bank swapping enables switching the address ranges of banks 0 and 1 of the code flash memory by using flash functions of FSL Type 01 and the selection is only applicable in user mode when bank programming is used for the user program. Figure 2-4 is a schematic view of bank swapping.

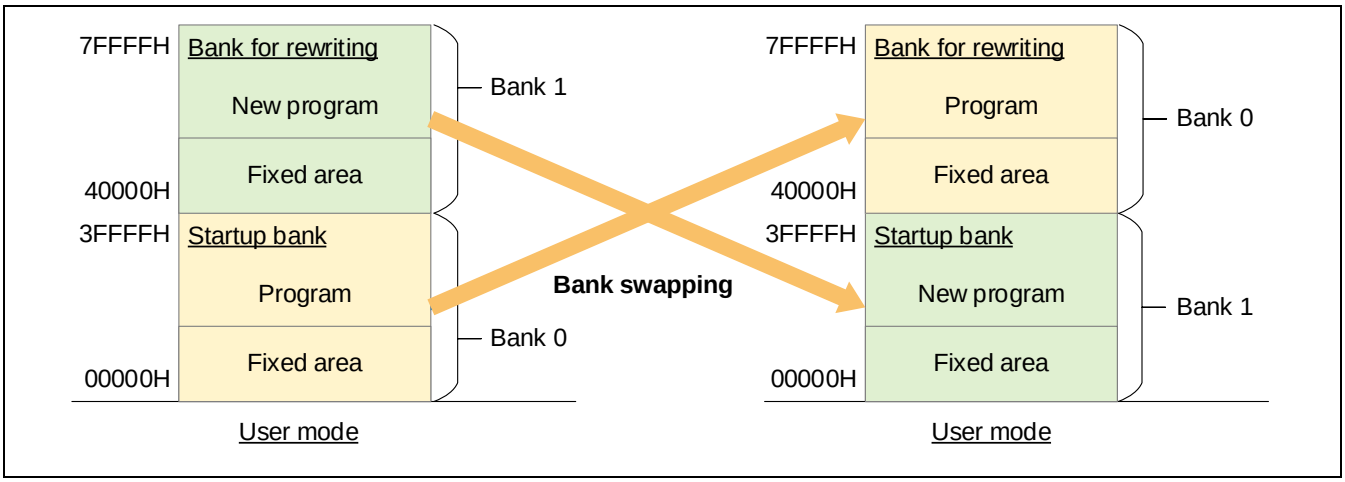

Figure 2-4 Schematic View of Bank Swapping

Caution:If you are not using bank programming, do not use bank swapping.

Table 2-1 shows a list of flash functions of FSL Type 01 which realize bank swapping. The timing and internal processing with which bank swapping is executed differs with the function used.

Table 2-1 List of Flash Functions of FSL Type 01 which Realize Bank Swapping

Function Name	Operation
FSL_InvertBootFlag	Inverts the current value of the boot flag. After a reset, the address ranges for banks 0 and 1 are switched and startup is from the bank that the boot flag indicates.
FSL_SwapBootCluster	Proceeds with bank swapping by moving the program counter to the address in the area after bank swapping that has been registered in the reset vector. The boot flag is not inverted.
FSL_SwapActiveBootCluster	Inverts the current value of the boot flag to proceed with bank swapping.

2.5 Procedures for Executing Bank Programming

(1) Procedures for executing bank programming

You can select the status check mode during bank programming of the bank for rewriting. Figure 2-5 and Figure 2-6 respectively show examples of the flows for the status check internal mode and the status check user mode. For details on the status check modes, refer to the user's manual for FSL Type 01.

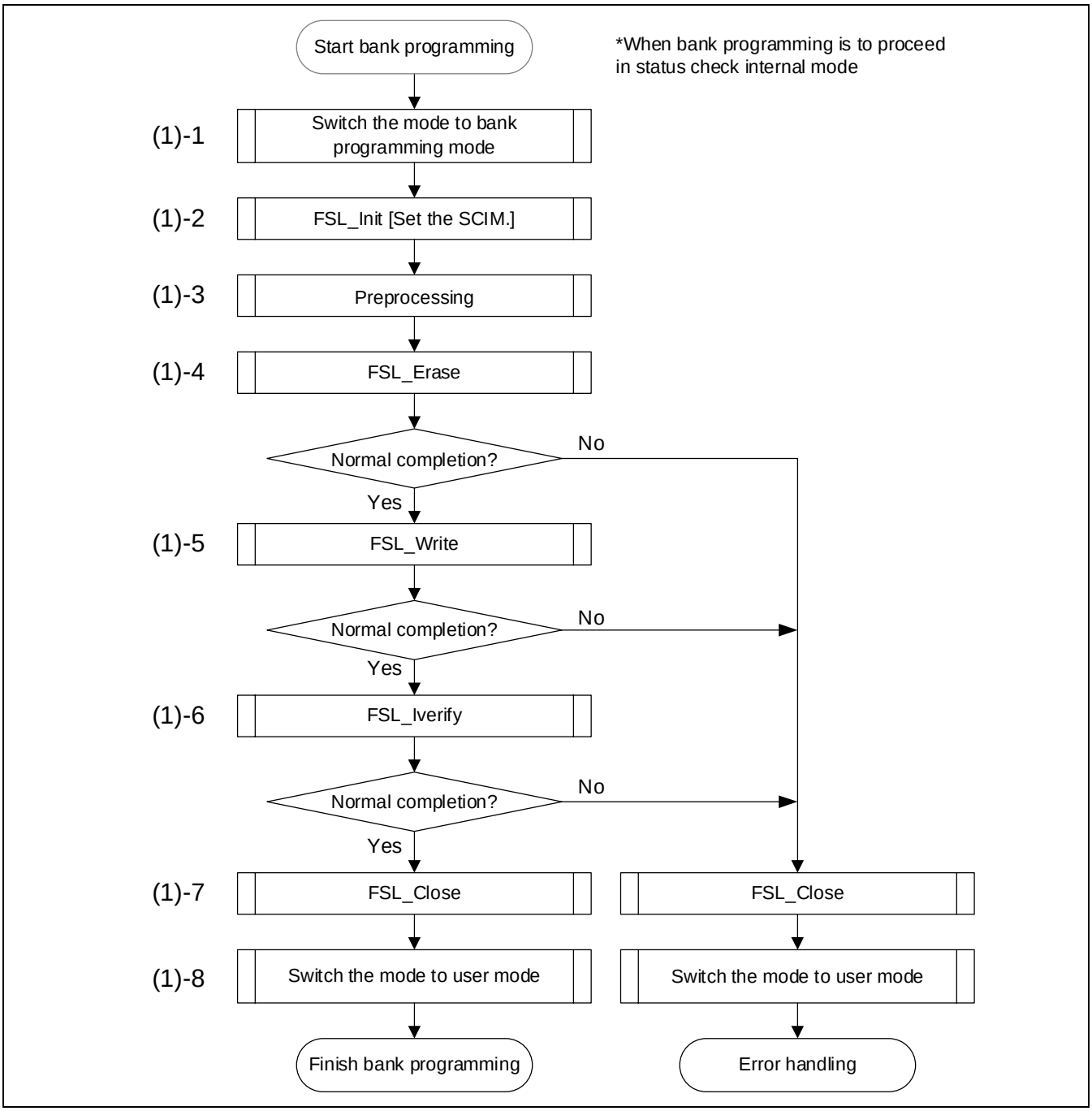

Figure 2-5 Example of the Flow for Status Check Internal Mode

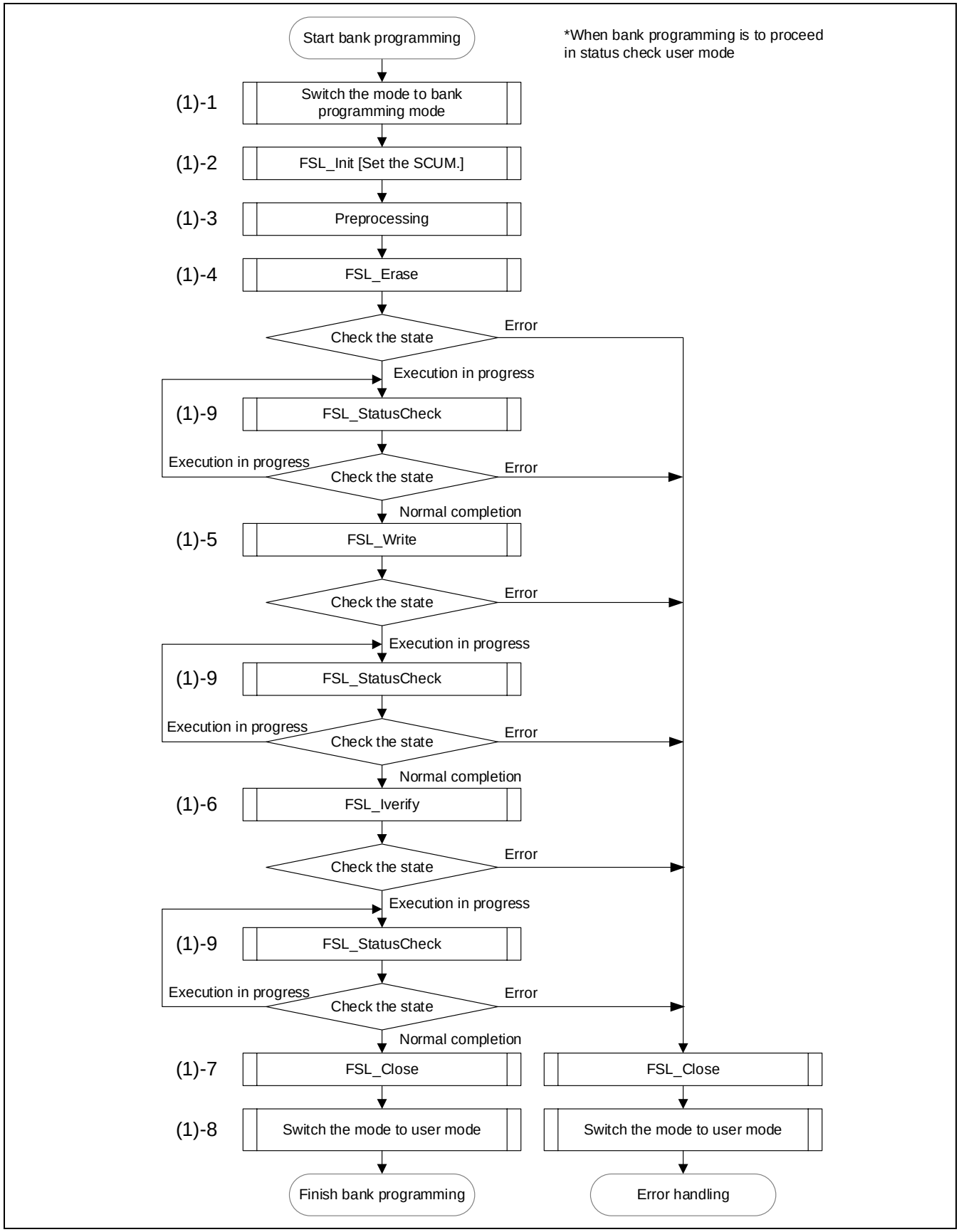

Figure 2-6 Example of the Flow for Status Check User Mode

The following describes details of Figure 2-5, Example of the Flow for Status Check Internal Mode, and Figure 2-6, Example of the Flow for Status Check User Mode.

(1)-1 Switching the mode

Set the bank programming mode.

(1)-2 Initializing flash self-programming

Executing the FSL_Init function

– Set the status check mode.

In Figure 2-5, set the status check internal mode.

In Figure 2-6, set the status check user mode.

(1)-3 Preprocessing

- Starting the flash environment (executing the FSL_Open function)
- Processing to prepare the flash functions (executing the FSL_PrepareFunctions function)

(1)-4 Erasing the area for the new program in the bank for rewriting

Execute the FSL_Erase function to erase all blocks of the area for the new program in the bank for rewriting.

(1)-5 Writing the new program to the area for the new program in the bank for rewriting

Execute the FSL_Write function and write the new program (such as program code received by the communications program) to the area for the new program in the bank for rewriting.

(1)-6 Verifying the area for the new program in the bank for rewriting

Execute the FSL_IVerify function to verify the area to which the new program has been written.

(1)-7 End processing

Execute the FSL_Close function to finish flash self-programming.

(1)-8 Switching the mode

Set the user mode.

(1)-9 Checking the state

When the status check user mode is used, status checking must be performed until control of the code flash memory is finished.

In Figure 2-5, status checking is not required since control of the code flash memory is finished when the program has returned from all functions and the processing by each of the functions has also been finished.

2.6 Procedures for Executing Bank Swapping

The FSL_InvertBootFlag, FSL_SwapBootCluster, and FSL_SwapActiveBootCluster flash functions each realize bank swapping.

For FSL_InvertBootFlag, bank swapping will proceed after a reset following execution of this function.

For FSL_SwapBootCluster and FSL_SwapActiveBootCluster, bank swapping will proceed during the execution of the function. For these flash functions that realize bank swapping in this way, execution must be in user mode and status check internal mode.

The following describes the procedures for executing each of the functions and details of their operations.

(1) Procedure for executing bank swapping by using the FSL_InvertBootFlag function

Figure 2-7 shows an example of the execution of bank swapping due to the generation of a reset after the FSL_InvertBootFlag function has been executed.

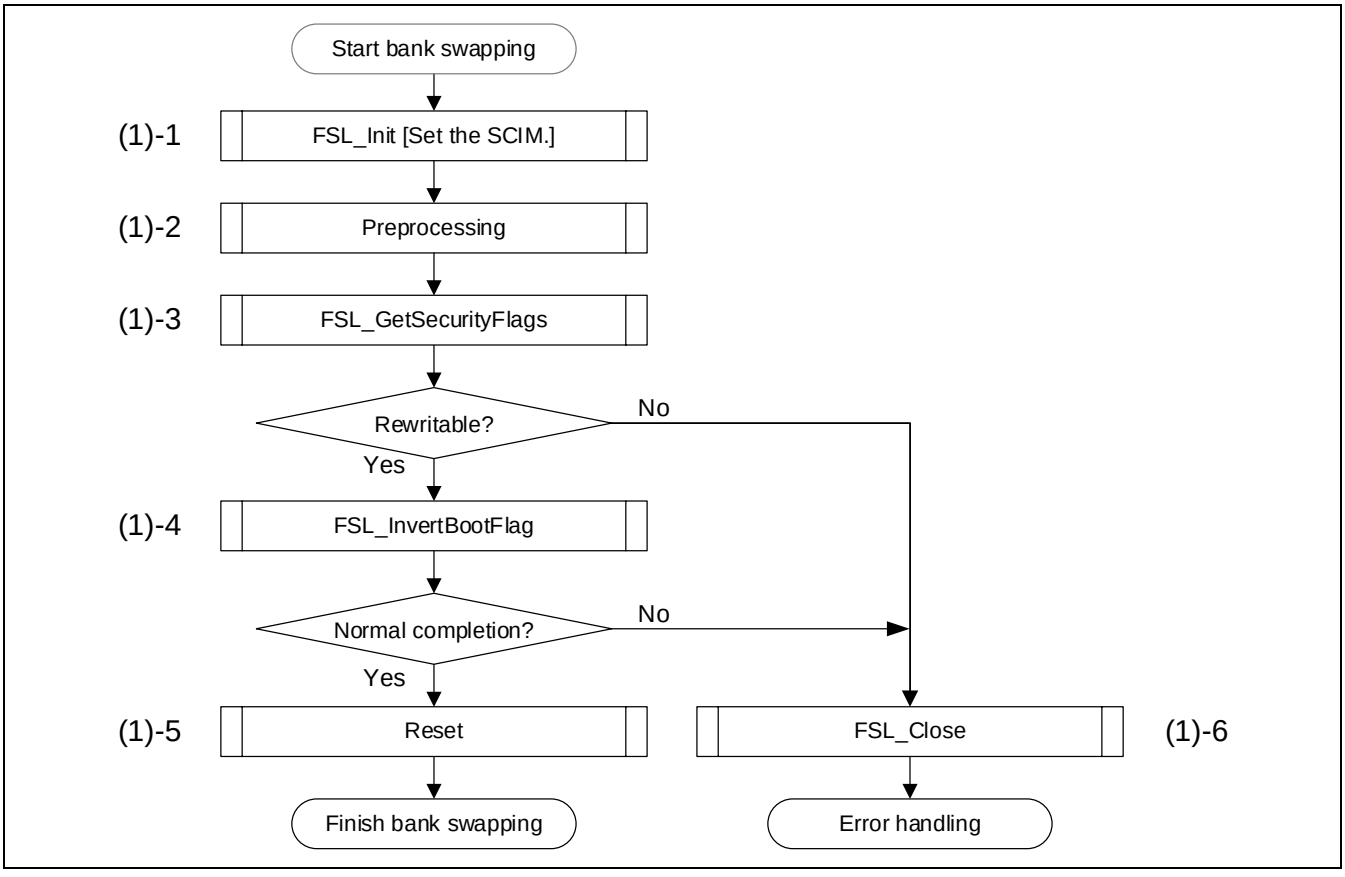

Figure 2-7 Example of the Flow for Bank Swapping by Using the FSL_InvertBootFlag Function

The following describes details of Figure 2-7, Example of the Flow for Bank Swapping by Using the FSL_InvertBootFlag Function.

(1)-1 Initializing flash self-programming

Executing the FSL_Init function

– Set the status check internal mode.

(1)-2 Preprocessing

- Starting the flash environment (executing the FSL_Open function)

- Processing to prepare the flash functions (executing the FSL_PrepareFunctions function)

- Processing to prepare the flash function (an extended function) (executing the FSL_PrepareExtFunctions function)

(1)-3 Checking the security bit (recommended)

Execute the FSL_GetSecurityFlags function to acquire information on the security flag and check that bit 1 of the information on the security flag (rewrite-prohibited flag in boot cluster 0) is 1 (enabled).

Remark: If bit 1 (rewrite-prohibited flag in boot cluster 0) is 0 (disabled), a protection error will be returned in response to execution of the FSL_InvertBootFlag function in step (1)-4 below.

(1)-4 Setting the boot flag

Execute the FSL_InvertBootFlag function to switch the boot flag.

(1)-5 Generating an event

Generate a reset to switch the startup bank and bank for rewriting. After the reset, the startup bank will be the other bank from that before the reset.

(1)-6 End processing

Execute the FSL_Close function to finish flash self-programming.

(2) Procedure for executing bank swapping by using the FSL_SwapBootCluster function

While the FSL_SwapBootCluster function is executed, bank swapping proceeds by moving the program counter to the address in the area after bank swapping that has been registered in the reset vector. The boot flag is not inverted.

Figure 2-8 shows an example of the flow for bank swapping by using the FSL_SwapBootCluster function.

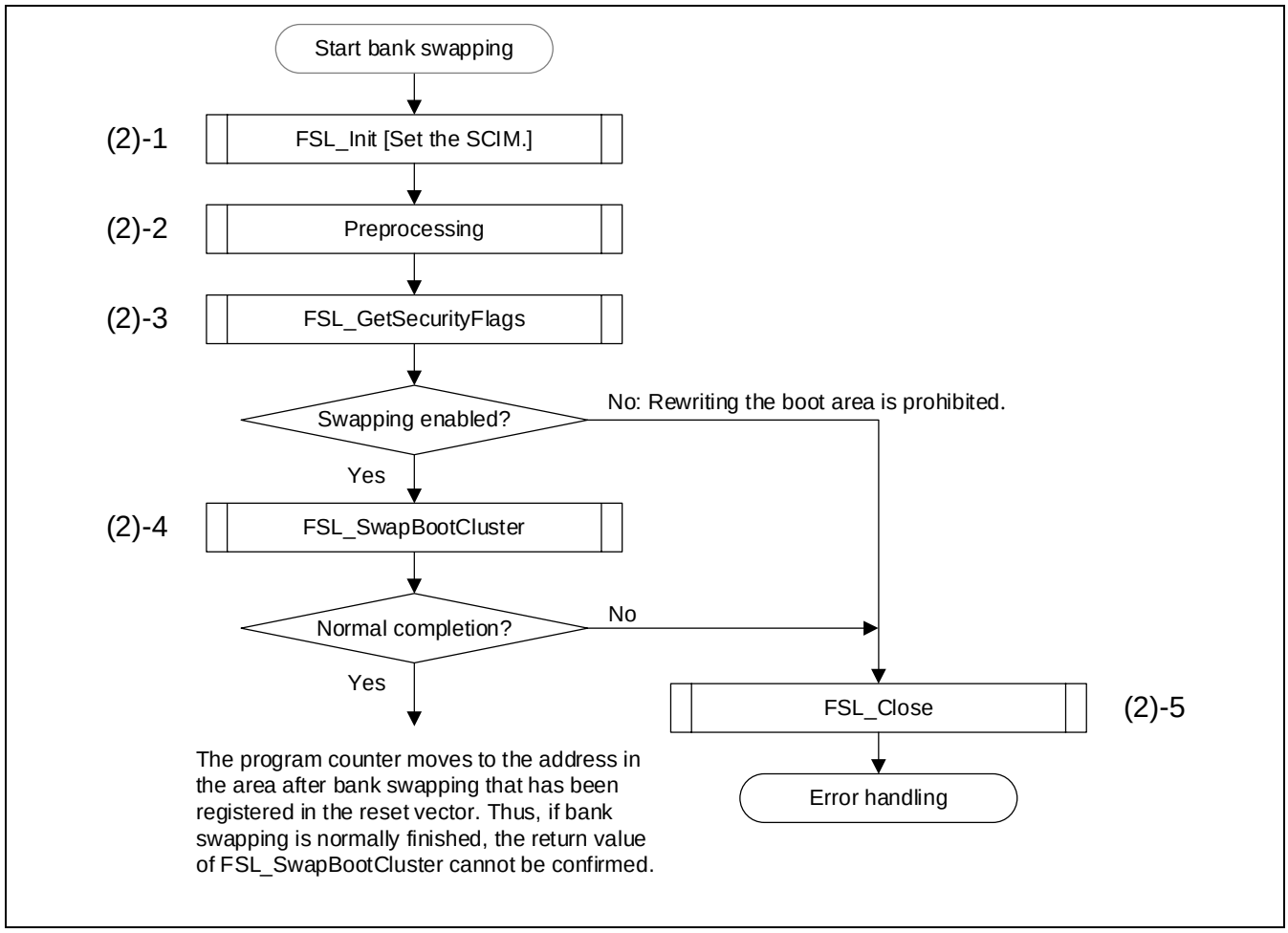

Figure 2-8 Example of the Flow for Bank Swapping by Using the FSL_SwapBootCluster Function

The following describes details of Figure 2-8, Example of the Flow for Bank Swapping by Using the FSL_SwapBootCluster Function.

(2)-1 Initializing flash self-programming

Executing the FSL_Init function

– Set the status check internal mode.

(2)-2 Preprocessing

- Starting the flash environment (executing the FSL_Open function)
- Processing to prepare the flash functions (executing the FSL_PrepareFunctions function)
- Processing to prepare the flash function (an extended function) (executing the FSL_PrepareExtFunctions function)

(2)-3 Checking the security bit (recommended)

Execute the FSL_GetSecurityFlags function to acquire information on the security flag and check that bit 1 of the information on the security flag (rewrite-prohibited flag in boot cluster 0) is 1 (enabled).

Remark: If bit 1 (rewrite-prohibited flag in boot cluster 0) is 0 (disabled), a protection error will be returned in response to execution of the FSL_SwapBootCluster function in step (2)-4 below.

(2)-4 Proceeding with bank swapping

Execute the FSL_SwapBootCluster function and switch the startup bank and bank for rewriting by moving the program counter to the address in the area after bank swapping that has been registered in the reset vector. The boot flag is not inverted.

(2)-5 End processing

Execute the FSL_Close function to finish flash self-programming.

Caution 1. Following normal execution of this function, the program counter moves to the address in the area after bank swapping that has been registered in the reset vector. Any subsequent processing in the bank that had been selected before the function call will not be executed.

2. This function does not invert the boot flag. If a reset is applied, the startup bank becomes that which matches the setting of the boot flag.
3. When the FSL_ChangeInterruptTable function is executed to change the interrupt destinations, the states of the interrupts in terms of entering the area changed by the FSL_ChangeInterruptTable function are retained even after the program counter has moved to the new address registered in the reset vector. To move the program counter to the addresses that should be registered in the restored reset vector, execute the FSL_RestoreInterruptTable function to restore the interrupt destinations before executing the FSL_SwapBootCluster function.
4. This function cannot be executed from the ROM. To use this function, allocate the code to the FSL_RCD section in the RAM.
5. To maintain the state after swapping even if a reset is applied, execute the FSL_InvertBootFlag function before executing the FSL_SwapBootCluster function in step (2)-4 above.

(3) Procedure for executing bank swapping by using the FSL_SwapActiveBootCluster function

When the FSL_SwapActiveBootCluster function is executed, it proceeds with bank swapping by inverting the current value of the boot flag.

Figure 2-9 shows an example of the flow for bank swapping by using the FSL_SwapActiveBootCluster function.

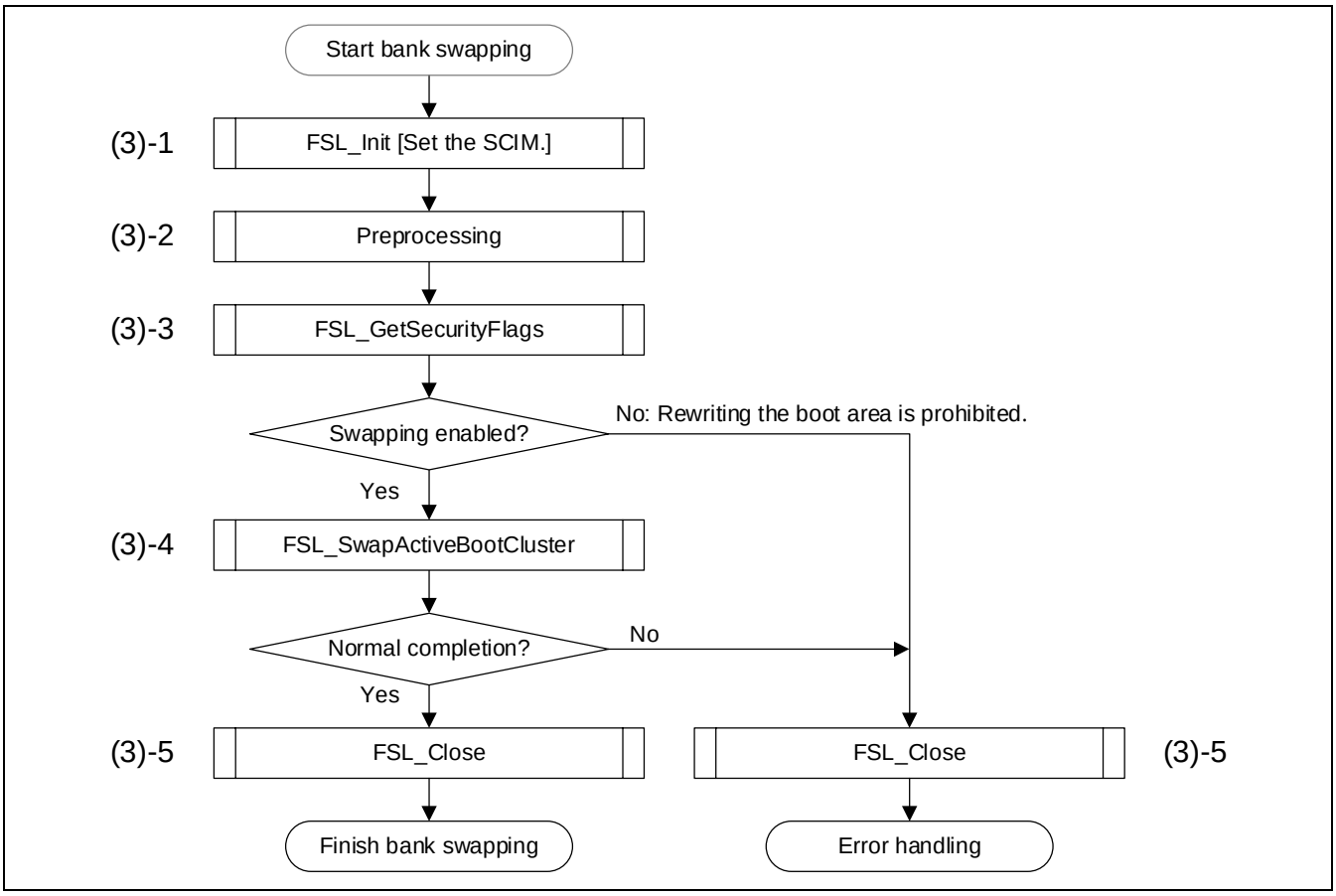

Figure 2-9 Example of the Flow for Bank Swapping by Using the FSL_SwapActiveBootCluster Function

The following describes details of Figure 2-9, Example of the Flow for Bank Swapping by Using the FSL_SwapActiveBootCluster Function.

(3)-1 Initializing flash self-programming

Executing the FSL_Init function

– Set the status check internal mode.

(3)-2 Preprocessing

- Starting the flash environment (executing the FSL_Open function)

- Processing to prepare the flash functions (executing the FSL_PrepareFunctions function)

- Processing to prepare the flash function (an extended function) (executing the FSL_PrepareExtFunctions function)

(3)-3 Checking the security bit (recommended)

Execute the FSL_GetSecurityFlags function to acquire information on the security flag and check that bit 1 of the information on the security flag (rewrite-prohibited flag in boot cluster 0) is 1 (enabled).

Remark: If bit 1 (rewrite-prohibited flag in boot cluster 0) is 0 (disabled), a protection error will be returned in response to execution of the FSL_SwapActiveBootCluster function in step (3)-4 below.

(3)-4 Proceeding with bank swapping

Execute the FSL_SwapActiveBootCluster function to invert the current value of the boot flag and switch the startup bank and bank for rewriting.

(3)-5 End processing

Execute the FSL_Close function to finish flash self-programming.

Caution 1. This function cannot be executed from the ROM. To use this function, allocate the code to the FSL_RCD section in the RAM.

2. After the execution of this function, the interrupt vector table allocated to the top of the startup bank is changed to that allocated to the top of the bank for rewriting and destination addresses of interrupt vectors are also changed. To use interrupt processing in the ROM before and after executing FSL_SwapActiveBootCluster, consider that the operation will also switch any interrupt vectors in the ROM for which operation is in progress.

3. Since this function switches banks without issuing a reset and returns to the address from which it was executed, user processing for calling the FSL_SwapActiveBootCluster function must be allocated to the RAM.

2.7 Functions that are Executable during the Use of Bank Programming

Flash functions that are executable during the use of bank programming differ with the settings for the bank mode and the status check mode. Table 2-2 shows a list of functions that are executable during the use of bank programming. When the area for a new program in the bank for rewriting is rewritten in bank programming mode, flash functions must be allocated to the ROM. Since extended functions are also allocated to the ROM, when those functions are executed in user mode, only the status check internal mode (SCIM) where user processing is allocated to the ROM is available. For details on each of the flash functions, refer to the user's manual for FSL Type 01.

Table 2-2 List of Functions that are Executable during the Use of Bank Programming

◎: Bank programming, √: Usable, —: Not used

Flash Function	Execution Mode		
	User Mode	Bank Programming Mode	
	SCIM*3	SCIM	SCUM
FSL_Init	√	√	√
FSL_Open	√	√	√
FSL_Close	√	√	√
FSL_PrepareFunctions	√	√	√
FSL_PrepareExtFunctions	√	√	√
FSL_ChangeInterruptTable	√	—	—
FSL_RestoreInterruptTable	√	—	—
FSL_BlankCheck	—*2	◎	◎
FSL_Erase	—*2	◎	◎
FSL_IVerify	—*2	◎	◎
FSL_Write	—*2	◎	◎
FSL_GetSecurityFlags	√	√	√
FSL_GetBootFlag	√	√	√
FSL_GetSwapState	√	√	√
FSL_GetBlockEndAddr	√	√	√
FSL_GetFlashShieldWindow	√	√	√
FSL_SwapBootCluster	√*1	—	—
FSL_SwapActiveBootCluster	√*1	—	—
FSL_InvertBootFlag	√	—	—
FSL_SetBlockEraseProtectFlag	√	—	—
FSL_SetWriteProtectFlag	√	—	—
FSL_SetBootClusterProtectFlag	√	—	—
FSL_SetFlashShieldWindow	√	—	—
FSL_StatusCheck	—	—	√
FSL_StandBy	—	—	√
FSL_WakeUp	—	—	√
FSL_ForceReset	√	√	√
FSL_GetVersionString	√	√	√

Notes: 1. The FSL_RCD (segment or section) must be allocated to the RAM.
2. The bank for rewriting is rewritten in bank programming mode.
3. For interrupts that may be used during processing, the interrupt destinations must be changed to the RAM.

2.8 Points for Caution on Bank Programming

- If the rewrite-prohibited flag in boot cluster 0 is 0 (disabled), bank swapping cannot proceed.
- To use bank swapping, set the same value as “000C0H to 000C3H” at addresses “400C0H to 400C3H”.
- Set VDD ≥ 2.7 V for bank programming. Proceeding with bank programming in LP or LV mode is prohibited.

For other points for caution, refer to the user’s manual for the device.

RL78 Family Flash Self-Programming Library Type 01
Additional Document for User's Manual
Bank Programming for the RL78/I1C (512 KB)

Publication Date: Rev.1.00 Feb.02.21

Published by: Renesas Electronics Corporation

RL78 Family

Renesas Electronics Corporation

R20UT4871EJ0100