RH850/U2A-EVA Group

Over-the-air(OTA) Software Update

Introduction

This application note describes the Over-the-air (OTA) software update on RH850/U2A-EVA series in the automotive single chip microcomputer by Renesas Electronics. This application note describes receiving update software, programming/Erasure of flash memory and switching of program as OTA software update. RS-CANFD is used as an interface to external device. The reprograming program is assumed to be on the user area of incorporated code flash.

This application note does not support security function that required to implement OTA software update in vehicle. For security, it is necessary that built on your system.

Aim of this document and software is to provide supplemental information for the function on RH850/U2A-EVA devices. It is not intended to implement in the design for mass production.

There is no guarantee to update in this document and software to reflect the latest manual, errata, technical update and development environment. You are fully responsible for the incorporation or any other use of the information of this document in the design of your product or system, and please refer to latest manual, errata, technical update and development environment.

Target device

- RH850/U2A-EVA Group

Target integrated development environment

CS+(by Renesas electronics corporation)

Version : V8.07.00

Device file : DR7F702300.DVF

: DR7F702301.DVF

: DR7F702302.DVF

Reference Document

The User's Manual provides information about functional and electrical behavior of the device.

At the release time of this application note the following manual version available:

RH850/U2A-EVA Group User's Manual (Rev1.20): R01UH0864EJ0120

RH850/U2A-EVA Group User's Manual Flash Memory User's Manual (Rev1.10): R01UH0832EJ0110

Driver

- Renesas Flash Driver RFD28F V.1.00.01 Free package (included in sample code)

For the details about RFD, please contact us at this location(www.renesas.com/contact/).

Contents

1. Background and Use case 4

2. OTA capability of RH850/U2A 5
2.1 Flash specification 5
2.1.1 Single Map Mode 6
2.1.2 Double Map Mode 7
2.2 Address remapping function 8
2.2.1 Remapping using GCFU 8
2.2.2 Address swap method of Double Map Mode 9
2.3 FACI capability 10
2.3.1 BGO(Background operation) support 10
2.3.2 Suspend function 10

3. OTA software update method using RH850/U2A 11
3.1 Method for remapping of code flash read address (Single map mode) 11
3.2 Method for hardware swapping of address map between code flash memory banks(Double map mode) 12
3.3 Note on security support 13

4. Configuration for Sample software 14
4.1 Hardware configuration 14
4.2 CAN FD communication 14
4.2.1 CAN FD Communication Specifications 14
4.2.2 CAN FD command specification (from the U2A perspective) 15
4.3 Port(LED) specification 15

5. Software operation in single map mode(with GCFU) 16
5.1 Specification 16
5.1.1 Software specification 16
5.1.2 Operation overview 17
5.1.3 Overall sequence 18
5.1.4 Functions used 20
5.1.5 Operation mode 20
5.1.6 Memory mapping mode 20
5.1.7 Configuration for RFD 20
5.2 Operation procedure 21
5.2.1 (1) Bank switching 21
5.2.2 (2) Code flash reprogramming with background operation (BGO) function 24
5.2.3 (3) ID authentication 27
5.2.4 (4) Code Flash Erasure 28
5.2.5 (5) Download of write data 30
5.2.6 (6) Code flash programming 31
5.2.7 (7) Writing of boot bank information to data flash 33
5.3 Memory allocation 34

6. Software operation in double map mode(Bank swapping) 35
6.1 Specification 35
6.1.1 Software specification 35
6.1.2 Operation overview 37
6.1.3 Overall sequence 38
6.1.4 Functions used 41
6.1.5 Operation mode 41
6.1.6 Memory mapping mode 41
6.1.7 Configuration for RFD 41
6.2 Operation procedure 42
6.2.1 (1) Start Program 42
6.2.2 (2) Code flash reprogramming with background operation (BGO) function 43
6.2.3 (3) ID authentication 46
6.2.4 (4) Code Flash Erasure 47
6.2.5 (5) Download of write data 49
6.2.6 (6) Code flash programming 50
6.2.7 (7) Update of hardware property area 52
6.2.8 (8) Update of Switch Area 57
6.2.9 (9) Update of TAG Area 60
6.3 Memory allocation 62

7. Programming/Erasure is interrupted intentionally or unintentionally during software update 63

1. Background and Use case

Over-the-air (OTA) means wireless communication, this technology is generally used for software update and so on. Recently, OTA software update is drawn attention to make improvement control software easy in vehicle. OTA software update requires background update without disturbing vehicle control.

U2A-EVA devices include several functions for OTA software update in background.

The OTA has several software update assumptions of use.

The method called “Wait OTA” is download while vehicle is stopped and write the update program to flash memory.

The method called “Semi-non wait OTA” is download update program to external flash memory and update internal flash memory in MCU while vehicle is stopped.

The method called “Non wait OTA” allows to run software in one bank while updating software the other bank. With “Non wait OTA” method, software can be downloaded to the vehicle to be updated via wireless communication, and the software can be updated in the background without interfering with the control of the running vehicle.

Figure 1-1 shows use case of software updates.

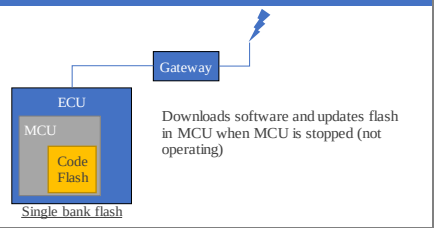
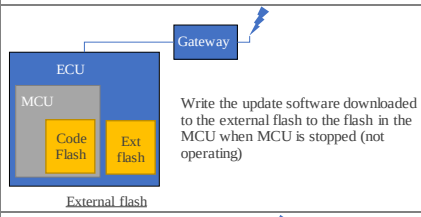
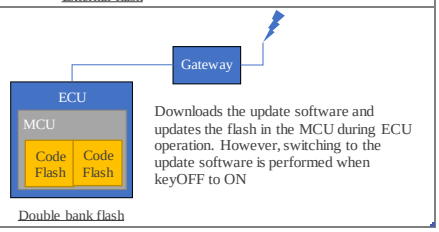
OTA methods	Backup program	When writing fail	Background operation	Timing of writing sofware to MCU	Images (MCU operation)
Wait OTA	None	Need to re-download	not possible	key-off	
Semi-non wait OTA	None	Need not to re-download (Depend on external flash)	Possible	key-off	
Non wait OTA	Yes	Need not to re-download	Possible	Key-on (During the softeware is runnning)	

Figure 1-1 Use case of software updates

RH850/U2A-devices have double map mode with multiple banks that separate the software area to be placed to realize “Non wait OTA”. Also, RH850/U2A-EVA devices have single map mode without separation of flash memory. When U2A is used in single map mode, software update in background is realized using GCFU(Global Calibration Function Unit).

For the details of mapping mode of U2A, please refer to 2.1 Single Map Mode.
For the details of GCFU, please refer to 2.2.1 Remapping using GCFU.

2. OTA capability of RH850/U2A

2.1 Flash specification

RH850/U2A-EVA devices contain three Flash Programing systems (FPSYS0, FPSYS1, FPSYS2)[Note1]. Each FPSYS is configured by Flash Memory and flash sequencer. Each flash sequencer contains Flash Application Command Interface (FACI).

Each U2A16, U2A8 and U2A6 incorporate 16MB, 8MB and 6MB of code flash. FPSYS2 is used only for data flash area of ICUMH. Figure 2-1 shows block diagram of U2A16 Flash Programming System structure.

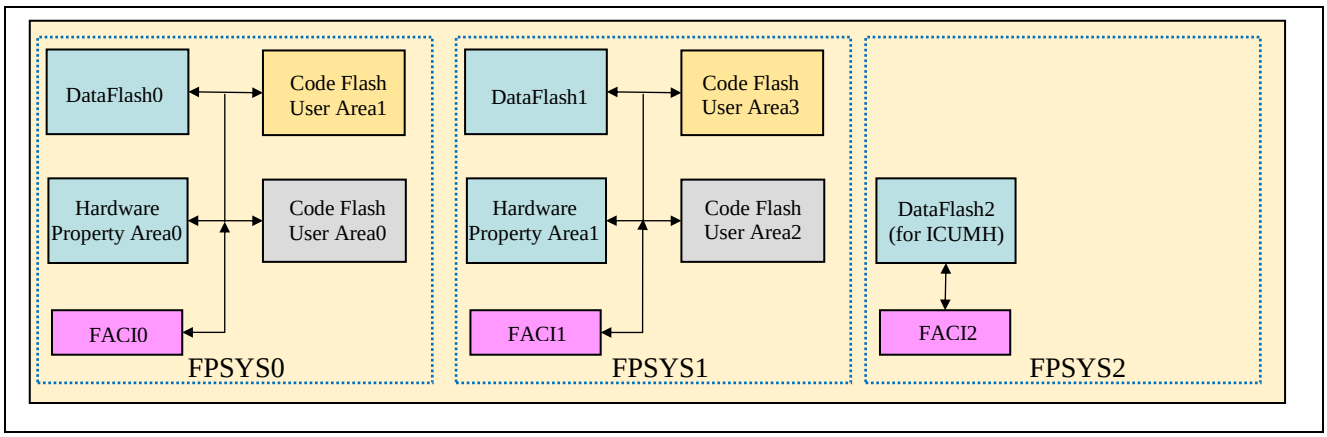


Figure 2-1 Block diagram of U2A16 Flash Programming System structure

RH850/U2A-EVA devices contain two types of code flash mapping mode (Single Map mode and Double Map mode). The mapping mode can be selected by setting of OPBT12.MAPMODE[1:0]. For the details, please refer to RH850/U2A-EVA Group User's Manual: Hardware "51.12.18 OPBT12 — Option Byte 12".

- Single Map Mode

Hardware remapping of User Area read address is supported by GCFU (Overlay function).

- Double Map Mode

Hardware swapping of address map between Code Flash Memory Banks is supported.
The Code Flash Memory Banks are mapped into two areas. One is valid area and the other is invalid area.
The Banks in the valid area and invalid area can be swapped by Option Bytes setting.
The user program can be executed from the banks in valid area while the banks of invalid area are being programmed / erased.

Note1 FPSYS1 is available only for U2A16.

2.1.1 Single Map Mode

This mode allocates code flash area continuously. Each U2A16, U2A8 and U2A6 are used 16MB, 8MB and 6MB for user program area. Therefore, large size code flash can be used in this mode. Single map mode is efficient to use code flash for partial program update. Figure 2-2 shows the code flash (User area) structure of single map mode.

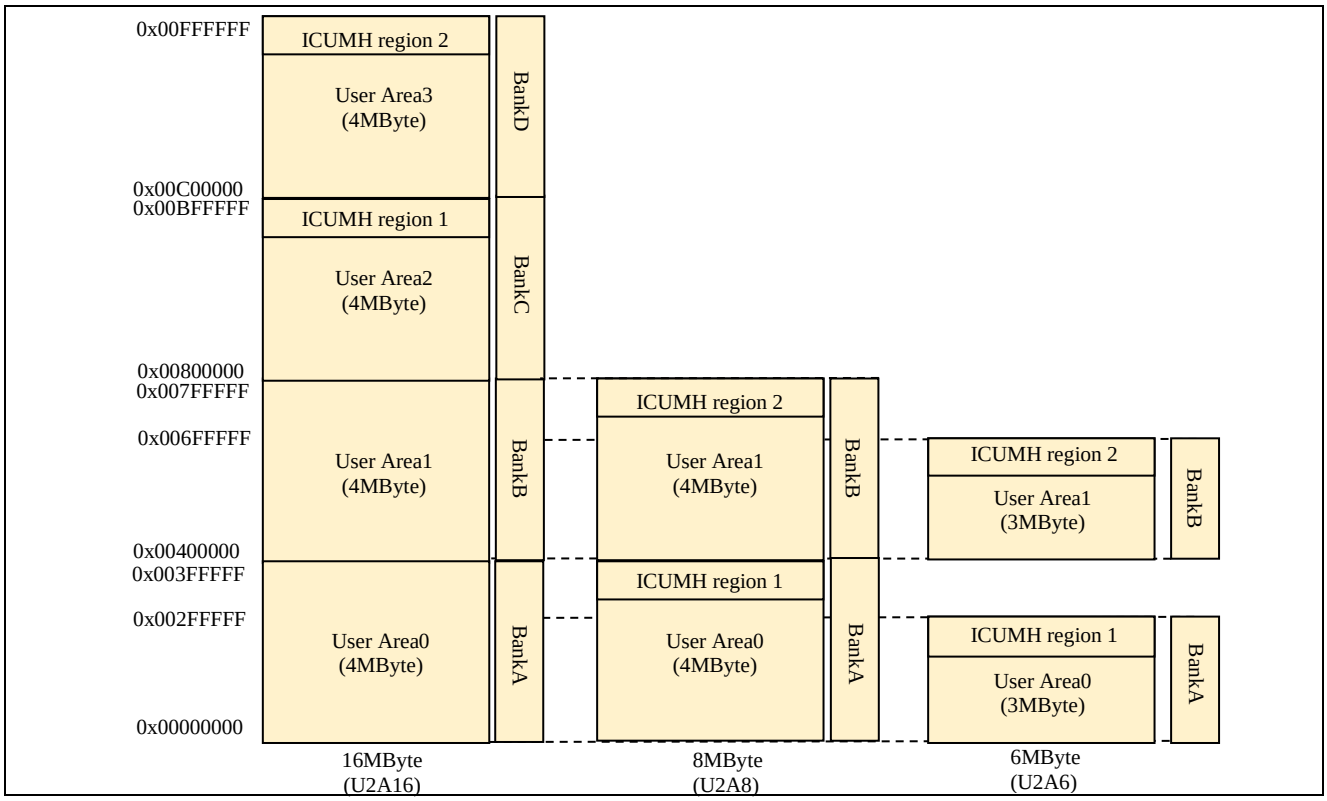


Figure 2-2 Code flash(User area) structure of single map mode[Note1].

[Note1] The number of ICUMH region and address are set by option byte. When ICUMH is disabled, ICUMH regions are also disabled.

2.1.2 Double Map Mode

This mode divides the code flash area into two areas. Code flash is mapped into two areas for each FPSYS, one is the valid area, the other is invalid area, and the valid or invalid can be switched by setting of OPBT13.DBMAPSW0/1. Each U2A16 and U2A8 are used 8MB and 4MB for user program.

Double map mode is efficient to update the entire code flash. Code flash is duplicated, and valid and invalid area can be switched in this mode. Therefore, all access master can always access same address to code flash. When reverting to the program before update after updating, it makes easier to revert due to backup program exists in invalid area.

Code flash valid area in double map mode is possible to be selected with OPBT13.DBMAPSW0/1. Valid area is settable each FPSYS individually.

By swapping each area between valid and invalid areas, the address of valid area is always same.

During program is running valid area, rewriting the invalid area is realized as software update, after the software is updated, the valid and invalid area is switched. Refer to “2.2.2 Address swap method of Double Map Mode” for valid area switching method.

Figure 2-3 shows code flash (User area) structure of double map mode.

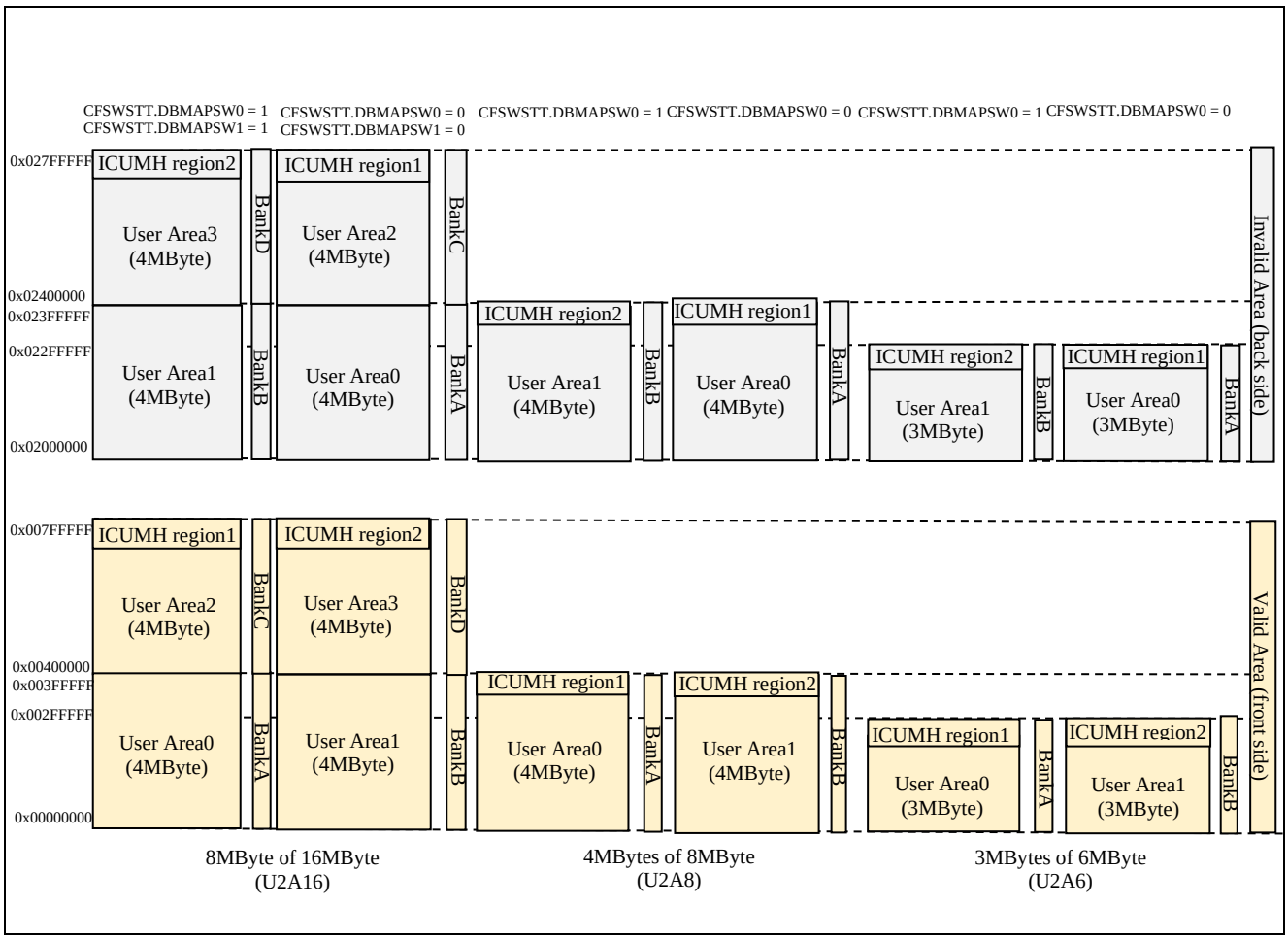


Figure 2-3 Code flash(User area) structure of double map mode[Note1].

Note1 ICUMH region is set by option byte. When ICUMH is disabled, ICUMH regions are also disabled.

2.2 Address remapping function

2.2.1 Remapping using GCFU

U2A supports a GCFU (Global Calibration Function Unit), which can translate read access to code flash. In single-map mode, GCFU translates read access from CPU for remap the program before and after the update. This function is efficient for partial program update. Figure 2-4 shows the flash access remapped by GCFU.

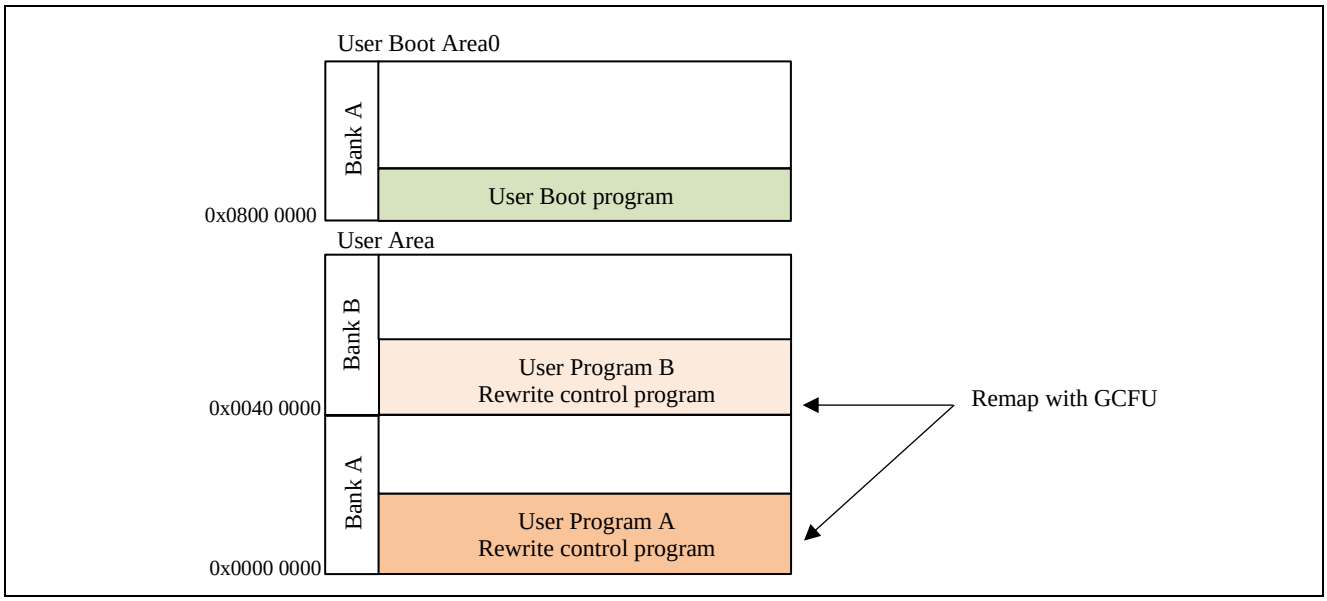


Figure 2-4 Flash access remapped by GCFU

CPU accesses to flash memory as logical address. GCFU remaps address of flash memory. FACI accesses to flash memory as physical address due to FACI does not depend on logical address remapped by GCFU. Figure 2-5 shows bus structure related flash memory accesses.

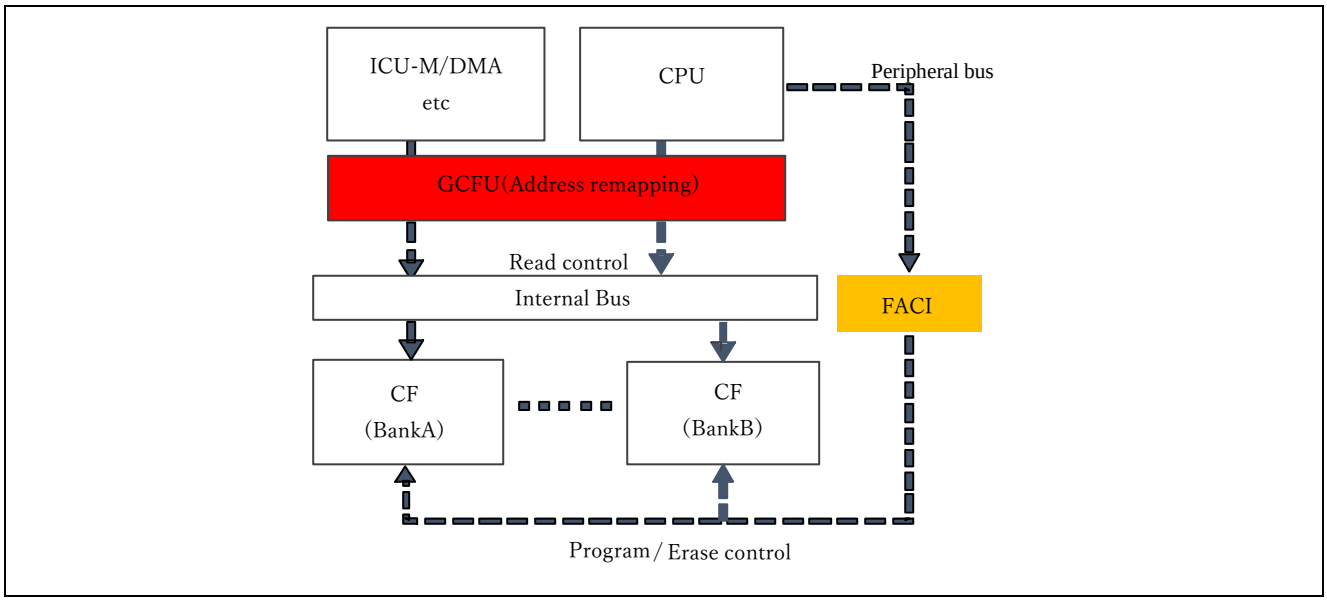


Figure 2-5 Bus structure related flash memory accesses.

For the example of address remapping operation by using GCFU, please refer to 5.1.2 Operation overview.

(a) Note

The access from FACI for code flash depends on physical address. Therefore, accessibility of code flash during programming/erasure also depends on physical address. When flash address is remapped by GCFU, note that BGO(Background operation) function and Multi FPSYS operation described later are based on the physical address, so you should be careful. For the details, please refer to RH850/U2A-EVA Group User's Manual: Hardware Section 51, Flash Memory.

2.2.2 Address swap method of Double Map Mode

RH850/U2Ax has 9 blocks[Note1] of hardware property area in data flash for hardware configuration including valid area setting of code flash in double map mode. Figure 2-6 shows overview of the hardware property area (Switching related area).

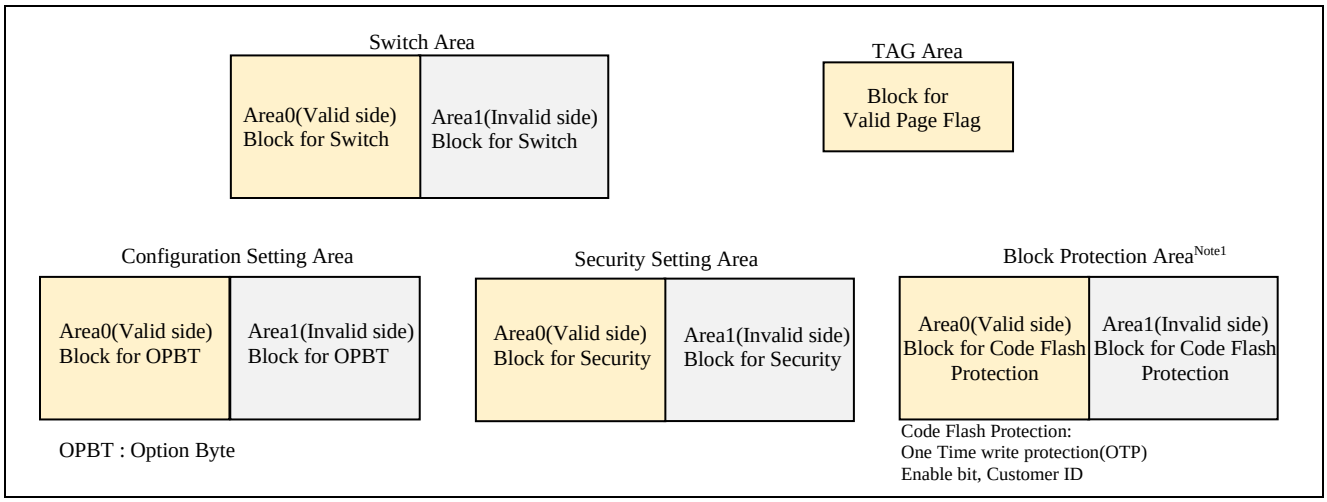


Figure 2-6 Overview of the hardware property area (shipping value)

Configuration Setting Area, Security Setting Area and Block protection Area for FPSYS0/1[Note1] are consisted with two areas (Area0 and Area1). Valid areas are mapped in front side and invalid areas are mapped in back side. Valid area cannot be updated whereas invalid area can be updated. These areas are used for before and after update setting. Area 0 is selected as valid area and Area 1 is selected as invalid area that erased at the shipping.

Switch Area consists of two areas (Area 0 and Area 1). This area consists switch setting and various flags for each Configuration Setting Area, Security Setting Area, Block protection Area0/1. Area 0 is selected as valid area and Area 1 is selected as invalid area at the shipping.

TAG Area is used for valid area setting of Switch Area. Figure 2-7 shows details of Configuration Setting, Security Setting, Block Protection, Switch, and TAG Area structure.

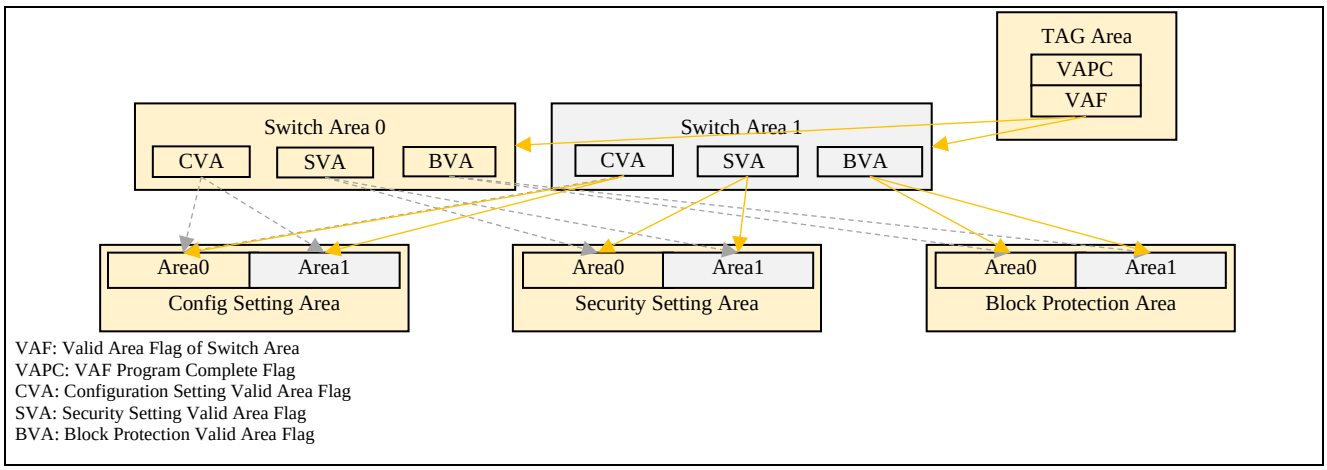


Figure 2-7 Details of Config Setting, Security Setting, Block Protection, Switch and TAG Area structure.

Valid area of Switch Area is selected by VAF in TAG Area. Valid area of Configuration Setting Area, Security Setting Area, and Block Protection Area are selected by CVA, SVA and BVA in valid area of Switch Area respectively. The other flags are prepared to indicate progress of programming / erasure of each data to improve robustness against unintentional interruption during programming / erasure of each data (Note that the complete robustness is not guaranteed).

[Note1] Block Protection Area for FPSYS1 is available only for U2A16.

2.3 FACI capability

2.3.1 BGO(Background operation) support

The FACI supports the programming/erasure processing suspension/resumption and BGO (Background Operation)/Multi FPSYS Operation.

The readability of flash memory during programming/erasure within one FPSYS is as follows.

– Code Flash read is possible during Data Flash programming/erasure

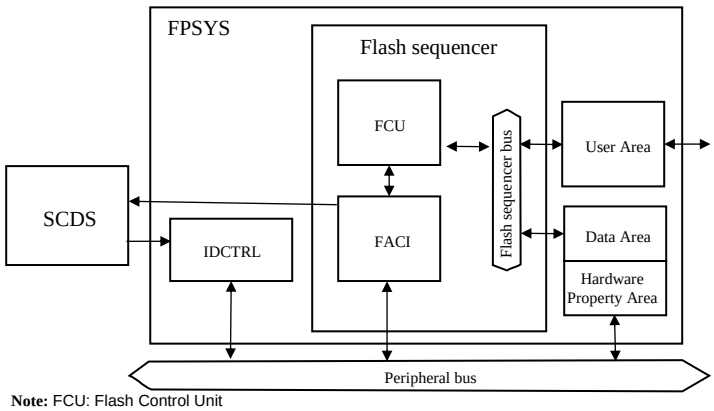
– Code Flash read from other bank is possible during programming/erasure of a bank of Code Flash

– Data Flash read from other Data Area is possible during programming/erasure of a Data Area of Data Flash

– Data Flash read is possible during Code Flash programming/erasure

Multi FPSYS Operation support

– The Flash memories that are belonging to the different FPSYS can be programmed/erased simultaneously. Figure 2-8 shows structure of Flash Programing Systems



Note: FCU: Flash Control Unit
FACI: Flash Application Command Interface
FPSYS: Flash Programming System
SCDS: System Configuration Data Storage (FPSYS0 only)
IDCTRL: ID control block (FPSYS0 only)
User Area: see the *RH850/U2A-EVA Group User's Manual: Hardware Section 51, Flash Memory*
Data Area: see the *RH850/U2A-EVA Group User's Manual: Hardware Section 51, Flash Memory*
Hardware Property Area: see the *RH850/U2A-EVA Group User's Manual: Hardware Section 51, Flash Memory*

Figure 2-8 Structure of Flash Programming System

2.3.2 Suspend function

FACI supports suspend programming/Erasure operation due to OTA software update is assumed low priority task than the other high priority tasks and can handle the interrupt of high priority task.

This application note and sample software does not include suspend operation during programing code flash. When execute software update on your software, implement the suspend operation and handle the other tasks.

3. OTA software update method using RH850/U2A

3.1 Method for remapping of code flash read address (Single map mode)

Figure 3-1 shows the method for reprogram using the .

This section describes the method for remapping of code flash in single map mode using GCFU. The Upper bank and lower bank of code flash in the same FPSYS are described as each “lower area” and “upper area” in this section.

The rewriting control program (Reprog code) is allocated to lower area and updatable area is allocated to upper area in single map mode.

(1) Before update
Prohibit program fetch to application software on upper area (updatable).

(2) Updating
Update software is programmed upper area (updatable) by reprog code on lower area (not updatable).

(3) After update
After reset released, enable program fetch to upper area where programmed by reprog code and remap the address of code flash using GCFU with microcontroller reset.

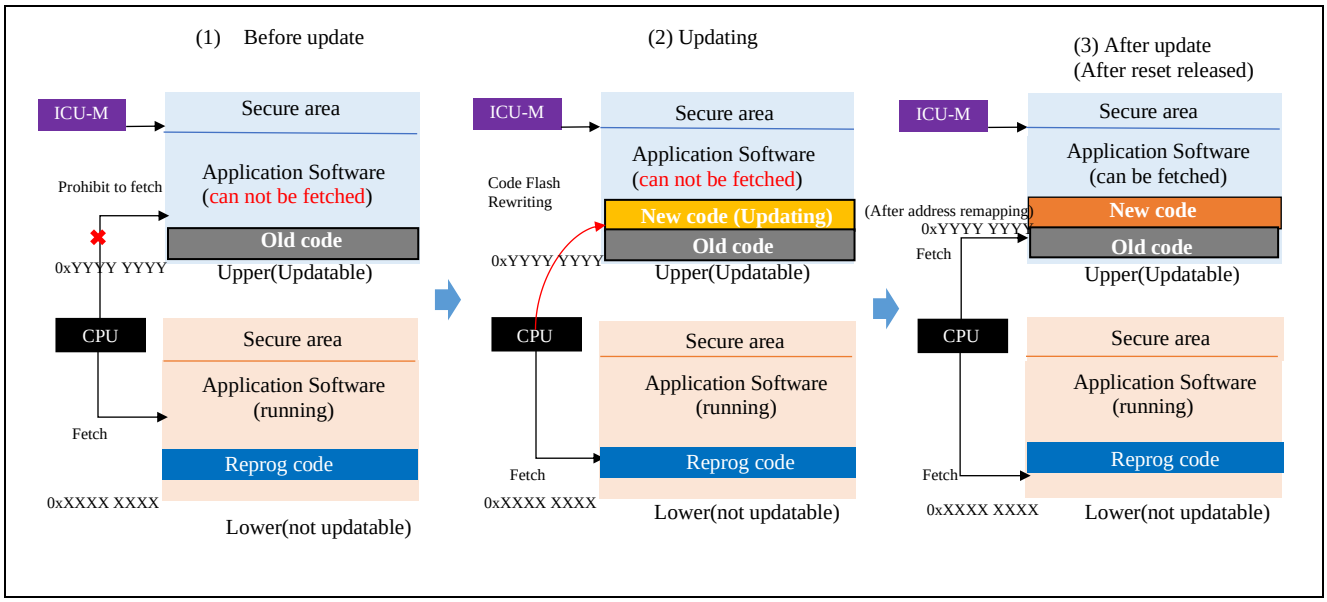


Figure 3-1 Method for reprogram diff code

Note

- The access of code flash area same as the bank during code flash programming is prohibited. Therefore, the source code to be executed during code flash programing must copied to secure RAM and RAM execution. (The size of secure RAM is limited.)
- The application that allocated to updatable area is needed to be considered that functions are separate into blocks due to programming/erasure are executed by 4 bytes unit.
- When data flash in same FPSYS is needed to program/erase during code flash program/erase, use the suspend function or forced stop command.

In this application note (5. Software operation in single map mode(with GCFU)), reprog code is allocated to block 0 to 3 in BankA and updatable area is allocated to block 0 to 3 in BankB and address of block 0 to 3 in BankA and BankB is remapped. When implemented the method for reprograming in your software, it is needed to be consider separation the functions into blocks. Also, this application note and software are not support security functions (e.g. ICU-MH). When implemented in your software, implement RAM execution as necessary.

3.2 Method for hardware swapping of address map between code flash memory banks(Double map mode)

Figure 3-2 shows the method for bank swapping.

This section describes the method for bank swapping in double map mode. The lower bank and upper bank of code flash in the same FPSYS are described as each “lower area” and “upper area” in this section.

(1) Updating
Update software is programmed upper area (invalid area) by reprog code on lower area (valid area).

(2) After programmed
After update of upper area is completed, configure settings for swapping address map of valid area and invalid area.

(3) After reset released
After reset released, address map is swapped with microcomputer reset and update application on lower area is executed.

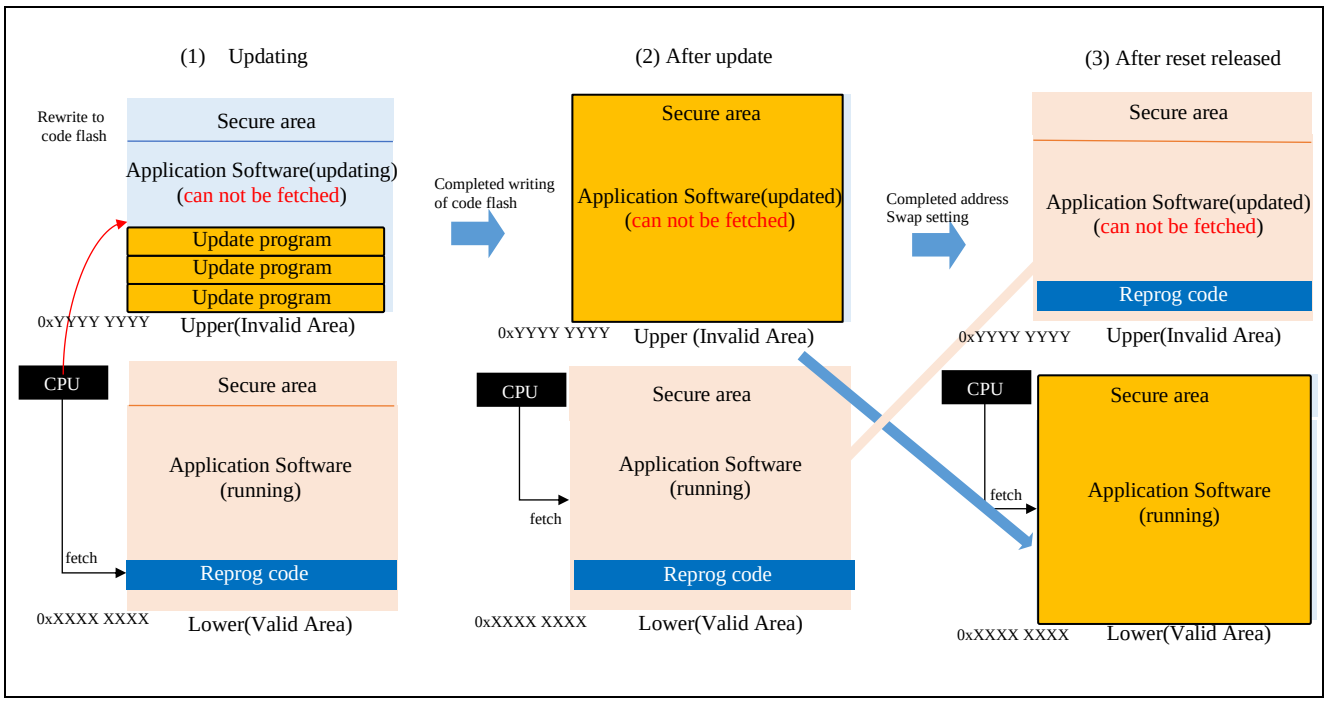


Figure 3-2 Method for bank swapping

Note

When data flash in same FPSYS is needed to program/erase during code flash program/erase, use the suspend function or forced stop command.

3.3 Note on security support

Software update with OTA (Over-the -air) technology require robust security measure to respond to threats of cyber-attack to vehicle.

This application note and sample software assigns CPU as access master for flash memory. When implementing security measures for in-vehicle networks, access master for flash memory requires ICU-MH during software update. Therefore, cyber security measure must be implemented with ICU-MH responsible for access master of flash memory in the OTA software update.[Note1] Figure 3-3 shows the diagram of access master.

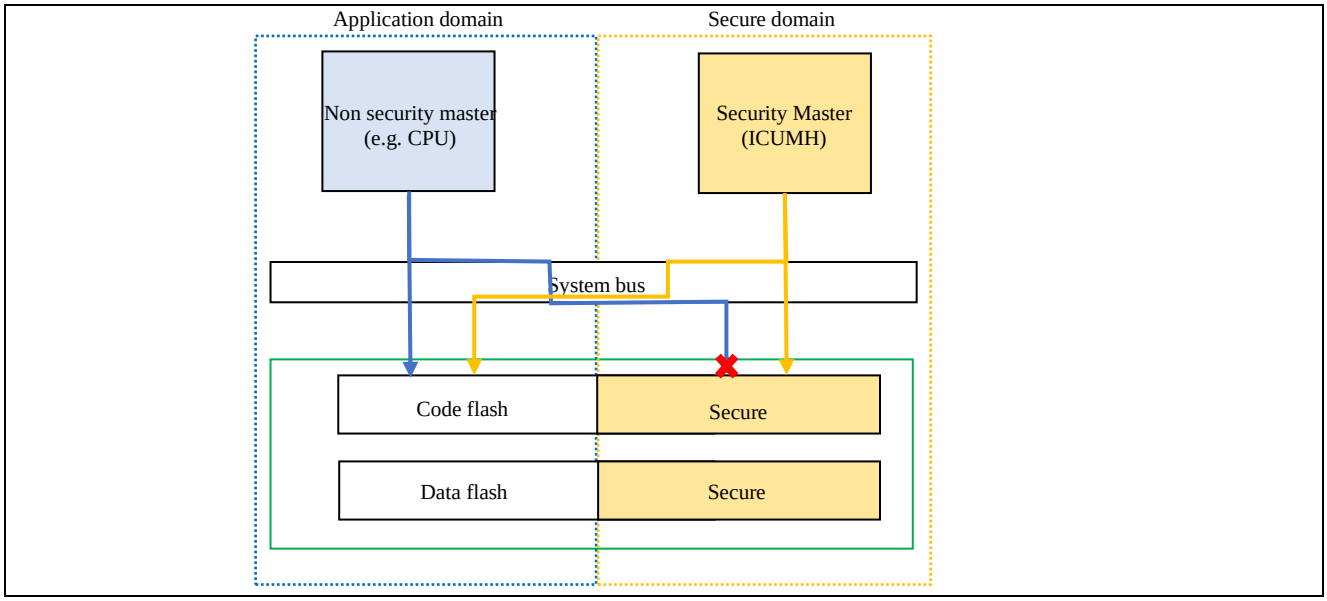


Figure 3-3 Diagram of access master.

U2A-EVA devices have guard functions for protection of flash memory. The guard functions are different for each access master of FACI and flash memory. [Note1]

In this application note, the update program from external device is received and programmed to code flash of U2A. The update program from external device must be encrypted for security measure. Also, authentication of encryption keys, detection for tampering and so on must be implemented. [Note1]

Figure 3-4 shows diagram of software update with encryption.

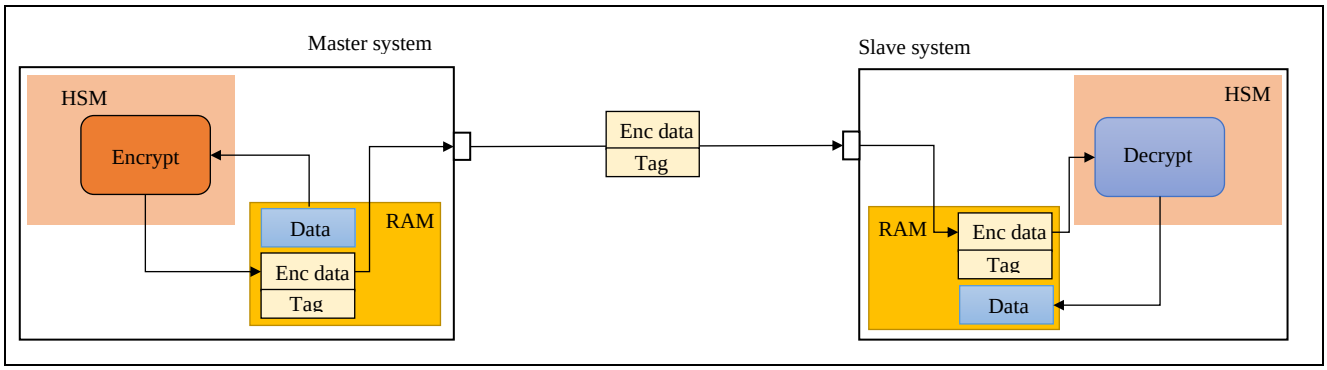


Figure 3-4 Diagram of communication with encryption

[Note1] This application note does not support security function that required to implement OTA software update in vehicle. For security, it is necessary that built on your system.

4. Configuration for Sample software

This application note provides the two types of sample software such as single map mode and double map mode.

The detail of the software, please refer to below.

- 5. Software operation in single map mode(with GCFU)
- 6. Software operation in double map mode(Bank swapping)

4.1 Hardware configuration

Figure 4-1 shows the hardware configuration of sample software. This is common configuration for each example.

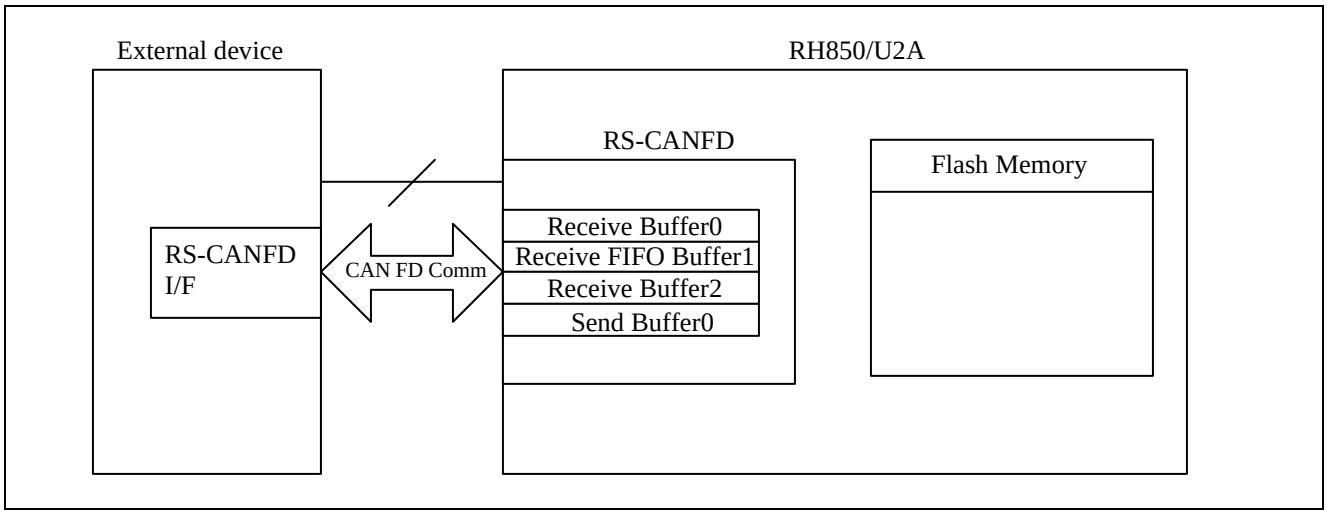


Figure 4-1 Hardware configuration

4.2 CAN FD communication

The specifications of CAN FD communication are explained below. The specification is common to both 5. Software operation in single map mode(with GCFU) and 6. Software operation in double map mode(Bank swapping)

.

The data written to code flash is stored to RAM via RS-CANFD(ch0) communication.

- The target device executes rewriting of code flash when received specified ID and data that transmit by external device via CAN FD. These ID and data are described “CAN FD command” in this application note.

4.2.1 CAN FD Communication Specifications

- Channel 0 is used.
- The communication speed is set to 1 Mbps for the nominal bit rate and 2 Mbps for data bit rate.
- Communication frame is set to CAN FD frame.
- The reception rule number of channel 0 is set to 2 to store each CAN FD command sent from the external device.

4.2.2 CAN FD command specification (from the U2A perspective)

- When U2A receives “Rewrite start command” by external device, rewriting of code flash is going to be started.
- When U2A receives “Write data download command”, the write data is transmitted to U2A from external device.
- When U2A transmits “Write data request command”, write data is requested to external device.
- When U2A transmits “Rewrite end command”, rewriting operation is finished.

Table 4-1 shows the CAN FD command specification.

Table 4-1 CAN FD command specification

Buffer	Channel	Command	Transmit/Receive	Standard ID	Length	Data
0	0	Rewrite start command	Receive	H’100	1Byte	H’00
1	0	Write data download command	Receive	H’110	64Byte	Data to write code flash (64Byte x 8)
1	0	Write data request command	Transmit	H’111	1Byte	H’11
2	0	Rewrite end command	Transmit	H’121	1Byte	H’22

4.3 Port(LED) specification

In this application note, LEDs connected U2A are blinked by port(P24_4/P24_5) output while program execute. The specification is common to both 5. Software update using the GCFU and 6. Software update using the hardware remapping.

Table 4-2 Port specification

Bank A operation	Bank B Operation	P24_4	P24_5
Operating rewrite program	-	LED blinking	-
-	Operating rewrite program	-	LED blinking

5. Software operation in single map mode(with GCFU)

5.1 Specification

5.1.1 Software specification

- This application note supports that user area in code flash is rewritten with OTA 2 bank configuration in single map mode. Table 5-1 shows the memory allocation.
- When the bank A is active, bank A is operated as 0x00000000 to 0x0000FFFF of logical address using GCFU function (Global Calibration Function Unit).
- When the bank B is active, bank B is operated as 0x00000000 to 0x0000FFFF of logical address using GCFU.
- The flag which bank is activate are stored in data flash. Address remapping is operated with this flag.
- The rewritable area is for a different bank from one in which the program operating.
- The mapping mode of code flash is single map mode, operation mode is user boot mode0 and boot area is user boot area0.
- The flash operation performed using the Renesas Flash Driver (RFD28F).
- Bank C and Bank D of code flash are out of scope of operation in this example.

Table 5-1 Memory allocation

Area	Physical address	Block	Bank	Size	OTA target
User boot Area 0	0x0800 0000 – 0x0800 FFFF	User boot Area0	BankA	64KB	Not-target
User Area0	0x0000 0000 – 0x0000 FFFF	Block 0-3	BankA	64KB	Target
User Area1	0x0040 0000 – 0x0040 FFFF	Block 0-3	BankB	64KB	Target

5.1.2 Operation overview

Activate bank is judged from the information for next wake up stored in data flash. When the active bank is A, address remapping does not perform. When the active bank is B, address remapping is performed. The rewriting operation of each case by active bank is shown below.

(a) The case of bank A is active

Address remapping by GCFU is not performed (Physical address=Logical address). The “Regrog code” rewrites the area for update area using FACI.

Figure 5-1 shows the writing operation in the case of Bank A is active

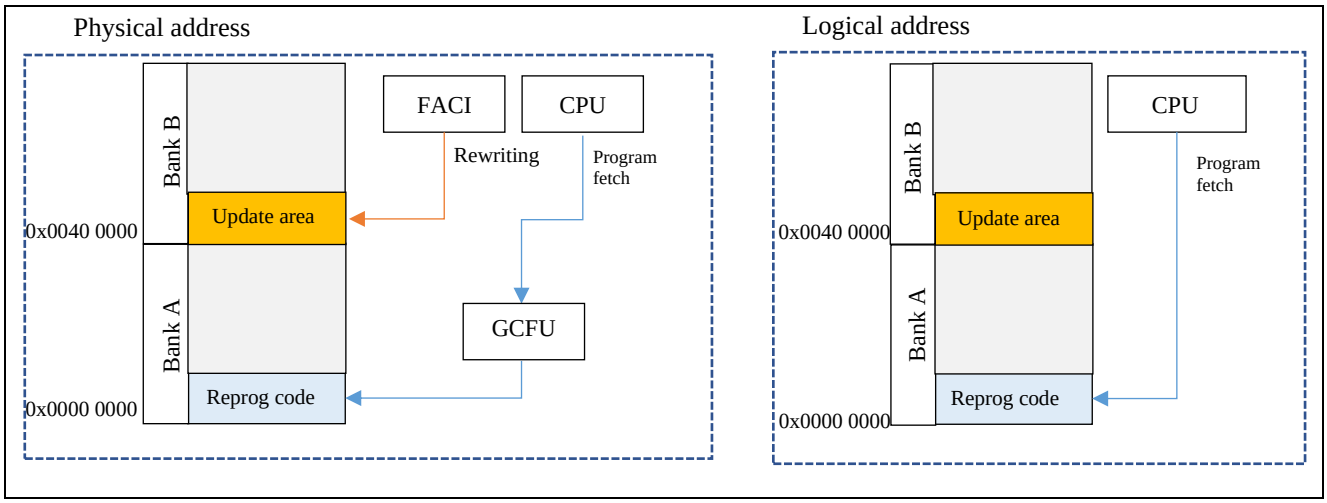


Figure 5-1 Rewriting operation in the case of Bank A is active

(b) The case of bank B is active

Address remapping is performed (Physical address != Logical address). The “reprog code” rewrites the area for BankA using FACI. Program fetch from CPU is executed to same address as when BankA is active and GCFU remaps the address of instruction fetch destination from CPU.

Figure 5-2 shows the writing operation in the case of Bank B is active.

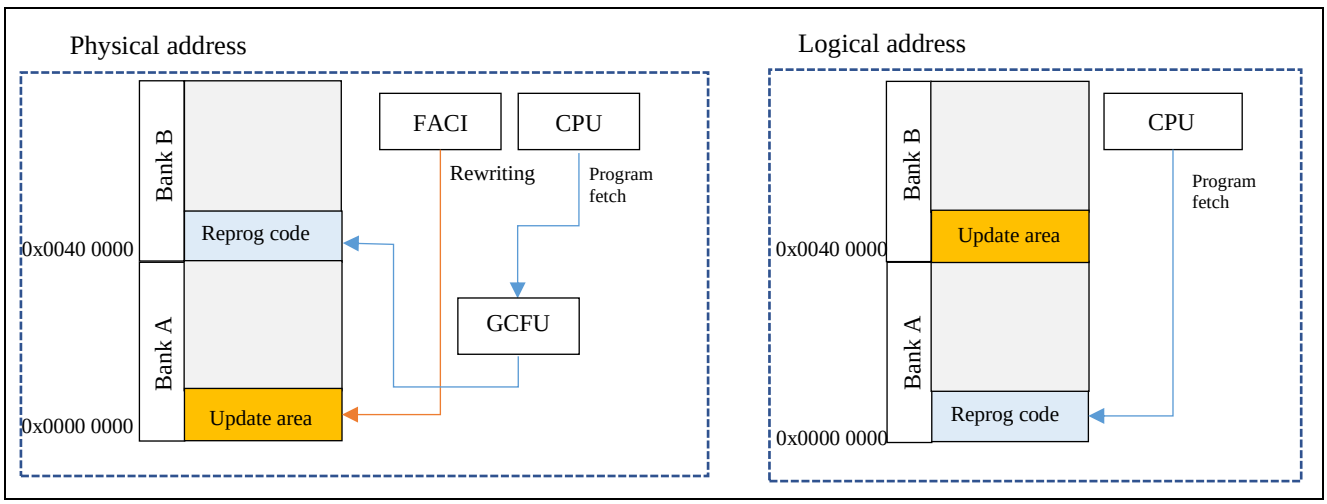


Figure 5-2 Rewriting operation in the case of Bank B is active

5.1.3 Overall sequence

Figure 5-3, Figure 5-4 shows overall sequence.

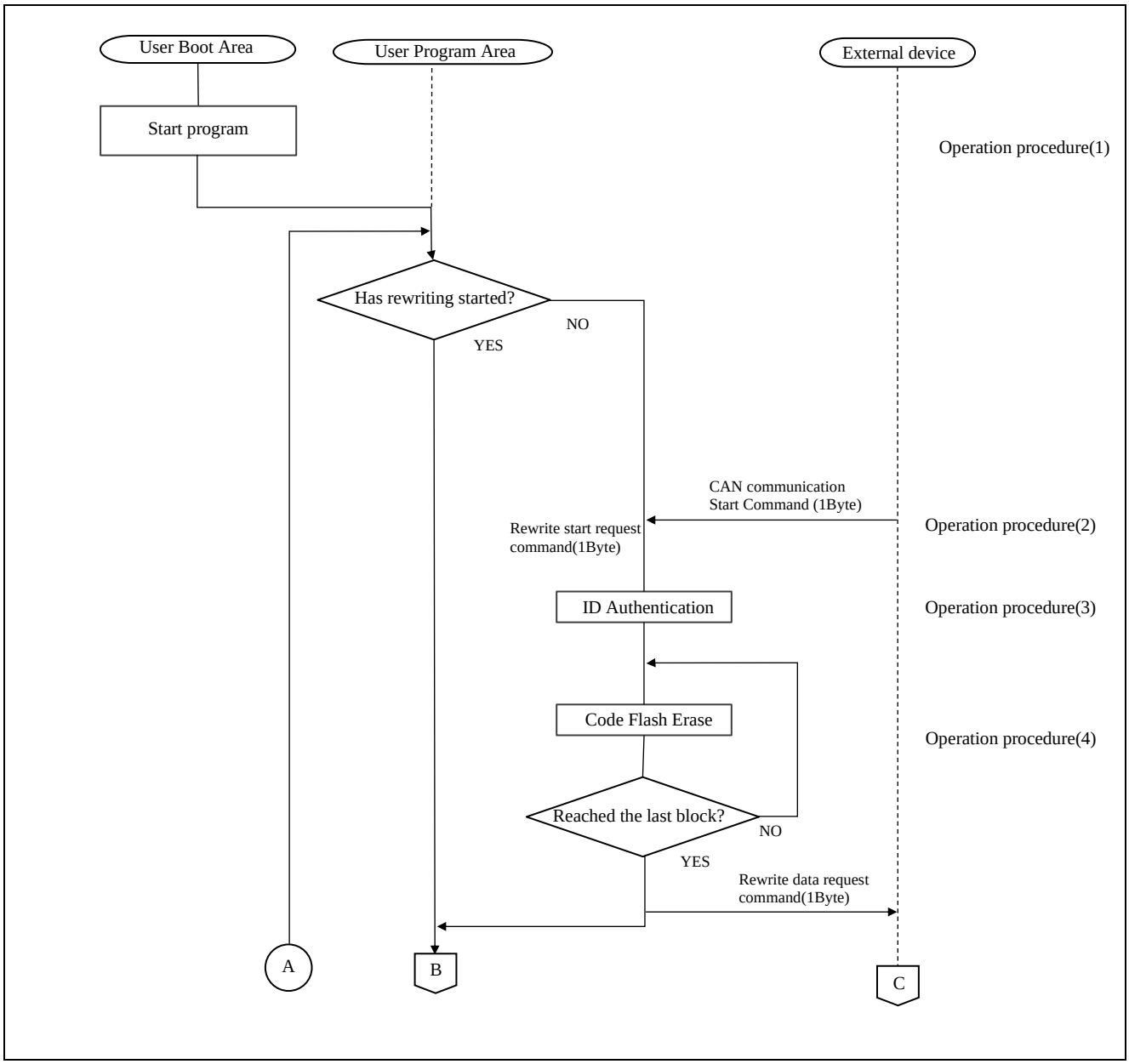


Figure 5-3 Overall sequence(1)

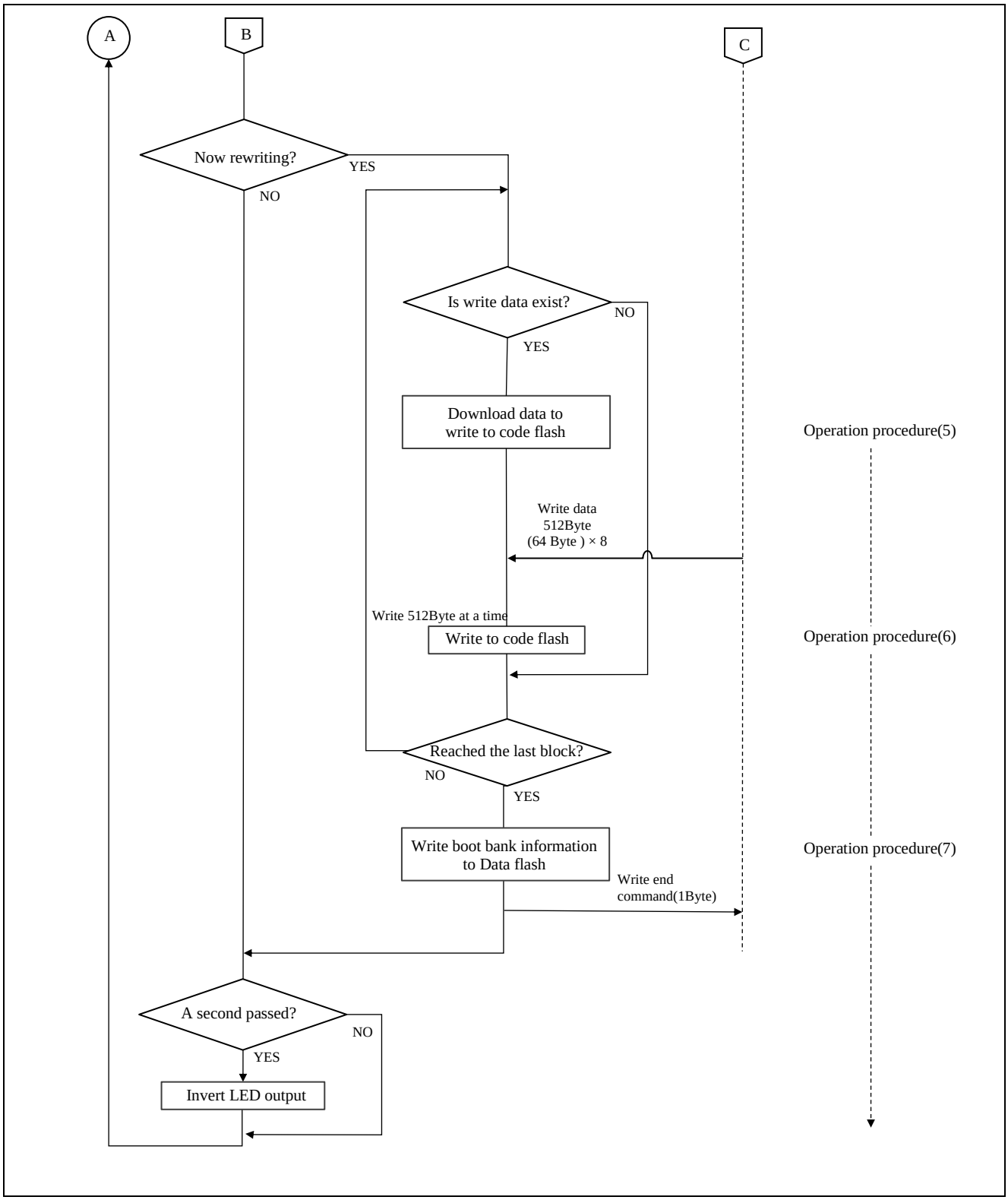


Figure 5-4 Overall sequence(2)

5.1.4 Functions used

- CANFD Interface(RS-CANFD)
- FACI
- GCFU
- Port

5.1.5 Operation mode

Operation mode is user boot mode 0 in this application note. After reset release, instruction fetch of PE0 is carried out from the user boot area in user boot mode 0.

The operation mode can be selected with mode pins and option byte. Option byte can be set by using Renesas Flash Programmer for RH850 family or configuration setting command of IDE(e.g. CS+, GHS Multi). Table 5-2 shows the operation mode selection.

Table 5-2 Operation mode selection

Pins			Option byte		Operation mode	Boot area
FLMD0	FLMD1	_TRST	STMSEL1	STMSEL0		
0	X	0	0	1	User boot mode0	User boot Area0

5.1.6 Memory mapping mode

The memory mapping mode is single map mode in this operation example. Table 5-3 shows the memory mapping selection.

Table 5-3 Memory mapping selection

Option byte setting		Mapping mode
MAPMODE1	MAPMODE 0	
0	1	Single Map mode

5.1.7 Configuration for RFD

Macro definitions for RFD in this operation example are shown in Table 5-4.

Table 5-4 Macro definitions

Macro definition	Setting value	description
R_RFD_VALUE_FORCED_STOP_TIMEOUT	Refer to sample program	Time out value of forced stop command
R_RFD_ERASURE_SUSPENDED_MODE	R_RFD_ERASURE_PRIORITY_MODE	Suspension-priority mode
R_RFD_REG_FAEINT_CFAEIE R_RFD_REG_FAEINT_CMDLKIE R_RFD_REG_FAEINT_DFAEIE R_RFD_REG_FAEINT_ECRCTIE	R_RFD_DISABLE	Does not generate the FLERR interrupt
R_RFD_CONTROL_TARGET_DATAFLASH	R_RFD_ENABLE	Data flash memory is control target by RFD28F
R_RFD_CONTROL_TARGET_CODEFLASH	R_RFD_ENABLE	Code flash memory is control target by RFD28F
R_RFD_MAPMODE	R_RFD_SINGLE	Single map mode
R_RFD_BLOCK_PROTECTION_AREA1	R_RFD_ENABLE	Block Protection Area 1 exists
R_RFD_FPMON_CHECK	R_RFD_ENABLE	Confirmation for FPMON register is valid

5.2 Operation procedure

Operation procedure (1) to (7) correspond to ”5.1.3 Overall sequence”.

5.2.1 (1) Bank switching

After reset start, the start program on the user boot area reads data flash to judge the boot bank, and switches to appropriate bank and jumps to the user program. Figure 5-5 shows the start program location.

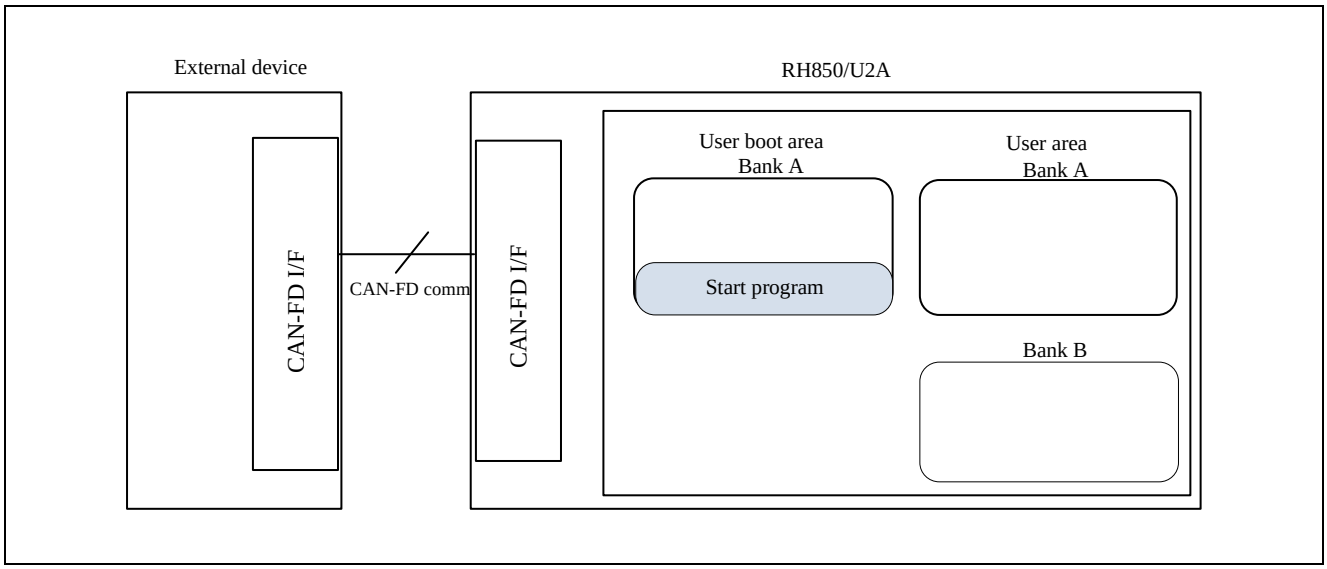


Figure 5-5 Startup program location

Table 5-5 “main_pm0()”

Function	Argument	Description
main_pm0()	N/A	After judged the boot bank, switches to appropriate bank and jumps to the user program.

Figure 5-6 shows flowchart of main_pm0.

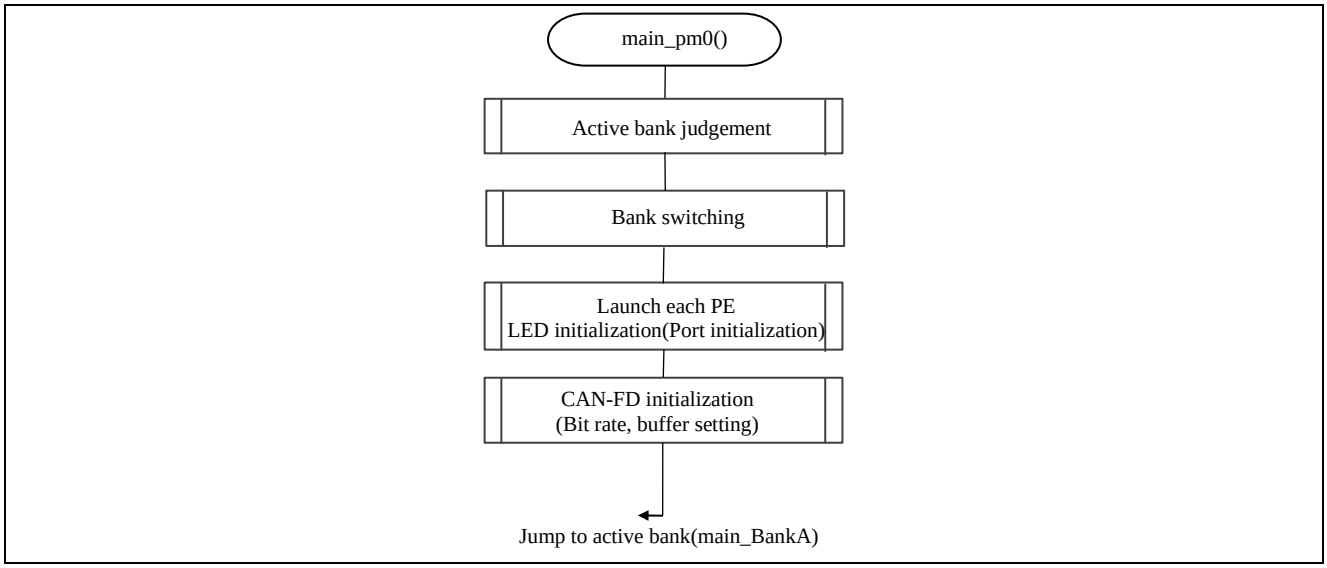


Figure 5-6 Flowchart of main_pm0()

“Operation procedure (1)”

Figure 5-7 shows flowchart of “Bank switching function”.

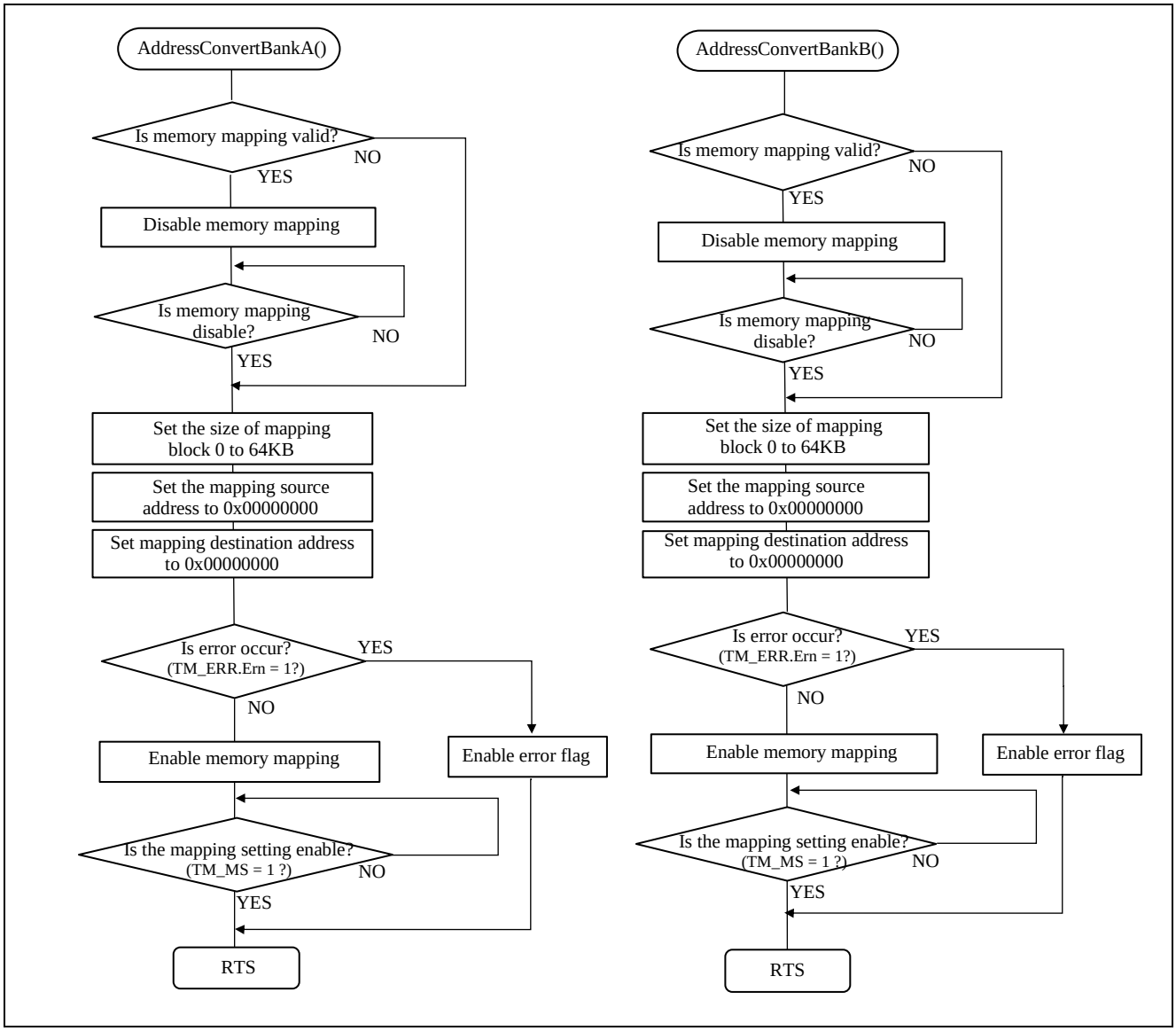


Figure 5-7 Flowchart of “Bank switching function”

5.2.2 (2) Code flash reprogramming with background operation (BGO) function

The “Rewrite control program” can be performed without transmitting to RAM by rewrite target area being different from active bank. Bank B area of code flash is written by “Rewrite control program” in the case of bank A is active.

The main routine executes writing update program to code flash after the rewrite start command is received while constantly blinks LED.

Figure 5-8 shows the program location of each Bank.

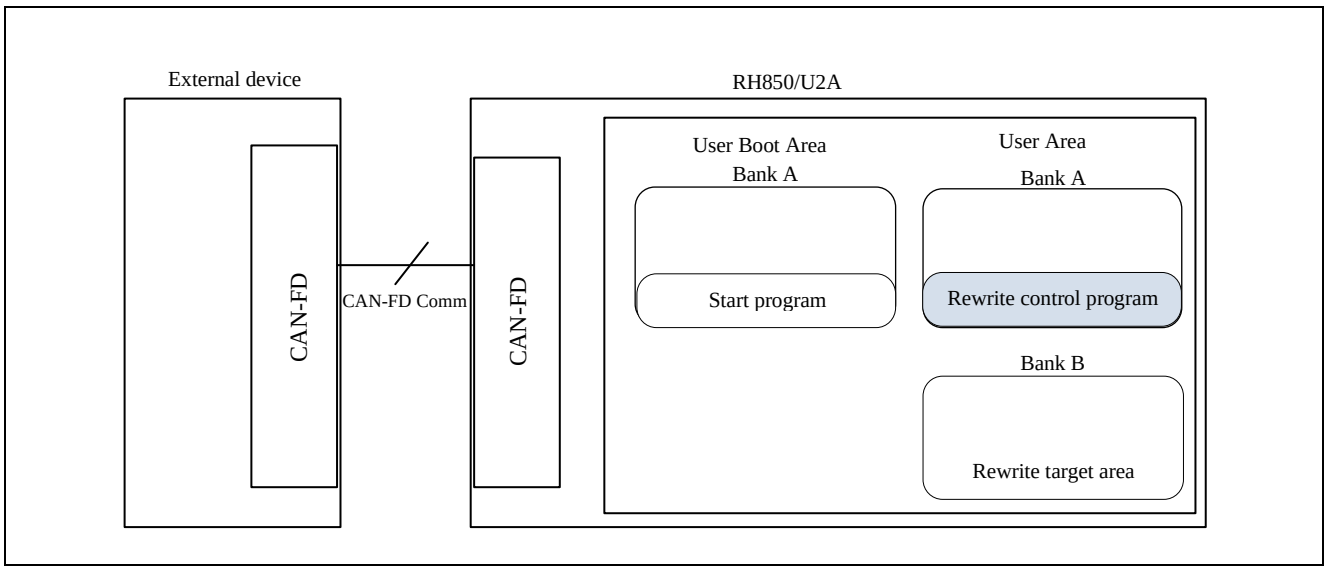


Figure 5-8 Program location of each Bank

Function description

Table 5-6 “main_BankA()”

Function	Argument	Description
main_BankA ()	N/A	The “Rewrite control program” is executed after the rewrite start command is received while constantly blinks LED.

Figure 5-9 shows the flowchart of “main_BankA()” function.

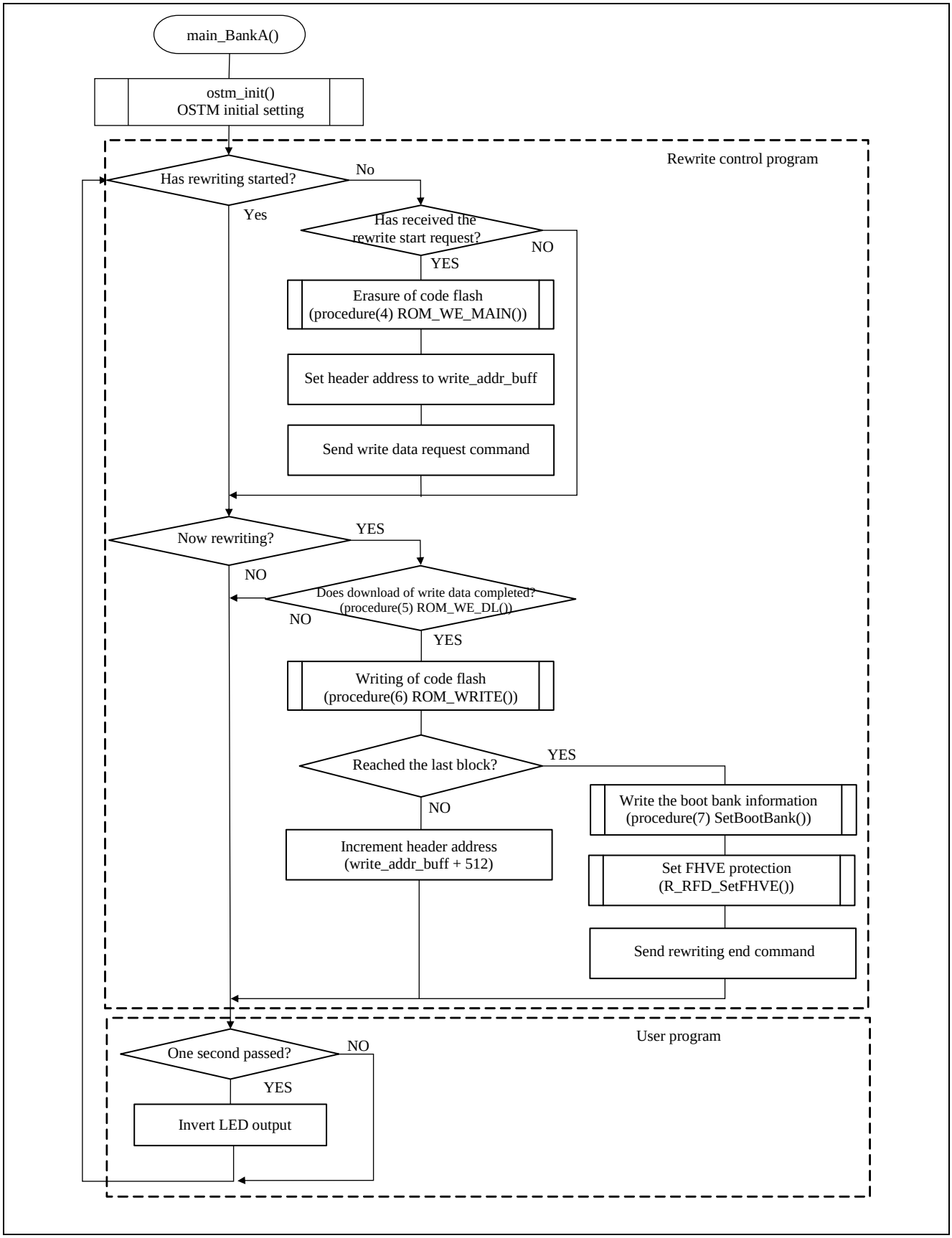


Figure 5-9 Flowchart of “main_BankA()” function

“Operation procedure (2)”

Function description

Table 5-7 “ROM_WE_MAIN()” function

Function	Argument	Description
ROM_WE_MAIN()	N/A	Execute some function such as ID authentication, erasure of code flash, download of writing data and programing of code flash.

Figure 5-10 shows flowchart of “ROM_WE_MAIN()” function.

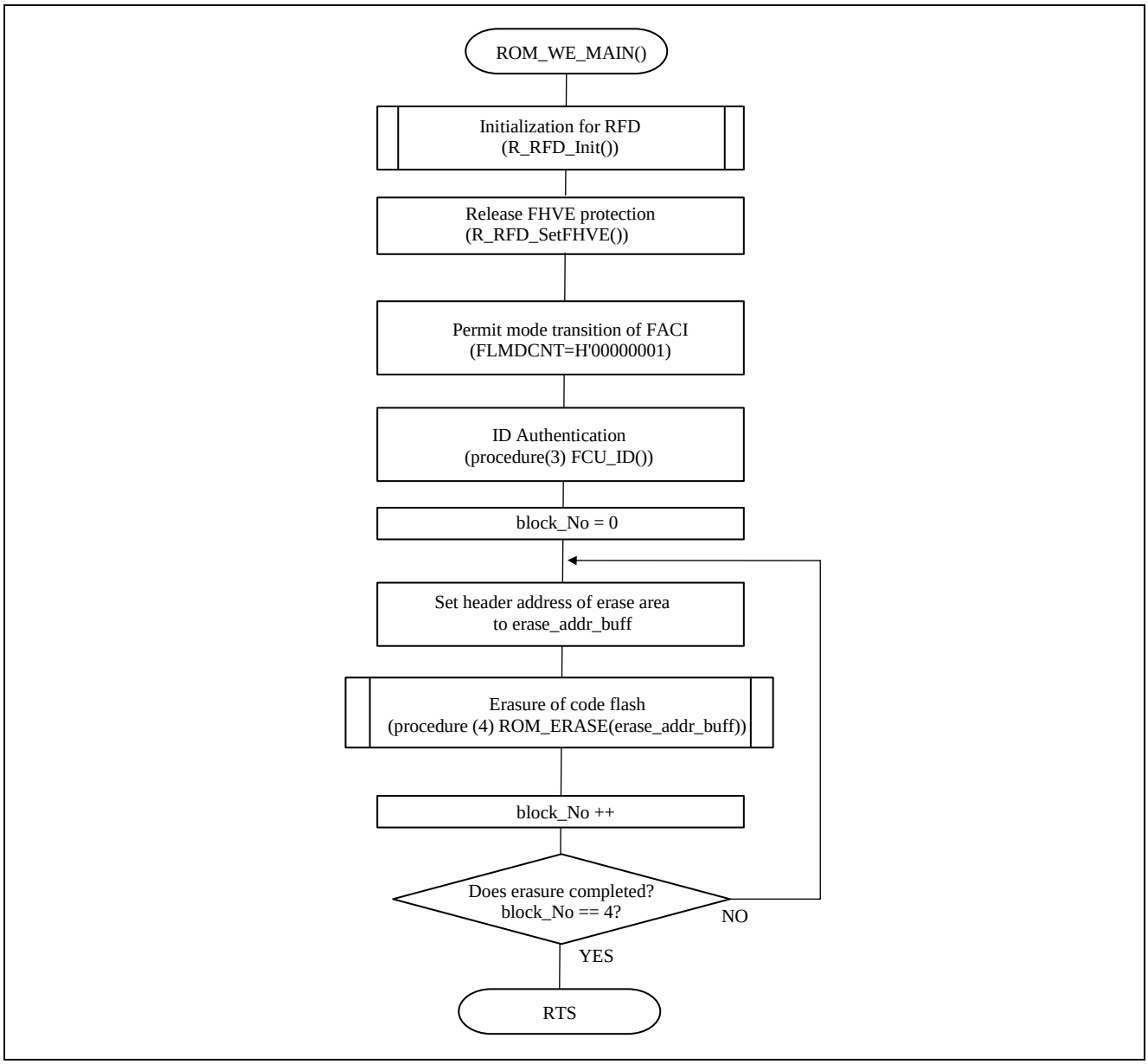


Figure 5-10 Flowchart of “ROM_WE_MAIN()” function

Included operation procedure (3) and (4).

5.2.3 (3) ID authentication

ID authentication in “Rewriting control program” is executed by comparing the preset 256bits ID and customer ID. In this example of operation, the IDs are set “0” for the first byte and “F” for other. The customer ID can be set by using Renesas Flash Programmer for RH850 family or configuration setting command of IDE(e.g. CS+, GHS Multi).

Function description

Table 5-8 “FCU_ID() function”

Function	Argument		Description
FCU_ID()	N/A		Execute ID authentication.
API	Argument		Description
R_RFD_IDAuth	T_en_IDType	R_RFD_ID_CUSTIDA/B/C	Execute ID authentication.
	T_u1*	Pointer for ID data	

Figure 5-11 shows flowchart of “FCU_ID()” function.

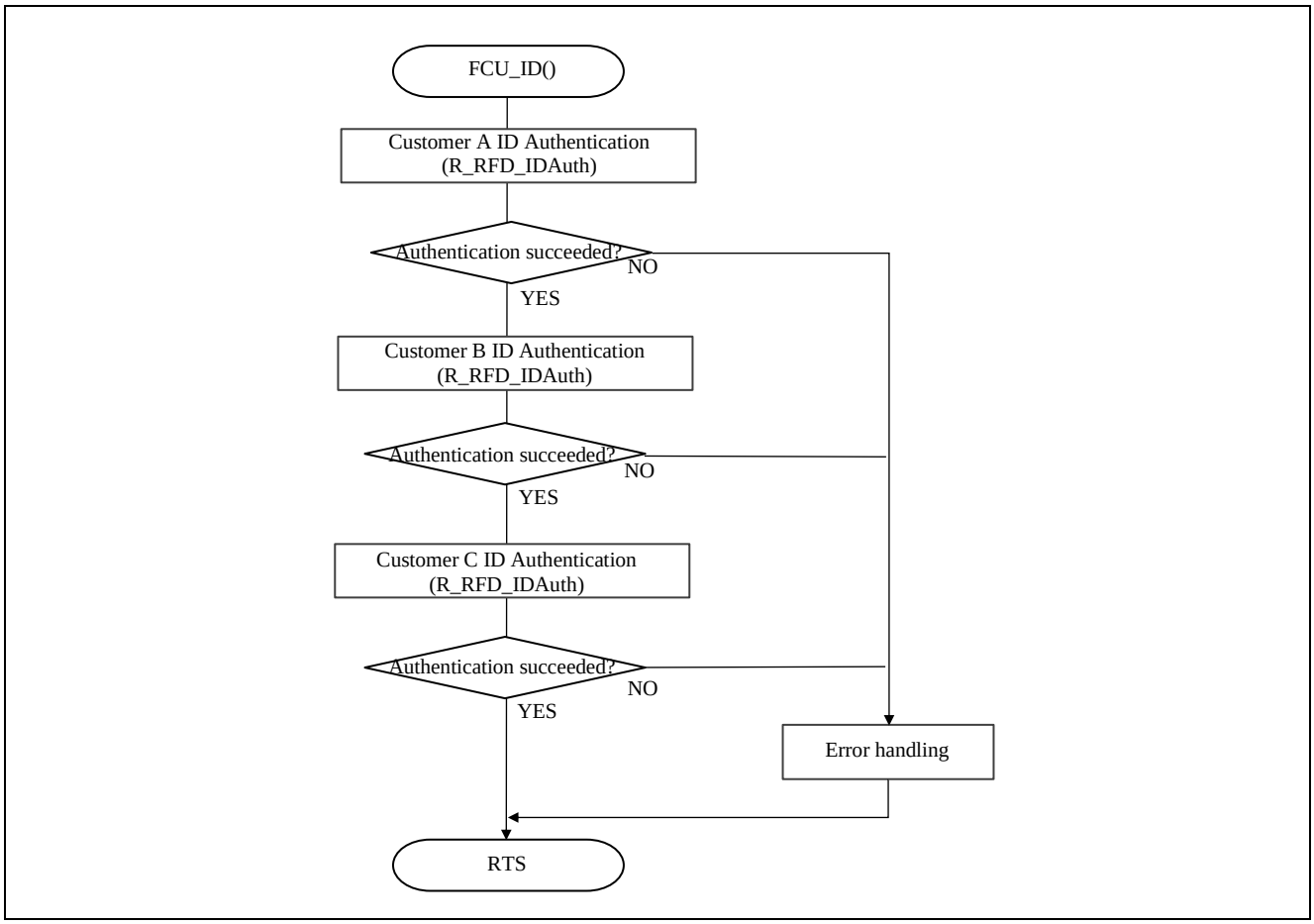


Figure 5-11 Flowchart of “FCU_ID()” function

“Operation procedure (3)”

5.2.4 (4) Code Flash Erasure

ROM_ERASE() function in “Rewrite control program” is executed after ID authentication was succeeded.

Issue the “Erase Command” of FACI to erase the contents of code flash memory area specified by the parameter.

Function description

Table 5-9 “ROM_ERASE()” function

Function	Argument		Description
ROM_ERASE()	erase_addr	Header address of erasure area	Erases the specified area of Code Flash
API	Argument		Description
R_RFD_ShiftToPEMode	T_u2_FACI	R_RFD_FACI0	Shift the target memory mode in the target FACI to P/E mode.
	T_en_FACIMode	R_RFD_MODE_CFPE	
R_RFD_ShiftToReadMode	T_u2_FACI	R_RFD_FACI0	Shift the target memory mode in the target FACI to read mode.
R_RFD_GetFaciStatus	T_u2_FACI	R_RFD_FACI0	Check whether target FACI is command lock or error occurred.
R_RFD_GetFaciSequenceRead	T_u2_FACI	R_RFD_FACI0	Check whether the target FACI is in ready state.
R_RFD_StatusClear	T_u2_FACI	R_RFD_FACI0	Clear FACI status
R_RFD_ForcedStopAndErrorClear	T_u2_FACI	R_RFD_FACI0	Issue forced stop commend
R_RFD_EraseCFRequest	T_u4_RfdAddress	Header address of erasure area	Erase the contents of the code flash memory.

Figure 5-12 shows the flowchart of “ROM_ERASE()” function.

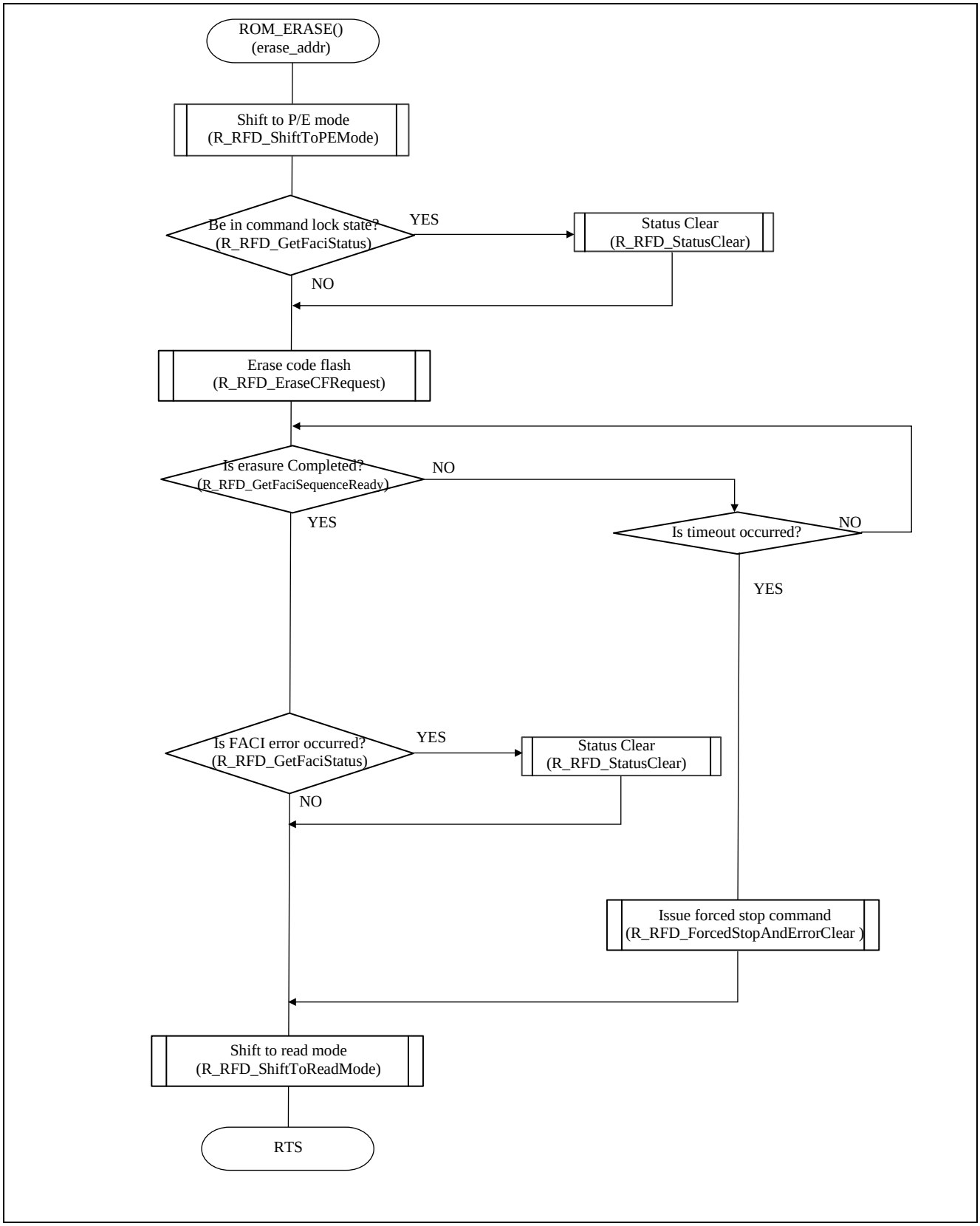


Figure 5-12 Flowchart of “ROM_ERASE()” function

Operation Procedure (4)

5.2.5 (5) Download of write data

Execute the “ROM_WE_DL” function and transmit the write data request command. External device transmits the 512byte data to U2A when receive the write data request command. The data from external device is saved on RAM in this function.

Function description

Table 5-10 “ROM_WE_DL()”function

Function	Argument	Description
ROM_WE_DL()	N/A	Download the write data from external device

Figure 5-13 shows the flowchart of “ROM_WE_DL()” function.

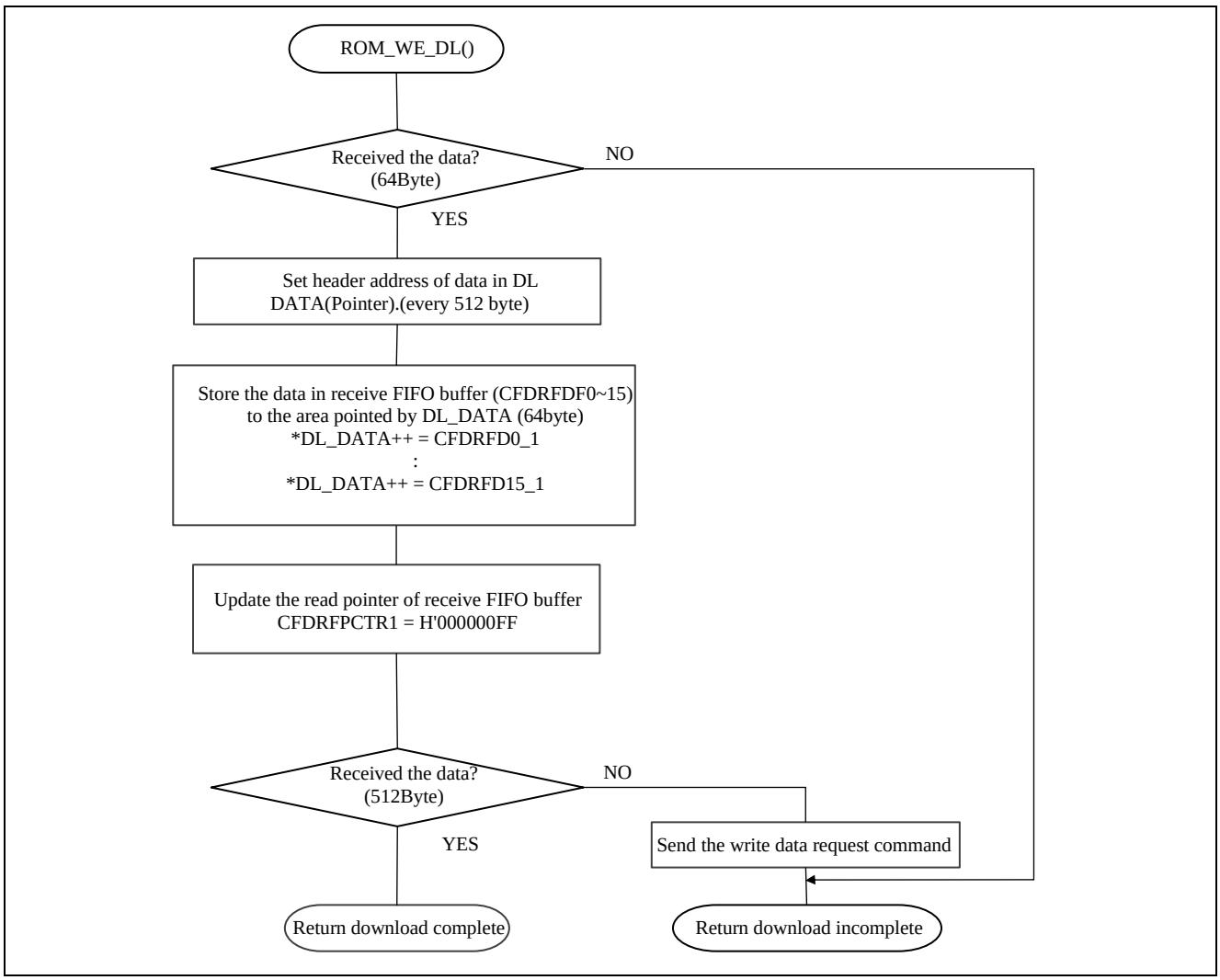


Figure 5-13 Flowchart of “ROM_WE_DL()” function

“Operation procedure (5)”

5.2.6 (6) Code flash programming

The data that received from external device is programmed to code flash by using “ROM_Write()” function. Issue the “Program Command” of FACI to write to code flash area specified by the parameter. The programming is finished when the limit size (64 Kbyte) is reached.

Function description

Table 5-11 “ROM_WRITE()” function

Function	Argument		Description
ROM_WRITE()	write_addr	Header address for writing area	Write the data to the code flash memory. (512Byte unit)
API	Argument		Description
R_RFD_ShiftToPEMode	T_u2_FACI	R_RFD_FACI0	Shift the target memory mode in the target FACI to P/E mode.
	T_en_FACIMode	R_RFD_MODE_CFPE	
R_RFD_ShiftToReadMode	T_u2_FACI	R_RFD_FACI0	Shift the target memory mode in the target FACI to read mode.
R_RFD_GetFaciStatus	T_u2_FACI	R_RFD_FACI0	Check whether target FACI is command lock or error occurred.
R_RFD_GetFaciSequenceRead	T_u2_FACI	R_RFD_FACI0	Check whether the target FACI is in ready state.
R_RFD_StatusClear	T_u2_FACI	R_RFD_FACI0	Clear FACI status
R_RFD_ForcedStopAndErrorClear	T_u2_FACI	R_RFD_FACI0	Issue forced stop commend
R_RFD_WriteCFRequest	T_u4_RfdAddress	Header address of erasure area	Writes the contents of the code flash memory.

Figure 5-14 shows the flowchart of “ROM_WRITE()” function.

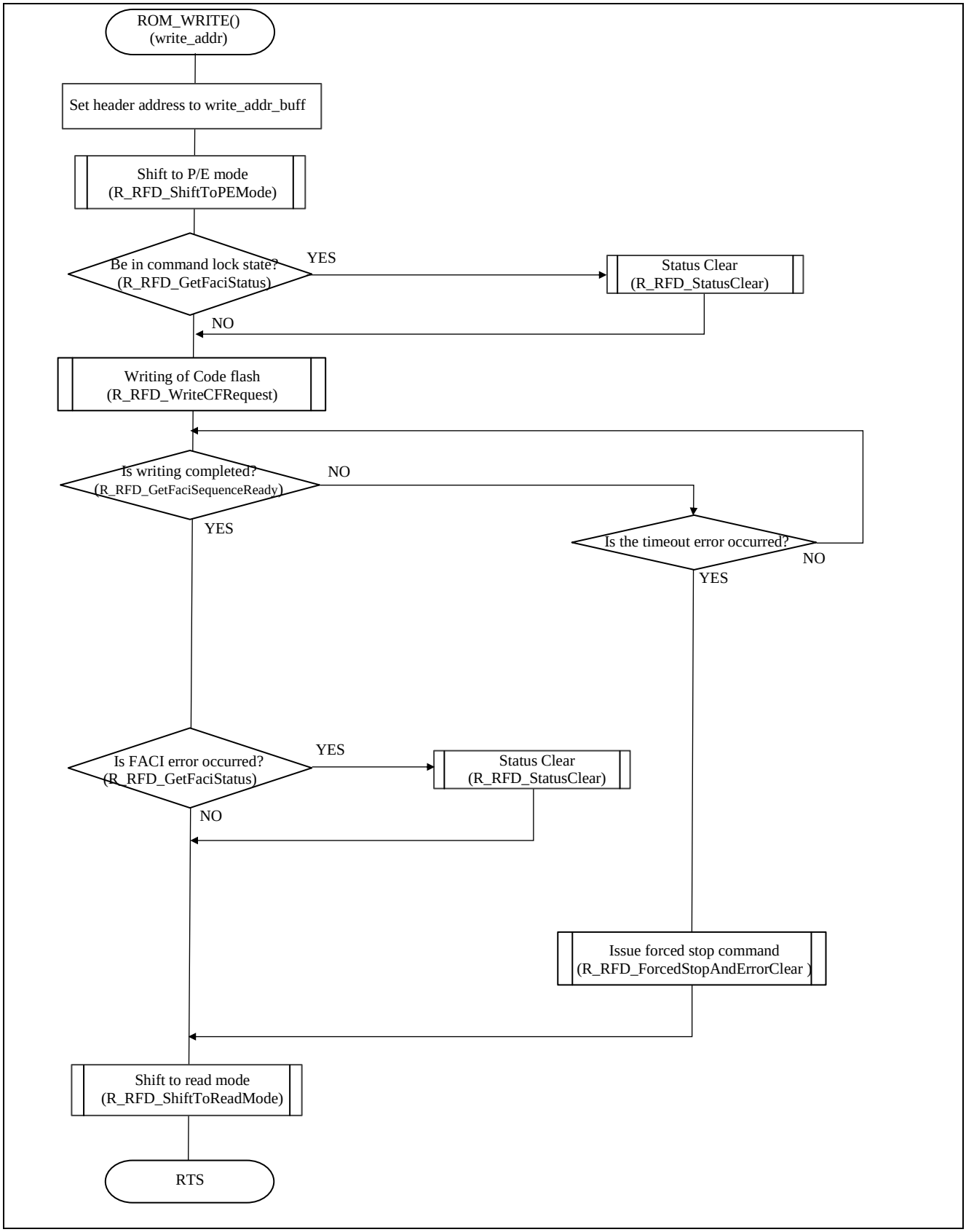


Figure 5-14 Flowchart of “ROM_WRITE()” function

“Operation procedure (6)”

5.2.7 (7) Writing of boot bank information to data flash

After the rewriting of code flash is completed, the boot bank information is written to the data flash.

Function Description

Table 5-12 “SetBootBank ()” function

Function	Argument		Description
SetBootBank ()	N/A		The boot bank for the next wakeup is judged and the information is written to data flash.
API	Argument		description
R_RFD_ShiftToPEMode	T_u2_FACI	R_RFD_FACI0	Shift the target memory mode in the target FACI to P/E mode.
	T_en_FACIMode	R_RFD_MODE_CFPE	
R_RFD_ShiftToReadMode	T_u2_FACI	R_RFD_FACI0	Shift the target memory mode in the target FACI to read mode.
R_RFD_EraseDFRequest	T_u4_RfdAddress	Header address	Erasure of data flash
R_RFD_WriteDFRequest)	T_u4_RfdAddress	Header address	Programming of data flash

Figure 5-15 shows the flowchart of “SetBootBank()” function.

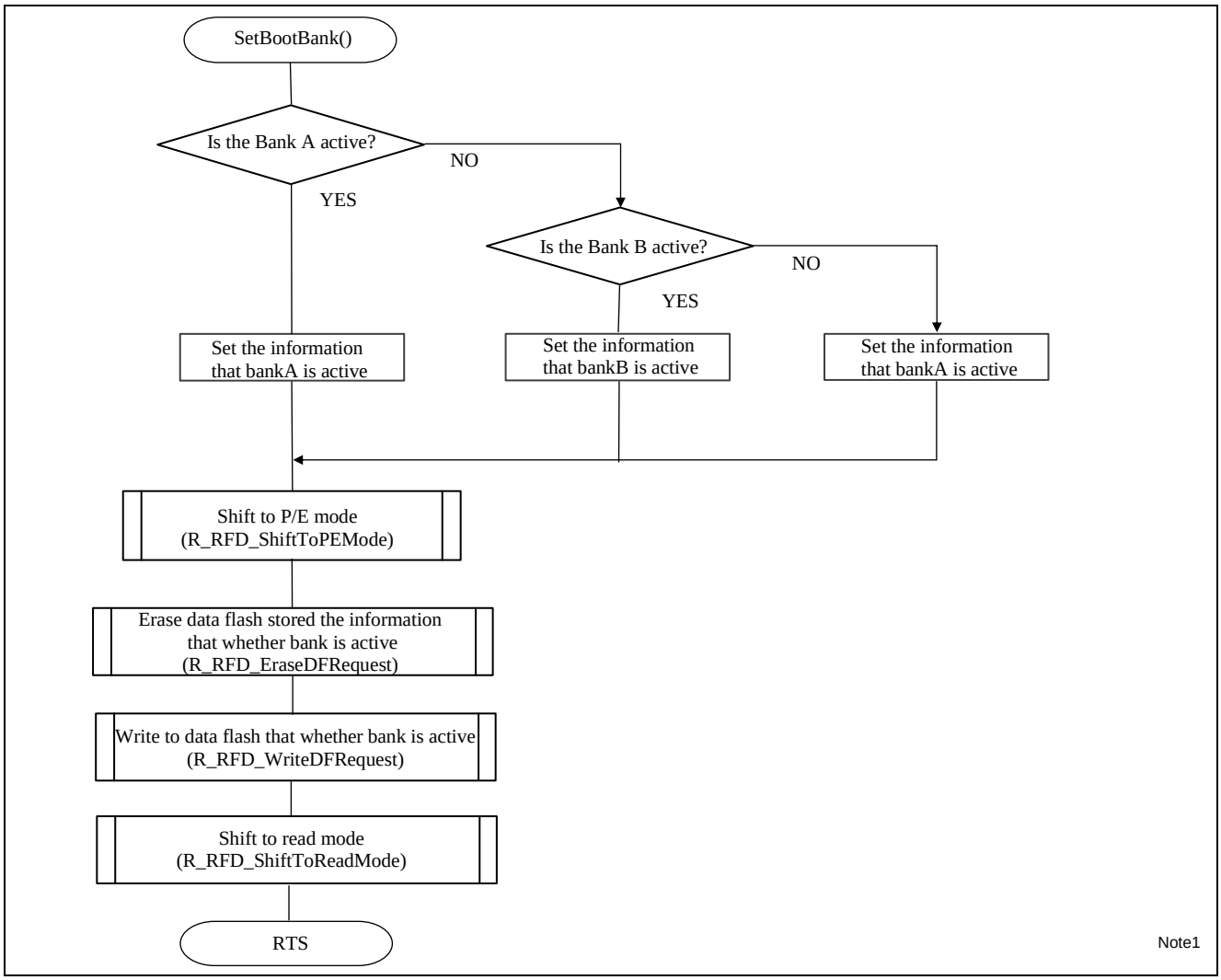


Figure 5-15 Flowchart of SetBootBank () function

“Operation procedure (7)”

Note1 ID Authentication for data flash is needed when S_OPBT4.DPROT is set to 0.

5.3 Memory allocation

Table 5-13 shows memory allocation in single map mode.

Table 5-13 Memory allocation

Area	Address	Section	Access
Code Flash User Area0(Bank A)	H'0000_0000	.BankA.text	PE0
		.BankA.const	
		.FlashControl.text	
		.R_RFD_RODATA_EXTRA.const	
		.R_RFD_CODE_COMMON.text	
		.R_RFD_CODE_COMMON_RAM_NO_BGO.text	
		.R_RFD_CODE_USEROWN_COMMON.text	
		.R_RFD_CODE_DF.text	
		.R_RFD_CODE_CF.text	
		.R_RFD_CODE_CF_RAM_NO_BGO.text	
		.R_RFD_CODE_EXTRA.text	
		.R_RFD_RODATA_VERSION_DF.const	
		.R_RFD_RODATA_VERSION_CF.const	
		.R_RFD_RODATA_VERSION_COMMON.const	
	H'0002_0000	RESET_PE1	PE1
		EIINTTBL_PE1	
		.const	
		.INIT_DSEC.const	
		.INIT_BSEC.const	
		.text.cmn	
		.text	
		.data	
Code Flash User Area2(Bank C) ※Available only for U2A16	H'0080_0000	RESET_PE2	PE2
		EIINTTBL_PE2	
		.const	
		.INIT_DSEC.const	
		.INIT_BSEC.const	
		.text.cmn	
		.text	
		.data	
Code Flash User Area3(Bank D) ※Available only for U2A16	H'00C0_0000	RESET_PE3	PE3
		EIINTTBL_PE3	
		.const	
		.INIT_DSEC.const	
		.INIT_BSEC.const	
		.text.cmn	
		.text	
		.data	
Code Flash User boot area0(Bank A)	H'0800_0000	RESET_PE0	PE0
		EIINTTBL_PE0	
		.const	
		.INIT_DSEC.const	
		.INIT_BSEC.const	
		.text.cmn	
		.data	
		.text	
	H'0800_4000	.text	PE0～PE3
LRAM(PE3) ※Available only for U2A16	H'FD60_0000	.data.R	PE3
		.bss	
		.stack.bss	
LRAM(PE2) ※Available only for U2A16	H'FD80_0000	.data.R	PE2
		.bss	
		.stack.bss	
LRAM(PE1)	H'FDA0_0000	.data.R	PE1
		.bss	
		.stack.bss	
LRAM(PE0)	H'FDC0_0000	.bss	PE0
		.BankA.bss	
		.data.R	
		.stack.bss	
		.R_RFD_BSS.bss	
Data Flash(Block 0)	H'FF20_0000	.BootBankInfo.const	PE0

6. Software operation in double map mode(Bank swapping)

6.1 Specification

6.1.1 Software specification

- In this operation, update program is written to code flash in double map mode.
- Code flash is divided into two areas between valid area and invalid area and these areas are swapped by settings of hardware property areas.
- Update software is written to invalid area during program is running in valid area.
- Target of software update is user area 0 and 1 (BankA, BankB) of code flash. The address of code flash between BankA and BankB is swapped by settings of hardware property area described below.
- Extended data areas of code flash (User boot area, Product info Area, ECC Test Area) between BankA and BankB are also swapped by settings of hardware property area.
- Table 6-1 shows memory allocation of code flash when BankA is valid. In this operation, BankC and BankD of code flash are not written and swapped.
- User Boot Area can be written only in serial programming mode. BankA and BankB of User Boot Area is needed to be written the same program in advance. The Product Info Area stores Product information. This area is not rewritable.

Table 6-1 Memory allocation of code flash (When BANK A is valid)

Code Flash Area	Physical address	Block	Valid/invalid	Size	OTA target
User Area 0 (U2A16/U2A8)	0x0000 0000 – 0x003F FFFF	Block 0-69	Valid	4MB	Non-target
User Area 0 (U2A6)	0x0000 0000 – 0x002F FFFF	Block 0-53		3MB	
User Area 1 (U2A16/U2A8)	0x0200 0000 – 0x023F FFFF	Block 0-69	Invalid	4MB	Target
User Area 1 (U2A6)	0x0200 0000 – 0x022F FFFF	Block 0-53		3MB	
User Boot Area 0	0x0800 0000 – 0x0800 FFFF	User Boot Area 0	Valid	64KB	Non-target
User Boot Area 1	0x0A00 0000 – 0x0A00 FFFF	User Boot Area 1	Invalid	64KB	Non-target
Product info Area 0	0x0803 0000 – 0x0803 7FFF	Product info Area 0	Valid	32KB	Non-target
Product info Area 1	0x0A03 0000 – 0x0A03 7FFF	Product info Area 1	Invalid	32KB	Non-target
ECC Test Area 0	0x0805 0000 – 0x0805 FFFF	ECC Test Area 0	Valid	64KB	Non-target
ECC Test Area 1	0x0A05 0000 – 0x0A05 FFFF	ECC Test Area 1	Invalid	64KB	Non-target

- Configuration setting area, Security setting area, Block protection area0/1, Switch Area consist valid area and invalid area and these areas can be swapped.
- Address of code flash is swapped by rewriting of Configuration setting area, Switch Area and TAG Area in this operation.
- In double map mode, code flash between BankA and BankB is swapped by setting of OPBT13.DBMAPSW0 on Configuration Setting Area.
- Valid area of Configuration setting area is set by CVA on Switch Area. Valid area of Switch area is set by VAF on TAG Area. Valid area of Configuration Setting Area is swapped by rewriting of OPBT13.DBMAPSW0 of invalid area on Configuration Setting Area and settings of Switch Area and TAG Area in this operation.
- After the above settings are completed, address swapping is executed from reset release.
- Note that the Hardware Property handled by this function is only the “Configuration Setting Area”. When you want to operate the “Security Setting Area”,” Block Protection Area0”, and “Block Protection Area1”, the processing procedure is the same and only the operation target changes. Therefore, please process according to each.

Table 6-2 Memory allocation (At the shipping)

Date Flash Area	Physical address	Block	Valid/invalid	Size	OTA target
Hardware Property Area0	0xFF32 0000- 0xFF32 07FF	Extended Data Area	Valid	2KB	Non-target
	0xFF32 0800- 0xFF32 0FFF	Configuration Setting Area0	Valid	2KB	Non-target
	0xFF32 1000 0xFF32 17FF	Configuration Setting Area1	Invalid	2KB	Target
	0xFF32 1800- 0xFF32 1FFF	Security Setting Area0	Valid	2KB	Non-target
	0xFF32 2000- 0xFF32 27FF	Security Setting Area1	Invalid	2KB	Non-target
	0xFF32 2800- 0xFF32 2FFF	Block Protection Area for FPSYS0 Area0	Valid	2KB	Non-target
	0xFF32 3000- 0xFF32 37FF	Block Protection Area for FPSYS0 Area1	Invalid	2KB	Non-target
	0xFF32 5000- 0xFF32 67FF	Erase Counter Area for User Area 0, User Boot Area0	Valid	2KB×3	Non-target
	0xFF32 6800- 0xFF32 7FFF	Erase Counter Area for User Area 1, User Boot Area1	Invalid	2KB×3	Non-target
	0xFF37 3800- 0xFF37 37FF	Switch Area 0	Valid	2KB	Non-target
	0xFF37 4000- 0xFF37 47FF	Switch Area 1	Invalid	2KB	Target
	0xFF37 4800- 0xFF37 4FFF	TAG Area	Valid	2KB	Target
Hardware Property Area1	0xFF34 0000- 0xFF34 0FFF	Block Protection Area for FPSYS1 Area0	Valid	2KB	Non-target
	0xFF34 0800- 0xFF34 0FFF	Block Protection Area for FPSYS1 Area1	Invalid	2KB	
	0xFF34 1000- 0xFF34 27FF	Erase Counter Area for User Area 2, User Boot Area0	Valid	2KB×3	
	0xFF34 2800- 0xFF34 3FFF	Erase Counter Area for User Area 3, User Boot Area1	Valid	2KB×3	

6.1.2 Operation overview

Figure 6-1 shows overview of software update operation.

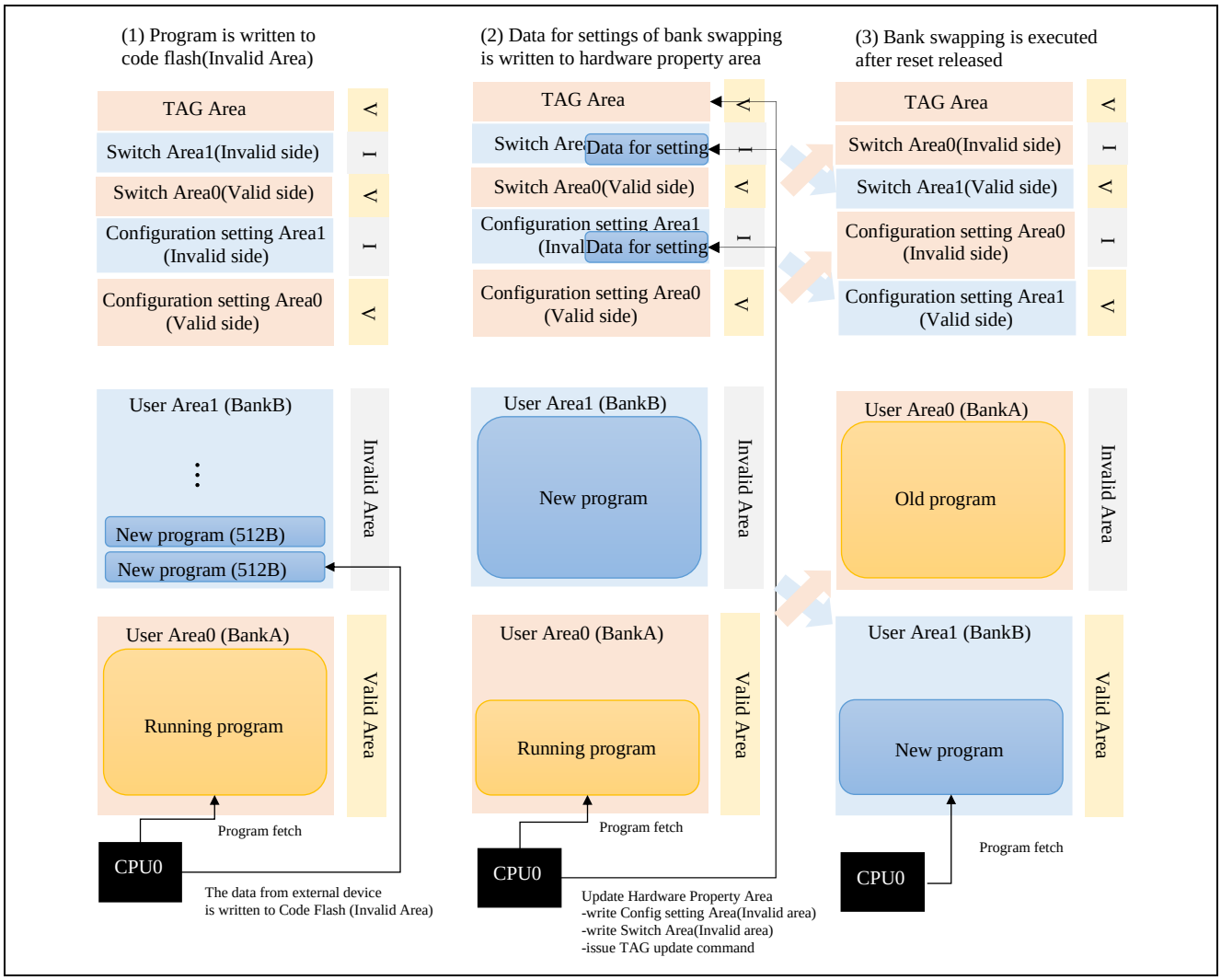


Figure 6-1 Overview of software update operation

6.1.3 Overall sequence

Figure 6-2, Figure 6-3 and Figure 6-4 shows overall sequence.

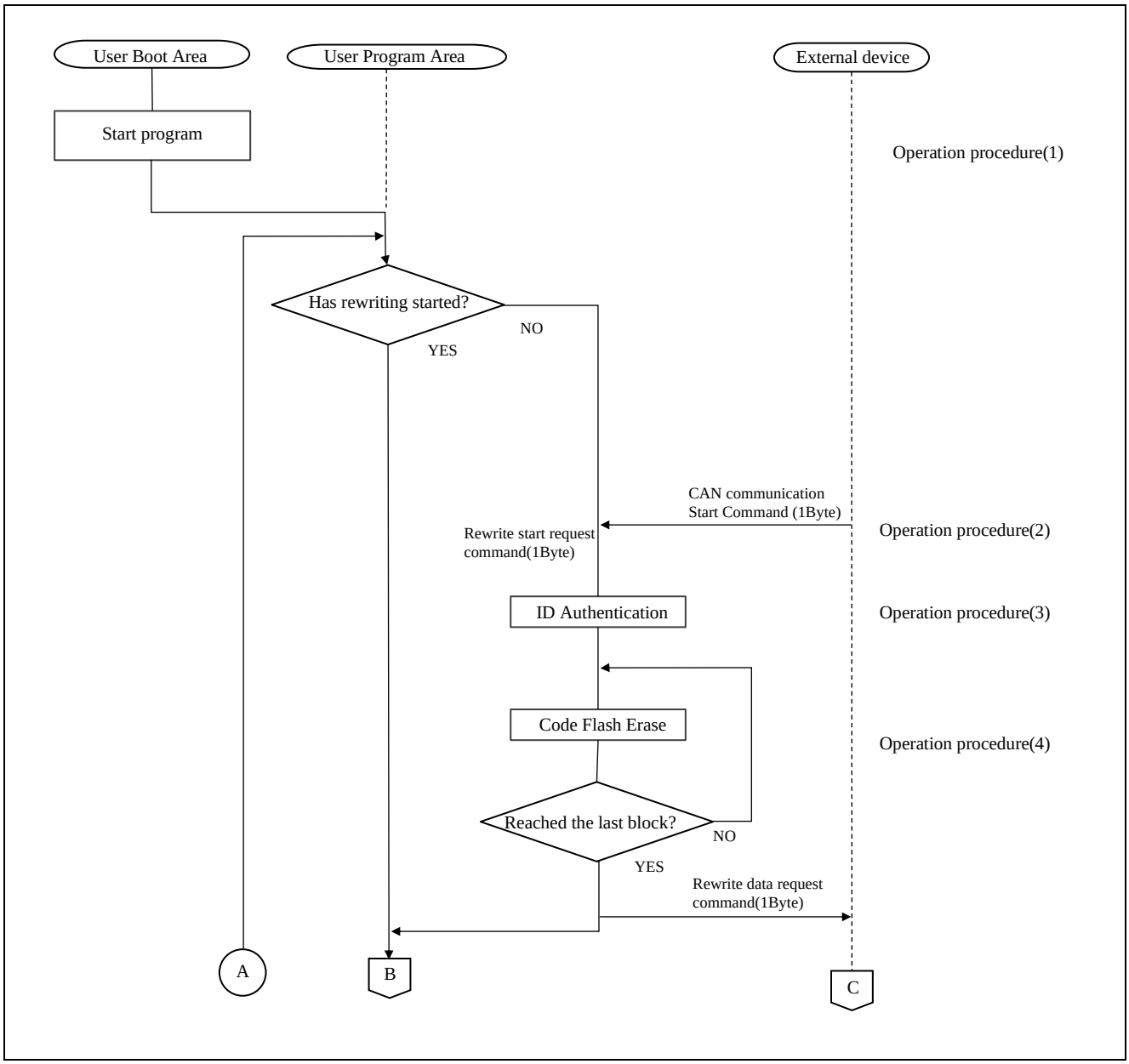


Figure 6-2 Overall sequence (1)

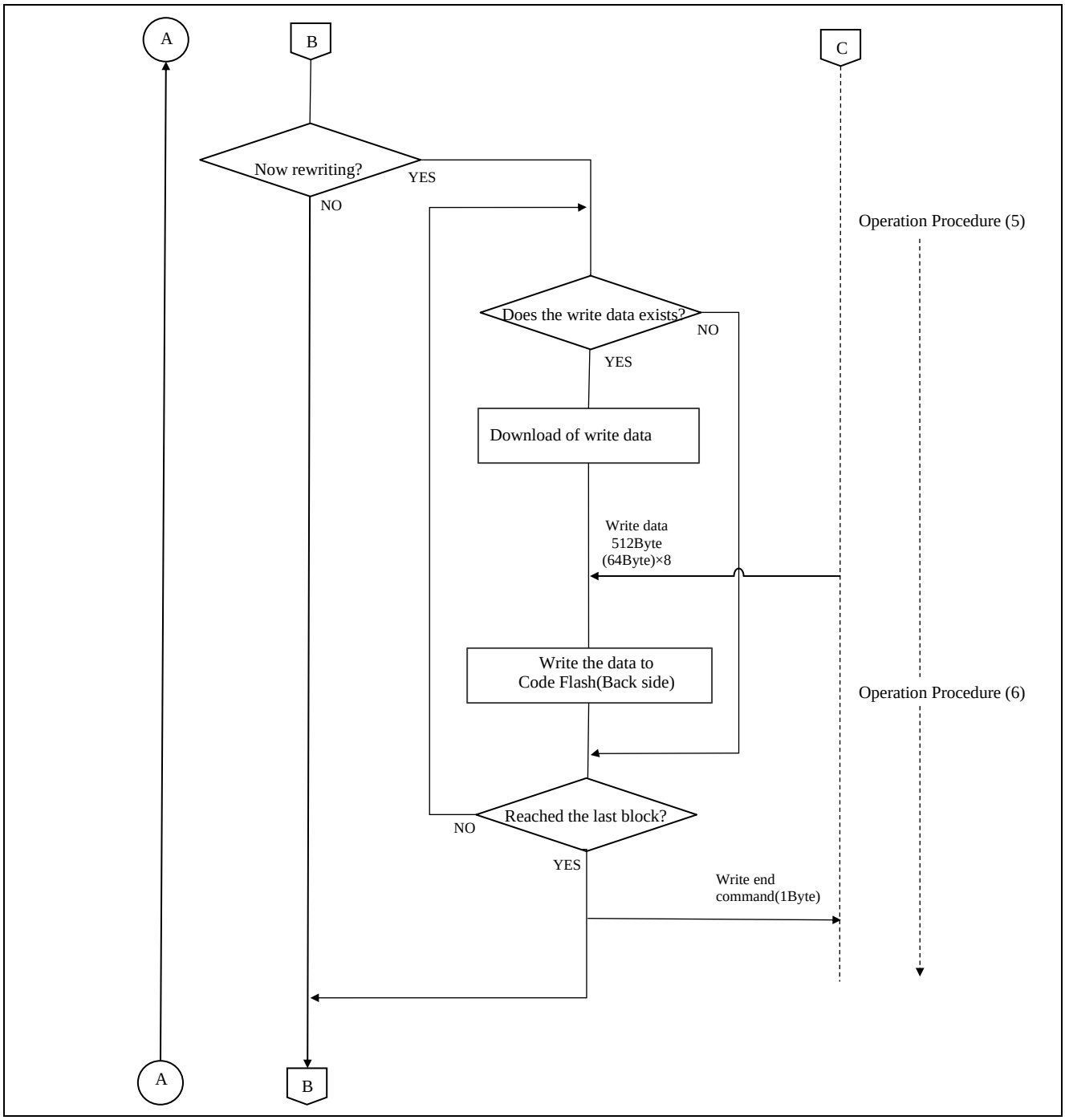


Figure 6-3 Overall sequence (2)

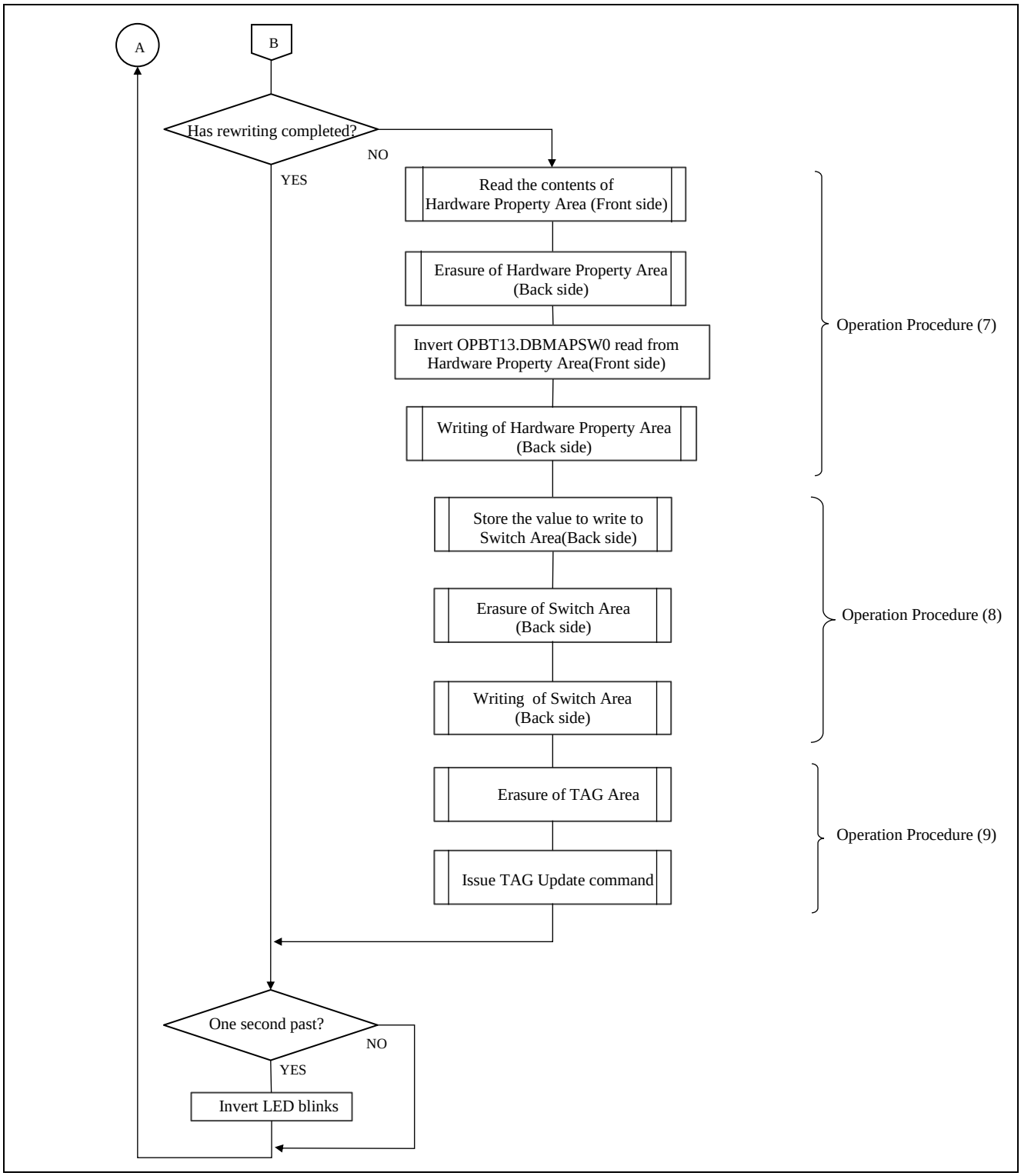


Figure 6-4 Overall sequence (3)

6.1.4 Functions used

- CANFD Interface(RS-CANFD)
- FACI
- GCFU
- Port

6.1.5 Operation mode

Operation mode is user boot mode 0 in this operation. After reset release, instruction fetch of PE0 is carried out from the user boot area in user boot mode 0.

The operation mode can be selected with mode pins and option byte. Option byte can be set by using Renesas Flash Programmer for RH850 family or configuration setting command of IDE(e.g. CS+, GHS Multi). Table 6-3 shows the operation mode selection.

Table 6-3 Operation mode selection

Pins			Option byte		Operation mode	Boot area
FLMD0	FLMD1	_TRST	STMSEL1	FLMD0		
0	X	0	0	0	User boot mode0	User boot Area0

6.1.6 Memory mapping mode

The memory mapping mode is double map mode in this operation example. Table 6-4 shows the memory mapping selection.

Table 6-4 Memory mapping selection

Option byte setting		Mapping mode
MAPMODE1	MAPMODE 0	
0	1	Double Map Mode

6.1.7 Configuration for RFD

Macro definitions for RFD in this operation example are shown in Table 6-5.

Table 6-5 Macro definition

Macro definition	Setting value	description
R_RFD_VALUE_FORCED_STOP_TIMEOUT	Refer to sample program	Time out value of forced stop command
R_RFD_ERASURE_SUSPENDED_MODE	R_RFD_ERASURE_PRIORITY _MODE	Suspension-priority mode
R_RFD_REG_FAEINT_CFAEIE R_RFD_REG_FAEINT_CMDLKIE R_RFD_REG_FAEINT_DFAEIE R_RFD_REG_FAEINT_ECRCTIE	R_RFD_DISABLE	Does not generate the FLERR interrupt
R_RFD_CONTROL_TARGET_DATAFLASH	R_RFD_ENABLE	Data flash memory is control target by RFD28F
R_RFD_CONTROL_TARGET_CODEFLASH	R_RFD_ENABLE	Code flash memory is control target by RFD28F
R_RFD_MAPMODE	R_RFD_DOUBLE	Double map mode
R_RFD_BLOCK_PROTECTION_AREA1	R_RFD_ENABLE	Block Protection Area 1 exists
R_RFD_FPMON_CHECK	R_RFD_ENABLE	Confirmation for FPMON register is valid

6.2 Operation procedure

Operation procedure (1) to (9) correspond to ”6.1.3 Overall sequence”.

6.2.1 (1) Start Program

After reset start, the start program on the user boot area is executed. Figure 6-5 shows the start program location.

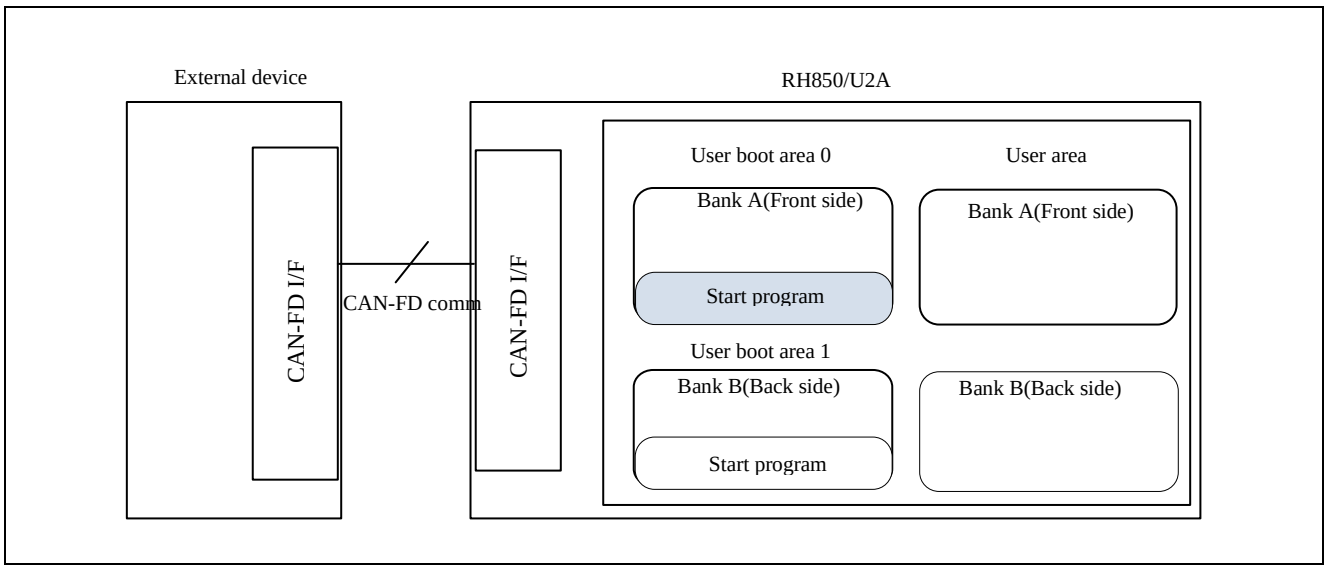


Figure 6-5 Startup program location

Function description

Table 6-6 “main_pm0() function”

Function	Argument	Description
main_pm0()	N/A	Jumps to the user program.

Figure 6-6 shows flowchart of main_pm0.

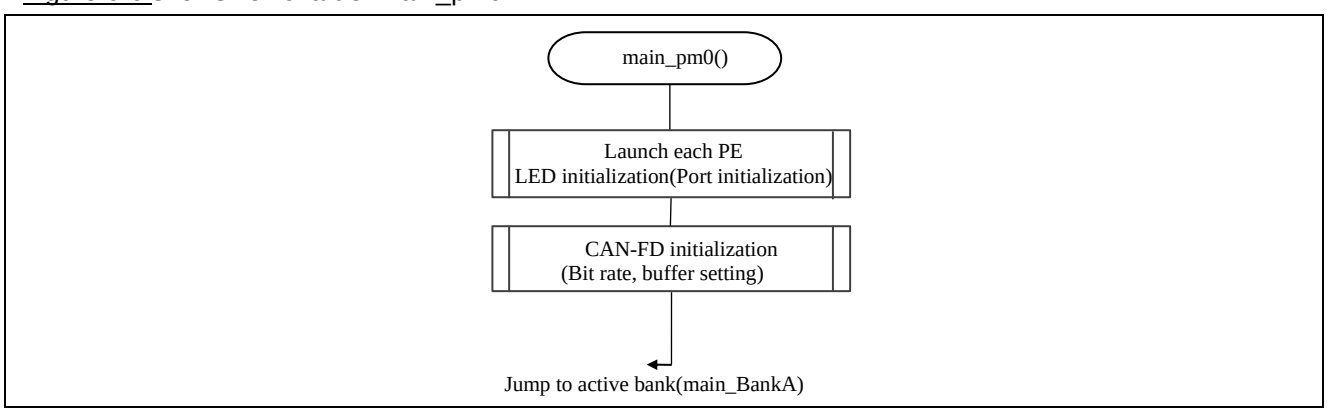


Figure 6-6 Flowchart of main_pm0

“Operation procedure (1)”

6.2.2 (2) Code flash reprogramming with background operation (BGO) function

Back side (invalid area) of code flash is target of rewriting. Figure 6-7 shows program location of each Bank when BankA is valid.

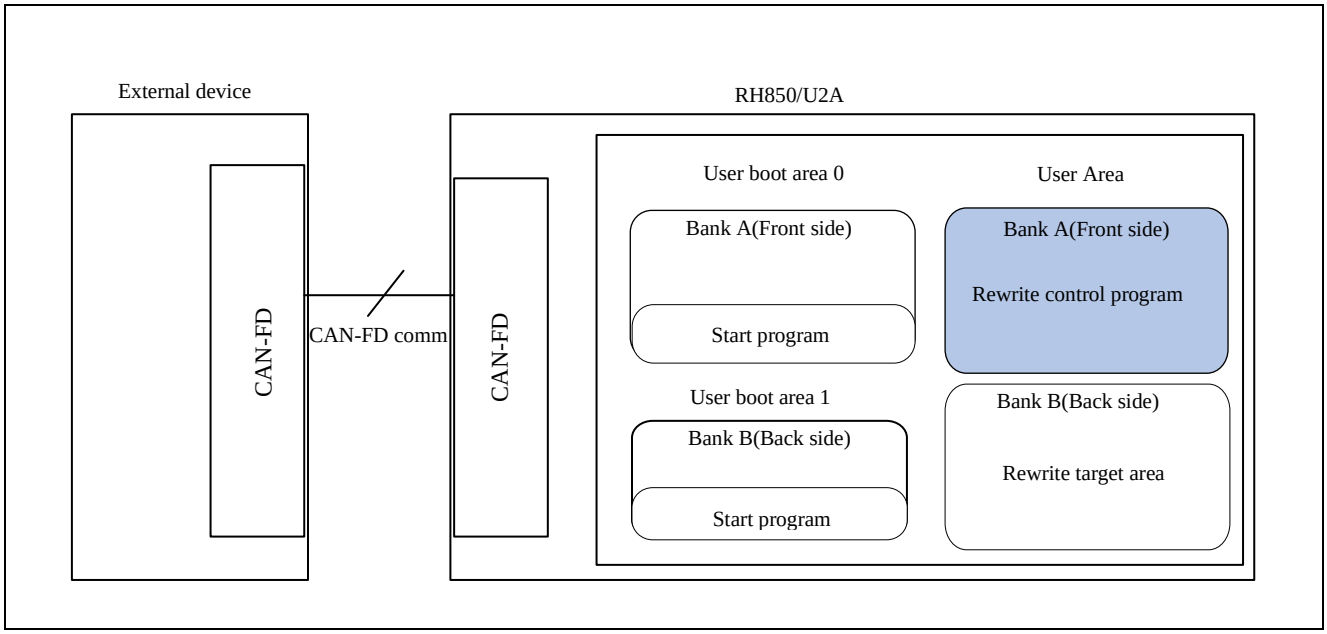


Figure 6-7 Program location of each Bank

Function description

Table 6-7 “main_BankA() function”

Function	Argument	Description
main_BankA()	N/A	The “Rewrite control program” is executed after the rewrite start command is received while constantly blinks LED.

Figure 6-8 shows flowchart of “main_BankA()” function.

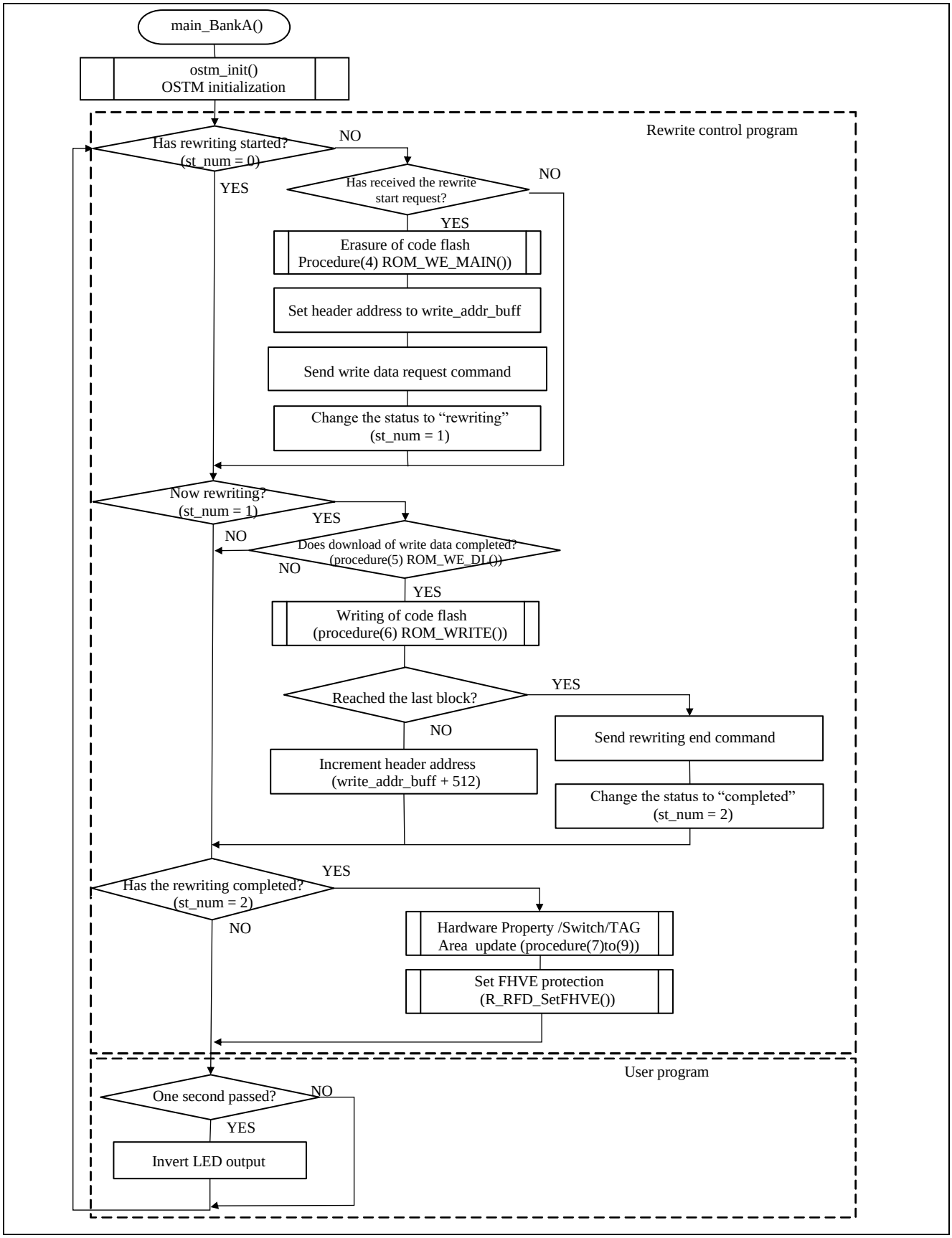


Figure 6-8 Flowchart of “main_BankA()” function

“Operation procedure (2)”

Function description

Table 6-8 “ROM_WE_MAIN()” function

Function	Argument	Description
ROM_WE_MAIN()	N/A	Execute some function such as ID authentication, erasure of code flash, download of writing data and programing of code flash.

Figure 6-9 shows flowchart of “ROM_WE_MAIN()” function.

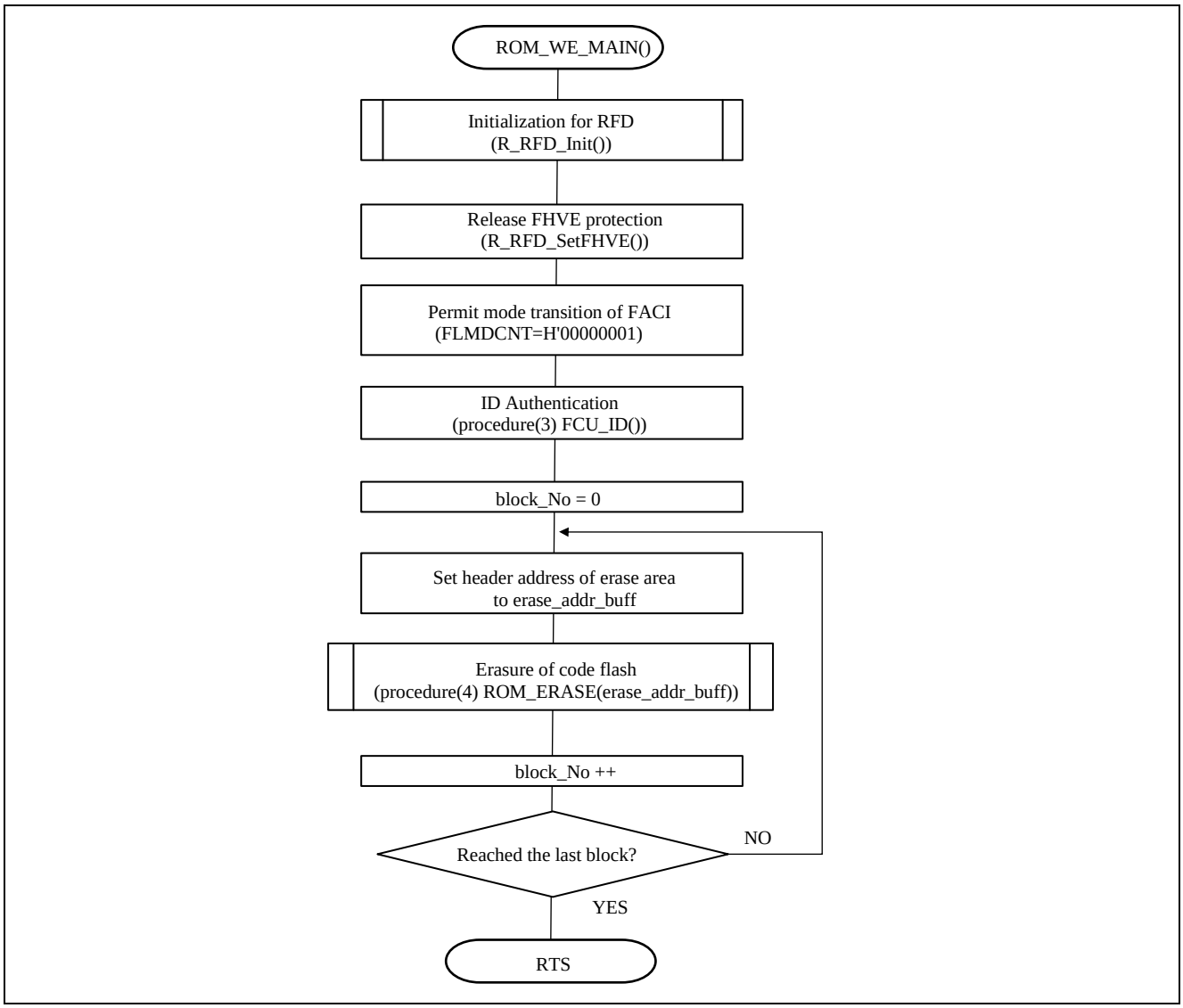


Figure 6-9 Flowchart of ”ROM_WE_MAIN()” function

Included operation procedure (3) and (4)

6.2.3 (3) ID authentication

ID authentication in “Rewriting control program” is executed by comparing the preset 256bits ID and customer ID. In this example of operation, the IDs are set “0” for the first byte and “F” for other. The customer ID can be set by using Renesas Flash Programmer for RH850 family or configuration setting command of IDE(e.g. CS+, GHS Multi).

Function description

Table 6-9 “FCU_ID() function”

Function	Argument		Description
FCU_ID()	N/A		Execute ID authentication.
API	Argument		Description
R_RFD_IDAuth	T_en_IDType	R_RFD_ID_CUSTIDA/B/C	Execute ID authentication.
	T_u1*	Pointer for ID data	

Figure 6-10 shows flowchart of “FCU_ID()” function.

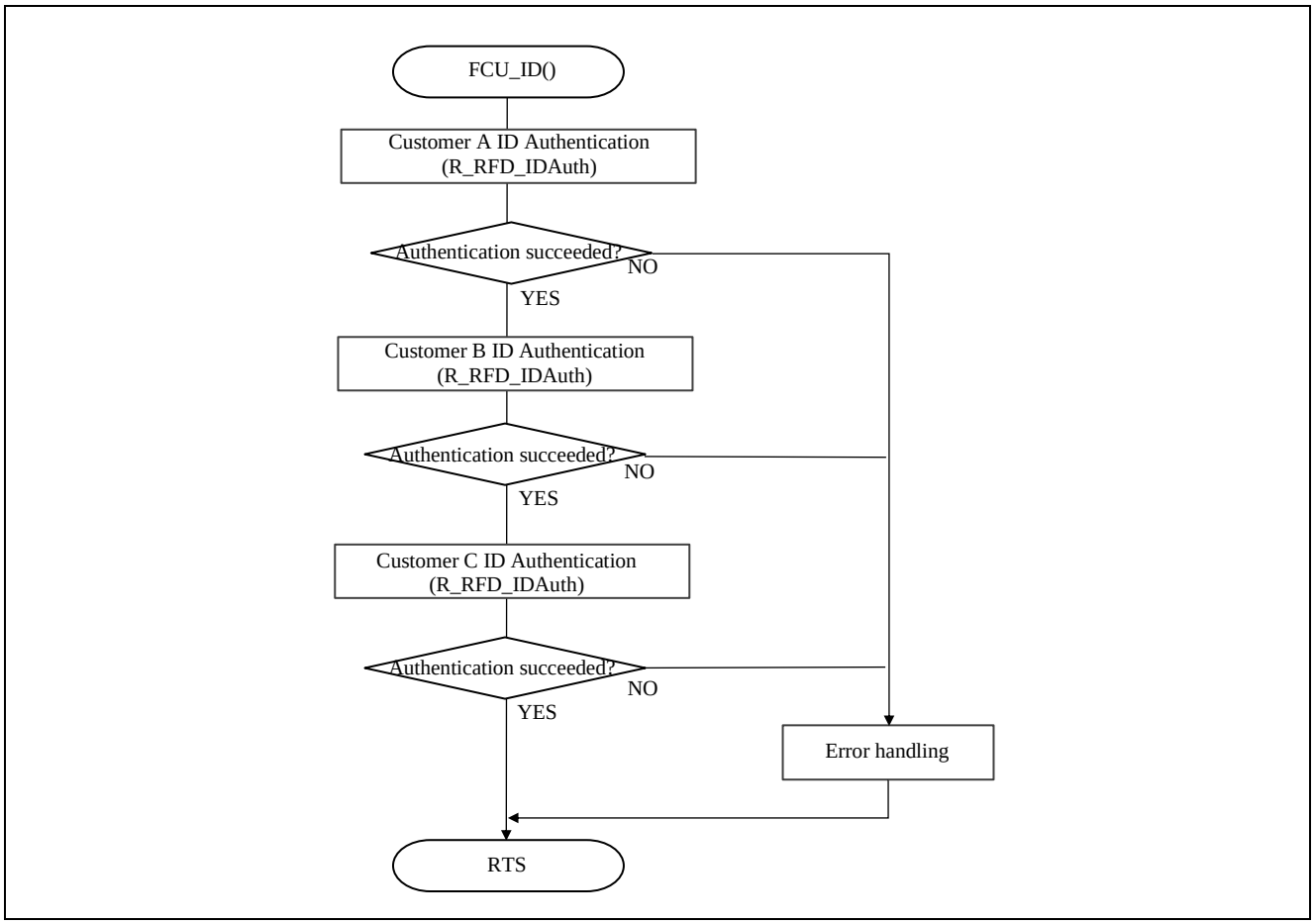


Figure 6-10 Flowchart of “FCU_ID()” function

“Operation procedure (3)”

6.2.4 (4) Code Flash Erasure

ROM_ERASE() function in “Rewrite control program” is executed after ID authentication was succeeded.

Issue the “Erase Command” of FACI to erase the contents of code flash memory area specified by the parameter.

Function description

Table 6-10 “ROM_ERASE()” function

Function	Argument		Description
ROM_ERASE()	erase_addr	Header address of erasure area	Erases the specified area of Code Flash
API	Argument		Description
R_RFD_ShiftToPEMode	T_u2_FACI	R_RFD_FACI0	Shift the target memory mode in the target FACI to P/E mode.
	T_en_FACIMode	R_RFD_MODE_CFPE	
R_RFD_ShiftToReadMode	T_u2_FACI	R_RFD_FACI0	Shift the target memory mode in the target FACI to read mode.
R_RFD_GetFaciStatus	T_u2_FACI	R_RFD_FACI0	Check whether target FACI is command lock or error occurred.
R_RFD_GetFaciSequenceRead	T_u2_FACI	R_RFD_FACI0	Check whether the target FACI is in ready state.
R_RFD_StatusClear	T_u2_FACI	R_RFD_FACI0	Clear FACI status
R_RFD_ForcedStopAndErrorClear	T_u2_FACI	R_RFD_FACI0	Issue forced stop commend
R_RFD_EraseCFRequest	T_u4_RfdAddress	Header address of erasure area	Erase the contents of the code flash memory.

Figure 6-11 shows the flowchart of “ROM_ERASE()” function.

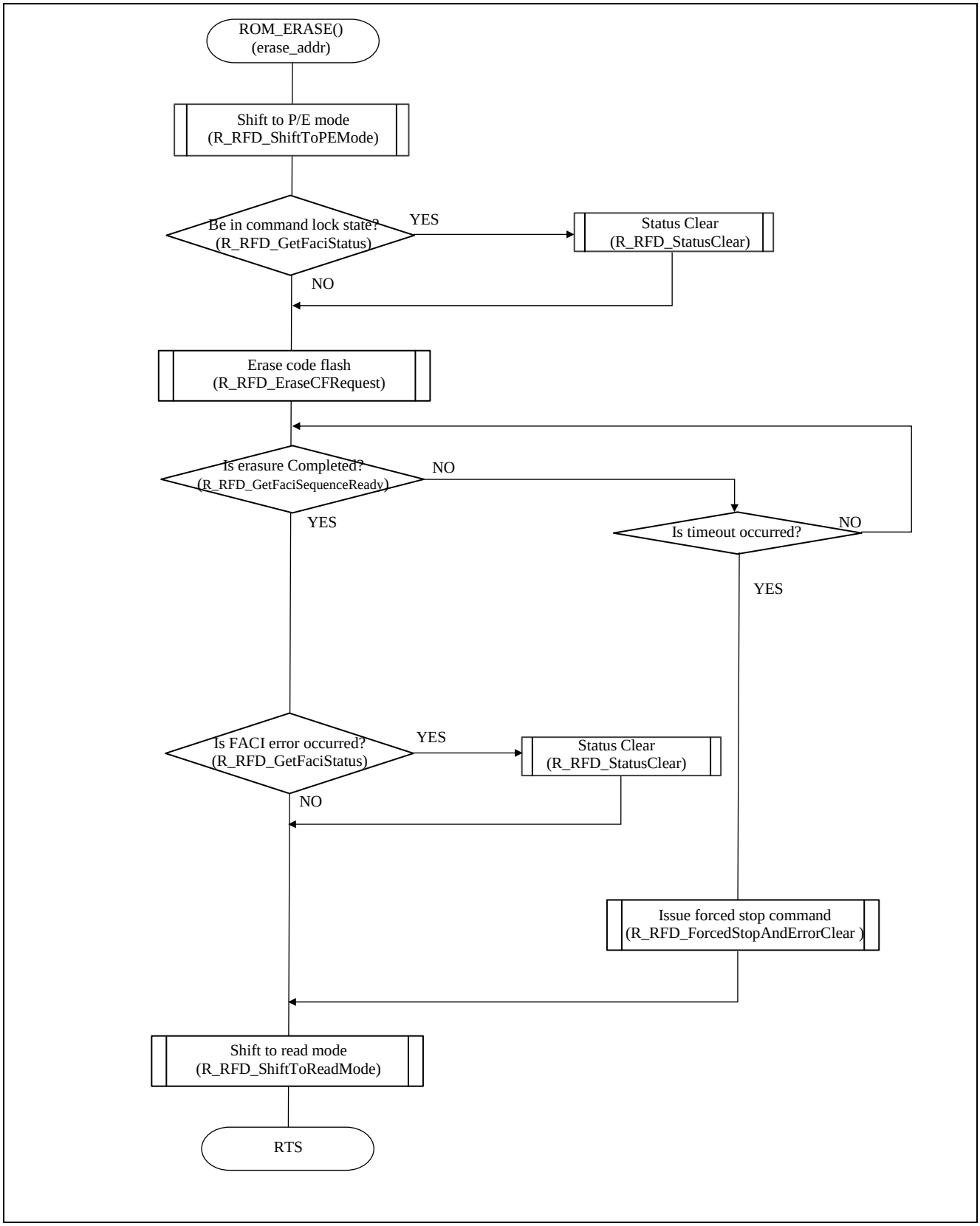


Figure 6-11 Flowchart of “ROM_ERASE()” function

Operation Procedure (4)

6.2.5 (5) Download of write data

Execute the “ROM_WE_DL” function and transmit the write data request command. External device transmits the 512byte data to U2A when receive the write data request command. The data from external device is saved on RAM in this function.

Function description

Table 6-11 “ROM_WE_DL()”function

Function	Argument	Description
ROM_WE_DL()	N/A	Download the write data from external device

Figure 6-12 shows the flowchart of “ROM_WE_DL()” function.

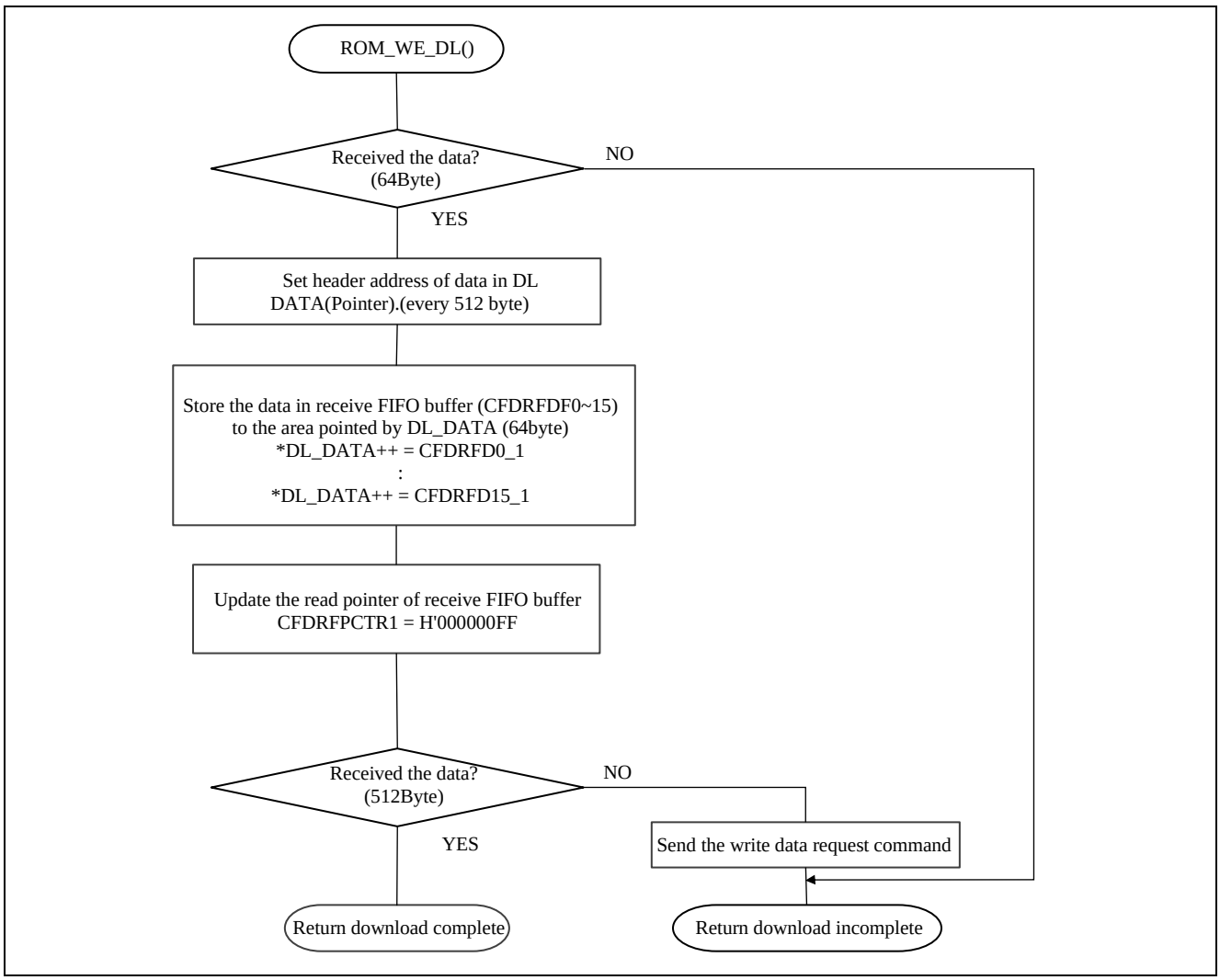


Figure 6-12 Flowchart of “ROM_WE_DL()” function

“Operation procedure (5)”

6.2.6 (6) Code flash programming

The data that received from external device is programmed to code flash by using “ROM_Write()” function. Issue the “Program Command” of FACI to write to code flash area specified by the parameter. The programming is finished when the limit size (64 Kbyte) is reached.

Function description

Table 6-12 “ROM_WRITE()” function

Function	Argument		Description
ROM_WRITE()	write_addr	Header address for writing area	Write the data to the code flash memory. (512Byte unit)
API	Argument		Description
R_RFD_ShiftToPEMode	T_u2_FACI	R_RFD_FACI0	Shift the target memory mode in the target FACI to P/E mode.
	T_en_FACIMode	R_RFD_MODE_CFPE	
R_RFD_ShiftToReadMode	T_u2_FACI	R_RFD_FACI0	Shift the target memory mode in the target FACI to read mode.
R_RFD_GetFaciStatus	T_u2_FACI	R_RFD_FACI0	Check whether target FACI is command lock or error occurred.
R_RFD_GetFaciSequenceRead	T_u2_FACI	R_RFD_FACI0	Check whether the target FACI is in ready state.
R_RFD_StatusClear	T_u2_FACI	R_RFD_FACI0	Clear FACI status
R_RFD_ForcedStopAndErrorClear	T_u2_FACI	R_RFD_FACI0	Issue forced stop commend
R_RFD_WriteCFRequest	T_u4_RfdAddress	Header address of erasure area	Writes the contents of the code flash memory.

Figure 6-13 shows the flowchart of “ROM_WRITE()” function.

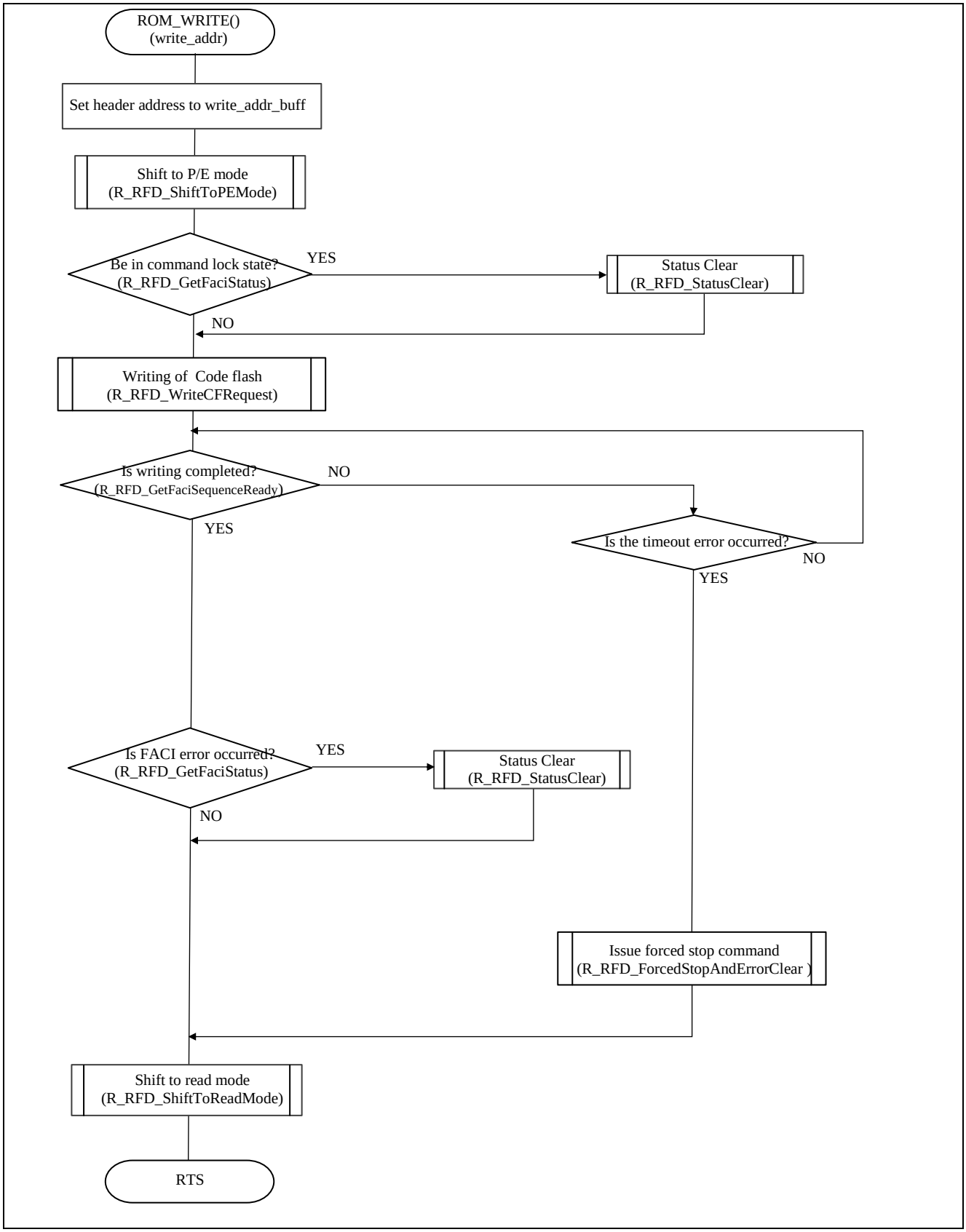


Figure 6-13 Flowchart of “ROM_WRITE()” function

“Operation procedure (6)”

6.2.7 (7) Update of hardware property area

Rewriting of hardware property area, switch area and TAG update are executed by “Sample_PropertyAreaControl” function. Note that the Hardware Property handled by this function is only the “Configuration Setting Area” in this operation. When you want to operate the “Security Setting Area”,” Block Protection Area0”, and “Block Protection Area1”, the processing procedure is the same and only the operation target changes. Therefore, please process according to each.

Table 6-13 “Sample_PropertyAreaControl ()” function

Function	Argument		Description
Sample_PropertyAreaControl	N/A	-	Execution of bank swapping
API	Argument		Description
R_RFD_ShiftToPEMode	T_u2_FACI	R_RFD_FACI0	Shift to P/E mode
	T_en_FACIMode	R_RFD_MODE_DFPE	
R_RFD_ShiftToReadMode	T_u2_FACI	R_RFD_FACI0	Shift to read mode
R_RFD_GetFaciStatus	T_u2_FACI	R_RFD_FACI0	Check whether target FACI is command lock or error occurred.
R_RFD_GetFaciSequenceReady	T_u2_FACI	R_RFD_FACI0	Check whether the target FACI is in ready state.
R_RFD_StatusClear	T_u2_FACI	R_RFD_FACI0	Clear FACI status
R_RFD_ErasePropertyRequest	T_u4_RfdAddress	Start address for erasure of one area in the hardware property area.	Erases the contents of the Configuration Setting Area within Hardware Property Area.
R_RFD_WritePropertyRequest	T_u4_RfdAddress	Start address for writing area	Writes the contents of the Configuration Setting Area within Hardware Property Area.
	T_pu4_RfdBuffer	Buffer address where write data is stored	
R_RFD_EraseSwitchRequest	N/A	-	Erases the contents of the Switch Area within Hardware Property Area.
R_RFD_WriteSwitchRequest	T_pu4_RfdBuffer	Start address for writing area	Writes the contents of the Switch Area within Hardware Property Area.
R_RFD_EraseTagRequest	N/A	-	Erases the contents of TAG area.
R_RFD_UpdateTagRequest	N/A	-	Updates the contents of TAG area.
R_RFD_ForcedStopAndErrorClear	T_u2_FACI	R_RFD_FACI0	Issue forced stop commend

(1) Read from Configuration Setting Area (Front side)

Read the all contents from Configuration setting area (Front side) and create the data for writing to Configuration setting area (Back side).

Figure 6-14 shows flowchart of reading the contents of Configuration Setting Area (Front side) part of “Sample_PropertyAreaControl()” function.

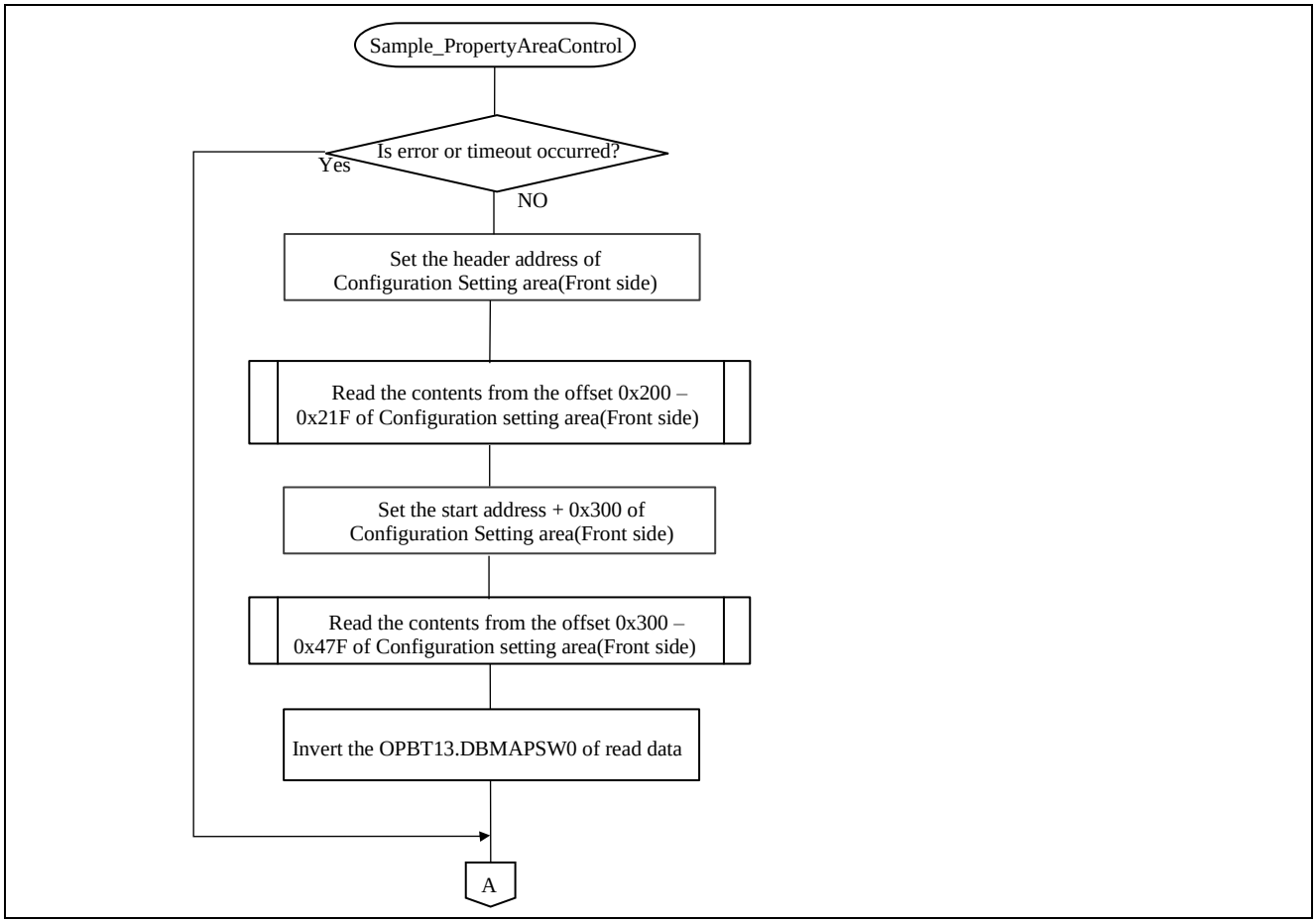


Figure 6-14 Flowchart of reading the contents of Configuration Setting Area (Front side)

(2) Erasure of Configuration Setting Area (Back side)

Erase the contents of Configuration setting area (Back side).

Figure 6-15 shows flowchart of erasure the contents of Configuration Setting Area (Back side) part of ”Sample_PropertyAreaControl()” function.

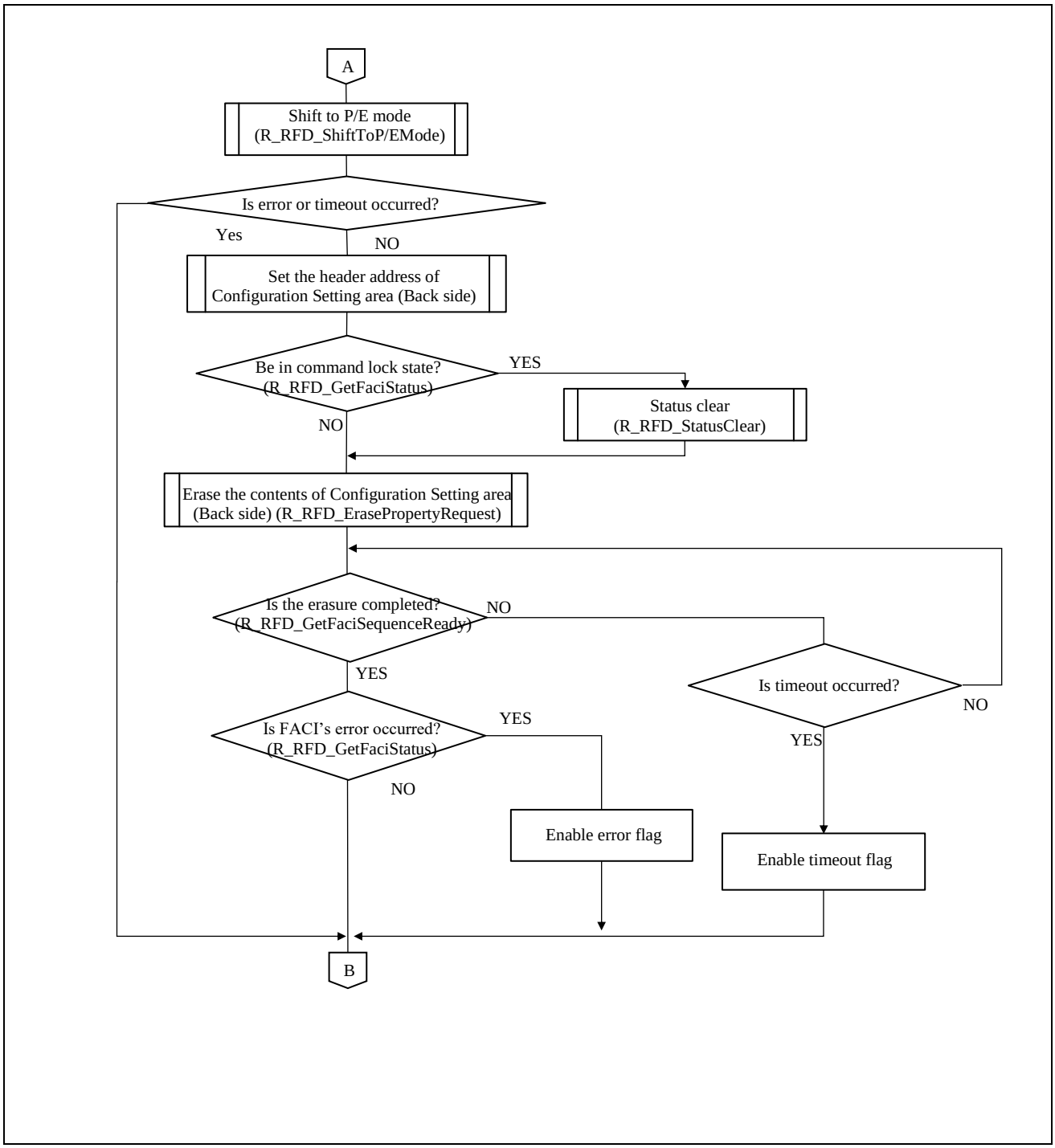


Figure 6-15 Flowchart of erasure the contents of Configuration Setting Area (Back side)

(3) Writing of Configuration Setting Area (Back side)
Write the data to the Configuration Setting Area(Back side).

Figure 6-16, Figure 6-17 shows flowchart of writing the data to Configuration Setting Area (Back side) part of ”Sample_PropertyAreaControl()” function.

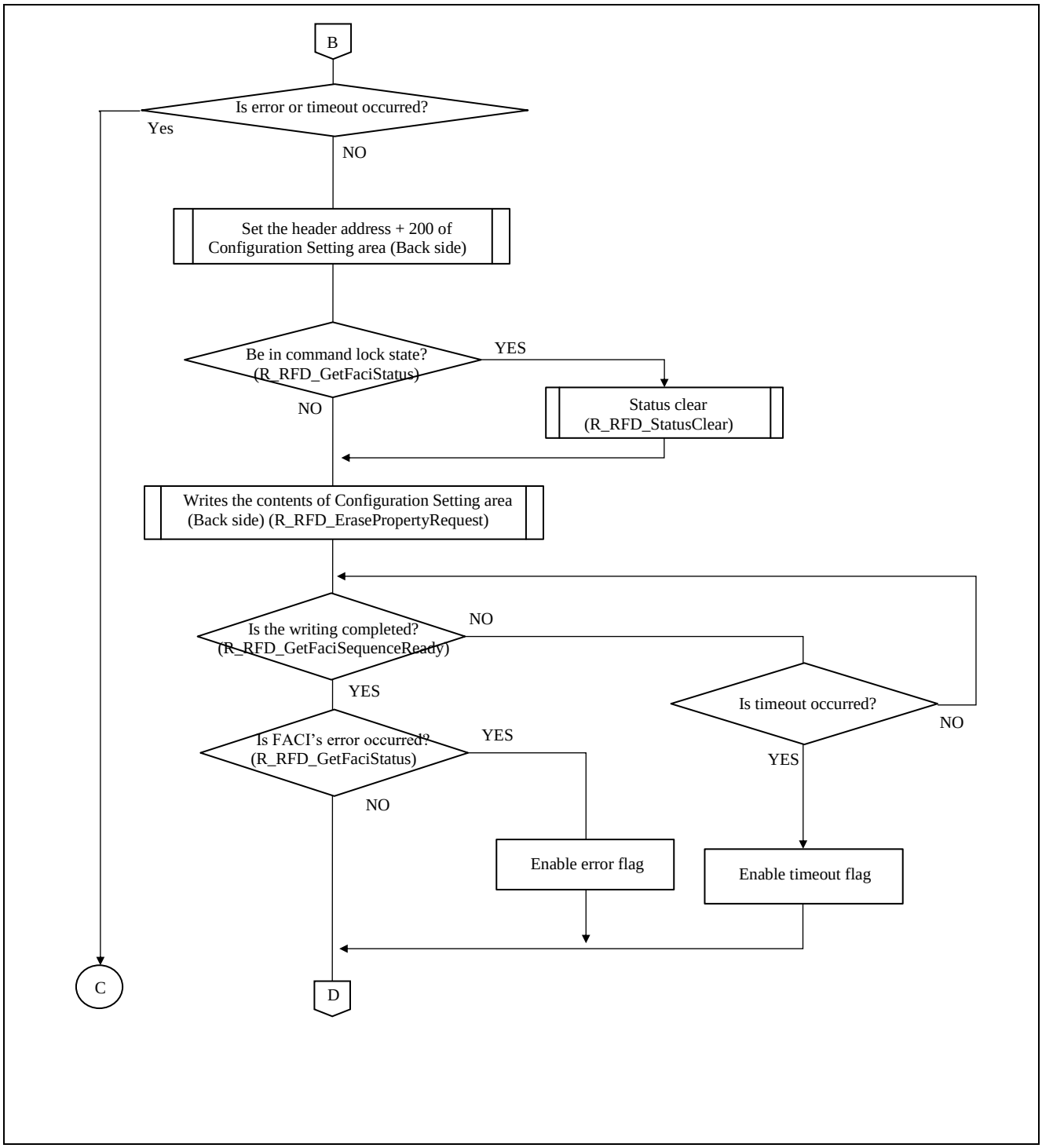


Figure 6-16 Flowchart of writing the data to Configuration Setting Area (Back side) (1)

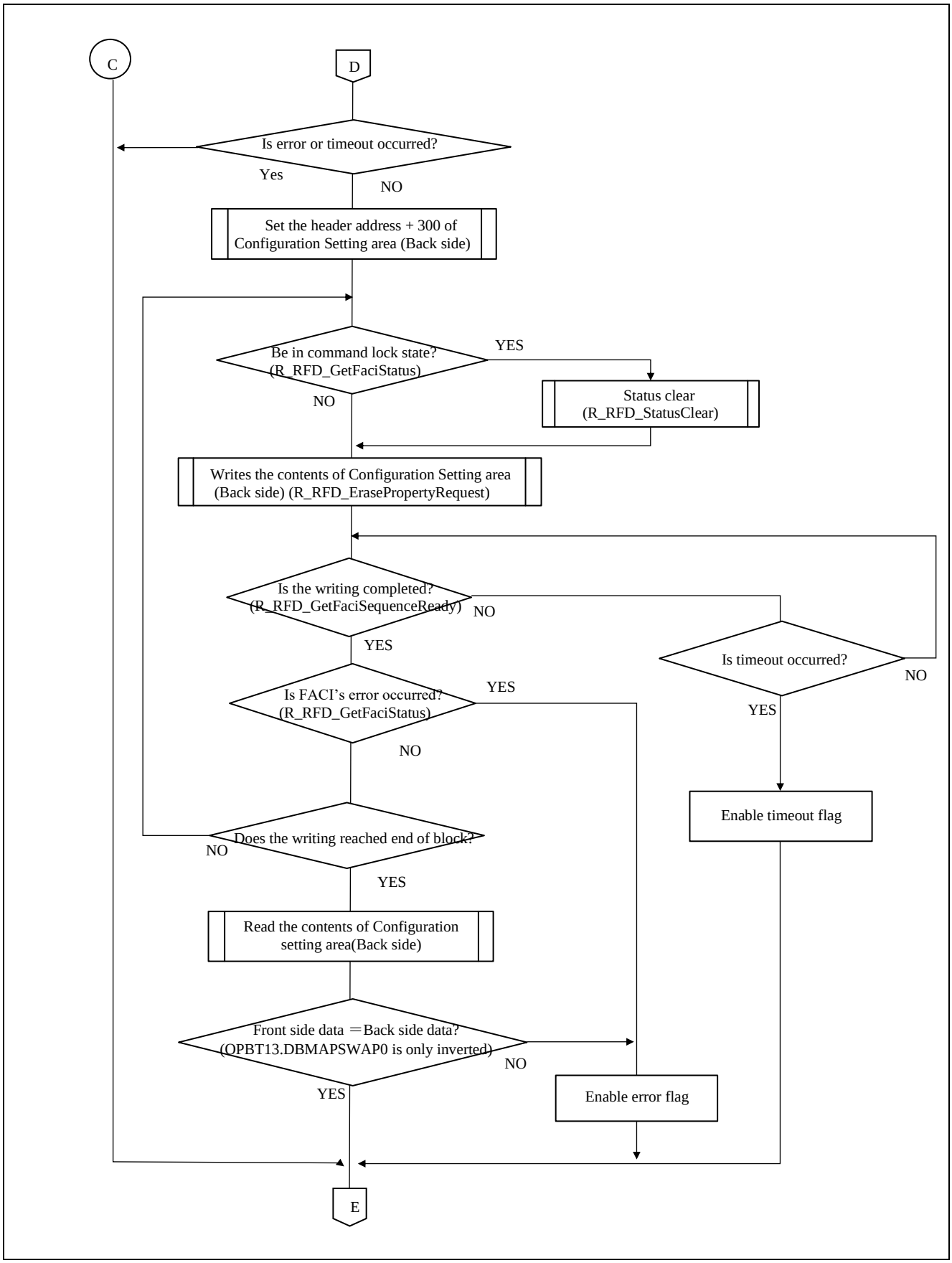


Figure 6-17 Flowchart of writing the data to Configuration Setting Area (Back side) (2)

6.2.8 (8) Update of Switch Area

(1) Store the setting value for Switch Area

Read the values of CFGVA, SECVA, BPVA0 and BPVA1 of the FSWASTAT register to obtain the area indicating the current status. In this operation, only the configuration area is swapped. When you want to operate the “Security Setting Area”,” Block Protection Area0”, and “Block Protection Area1”, the processing procedure is the same and only the operation target changes. Therefore, please process according to each.

Figure 6-18 shows flowchart of setting the value for CVA, SVA, BVA0 and BVA1 in Switch Area (To Back side) part of ”Sample_PropertyAreaControl()” function.

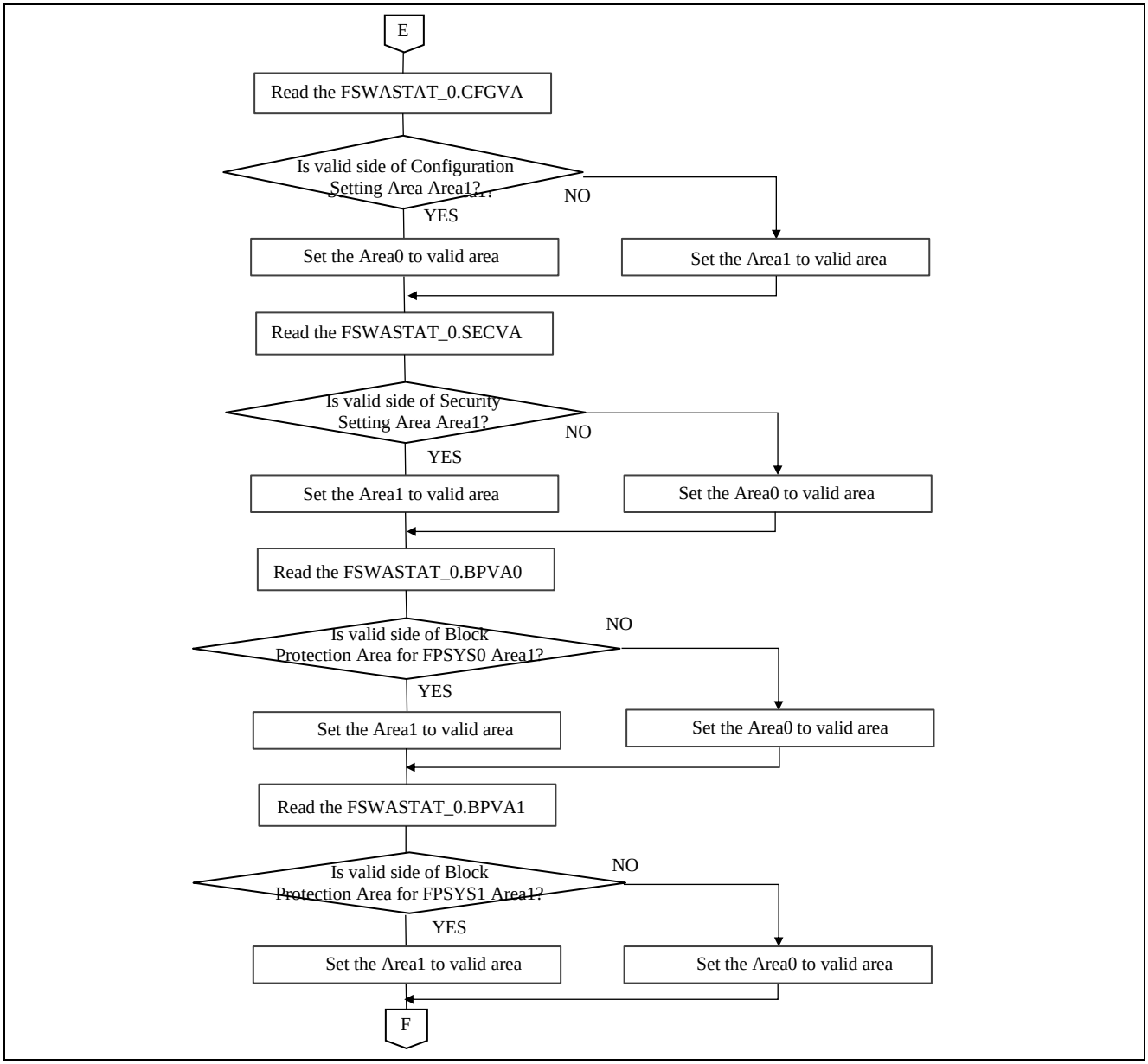


Figure 6-18 Flowchart of Setting the value for Switch Area (To Back side)

(2) Erasure of Switch Area(Back side)

Erase the contents of Switch area(Back side).

Figure 6-19 shows flowchart of erasure the contents of Switch Area (Back side) part of ”Sample_PropertyAreaControl()” function.

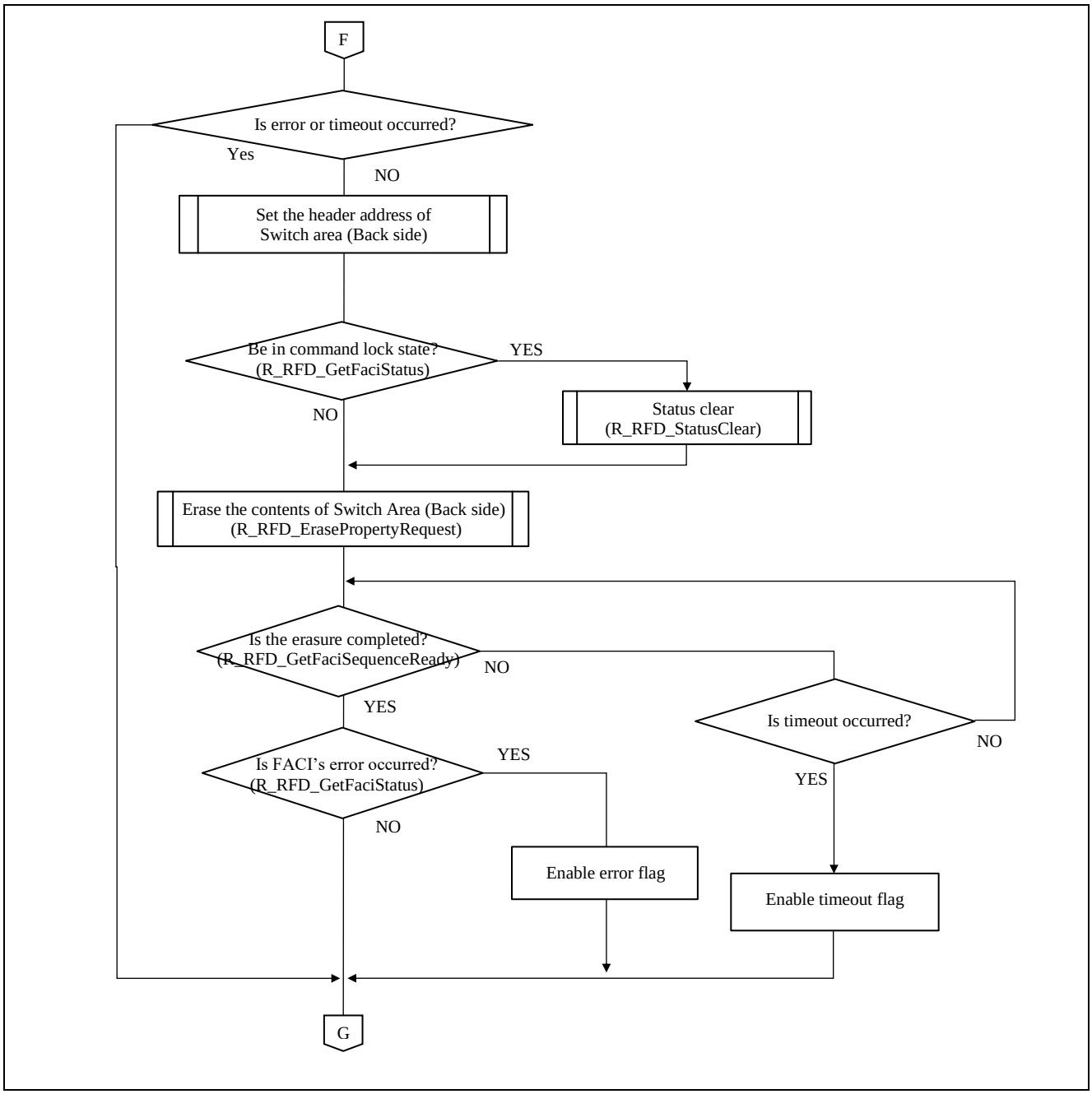


Figure 6-19 Flowchart of erasure the contents of Switch Area (Back side)

(3) Writing of Switch Area(Back side)

Write the data to Switch area (Back side).

Figure 6-20 shows flowchart of writing the contents of Switch Area (Back side) part of" Sample_PropertyAreaControl()" function.

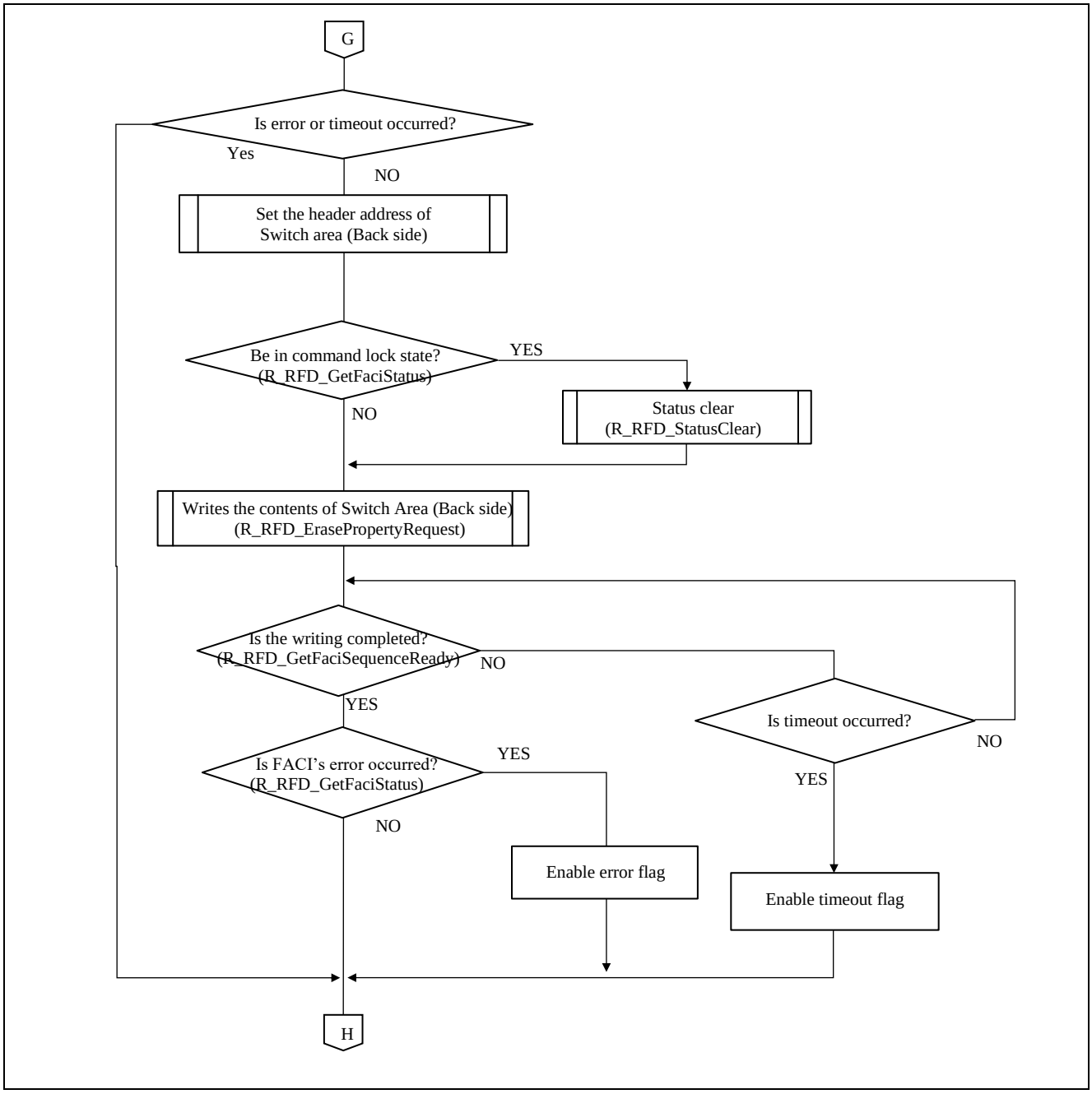


Figure 6-20 Flowchart of writing the contents of Switch Area (Back side)

6.2.9 (9) Update of TAG Area

(1) Erasure of TAG Area

Erase the contents of the Tag Area. Figure 6-21 shows flowchart of erasure the contents of TAG Area part of ”Sample_PropertyAreaControl()” function.

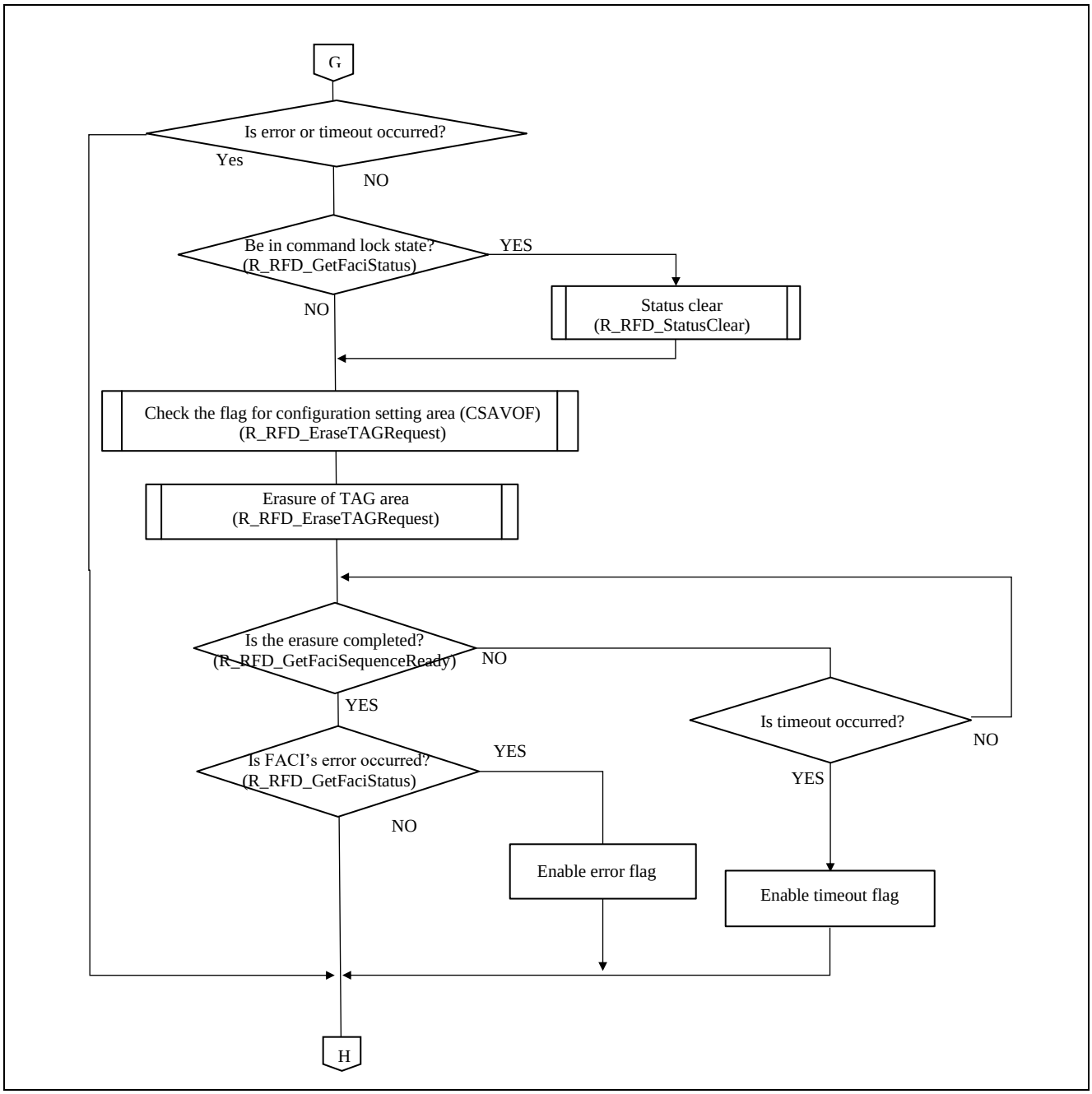


Figure 6-21 Flowchart of erasure the contents of TAG Area

(2) TAG Update

By updating the Tag information using FACI's "Tag Update Command", the Switch Area on the currently Front side (valid side) is set to the Back side (invalid side), and the Switch Area on the currently Back side (invalid side) is set to the Front side (valid side). Figure 6-22 shows flowchart of update the contents of Tag Area part of “Sample_PropertyAreaControl()”.

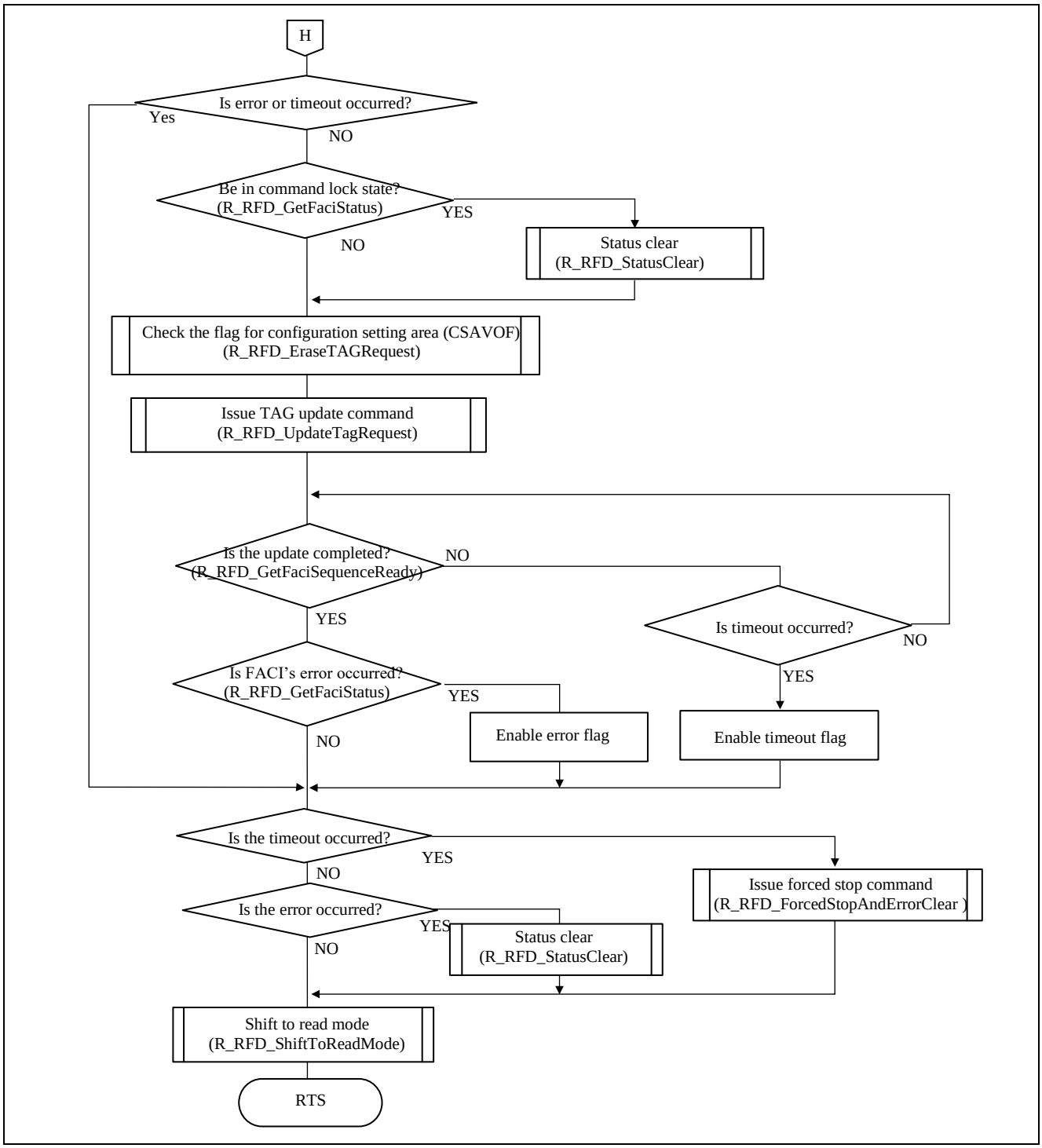


Figure 6-22 Flowchart of update the contents of Tag Area

6.3 Memory allocation

Table 6-14 shows memory allocation in double map mode.

Table 6-14 Memory allocation

Area	Address	Section	Access
CodeFlash User Area0/1 (Bank A/B)	H'0000_0000	.BankA.text	PE0
		.BankA.const	
		.FlashControl.text	
		.R_RFD_RODATA_EXTRA.const	
		.R_RFD_CODE_COMMON.text	
		.R_RFD_CODE_COMMON_RAM_NO_BGO.text	
		.R_RFD_CODE_USEROWN_COMMON.text	
		.R_RFD_CODE_DF.text	
		.R_RFD_CODE_CF.text	
		.R_RFD_CODE_CF_RAM_NO_BGO.text	
		.R_RFD_CODE_EXTRA.text	
		.R_RFD_RODATA_VERSION_DF.const	
		.R_RFD_RODATA_VERSION_CF.const	
		.R_RFD_RODATA_VERSION_COMMON.const	
	H'0000_8000	RESET_PE1	PE1
		EIINTTBL_PE1	
		.const	
		.INIT_DSEC.const	
		.INIT_BSEC.const	
		.text.cmn	
		.text	
		.data	
CodeFlash User Area2 (Bank C) ※Available only for U2A16	H'0040_0000	RESET_PE2	PE2
		EIINTTBL_PE2	
		.const	
		.INIT_DSEC.const	
		.INIT_BSEC.const	
		.text.cmn	
		.text	
		.data	
	H'0042_0000	RESET_PE3	PE3
		EIINTTBL_PE3	
		.const	
		.INIT_DSEC.const	
		.INIT_BSEC.const	
		.text.cmn	
		.text	
		.data	
CodeFlash User Boot Area0/1 (Bank A/B)	H'0800_0000	RESET_PE0	PE0
		EIINTTBL_PE0	
		.const	
		.INIT_DSEC.const	
		.INIT_BSEC.const	
		.text.cmn	
		.data	
		.text	
	H'0800_4000	.text	PE0～PE3
LRAM (PE3) ※Available only for U2A16	H'FD60_0000	.data.R	PE3
		.bss	
		.stack.bss	
LRAM (PE2) ※Available only for U2A16	H'FD80_0000	.data.R	PE2
		.bss	
		.stack.bss	
LRAM (PE1)	H'FDA0_0000	.data.R	PE1
		.bss	
		.stack.bss	
LRAM (PE0)	H'FDC0_0000	.bss	PE0
		.BankA.bss	
		.data.R	
		.stack.bss	
		.R_RFD_BSS.bss	

7. Programming/Erasure is interrupted intentionally or unintentionally during software update

If programming or erasure is interrupted intentionally or unintentionally during programing or erasure of code flash or hardware property area, the programing or erasure state of the flash memory with undefined data cannot be verified or checked. Erase the area again to ensure that the corresponding area is completely erased before using. Managing progress with user flag on data flash is recommended to detect intentional or unintentional interruption and to check the validity of the effective area (whether the software is running on intended Bank or not) during software update.

For the code flash area where programming or erasure was interrupted, the blank check function cannot judge whether the area is erased successfully or not. Therefore, managing progress with user flag on data flash is recommended. Erase the area again with the user flag to prove that corresponding area is completely before using.

For the hardware property area where programing or erasure was interrupted, valid area flags on switch area (CVA, SVA, BVAn) and valid area flash on TAG area (VAPC) sometimes do not reliable. Erase the area again with the user flag to prove that corresponding area is completely before using.

Figure 7-1 shows diagram of managing progress with user flag.

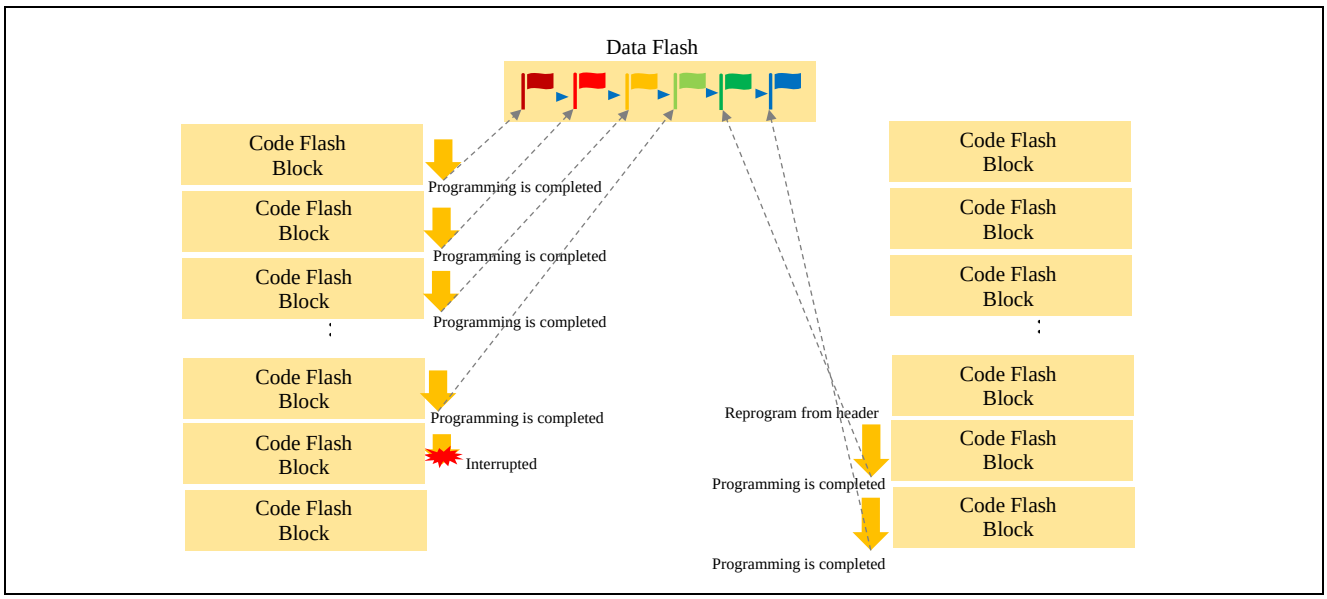


Figure 7-1 Diagram of managing progress with user flag

History

Rev.	Issued date	History	
		Page	Description
1.00	2021.08.05	All	Issued first version.
1.10	2022.04.01	All	RH850/U2A6 is added to the products covered by this document.

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.

—

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/.