User's Manual

RFD28F
Renesas Flash Driver

[For Free Package]

User's Manual

RENESAS MCU for 28nm process

RH850 Family
RH850/U2A16

Renesas Electronics
www.renesas.com

Ver.1.00E 2020.6

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 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.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; 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.

6. 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.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. 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.
8. 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.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall 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.
10. 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.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(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.4.0-1 November 2017)

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/.

How to Use This Manual

- Readers

  This manual is intended for users who develop systems operating data flash memory on the RH850/U2A16.

- Purpose

  This manual is intended for users to understand the functions of Renesas Flash Driver "RFD28F" manufactured by Renesas Electronics, described the construction listed below

- Construction

  1. Introduction
  2. System Construction
  3. RFD28F API
  4. Sample Program
  5. Note for RFD28F

- How to Read this Manual

  It is assumed that the readers of this manual have general knowledge in the fields of electrical engineering, logic circuits, microcontrollers, C language, and assemblers.

  To understand the hardware functions of the RH850/U2A16.

  - Refer to the User's Manual of each product.

- Conventions

  - Data significance

    Higher digits on the left and lower digits on the right

  - Note

    Footnote for item marked with Note in the text

  - Caution

    Information requiring particular attention

  - Remark

    Supplementary information

  - Numeric representation

    Decimal ... XXXX

    Hexadecimal ... 0xXXXX

  - Prefixes indicating power of 2 (address space and memory capacity)

    K (kilo) $2^{10} = 1024$

    M (mega) $2^{20} = 1024^2$

- Related Documents

  The related documents indicated in this publication may include preliminary versions. However, preliminary versions are not marked as such.

No	Document Title	Document Number
1	RH850/U2A-EVA Group User’s Manual: Hardware	R01UH0820EJ0100
2	RH850/U2A-EVA Flash Memory User’s Manual: Hardware	R01UH0832EJ0100

Index

Abbreviations

Abbreviations	Description
API	Application Program Interface
BGO	Background Operation. A function that can read the contents of the flash memory even during writing / erasing processing to the flash memory. There are conditions under which this function can be used, see the user's manual of the hardware.
ECC	Error Check and Correct
FACI	Flash Access Control Interface It is an interface for operating flash memory.
FCU	Flash Control Unit Basic control of flash memory reprogramming is executed by a part of the flash sequencer.
OTP	One Time Programming This function protects against code flash memory update. The code flash memory area where the OTP function is set cannot be erased again.
RAM	Random Access Memory Randomly accessible volatile memory. This memory holds the value to be changed during program execution.
RFD28F	Renesas Flash Driver for RV28F (Renesas MCU for 28nm process) The name of this product.
ROM	Read Only Memory Non-volatile memory. It is memory that cannot change contents.

Terminology

Terminology	Description
Code Flash Memory	Flash memory for storing application code and constant data
Data Flash Memory	Flash memory to store data
Flash Extra Area	Generic name of configuration setting area, security setting area, block protection setting area
P/E mode	Programming/Erasure mode. It is a mode that writing(programming) or erasing are performed. The opposite of this mode is the read mode. The read mode is the mode to read.
Self-Programming	Execute the program and rewrite the flash memory not using the external flash programming tool.
Serial Programming	To rewrite the flash memory using an external flash programming tool.

1 Introduction

1.1 Product Outline

The purpose of this document is to describe the information related to Renesas Flash Driver (RFD28F) for operating the data flash memory for RH850/U2A16.

1.2 Product Contents

RFD28F includes as follows

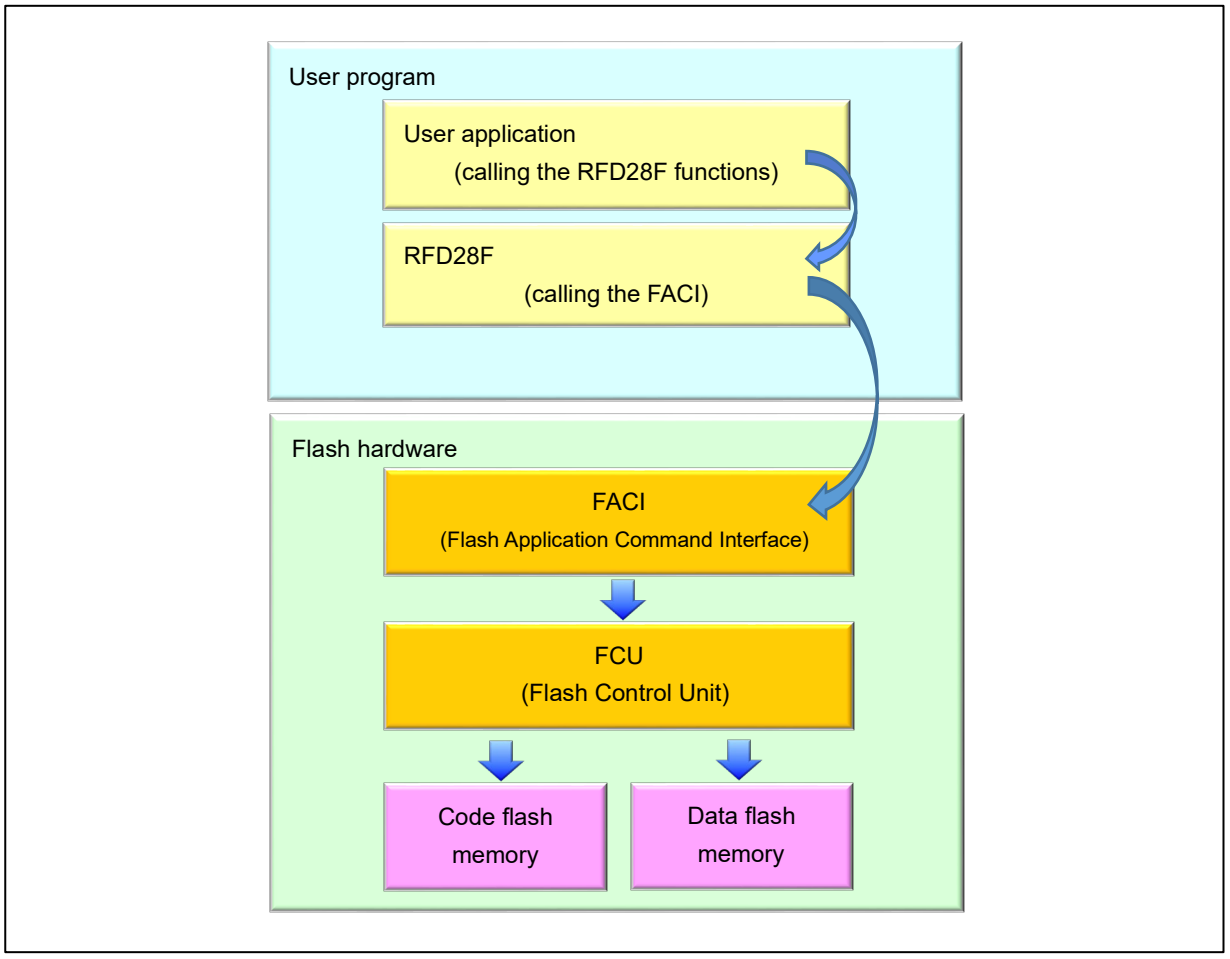
- Files of the RFD28F for working with data flash memory, code flash memory and hardware property area (source code).
- Sample projects
  - Sample project for erasure and writing the contents to the data flash memory, code flash memory and hardware property area (executed from code flash memory).

1.3 Product Feature

The RFD28F provides simple operation as an API (RFD28F function). That is, the user can handle flash memory operations by calling the RFD28F function for the processing to be implemented in order within the application.

This RFD28F handles calls of the “Flash Application Command Interface (FACI)” in the flash hardware. Figure 1-1 is an image of how a user application interacts with data flash memory and code flash memory.

Figure 1-1 Image of Operation of Flash Memory



The flash operation is performed based on the processing flow using the FACI command defined by hardware. Although one API can be prepared for each processing flow, RFD28F is to realize by combining multiple APIs and processing to be performed by users to one processing flow. This is because there are cases where the timeout value varies depending on time and case, such as processing dependent on the user application, for example, timeout processing, and such a configuration is adopted so as to flexibly correspond to these. RFD28F provide a sample program that is a combination of multiple APIs and what should be done by the user. Please refer to these sample program when embedding the flash operation process into the application.

1.4 Environment

- Host Machine

  The Environment does not depend on the host machine, but it requires an environment in which the C compiler package and debugger can run (Development and testing was carried out in Windows 10 Enterprise).

- C-Compiler Package

  The C compiler package under operation check is as follows.

Package	Maker	Version
MULTI (CCRH850)	Green Hills Software, Inc.	V2018.1.5
Package	Maker	Version
CC-RH (CS+)	Renesas Electronics.	V2.01.00

- Debugger

  The Debugger under operation check is as follows.

Package	Maker	Version
MULTI	Green Hills Software, Inc.	V7.1.6
Package	Maker	Version
CS+	Renesas Electronics.	V8.03.00

- Target MCU
  - RH850/U2A16 (400MHz)

    In the case of RH850/U2A16, there are three types of packages “FBGA516”. “FBGA373” and “FBGA292”, but both are equivalent for the flash operation function, there is no problem using either package.

- C-Compiler Option

  The C compiler option under operation check is as follows.

  [MULTI (CCRH850)]

  - -c
  - -cpu=rh850g4mh
  - -delete
  - --diag_error 193
  - -dual_debug
  - -g
  - -gsize
  - -ignore_callt_state_in_interrupts
  - -inline_prologue
  - -large_sda
  - -no_callt
  - --no_commons
  - -Ogeneral
  - -prepare_dispose
  - --prototype_errors
  - -reserve_r2
  - -sda=all
  - --short_enum
  - -shorten_loads
  - -shorten_moves
  - -Wshadow
  - -Wundef
  - -passsource
  - -additional_sda_reg=0
  - -rh850_abi=ghs2014

  [CC-RH (CS+)]

  - -c
  - -Xcpu=g4mh
  - -Odelete_static_func
  - -g
  - -Odefault
  - -Xreserve_r2
  - -Xpass_source
  - -Xasm_option=-Xprn_path

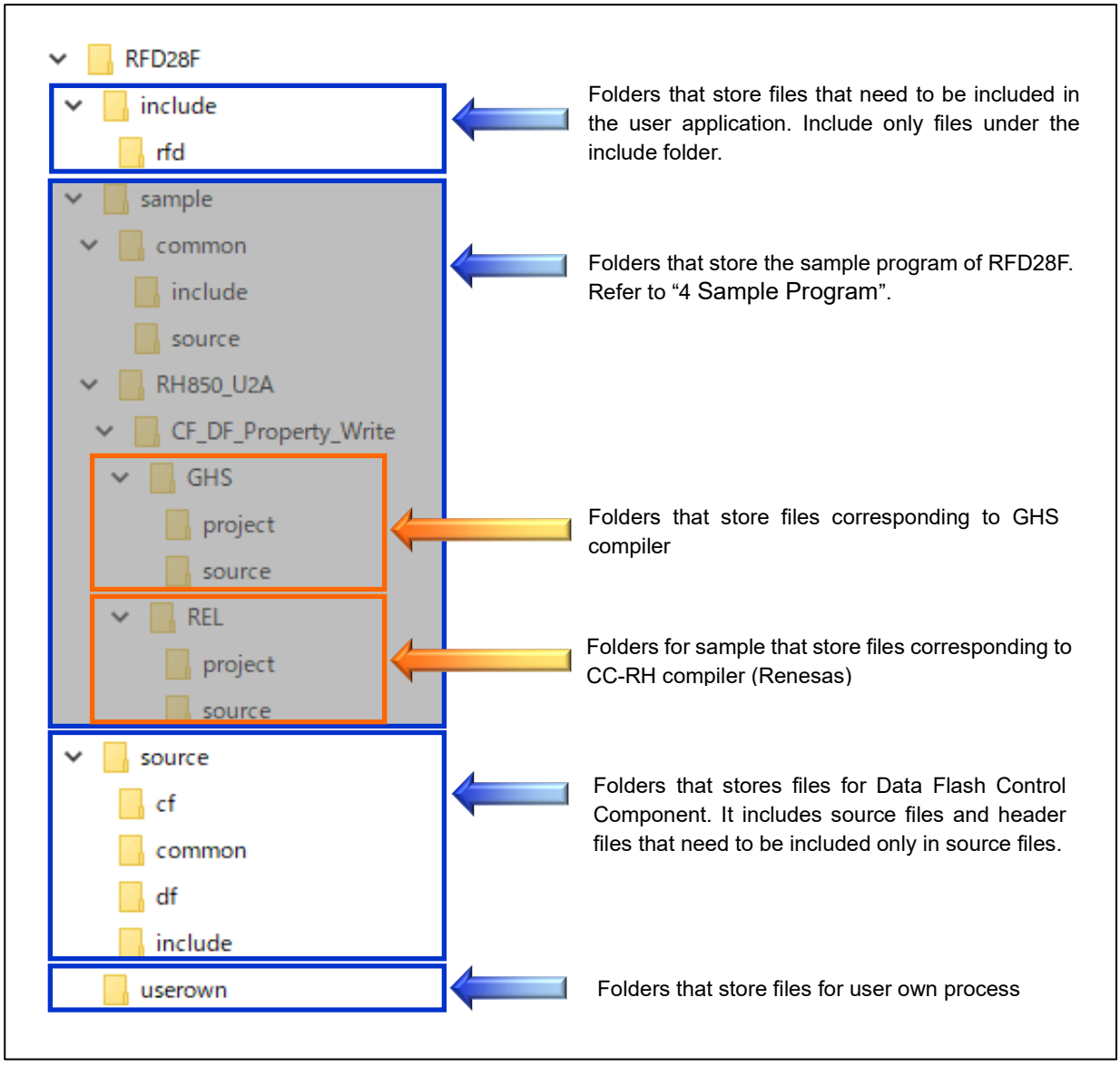
2 System Construction

2.1 File Structure

2.1.1 Folder Structure

The folder structure of the RFD28F is as follows.

Figure 2-1 Folder Structure



2.1.2 File List

2.1.2.1 Source File List

- Source files to allocated in “RFD28F/GHS/source/common”

No	Source File Name	Summary
1	r_rfd_common_api.c	File defining APIs of Common Flash Control Component in the RFD28F.
2	r_rfd_common_definition.c	Files that required definitions in Common Flash Control Component in the RFD28F.

- Source files to allocated in “RFD28F/userown”

No	Source File Name	Summary
1	r_rfd_common_userown.c	File defining API describing user own process.

- Source files to allocated in “RFD28F/GHS/source/df”

No	Source File Name	Summary
1	r_rfd_df_api.c	File defining APIs of Data Flash Control Component in the RFD28F.

- Source files to allocated in “RFD28F/GHS/source/cf”

No	Source File Name	Summary
1	r_rfd_cf_api.c	File defining APIs of Code Flash Control Component in the RFD28F.
2	r_rfd_cf_definition.c	Files that required definitions in Code Flash Control Component in the RFD28F.

2.1.2.2 Header File List

- Header files to allocated in “RFD28F/GHS/include”

No	Header File Name	Summary
1	r_rfd.h	The Header file that needs to be included when using RFD28F.
2	r_rfd_common.h	The Header file that needs to be included when using Common Flash Control Component in the RFD28F.
3	r_rfd_config.h	The Header file for user setting.
4	r_rfd_df.h	The Header file that needs to be included when using Data Flash Control Component in the RFD28F.
5	r_rfd_cf.h	The Header file that needs to be included when using Code Flash Control Component in the RFD28F.

・ Header files to allocated in “RFD28F/GHS/include/rfd”

No	Header File Name	Summary
1	r_rfd_common_api.h	The File defining prototype declaration of API of Common Flash Control Component in the RFD28F.
2	r_rfd_common_error.h	The file defining the return value and error value of API of Common Flash Control Component in the RFD28F.
3	r_rfd_common_types.h	The file that defines the type of variables and structure of the Common Flash Control Component in the RFD28F.
4	r_rfd_common_version.h	File defining version information of the Common Flash Control Component in the RFD28F.
5	r_rfd_compiler.h	The file defining compiler specification used by RFD28F.
6	r_rfd_constant.h	The file defining constants used by RFD28F.
7	r_rfd_device.h	The file defining Hardware-specific definition that used by RFD28F.
8	r_rfd_memmap.h	The file defining section information that used by RFD28F.
9	r_rfd_types.h	The file that defines the type of variables and structure that used by RFD28F.
10	r_rfd_df_api.h	The File defining prototype declaration of API of the Data Flash Control Component in the RFD28F.
11	r_rfd_df_error.h	The file defining the return value and error value of API of the Data Flash Control Component in the RFD28F.
12	r_rfd_df_types.h	The file that defines the type of variables and structure of the Data Flash Control Component in the RFD28F.
13	r_rfd_df_version.h	File defining version information of the Data Flash Control Component in the RFD28F.
14	r_rfd_cf_api.h	The File defining prototype declaration of API of the Code Flash Control Component in the RFD28F.
15	r_rfd_cf_error.h	The file defining the return value and error value of API of the Code Flash Control Component in the RFD28F.
16	r_rfd_cf_types.h	The file that defines the type of variables and structure of the Code Flash Control Component in the RFD28F.
17	r_rfd_cf_version.h	File defining version information of the Code Flash Control Component in the RFD28F.

・ Header files to allocated in “RFD28F/GHS/source/include”

No	Header File Name	Summary
1	r_rfd_common_local.h	The file that defines values to be used in the internal functions of the Common Flash Control Component in the RFD28F.
2	r_rfd_df_local.h	The file that defines values to be used in the internal functions of the Data Flash Control Component in the RFD28F.
3	r_rfd_cf_local.h	The file that defines values to be used in the internal functions of the Code Flash Control Component in the RFD28F.

2.2 How to construct

There is a project file in the following folder.

- RFD28F/sample/RH850_U2A/CF_DF_Property_Write/GHS/project
- RFD28F/sample/RH850_U2A/CF_DF_Property_Write/REL/project

Since RFD28F is provided as a source file, it is not being built only RFD28F, but will be built at the same time as the sample program (user application).

2.3 Resource Usage

2.3.1 Hardware Function List

Base Address	Offset	Register Name	Function name / Note
FACI0 :0xffa10000	0x0	FPMON_0	Flash Pin Monitor Register 0
	0x10	FASTAT_0	Flash Access Status Register 0
	0x14	FAEINT_0	Flash Access Error Interrupt Enable Register 0
	0x30	FSADDR_0	Flash Start Address Register 0
	0x34	FEADDR_0	Flash End Address Register 0
	0x40	FCVAPROT_0	Code Flash Valid Area Protection Register 0
	0x80	FSTATR_0	Flash Status Register 0
	0x84	FENTRYR_0	Program/Erase Mode Entry Register 0
	0x8C	FSUINITR_0	Flash Sequencer Set-up Initialize Register 0
	0xA0	FCMDR_0	Flash Sequencer Command Register 0
	0xA4	FCMDMON_0	Flash Sequencer Command Monitor Register 0
	0xA8	FSWASTAT_0	Switch Area Status Register 0
	0xC0	FPESTAT_0	Program/Erase Error Status Register 0
	0xD0	FBCCNT_0	Blank Check Control Register 0
	0xD4	FBCSTAT_0	Blank Check Status Register 0
	0xD8	FPSADDR_0	Programmed Area Start Address Register 0
	0xE0	FCPSR_0	FCU Process Switch Register 0
	0x100	FECCEMON_0	Flash ECC Encoder Monitor Register 0
	0x104	FECCTMD_0	Flash ECC Test Mode Register 0
	0x108	FDMYECC_0	Flash Dummy ECC Register 0
FACI1 :0xffa14000	0x0	FPMON_1	Flash Pin Monitor Register 1
	0x10	FASTAT_1	Flash Access Status Register 1
	0x14	FAEINT_1	Flash Access Error Interrupt Enable Register 1
	0x30	FSADDR_1	Flash Start Address Register 1
	0x34	FEADDR_1	Flash End Address Register 1
	0x40	FCVAPROT_1	Code Flash Valid Area Protection Register 1
	0x80	FSTATR_1	Flash Status Register 1
	0x84	FENTRYR_1	Program/Erase Mode Entry Register 1
	0x8C	FSUINITR_1	Flash Sequencer Set-up Initialize Register 1
	0xA0	FCMDR_1	Flash Sequencer Command Register 1
	0xA4	FCMDMON_1	Flash Sequencer Command Monitor Register 1
	0xA8	FSWASTAT_1	Switch Area Status Register 1
	0xC0	FPESTAT_1	Program/Erase Error Status Register 1
	0xD0	FBCCNT_1	Blank Check Control Register 1

Base Address	Offset	Register Name	Function name / Note
FACI1 :0xffa14000	0xD4	FBCSTAT_1	Blank Check Status Register 1
	0xD8	FPSADDR_1	Programmed Area Start Address Register 1
	0xE0	FCPSR_1	FCU Process Switch Register 1
	0x100	FECCEMON_1	Flash ECC Encoder Monitor Register 1
	0x104	FECCTMD_1	Flash ECC Test Mode Register 1
	0x108	FDMYECC_1	Flash Dummy ECC Register 1
FACI2 :0xffa18000	0x0	FPMON_2	Flash Pin Monitor Register 2
	0x10	FASTAT_2	Flash Access Status Register 2
	0x14	FAEINT_2	Flash Access Error Interrupt Enable Register 2
	0x30	FSADDR_2	Flash Start Address Register 2
	0x34	FEADDR_2	Flash End Address Register 2
	0x40	FCVAPROT_2	Code Flash Valid Area Protection Register 2
	0x80	FSTATR_2	Flash Status Register 2
	0x84	FENTRYR_2	Program/Erase Mode Entry Register 2
	0x8C	FSUINITR_2	Flash Sequencer Set-up Initialize Register 2
	0xA0	FCMDR_2	Flash Sequencer Command Register 2
	0xA4	FCMDMON_2	Flash Sequencer Command Monitor Register 2
	0xA8	FSWASTAT_2	Switch Area Status Register 2
	0xC0	FPESTAT_2	Program/Erase Error Status Register 2
	0xD0	FBCCNT_2	Blank Check Control Register 2
	0xD4	FBCSTAT_2	Blank Check Status Register 2
	0xD8	FPSADDR_2	Programmed Area Start Address Register 2
	0xE0	FCPSR_2	FCU Process Switch Register 2
	0x100	FECCEMON_2	Flash ECC Encoder Monitor Register 2
	0x104	FECCTMD_2	Flash ECC Test Mode Register 2
	0x108	FDMYECC_2	Flash Dummy ECC Register 2
IDCTRL :0xffa08000	0x00	SPIDIN0	Serial Programmer ID Input Register 0
	0x04	SPIDIN1	Serial Programmer ID Input Register 1
	0x08	SPIDIN2	Serial Programmer ID Input Register 2
	0x0c	SPIDIN3	Serial Programmer ID Input Register 3
	0x10	SPIDIN4	Serial Programmer ID Input Register 4
	0x14	SPIDIN5	Serial Programmer ID Input Register 5
	0x18	SPIDIN6	Serial Programmer ID Input Register 6
	0x1c	SPIDIN7	Serial Programmer ID Input Register 7
	0x20	DFIDIN0	Data Flash ID Input Register 0
	0x24	DFIDIN1	Data Flash ID Input Register 1
	0x28	DFIDIN2	Data Flash ID Input Register 2
	0x2c	DFIDIN3	Data Flash ID Input Register 3

Base Address	Offset	Register Name	Function name / Note
IDCTRL :0xffa08000	0x30	DFIDIN4	Data Flash ID Input Register 4
	0x34	DFIDIN5	Data Flash ID Input Register 5
	0x38	DFIDIN6	Data Flash ID Input Register 6
	0x3c	DFIDIN7	Data Flash ID Input Register 7
	0x40	OCDIDIN0	OCDID Input Register 0
	0x44	OCDIDIN1	OCDID Input Register 1
	0x48	OCDIDIN2	OCDID Input Register 2
	0x4c	OCDIDIN3	OCDID Input Register 3
	0x50	OCDIDIN4	OCDID Input Register 4
	0x54	OCDIDIN5	OCDID Input Register 5
	0x58	OCDIDIN6	OCDID Input Register 6
	0x5c	OCDIDIN7	OCDID Input Register 7
	0x80	CUSTIDAIN0	Customer ID A Input Register 0
	0x84	CUSTIDAIN1	Customer ID A Input Register 1
	0x88	CUSTIDAIN2	Customer ID A Input Register 2
	0x8c	CUSTIDAIN3	Customer ID A Input Register 3
	0x90	CUSTIDAIN4	Customer ID A Input Register 4
	0x94	CUSTIDAIN5	Customer ID A Input Register 5
	0x98	CUSTIDAIN6	Customer ID A Input Register 6
	0x9c	CUSTIDAIN7	Customer ID A Input Register 7
	0xa0	CUSTIDBIN0	Customer ID B Input Register 0
	0xa4	CUSTIDBIN1	Customer ID B Input Register 1
	0xa8	CUSTIDBIN2	Customer ID B Input Register 2
	0xac	CUSTIDBIN3	Customer ID B Input Register 3
	0xb0	CUSTIDBIN4	Customer ID B Input Register 4
	0xb4	CUSTIDBIN5	Customer ID B Input Register 5
	0xb8	CUSTIDBIN6	Customer ID B Input Register 6
	0xbc	CUSTIDBIN7	Customer ID B Input Register 7
	0xc0	CUSTIDCIN0	Customer ID C Input Register 0
	0xc4	CUSTIDCIN1	Customer ID C Input Register 1
	0xc8	CUSTIDCIN2	Customer ID C Input Register 2
	0xcc	CUSTIDCIN3	Customer ID C Input Register 3
	0xd0	CUSTIDCIN4	Customer ID C Input Register 4
	0xd4	CUSTIDCIN5	Customer ID C Input Register 5
	0xd8	CUSTIDCIN6	Customer ID C Input Register 6
	0xdc	CUSTIDCIN7	Customer ID C Input Register 7
	0x1fc	IDST	ID Authentication Status Register
	0x200	RHSIFIDIN0	RHSIF ID Input Register 0

Base Address	Offset	Register Name	Function name / Note
IDCTRL :0xffa08000	0x204	RHSIFIDIN1	RHSIF ID Input Register 1
	0x208	RHSIFIDIN2	RHSIF ID Input Register 2
	0x20c	RHSIFIDIN3	RHSIF ID Input Register 3
	0x210	RHSIFIDIN4	RHSIF ID Input Register 4
	0x214	RHSIFIDIN5	RHSIF ID Input Register 5
	0x218	RHSIFIDIN6	RHSIF ID Input Register 6
	0x21c	RHSIFIDIN7	RHSIF ID Input Register 7
	0x3fc	IDST2	ID Authentication Status Register2
FHVE :0xff984800	0x00	FHVE3FP0	FHVE3 Control Register for FACI0
	0x04	FHVE15FP0	FHVE15 Control Register for FACI0
	0x10	FHVE3FP1	FHVE3 Control Register for FACI1
	0x14	FHVE15FP1	FHVE15 Control Register for FACI1
	0x20	FHVE3FP2	FHVE3 Control Register for FACI2
	0x24	FHVE15FP2	FHVE15 Control Register for FACI2

Accessed only when using Code Flash Control Component

Base Address	Offset	Register Name	Function name / Note
ECCDF :0xffc62c00	0x04	DFERSTR	Data Flash error status register
	0x08	DFERSTC	Data Flash error status clear register

The usage registers for each API are in attached file (RFD28F_UsingRegisterList_U2A.xlsx). The green part is the parent API and it describes the API which is called below it. The gray part means there is no use register.

2.3.2 Memory Map

- ROM

Section Name	Contents of allocation
.R_RFD_CODE_COMMON	The area in which the code of RFD28F for common flash control component executed on code flash. Basically, this area is allocated on ROM.
.R_RFD_CODE_COMMON_RAM_NO_BGO	The area in which the code of RFD28F for common flash control component executed on code flash. If the BGO has the possible conditions, it can be allocated on ROM, but if the BGO does not have the possible conditions, it must be allocated on RAM. For example, when reading the contents of code flash memory during writing / erasing, it can be read from another Bank, but it cannot be read from the same Bank. If we want to read from the same Bank, we need to allocate the code to read in RAM.
.R_RFD_CODE_USEROWN_COMMON	The area where the user-own code is allocated. Basically, this area is allocated on ROM.
.R_RFD_CODE_DF	The area in which the code of RFD28F for data flash control component executed on code flash.
.R_RFD_CODE_CF	The area in which the code for controlling code flash executed on code flash of the Code Flash Control Components of RFD28F. Basically, this area is allocated on ROM.
.R_RFD_CODE_CF_RAM_NO_BGO	The area in which the code for controlling code flash executed on code flash of the Code Flash Control Components of RFD28F. If the BGO has the possible conditions, it can be allocated on ROM, but if the BGO does not have the possible conditions, it must be allocated on RAM. For example, when reading the contents of code flash memory during writing / erasing, it can be read from another Bank, but it cannot be read from the same Bank. If we want to read from the same Bank, we need to allocate the code to read in RAM.
.R_RFD_CODE_EXTRA	The area in which the code for controlling hardware property executed on code flash of the Code Flash Control Components of RFD28F. Basically, this area is allocated on ROM.

Section Name	Contents of allocation
.R_RFD_ROSDATA_EXTRA	An area in which the address information of each VOFC existing in the Configuration Setting Area, Security Setting Area, Block Protection Area0, Block Protection Area1 in the Flash Extra Area is allocated (When placed in the rosdata attribute)
.R_RFD_RODATA_EXTRA	An area in which the address information of each VOFC existing in the Configuration Setting Area, Security Setting Area, Block Protection Area0, Block Protection Area1 in the Flash Extra Area is allocated (When placed in the rosdata attribute).
.R_RFD_ROSDATA_VERSION_COMMON	The area where version information of Common Flash Control Component is allocated (When placed in the rosdata attribute).
.R_RFD_RODATA_VERSION_COMMON	The area where version information of Common Flash Control Component is allocated (When placed in the rodata attribute).
.R_RFD_ROSDATA_VERSION_DF	The area where version information of Data Flash Control Component is allocated (When placed in the rosdata attribute).
.R_RFD_RODATA_VERSION_DF	The area where version information of Data Flash Control Component is allocated (When placed in the rodata attribute).
.R_RFD_ROSDATA_VERSION_CF	The area where version information of Code Flash Control Component is allocated (When placed in the rosdata attribute).
.R_RFD_RODATA_VERSION_CF	The area where version information of Code Flash Control Component is allocated (When placed in the rodata attribute).
.R_RFD_CODE_EX_PROT	The area is for preventing occurrence of ECC error where is allocating the last place in the code flash area.

This component can operate in both code flash area and RAM. However, it is necessary to allocate it within the range that the branch instruction can reach from the function that calls the interface of this component.

- RAM

Section Name	Contents of allocation
.R_RFD_SBSS	The area for allocating data having no initial value used by the RFD28F (When placed in the sbss attribute).
.R_RFD_BSS	The area for allocating data having no initial value used by the RFD28F (When placed in the bss attribute).

Although neither section is dependent on the allocation address, but it must be allocated within accessible from the interface of this component.

2.3.3 Code Size

Section Name	Size (Bytes)	
	GHS	Renesas
.R_RFD_CODE_COMMON	518 (0x206)	720 (0x2d0)
.R_RFD_CODE_COMMON_RAM_NO_BGO	918 (0x396)	1080 (0x438)
.R_RFD_CODE_DF	570 (0x23a)	662 (0x296)
.R_RFD_CODE_CF	220 (0xdc)	316 (0x13c)
.R_RFD_CODE_CF_RAM_NO_BGO	246 (0xf6)	288 (0x120)
.R_RFD_CODE_EXTRA	1610 (0x64a)	1908 (0x774)
.R_RFD_CODE_EX_PROT	64 (0x40)	64 (0x40)
.R_RFD_CODE_USEROWN_COMMON	Depending on user application	Depending on user application
Total	4146 (0x1032) + userown	5038 (0x13ae) + userown

2.3.4 Data Size for ROM

Section Name	Size (Bytes)	
	GHS	Renesas
.R_RFD_ROSDATA_EXTRA or .R_RFD_RODATA_EXTRA	312 (0x138)	312 (0x138)
.R_RFD_ROSDATA_VERSION_COMMON or .R_RFD_RODATA_VERSION_COMMON	20 (0x14)	20 (0x14)
.R_RFD_ROSDATA_VERSION_DF or .R_RFD_RODATA_VERSION_DF	20 (0x14)	19 (0x19)
.R_RFD_ROSDATA_VERSION_CF or .R_RFD_RODATA_VERSION_CF	20 (0x14)	19 (0x19)
Total	372 (0x174)	370 (0x172)

2.3.5 Data Size for RAM

Section Name	Size (Bytes)	
	GHS	Renesas
.R_RFD_SBSS or .R_RFD_BSS	8 (0x8)	8 (0x8)
Total	8 (0x8)	8 (0x8)

2.3.6 Stack Size

I/F Name	Used stack size (Bytes)	
R_RFD_Init	Min-case 28 (0x1c) Max-case 56 (0x38)	Min-case 28 (0x1c) Max-case 56 (0x38)
R_RFD_ShiftToReadMode	Min-case 8 (0x8) Max-case 28 (0x1c)	Min-case 8 (0x8) Max-case 28 (0x1c)
R_RFD_ShiftToPEMode	12 (0xc)	12 (0xc)
R_RFD_CheckPEMode	8 (0x8)	8 (0x8)
R_RFD_ForcedStopAndErrorClear	12 (0xc)	12 (0xc)
R_RFD_StatusClear	8 (0x8)	8 (0x8)
R_RFD_GetFaciSequenceReady	8 (0x8)	8 (0x8)
R_RFD_GetFaciStatus	8 (0x8)	8 (0x8)
R_RFD_SuspendPERequest	8 (0x8)	8 (0x8)
R_RFD_ResumePERequest	8 (0x8)	8 (0x8)
R_RFD_SetFHVE	12 (0xc)	12 (0xc)
R_RFD_DFIDAuth	44 (0x3c)	44 (0x3c)
R_RFD_EraseDFRequest	16 (0x10)	16 (0x10)
R_RFD_WriteDFRequest	20 (0x14)	20 (0x14)
R_RFD_BlankCheckDFRequest	16 (0x10)	16 (0x10)
R_RFD_GetBlankCheckResult	12 (0xc)	12 (0xc)
R_RFD_DFAddressToFaciNumber	4 (0x4)	4 (0x4)
R_RFD_IDAuth	52 (0x34)	52 (0x34)
R_RFD_EraseCFRequest	16 (0x10)	16 (0x10)
R_RFD_WriteCFRequest	20 (0x14)	20 (0x14)
R_RFD_ErasePropertyRequest	8 (0x8)	8 (0x8)
R_RFD_WritePropertyRequest	12 (0xc)	12 (0xc)
R_RFD_EraseSwitchRequest	16 (0x10)	16 (0x10)
R_RFD_WriteSwitchRequest	8 (0x8)	8 (0x8)
R_RFD_EraseTagRequest	56 (0x38)	56 (0x38)
R_RFD_UpdateTagRequest	56 (0x38)	56 (0x38)
R_RFD_EraseExtendedDataRequest	8 (0x8)	8 (0x8)
R_RFD_WriteExtendedDataRequest	16 (0x10)	16 (0x10)
R_RFD_CFAddressToFaciNumber	8 (0x8)	8 (0x8)

2.3.7 Response Time

Target : RH850/U2A16 [400MHz]

I/F Name	Execution time (us)		Condition
	GHS	REL	
R_RFD_Init	1.12	1.30	Execute in FACI0, FACI1 and FACI2 are in read mode
	7.15	7.71	Execute in FACI0, FACI1 and FACI2 are in P/E mode. if Flash sequencer is BUSY, the execution time of R_RFD_ForcedStopAndErrorClear is added for each FACI which Flash sequencer is BUSY.
R_RFD_ShiftToReadMode	0.60	0.59	Shift P/E mode to read mode.
	0.57	0.59	If it was already in read mode
	0.62	0.60	Shift P/E mode to read mode. However, if FHVEERR occurs, the execution time of R_RFD_ForcedStopAndErrorClear is added.
R_RFD_ShiftToPEMode	0.57	0.66	Shift read mode to P/E mode
	0.31	0.27	If it was already in P/E mode
R_RFD_CheckPEMode	0.23	0.18	-
R_RFD_ForcedStopAndErrorClear	1.46	1.56	Forced terminate (no timeout). In the worst case, 20us is added to this execution time by hardware processing
	39.43	47.19	The fastest reference value for timeout processing in R_RFD_ForcedStopAndErrorClear when setting 300 to R_RFD_VALUE_FORCED_STOP_TIMEOUT in configuration. Reference value assuming that the hardware access time is the fastest
R_RFD_StatusClear	0.14	0.10	Clear status
R_RFD_GetFaciSequenceReady	0.26	0.25	-
R_RFD_GetFaciStatus	0.33	0.39	When the FHVEERR bit (FSTATR register) is valid (shortest processing)
	0.52	0.60	When the CMDLK bit (FASTAT register), ERSSPD bit (FSTATR register) and PRGSPD bit (FSTATR register) are invalid (longest processing)
R_RFD_SuspendPERequest	0.27	0.27	Suspend writing process
	0.31	0.30	The case that there is no suspendable process
R_RFD_ResumePERequest	0.08	0.08	Resumed suspended write processing
R_RFD_SetFHVE	0.34	0.33	-

I/F Name	Execution time (us)		Condition
	GHS	REL	
R_RFD_DFIDAuth	0.91	0.91	Data to be authenticated placed in 4-byte alignment place
	1.30	1.38	Data to be authenticated placed in not 4-byte alignment place
R_RFD_EraseDFRequest	0.36	0.43	Require to erase a block
R_RFD_WriteDFRequest	0.39	0.40	Require to write 4bytes data
	2.83	2.82	Require to write 128bytes data
R_RFD_BlankCheckDFRequest	0.26	0.33	Require to search free space (for 4 Kbytes)
R_RFD_GetBlankCheckResult	0.33	0.33	Acquire free space search results when there are free spaces
	0.44	0.44	Acquire free space search results when there are no free spaces
R_RFD_DFAddressToFaciNumber	0.17	0.20	With FACI0, FACI1 and FACI2 validated, specify the address of FACI0
	0.22	0.20	With FACI0, FACI1 and FACI2 validated, specify the address of FACI0
	0.23	0.21	With FACI0, FACI1 and FACI2 validated, sspecify the FACI2 address
R_RFD_IDAuth	0.97	1.18	Data to be authenticated placed in 4-byte alignment place (Authenticate the RHSIFID which takes the most processing time)
	1.33	1.42	Data to be authenticated placed in not 4-byte alignment place (Authenticate the RHSIFID which takes the most processing time)
R_RFD_EraseCFRequest	0.31	0.27	Require to erase a block
R_RFD_WriteCFRequest	10.21	10.28	Require to write 512bytes data
R_RFD_CFAddressToFaciNumber	0.08	0.16	With FACI0 and FACI1 validated, specify the address of FACI0
	0.09	0.19	With FACI0 and FACI1 validated, specify the address of FACI1
R_RFD_ErasePropertyRequest	0.20	0.23	Require to erase the Configuration Setting Area
R_RFD_WritePropertyRequest	0.94	0.87	Require to write the Configuration Setting Area
R_RFD_EraseSwitchRequest	0.46	0.47	Require to erase the Switch Area
R_RFD_WriteSwitchRequest	0.81	0.75	Require to write the Switch Area
R_RFD_EraseTagRequest	2.81	2.87	Require to erase the Tag Area

I/F Name	Execution time (us)		Condition
	GHS	REL	
R_RFD_UpdateTagRequest	2.50	2.43	Require to write the Tag Area
R_RFD_EraseExtendedDataRequest	0.20	0.21	Require to erase a block
R_RFD_WriteExtendedDataRequest	0.33	0.30	Require to write 4bytes data
	2.73	2.80	Require to write 128bytes data

- Maximum Interrupt Disable Time

Maximum Interrupt Disable Time(us)		Condition
GHS	REL	
0.56	0.59	Interrupt disable time in processing to check correctness of all BPA1VOFC in Block Protection Area for FPSYS1 in R_RFD_EraseTagRequest and R_RFD_UpdateTagRequest.

2.3.8 Interrupt List

There are no need interruptions.

3 RFD28F API

3.1 API List

- For Common Flash Control

API Name	Outline
R_RFD_Init	The function to initialize the RFD28F.
R_RFD_ShiftToReadMode	The function to shift the mode to read mode of flash memory.
R_RFD_ShiftToPEMode	The function to shift the mode to P/E mode of flash memory.
R_RFD_CheckPEMode	The function to check whether P/E mode of the data flash.
R_RFD_ForcedStopAndErrorClear	The function to forced stop and clear an error of RFD28F.
R_RFD_StatusClear	The function to clear the status of RFD28F.
R_RFD_GetFaciSequenceReady	The function to check whether FACI is in Ready state.
R_RFD_GetFaciStatus	The function to acquire FACI state.
R_RFD_SuspendPERequest	The function to require for suspension of process for RFD28F.
R_RFD_ResumePERequest	The function to require to resume for suspending process to RFD28F.
R_RFD_SetFHVE	The function to enable or disable writing, erasure and blank checking to the flash memory.

- For Data Flash Control

API Name	Outline
R_RFD_DFIDAuth	The function to authenticate ID of Data flash memory.
R_RFD_EraseDFRequest	The function to erase the data of the data flash memory.
R_RFD_WriteDFRequest	The function to write the data to the data flash memory.
R_RFD_BlankCheckDFRequest	The function to check the blank of the data flash memory.
R_RFD_GetBlankCheckResult	The function to acquire the result of blank check of data flash memory.
R_RFD_DFAddressToFaciNumber	The function to acquires the FACI number of the data flash memory address.

・ For Code Flash Control

API Name	Outline
R_RFD_IDAuth	The function to authenticate ID.
R_RFD_EraseCFRequest	The function to erase the data of the data flash memory.
R_RFD_WriteCFRequest	The function to write the data to the code flash memory.
R_RFD_ErasePropertyRequest	The function to erase the Hardware property area for invalid side (Configuration setting area, Block protection setting area, Security setting area).
R_RFD_WritePropertyRequest	The function to write data to the Hardware property area for invalid side (Configuration setting area, Block protection setting area, Security setting area).
R_RFD_EraseSwitchRequest	The function to erase data of switch area for invalid side.
R_RFD_WriteSwitchRequest	The function to write data to switch area for invalid side.
R_RFD_EraseTagRequest	The function to erase data of TAG area.
R_RFD_UpdateTagRequest	The function to write data to TAG area.
R_RFD_EraseExtendedDataRequest	The function to erase the data of the Extended Data Area.
R_RFD_WriteExtendedDataRequest	The function to write the data to the Extended Data Area.
R_RFD_CFAddressToFaciNumber	The function to acquires the FACI number of the code flash memory address.
R_RFD_PropertyAddressToFaciNumber	The function to acquires the FACI number of the Hardware Property Area.

3.2 Data Type Definition

3.2.1 Data Type

Macro value	Type	Description
T_s1	signed char	1byte signed integer
T_u1	unsigned char	1byte unsigned integer
T_s2	signed short	2byte signed integer
T_u2	unsigned short	2byte unsigned integer
T_s4	signed long	4byte signed integer
T_u4	unsigned long	4byte unsigned integer
T_bl	unsigned char	Boolean (True/FALSE)
T_en_FACIMode	enum	FACI mode
T_en_Protect	enum	For enabled or disabled to writing / erasure / blank checking of data flash memory by FHVE register
T_u2_FACI	T_u2	Target FACI
T_u4_RFDReturn	T_u4	Return value
T_u4_RfdAddress	T_u4	Flash memory address
T_pu4_RfdBuffer	T_u4*	Pointer to the buffer that stores the data to be written
T_en_IDType	enum	Type of authentication ID

3.2.2 Global Pointer Variable

No definition

3.2.3 Global Variable

Global variables used in RFD28F are as follows

（1） bu4_CMN_FaciBaseAddress

Location	r_rfd_common_definition.c
Type & Name	T_u4 bu4_CMN_FaciBaseAddress
Default value	Non
Description	Register Base Address for FACI
Referenced by	

（2） bu4_CMN_FaciCommandArea

Location	r_rfd_common_definition.c
Type & Name	T_u4 bu4_CMN_FaciCommandArea
Default value	Non
Description	Command-issuing area for FACI
Referenced by	

3.2.4 Structure

Structures used in RFD28F are as follows

- faciRegisters

Location	r_rfd_device.h	
1st Element	volatile T_u1 FPMON	FPMON
2nd Element	volatile T_u1 reserve00[15]	Reserved (15 bytes)
3rd Element	volatile T_u1 FASTAT	FASTAT
4th Element	volatile T_u1 reserve01[3]	Reserved (3 bytes)
5th Element	volatile T_u1 FAEINT	FAEINT
6th Element	volatile T_u1 reserve02[3]	Reserved (3 bytes)
7th Element	volatile T_u1 reserve03[1]	Reserved (1 bytes)
8th Element	volatile T_u1 reserve04[7]	Reserved (7 bytes)
9th Element	volatile T_u1 reserve05[2]	Reserved (2 bytes)
10th Element	volatile T_u1 reserve06[14]	Reserved (14 bytes)
11th Element	volatile T_u4 FSADDR	FSADDR
12th Element	volatile T_u4 FEADDR	FEADDR
13th Element	volatile T_u1 reserve07[8]	Reserved (8 bytes)
14th Element	volatile T_u2 FCVAPROT	FCVAPROT
15th Element	volatile T_u1 reserve08[62]	Reserved (62 bytes)
16th Element	volatile T_u4 FSTATR	FSTATR
17th Element	volatile T_u2 FENTRYR	FENTRYR
18th Element	volatile T_u1 reserve09[6]	Reserved (6 bytes)
19th Element	volatile T_u2 FSUINITR	FSUINITR
20th Element	volatile T_u1 reserve10[10]	Reserved (10 bytes)
21st Element	volatile T_u1 reserve11[1]	Reserved (1 bytes)
22nd Element	volatile T_u1 reserve12[3]	Reserved (3 bytes)
23rd Element	volatile T_u1 reserve13[1]	Reserved (1 bytes)
24th Element	volatile T_u1 reserve14[3]	Reserved (3 bytes)
25th Element	volatile T_u2 FCMDR	FCMDR
26th Element	volatile T_u1 reserve15[2]	Reserved (2 bytes)
27th Element	volatile T_u2 FCMDMON	FCMDMON
28th Element	volatile T_u1 reserve16[2]	Reserved (2 bytes)
29th Element	volatile T_u1 FSWASTAT	FSWASTAT
30th Element	volatile T_u1 reserve17[23]	Reserved (23 bytes)
31st Element	volatile T_u2 FPESTAT	FPESTAT
32nd Element	volatile T_u1 reserve18[14]	Reserved (14 bytes)
33rd Element	volatile T_u1 FBCCNT	FBCCNT
34th Element	volatile T_u1 reserve19[3]	Reserved (3 bytes)

35th Element	volatile T_u1 FBCSTAT	FBCSTAT
36th Element	volatile T_u1 reserve20[3]	Reserved (3 bytes)
37th Element	volatile T_u4 FPSADDR	FPSADDR
38th Element	volatile T_u1 reserve21[4]	Reserved (4 bytes)
39th Element	volatile T_u2 FCPSR	FCPSR
40th Element	volatile T_u1 reserve22[2]	Reserved (2 bytes)
41st Element	volatile T_u1 reserve23[2]	Reserved (2 bytes)
42nd Element	volatile T_u1 reserve24[10]	Reserved (10 bytes)
43rd Element	volatile T_u1 reserve25[1]	Reserved (1 bytes)
44th Element	volatile T_u1 reserve26[3]	Reserved (3 bytes)
45th Element	volatile T_u1 reserve27[4]	Reserved (4 bytes)
46th Element	volatile T_u1 reserve28[8]	Reserved (8 bytes)
47th Element	volatile T_u2 FECCEMON	FECCEMON
48th Element	volatile T_u1 reserve29[2]	Reserved (2 bytes)
49st Element	volatile T_u2 FECCTMD	FECCTMD
50th Element	volatile T_u1 reserve30[2]	Reserved (2 bytes)
51st Element	volatile T_u2 FDMYECC	FDMYECC
Description	Structure of register of FACI. When accessing each register of FACI, set the base address (bu4_CMN_FaciBaseAddress) of FACI to the start address of this structure.	

3.2.5 Enumeration

- `T_en_FACIMode`

Enumerated type for mode of FACI

Symbol Name	Value	Description
R_RFD_MODE_READ	0x0000	Read mode
R_RFD_MODE_DFPE	0x0080	P/E mode of data flash memory
R_RFD_MODE_CFPE	0x0001	P/E mode of code flash memory

- `T_en_Protect`

Enumerated type for enabled or disabled to writing / erasure / blank checking of data flash memory by FHVE register

Symbol Name	Value	Description
R_RFD_FHVE_PROTECT_ON	0x0000	Programming / erasure / DataFlashBlankChecking disabled
R_RFD_FHVE_PROTECT_OFF	0x0001	Programming / erasure / DataFlashBlankChecking enabled

- `T_en_IDType`

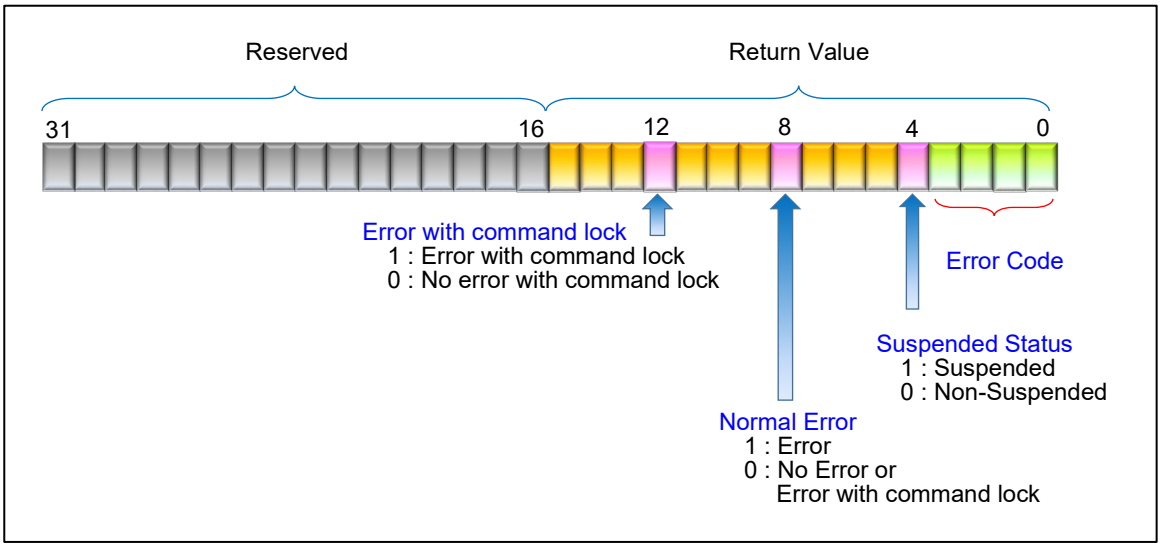
Enumeration for types of IDs for ID authentication

Symbol Name	Value	Description
R_RFD_ID_SPID	0x0000	Serial Programmer ID
R_RFD_ID_DFID	0x0001	Data Flash ID
R_RFD_ID_OCDID	0x0002	OCD ID
R_RFD_ID_CUSTIDA	0x0004	Customer ID A
R_RFD_ID_CUSTIDB	0x0005	Customer ID B
R_RFD_ID_CUSTIDC	0x0006	Customer ID C
R_RFD_ID_RHSIFID	0x0007	RHSIF ID

3.2.6 Macro definition

- Macro for return value

The rules for the return value are as follows.



Macro Name	Value	Description
R_RFD_OK	0x00000000	Refer to “3.4”.
R_RFD_STS_READY	0x00000001	Refer to “3.4”
R_RFD_STS_BUSY	0x00000002	Refer to “3.4”
R_RFD_STS_NOTBLANK	0x00000003	Refer to “3.4”
R_RFD_WARN_SWITCH_DATA	0x00000003	Refer to “3.4”
R_RFD_WARN_SECURITY_DATA	0x00000004	Refer to “3.4”
R_RFD_WARN_BLOCKPROTECTION0_DATA	0x00000005	Refer to “3.4”
R_RFD_WARN_BLOCKPROTECTION1_DATA	0x00000006	Refer to “3.4”
R_RFD_WARN_CONFIGURATION_DATA	0x00000007	Refer to “3.4”
R_RFD_WARN_ERASECOUNTER_DATA	0x00000008	Refer to “3.4”
R_RFD_WARN_INTERNAL_DATA	0x00000009	Refer to “3.4”
R_RFD_STS_ERASE_SUSPENDED	0x00000010	Refer to “3.4”
R_RFD_STS_WRITE_SUSPENDED	0x00000011	Refer to “3.4”
R_RFD_ERR_FHVE_PROTECT	0x00001000	Refer to “3.4”
R_RFD_ERR_ACCESS_ENV	0x00001001	Refer to “3.4”
R_RFD_ERR_INTERNAL_DATA	0x00001002	Refer to “3.4”
R_RFD_ERR_ERASE	0x00001003	Refer to “3.4”
R_RFD_ERR_WRITE	0x00001004	Refer to “3.4”
R_RFD_ERR_INTERNAL_HW	0x00001005	Refer to “3.4”
R_RFD_ERR_ERASECOUNTER_DATA	0x00001006	Refer to “3.4”
R_RFD_ERR_SWITCH_DATA	0x00001007	Refer to “3.4”
R_RFD_ERR_SECURITY_DATA	0x00001008	Refer to “3.4”
R_RFD_ERR_BLOCKPROTECTION0_DATA	0x00001009	Refer to “3.4”

R_RFD_ERR_BLOCKPROTECTION1_DATA	0x0000100a	Refer to “3.4”
R_RFD_ERR_CONFIGURATION_DATA	0x0000100b	Refer to “3.4”
R_RFD_ERR_REJECT	0x00000100	Refer to “3.4”
R_RFD_ERR_NOFACI	0x00000101	Refer to “3.4”
R_RFD_ERR_FACI_DEFINITION	0x00000102	Refer to “3.4”
R_RFD_ERR_IDAUTH	0x00000103	Refer to “3.4”
R_RFD_ERR_FLOW	0x00000104	Refer to “3.4”
R_RFD_ERR_FPMON	0x00000105	Refer to “3.4”

- Macro for specifying FACI to be operated

Macro Name	Value	Description
R_RFD_FACI0	0x0000	To be operated is FACI0
R_RFD_FACI1	0x0001	To be operated is FACI1
R_RFD_FACI2	0x0002	To be operated is FACI2
R_RFD_INVALID_FACI	0x00ff	FACI to be operated is invalid

- Macro for version number

Specify the version number in the following format.

Product Name 6 characters	Kind of component 2 or 3 characters	Kind of device 1 character	Kind of compiler 1 character	Version number 6 or 7 characters

1. Product Name
   - RFD28F : RFD28F
2. Kind of component
   - CMN : Common Flash Control Component
   - DF : Data Flash Control Component
   - CF : Code Flash Control Component

3 Kind of device
   - RH : RH850

4 Kind of compiler
   - X : Common to all compiler

5 Version number
   - (e.g.) E11001A : Tentative version (e.g. E1.10.01A)
   - (e.g.) V11001 : Release version (e.g. V1.10.01)

This version information is stored in a file (r_rfd_common_api.c) defining the API of the Common Flash Control Component in RFD28F.

This version information is stored in a file (r_rfd_df_api.c) defining the API of the Data Flash Control Component in RFD28F.

This version information is stored in a file (r_rfd_cf_api.c) defining the API of the Code Flash Control Component in RFD28F.

These versions information is allocated in .R_RFD_ROSDATA_VERSION or .R_RFD_RODATA_VERSION and can be acquired by reading the address of that section.

3.3 Configurable Software

3.3.1 Timeout value

In "R_RFD_ForcedStopAndErrorClear", set the timeout value in case the long-time processing does not return due to sticking of register value etc. The timeout value is defined by #define in the header file "r_rfd_config.h". The definition is as follows.

- `#define R_RFD_VALUE_FORCED_STOP_TIMEOUT` *`CountValue`*

Specifies the *CountValue* as timeout count value. The timeout count value is the loop count of the C language for loop. The assembler instructions for loop processing are as follows for example.

Figure 3-1 Process for timeout count

```
.L432:
ld.w    sdaoff23(_bu4_CMN_FaciBaseAddress)[gp],ep
sld.w   128[ep],r19
shr     16,r19
cmovl   0,r10,r10
bl      .L362
loop    r29,.L432
.L362:
```

Although the processing speed changes slightly depending on the pipeline control state etc., please decide the value to specify as the timeout count according to the number of instructions and set frequency.

Reference value : When the timeout value is 300, 30.3 us (@RH850/U2A16 400MHz) at the shortest on theoretical.

The default value (the setting value of the configuration file included in the package) is 300.

3.3.2 Erasure suspend processing mode

If the P/E suspend command is issued while the erasing process is in progress, it decides whether to give priority to the suspend process or erase process. The setting is defined with #define in the header file "r_rfd_config.h". The definition is as follows.

・ `#define R_RFD_ERASURE_SUSPENDED_MODE` *MODE*

The value that can be specified as *MODE* is as follows.

Setting value	value	Description
R_RFD_SUSPENTION_PRIORITY_MODE	0	Suspension-priority mode
R_RFD_ERASURE_PRIORITY_MODE	1	Erasure-priority mode

The default value (the setting value of the configuration file included in the package) is “R_RFD_SUSPENSION_PRIORITY_MODE”.

When the P/E suspend command is issued during the erasing process, all operate in the mode defined here. About detail of mode specification, refer to “RH850/U2A-EVA Flash Memory User’s Manual: Hardware” (Document No.2 on Page 4).

3.3.3 Setting value to FAEINT register

When an error occurs during flash access, set a value in the FAEINT register that determines whether FLERR interrupt is generated or not. Set values for each of the four bits (CFAEIE bit, CMDLKIE bit, DFAEIE bit, ECRCTIE bit) included in the FAEINT register. The setting is defined with #define in the header file "r_rfd_config.h". The definition is as follows.

- `#define R_RFD_REG_FAEINT_CFAEIE FLAG`
- `#define R_RFD_REG_FAEINT_CMDLKIE FLAG`
- `#define R_RFD_REG_FAEINT_DFAEIE FLAG`
- `#define R_RFD_REG_FAEINT_ECRCTIE FLAG`

The value that can be specified as *FLAG* is as follows.

Setting value	Value	Description
R_RFD_DISABLE	0	Does not generate the FLERR interrupt
R_RFD_ENABLE	1	Generate the FLERR interrupt

The default value (the setting value of the configuration file included in the package) are all “R_RFD_DISABLE”.

3.3.4 Data flash memory is control target by RFD28F ?

It defines whether to control data flash memory by RFD28F or not. This definition is defined by #define in the header file "r_rfd_config.h". The definition is as follows.

- `#define R_RFD_CONTROL_TARGET_DATAFLASH Switch`

The value that can be specified as *Switch* is as follows.

Setting value	value	Description
R_RFD_DISABLE	0	Data flash memory is NOT control target by RFD28F
R_RFD_ENABLE	1	Data flash memory is control target by RFD28F

The default value (the setting value of the configuration file included in the package) is “R_RFD_ENABLE”.

If “R_RFD_DISABLE” is set in this definition and the data flash control component is tried to build, a compiler output warning and no object is created.

3.3.5 Code flash memory is control target by RFD28F ?

It defines whether to control code flash memory by RFD28F or not. This definition is defined by #define in the header file "r_rfd_config.h". The definition is as follows.

- `#define R_RFD_CONTROL_TARGET_CODEFLASH` *`Switch`*

The value that can be specified as *`Switch`* is as follows.

Setting value	value	Description
R_RFD_DISABLE	0	Code flash memory is NOT control target by RFD28F
R_RFD_ENABLE	1	Code flash memory is control target by RFD28F

The default value (setting value of the configuration file included in the package) is "R_RFD_ENABLE".

If “R_RFD_DISABLE” is set in this definition and the code flash control component is tried to build, a compiler output warning and no object is created.

3.3.6 Map mode of Code flash memory

When controlling the code flash with RFD28F, define the mapping mode of the code flash memory address. It defines whether to control code flash memory by RFD28F or not. This definition is available when the defined value of "R_RFD_CONTROL_TARGET_CODEFLASH" is "R_RFD_ENABLE". When the definition value of "R_RFD_CONTROL_TARGET_CODEFLASH" is "R_RFD_DISABLE", this definition value is ignored. This definition is defined by #define in the header file "r_rfd_config.h". The definition is as follows.

- `#define R_RFD_MAPMODE Switch`

The value that can be specified as *Switch* is as follows.

Setting value	value	Description
R_RFD_SINGLE	0	Single map mode
R_RFD_DOUBLE	1	Double map mode

The default value (the setting value of the configuration file included in the package) is "R_RFD_SINGLE".

For details of the map mode, refer to the user's manual of the hardware.

3.3.7 Exists “Block Protection Area 1” ?

Defines whether "Block Protection Area 1" exists in the device to be used. If a FACI1 exists in the device, basically "Block Protection Area 1" exists, but some devices do not exist. In that case, RFD28F will not operate to this area. This definition is available when the defined value of “R_RFD_CONTROL_TARGET_CODEFLASH” is “R_RFD_ENABLE”. When the definition value of “R_RFD_CONTROL_TARGET_CODEFLASH” is “R_RFD_DISABLE”, this definition value is ignored. This definition is defined by #define in the header file "r_rfd_config.h". The definition is as follows.

- `#define R_RFD_BLOCK_PROTECTION_AREA1 Switch`

The value that can be specified as *Switch* is as follows.

Setting value	value	Description
R_RFD_DISABLE	0	Block Protection Area 1 does not exist
R_RFD_ENABLE	1	Block Protection Area 1 exists

The default value (the setting value of the configuration file included in the package) is “R_RFD_ENABLE”.

Please refer to the user's manual of the hardware to confirm if “Block Protection Area 1” exists. It is highly recommended to set R_RFD_ENABLE when “Block Protection Area 1” exists.

3.3.8 FPMON register

"FPMON register" exists as a write protection function to code flash memory. If the FPMON register indicates write prohibition to the code flash memory, it cannot enter P/E mode.

However, some devices do not have the FPMON register. Therefore, it defines whether confirmation of FPMON register is valid or invalid.

This definition is defined by #define in the header file "r_rfd_config.h". The definition is as follows.

- `#define R_RFD_FPMON_CHECK Switch`

The value that can be specified as *Switch* is as follows

Setting value	value	Description
R_RFD_DISABLE	0	Confirmation for FPMON register is invalid
R_RFD_ENABLE	1	Confirmation for FPMON register is valid

The default value (the setting value of the configuration file included in the package) is “R_RFD_ENABLE”.

For details of the FPMON register, refer to the user's manual of the hardware. It is highly recommended to set R_RFD_ENABLE when “FPMON register” exists.

3.3.9 Hardware depended information

Set the following information dependent on the hardware. This information is defined by #define in the header file "r_rfd_config.h".

- Required definition

Definition	Description
R_RFD_NUMBER_OF_FACI	Number of FACIs operated by RFD28F. The minimum value is 1, and the maximum value is the number of FACIs in the hardware (When operating all FACIs with RFD28F, it is the total number of FACIs in the hardware). The default value (setting value of the configuration file included in the package) is 3
R_RFD_DF_END	End address of data flash memory. Specify the end address of the data flash memory to be used. For example, if you only use the data flash memory handled by FACI0, specify the end address of the data flash memory addressed by FACI0. The default value (setting value of the configuration file included in the package) is 0xFF28FFFF.
R_RFD_CF_END	End address of code flash memory. Specify the end address of the code flash memory to be used. For example, if you only use the code flash memory handled by FACI0, specify the end address of the code flash memory addressed by FACI0. When double map mode is selected, please specify the address of the back side (Invalid side). The default value (setting value of the configuration file included in the package) is 0x00FFFFFF
R_RFD_BASE_ADDRESS_DFECC	Specified the address of “Data Flash ECC Control Register”. This information used for calculating address regarding Data Flash ECC that is necessary to read the data in the Hardware Property Area that exists on the data flash. The default value (setting value of the configuration file included in the package) is 0xFFC62C00.

- Definition required when operating FACI0 by RFD28F

Definition	Description
R_RFD_BASE_ADDRESS_FACI0	Register base address of FACI0. The default value (setting value of the configuration file included in the package) is 0xFFA10000
R_RFD_CMDAREA_FACI0	FACI0 command-issuing area. The default value (setting value of the configuration file included in the package) is 0xFFA20000

Definition	Description
R_RFD_REG_ADDRESS_FHVE3_0	Address of FHVE3FP0. The default value (setting value of the configuration file included in the package) is 0xFF984800.
R_RFD_REG_ADDRESS_FHVE15_0	Address of FHVE15FP0. The default value (setting value of the configuration file included in the package) is 0xFF984804
R_RFD_DF_BASE_FACI0	Start address of data flash memory controlled by FACI0. The default value (setting value of the configuration file included in the package) is 0xFF200000.

- Definition required when operating FACI1 by RFD28F

Definition	Description
R_RFD_BASE_ADDRESS_FACI1	Register base address of FACI1. The default value (setting value of the configuration file included in the package) is 0xFFA14000.
R_RFD_CMDAREA_FACI1	FACI1 command-issuing area. The default value (setting value of the configuration file included in the package) is 0xFFA30000.
R_RFD_REG_ADDRESS_FHVE3_1	Address of FHVE3FP1. The default value (setting value of the configuration file included in the package) is 0xFF984810.
R_RFD_REG_ADDRESS_FHVE15_1	Address of FHVE15FP1. The default value (setting value of the configuration file included in the package) is 0xFF984814.
R_RFD_DF_BASE_FACI1	Start address of data flash memory controlled by FACI1. The default value (setting value of the configuration file included in the package) is 0xFF240000.

- Definition required when operating FACI2 by RFD28F

Definition	Description
R_RFD_BASE_ADDRESS_FACI2	Register base address of FACI2. The default value (setting value of the configuration file included in the package) is 0xFFA18000.
R_RFD_CMDAREA_FACI2	FACI2 command-issuing area. The default value (setting value of the configuration file included in the package) is 0xFFA40000.
R_RFD_REG_ADDRESS_FHVE3_2	Address of FHVE3FP2. The default value (setting value of the configuration file included in the package) is 0xFF984820.

Definition	Description
R_RFD_REG_ADDRESS_FHVE15_2	Address of FHVE15FP2. The default value (setting value of the configuration file included in the package) is 0xFF984824.
R_RFD_DF_BASE_FACI2	Start address of data flash memory controlled by FACI2. The default value (setting value of the configuration file included in the package) is 0xFF280000

The definitions for RH850/U2A16 for the above items are predefined in the header file "r_rfd_config.h" (`#if defined / #elif defined` specification). Appropriate definition is set by using the compile option -D and specifying "-D__RH850_U2A__" depending on the hardware to be used. If want to customize it yourself, you not specified device name. then #else line will be valid and you can modify.

The setting that are predefined in the "r_rfd_config.h" is based on the premise that all the FACIs in each hardware are controlled by RFD28F. For example, if you want to control only FACI1 with RFD28F on hardware with two FACIs (FACI0, FACI1), you should be defined as follows. Be sure to set the address of the data flash to be controlled by RFD28F in continuous area. For example, when FACI0 and FACI2 are controlled, RFD28F interprets that the operation target Data Flash of FACI1 operates with FACI0.

```
#define  R_RFD_NUMBER_OF_FACI          1
#define  R_RFD_BASE_ADDRESS_FACI1      base_address_FACI1
#define  R_RFD_CMDAREA_FACI1           cmdarea_FACI1
#define  R_RFD_DF_BASE_FACI1           DF_start_address_FACI1
#define  R_RFD_DF_END                  DF_end_address
```

For "R_RFD_NUMBER_OF_FACI", the number of FACIs controlled by RFD28F is specified. Also, only the items for FACI1 and the common items (R_RFD_DF_END) are defined for the remaining items.

To set the above definition, either replace the `#if defined / #elif defined` setting below for each device or set the above definition under `#else` and change the compile option.

In addition, internal of RFD28F, in order to understand which FACI is defined and which FACI is undefined, the following definition is made from the above definition contents.

Definition	Description
R_RFD_VALID_FACI0	Definition of FACI0 is valid or invalid. In case of valid is TRUE, in case of invalid is FALSE.
R_RFD_VALID_FACI1	Definition of FACI1 is valid or invalid. In case of valid is TRUE, in case of invalid is FALSE.
R_RFD_VALID_FACI2	Definition of FACI2 is valid or invalid. In case of valid is TRUE, in case of invalid is FALSE.

3.4 Error Handling List

Processing for errors is not performed in RFD28F, but if an error occurs, RFD28F is returned as a return value. The list of return values is as follows.

No	Return Value	Summary
1	R_RFD_OK	Normal end. It is returned when there is no error or it is not in an abnormal state.
2	R_RFD_STS_ERASE_SUSPENDED	Erasure Suspended state. This is one of the returns values of the function of acquiring FACI state.
3	R_RFD_STS_WRITE_SUSPENDED	Programming Suspension state. This is one of the returns values of the function of acquiring FACI state.
4	R_RFD_STS_READY	Ready state. It is returned in not working FACI sequencer.
5	R_RFD_STS_BUSY	Busy state. It is returned if transition to P/E mode has not been done by confirming the transition to P/E mode, and if FACI sequencer is in progress.
6	R_RFD_ERR_ERASE	Erase Error state. This is one of the returns values of the function of acquiring FACI state.
7	R_RFD_ERR_WRITE	Write Error state. This is one of the returns values of the function of acquiring FACI state.
8	R_RFD_ERR_INTERNAL_HW	Internal Hardware Error. It is returned when the initialization of RFD28F is not done normally, or when the mode of flash memory is not changed to read mode normally, or when the process of forced termination is not done normally.
9	R_RFD_ERR_INTERNAL_DATA	Internal Data Error. This is one of the returns values of the function of acquiring FACI state. Check the SECDTCT bit, BPLDTCT bit, CFGDTCT bit and TBLDTCT bit of the FSTATR_n register. About detail of this error, refer to “RH850/U2A-EVA Flash Memory User’s Manual: Hardware” (Document No.2 on Page 4).
10	R_RFD_ERR_FHVE_PROTECT	FHVE Protect Error This error needs forced stop command for release command lock state.
11	R_RFD_ERR_ACCESS_ENV	Illegal Access Error. Check the ILGLERR bit of the FSTATR_n register. About detail of this error, refer to “RH850/U2A-EVA Flash Memory User’s Manual: Hardware” (Document No.2 on Page 4).
12	R_RFD_ERR_REJECT	Reject error. When requesting suspending process, it is returned when the request cannot be accepted.
13	R_RFD_ERR_NOFACI	Target FACI error. If a nonexistent FACI is specified in the API that specifies the target FACI as a parameter, return it.
14	R_RFD_ERR_FACI_DEFINITION	Error of the number of FACI. This error is returned if "the number of FACI in the hardware" and "the definition of FACI number" defined in the configuration do not match. For example, this error is returned if it is only FACI0 definition (R_RFD_BASE_ADDRESS_FACI0, R_RFD_CMDAREA_FACI0, R_RFD_DF_BASE_FACI0), even though “R_RFD_NUMBER_OF_FACI” is defined as 2.

No	Return Value	Summary
15	R_RFD_ERR_FPMON	FPMON value error. It returns when FPMON register value is 0x00.
16	R_RFD_ERR_ERASECOUNTER_DATA	ECC 2-bit error occurred in Erase counter area. It is returned when an ECC 2-bit error occurs when FACI reads the Erase counter area. - When FACI reads the erase counter area when the Area Erasure command is executed - When resuming Block Erasure / Area Erasure command that was interrupted by Programming / Erasure resumption command
17	R_RFD_ERR_SECURITY_DATA	ECC 2-bit error occurred in Security Setting Area. It is returned when an ECC 2-bit error occurs when FACI reads the Security Setting Area in “Property Program”, “Tag Update” and “Tag Erase” command processing.
18	R_RFD_ERR_BLOCKPROTECTION0_DATA R_RFD_ERR_BLOCKPROTECTION1_DATA	ECC 2-bit error occurred in Block Protection Area. It is returned when an ECC 2-bit error occurs when FACI reads the Block Protection Area in “Program (Code Flash/Extended Data Area)”, “Erase (Code Flash/Extended Data Area)”,” Multi Program (Extended Data Area)” ,”Multi Erase (Extended Data Area)”,”Property Program”, “Tag Update” and “Tag Erase” command processing. In the case of occurring error at the block protection Area0, “R_RFD_ERR_BLOCKPROTECTION0_DATA” will be returned, in the case of occurring error at the block protection Area1, “R_RFD_ERR_BLOCKPROTECTION1_DATA” will be returned.
19	R_RFD_ERR_CONFIGURATION_DATA	ECC 2-bit error occurred in Configuration Setting Area. It is returned when an ECC 2-bit error occurs when FACI reads the Configuration Setting Area in “Property Program”, “Tag Update” and “Tag Erase” command processing.
20	R_RFD_ERR_SWITCH_DATA	ECC 2-bit error occurred in Switch Area and TAG. It is returned when an ECC 2-bit error occurs when FACI reads the Switch Area and TAG in “Switch Program”,” Switch Erase”, “Tag Update” and “Tag Erase” command processing.
21	R_RFD_WARN_SWITCH_DATA	ECC 1-bit error occurred in Switch Area and TAG and it was corrected. It is returned when an ECC 1-bit error has been occurred and corrected when FACI has read the Switch Area and TAG in “Switch Program”,” Switch Erase”,” Tag Update” and “Tag Erase” command processing.
22	R_RFD_WARN_SECURITY_DATA	ECC 1-bit error occurred in Security Setting Area and it was corrected. It is returned when an ECC 1-bit error occurs when FACI reads the Security Setting Area in “Property Program”,” Tag Update”, “Tag Update” and “Tag Erase” command processing.

No	Return Value	Summary
23	R_RFD_WARN_BLOCKPROTECTION0_DATA R_RFD_WARN_BLOCKPROTECTION1_DATA	ECC 1-bit error occurred in Block Protection Area and it was corrected. It is returned when an ECC 1-bit error occurs when FACI reads the Block Protection Area in “Program (Code Flash/Extended Data Area)”, “Erase (Code Flash/Extended Data Area)”,” Multi Program (Extended Data Area)”,”Multi Erase (Extended Data Area)”,”Property Program”, “Tag Update” and “Tag Erase” command processing. In the case of occurring error at the block protection Area0, “R_RFD_WARN_BLOCKPROTECTION0_DATA” will be returned, in the case of occurring error at the block protection Area1, “R_RFD_WARN_BLOCKPROTECTION1_DATA” will be returned.
24	R_RFD_WARN_CONFIGURATION_DATA	ECC 1-bit error occurred in Configuration Setting Area and it was corrected. It is returned when an ECC 1-bit error occurs when FACI reads the Configuration Setting Area in “Property Program”,” Tag Update”, “Tag Update” and “Tag Erase” command processing.
25	R_RFD_WARN_ERASECOUNTER_DATA	ECC 1-bit error occurred in Erase counter area. It is returned when an ECC 1-bit error has been occurred and corrected when FACI reads the Erase counter area. - When FACI reads the erase counter area when the Area Erasure command is executed - When resuming Block Erasure / Area Erasure command that was interrupted by Programming / Erasure resumption command
26	R_RFD_WARN_INTERNAL_DATA	Internal Data Warning. This is one of the return values of the function of acquiring FACI state. This warning occurs if the TBLCRCT bit in the FSTATR register is valid. About detail, refer to “User’s Manual: Hardware Interface”
27	R_RFD_STS_NOTBLANK	Not Blank. It is returned if corresponding area is not blank during blank check.
28	R_RFD_ERR_IDAUTH	Failed ID authentication
29	R_RFD_ERR_FLOW	Flow Error. It is returned when blank check command is not executed at blank check right before.

3.5 API Reference

The functional specification (interface specification) of each subcomponent is as follows.

3.5.1 Common Flash Control Component

3.5.1.1 R_RFD_Init

・ Information

Syntax	`T_u4_RFDReturn R_RFD_Init (T_u2 i_u2_FaciFreqency);`	
Sync/Async	Sync	
Reentrancy	Non-Reentrant	
Parameters (IN)	T_u2	FACI Frequency. For RH850/U2A16, specified 0xffff as this value.
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	T_u4_RFDReturn	R_RFD_OK
		R_RFD_ERR_INTERNAL_HW
		R_RFD_ERR_FACI_DEFINITION
Description	Initialize RFD28F.	
Preconditions	-	
Remarks	-	

・ Detail Specification

Function1 Initialize RFD28F. Initialize contents are as follows.

- Set the parameter “i_u2_FaciFreqency” as the FCU clock value
- Initialize registers that require initialization

Function2 Check whether "Number of FACI operated by RFD28F (`R_RFD_NUMBER_OF_FACI`)" and "Number of definitions per FACI number (`R_RFD_BASE_ADDRESS_FACIx`, `R_RFD_CMDAREA_FACIx`, `R_RFD_DF_BASE_FACIx`)" set in the configuration match. If it does not match, “R_RFD_ERR_FACI_DEFINITION” is returned as a return value.

Function3 If the flash memory has already been in P/E mode before executing this API, execute “R_RFD_ForcedStopAndErrorClear” and then execute “R_RFD_ShiftToReadMode” internally to change the mode of flash memory from P/E mode to read mode. For detail, refer to “R_RFD_ForcedStopAndErrorClear”.

This check is performed for all FACIs operated by RFD28F.

Function4 Store the value of R_RFD_VALUE_FAEINT in the FAEINT register.

This check is performed for all FACIs operated by RFD28F.

Function5 When using RH850/U2A16, the frequency is fixed, so please specify 0xffff for this parameter.

Function6 When all the processing performed by this API is completed normally, “R_RFD_OK” is returned as a return value.

Function7 If the function “R_RFD_ForcedStopAndErrorClear” times out because of internally executing the “R_RFD_ForcedStopAndErrorClear”, “R_RFD_ERR_INTERNAL_HW” is returned as a return value.

Function8 As a result of executing "R_RFD_ShiftToReadMode" internally, if the return value is other than R_RFD_OK, "R_RFD_ERR_INTERNAL_HW" is returned as the return value.

・ Return Value

R_RFD_OK		
Macro number	:	0x00000000
meaning	:	Normal End. In Detail, refer to 3.4 Error Handling List.
treatment	:	No need.
R_RFD_ERR_INTERNAL_HW		
Macro number	:	0x00001005
meaning	:	Internal Hardware Error or Time out. In Detail, refer to 3.4 Error Handling List.
treatment	:	Retry from reset. If it occurs permanently, should correct configuration parameter (Time out value) or exchange device.
R_RFD_ERR_FACI_DEFINITION		
Macro number	:	0x00000102
meaning	:	Wrong configuration parameter (Hardware depended information). In Detail, refer to 3.4 Error Handling List.
treatment	:	Should correct configuration parameter (Hardware depended information).

3.5.1.2 R_RFD_ShiftToReadMode

・ Information

Syntax	`T_u4_RFDReturn R_RFD_ShiftToReadMode (T_u2_FACI i_u2_TargetFaci);`	
Sync/Async	Sync	
Reentrancy	Non-Reentrant	
Parameters (IN)	`T_u2_FACI`	Target FACI to be shifted to read mode.
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	`T_u4_RFDReturn`	R_RFD_OK
		R_RFD_ERR_INTERNAL_HW
		R_RFD_ERR_NOFACI
Description	Shift the flash memory mode in the target FACI to read mode.	
Preconditions	Flash state must be READY state..	
Remarks	-	

・ Detail Specification

Function1 Change the flash memory mode to the read mode in the target FACI specified by the parameter “i_u2_TargetFaci”. This processing is performed by operating the FENTRYR_n register. The value can be specified as parameter is as follows.

Value	Meanings
R_RFD_FACI0	The operation target is FACI0
R_RFD_FACI1	The operation target is FACI1
R_RFD_FACI2	The operation target is FACI2

For details about FACI, please refer to "3.6.1 About FACI".

Function2 When the parameter is specified a nonexistent FACI as a parameter “i_u2_TargetFaci”, “R_RFD_ERR_NOFACI” is returned as a return value.

Function3 Before executing this API, please set the flash state to READY state. Whether the flash memory state is READY state can be checked by executing "R_RFD_GetFaciSequenceReady".

Function4 If the flash access state is command lock state before executing this API, command lock is released in this function.

Function5 As a result of executing this API, when the flash memory mode is changed to read mode, “R_RFD_OK” is returned as a return value.

Function6 As a result of executing this API, when the flash memory mode is not changed to read mode, R_RFD_ERR_INTERNAL_HW is returned as a return value.

Function7 When this API is executed, R_RFD_ForcedStopAndErrorClear is executed to release this in the command locked state, and if it times out, "R_RFD_ERR_INTERNAL_HW" is returned as the return value.

Function8 A precondition of this API is that Initialization of RFD28F (R_RFD_Init) has been executed in advance and has completed normally.

- Return Value

R_RFD_OK		
Macro number	:	0x00000000
meaning	:	Normal End. In Detail, refer to 3.4 Error Handling List.
treatment	:	No need.
R_RFD_ERR_INTERNAL_HW		
Macro number	:	0x00001005
meaning	:	Internal Hardware Error or Time out. In Detail, refer to 3.4 Error Handling List.
treatment	:	Retry from reset. If it occurs permanently, should correct configuration parameter (Time out value) or exchange device.
R_RFD_ERR_NOFACI		
Macro number	:	0x00000101
meaning	:	Target FACI Error. In Detail, refer to 3.4 Error Handling List.
treatment	:	Should correct 1st parameter (FACI number) or configuration parameter (Hardware depended information).

3.5.1.3 R_RFD_ShiftToPEMode

・ Information

Syntax	`T_u4_RFDReturn R_RFD_ShiftToPEMode (T_u2_FACI i_u2_TargetFaci, T_en_FACIMode i_en_FaciMode);`	
Sync/Async	Sync	
Reentrancy	Non-Reentrant	
Parameters (IN)	`T_u2_FACI`	Target FACI to be checked the mode of flash memory.
	`T_en_FACIMode`	Target flash memory of shifting to P/E mode.
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	`T_u4_RFDReturn`	R_RFD_OK
		R_RFD_ERR_NOFACI
		R_RFD_STS_BUSY
Description	Shift the target memory mode in the target FACI to P/E mode.	
Preconditions	Flash state must be READY state..	
Remarks	-	

・ Detail Specification

Function1 Change the target memory mode specified by second parameter “i_en_FaciMode” in the target FACI specified by the first parameter “i_u2_TargetFaci” to the P/E mode. This processing is performed by operating the FENTRYR_n register. The value can be specified as the first and second parameter is as follows.

Value for the first parameter	Meanings
R_RFD_FACI0	The operation target is FACI0
R_RFD_FACI1	The operation target is FACI1
R_RFD_FACI2	The operation target is FACI2
Value for the second parameter	**Meanings**
R_RFD_MODE_DFPE	Change data flash memory to P/E mode
R_RFD_MODE_CFPE	Change code flash memory to P/E mode

For details about FACI, please refer to "3.6.1 About FACI".

Function2 When the parameter is specified a nonexistent FACI as the first parameter “i_u2_TargetFaci”, “R_RFD_ERR_NOFACI” is returned as a return value.

Function3 The second parameter “i_en_FaciMode” is assumed to be the correct value. If an illegal value is specified, the subsequent operation is indeterminate.

Function4 Before executing this API, please set the flash state to READY state. Whether the flash memory state is READY state can be checked by executing "R_RFD_GetFaciSequenceReady".

Function5 A precondition of this API is that Initialization of RFD28F (R_RFD_Init) has been executed in advance and has completed normally.

Function6 If the data flash memory is not in P/E mode, operate the FENTRYR_n register and set the target memory specified by the second parameter "i_en_FaciMode" to P/E mode. After that, it reads the FENTRYR_n register twice and returns R_RFD_OK if it is the same value as the second parameter "i_en_FaciMode” and returns R_RFD_STS_BUSY if it is a different value.

Function7 Whether the value is reflected to the FENTRYR_n register is checked by repeatedly executing "R_RFD_CheckPEMode" after executing this API, and check the return value of "R_RFD_CheckPEMode" is R_RFD_OK.

Function8 If the data flash memory is already in P/E mode, “R_RFD_OK” is returned as a return value without operating the FENTRYR_n register.

・ Return Value

R_RFD_OK		
Macro number	:	0x00000000
meaning	:	Normal End. In Detail, refer to 3.4 Error Handling List.
treatment	:	No need.
R_RFD_ERR_NOFACI		
Macro number	:	0x00000101
meaning	:	Target FACI Error. In Detail, refer to 3.4 Error Handling List.
treatment	:	Should correct 1st parameter (FACI number) or configuration parameter (Hardware depended information).
R_RFD_STS_BUSY		
Macro number	:	0x00000002
meaning	:	Transition to P/E mode has not been done. In Detail, refer to 3.4 Error Handling List.
treatment	:	Execute "R_RFD_CheckPEMode”.

3.5.1.4 R_RFD_CheckPEMode

・ Information

Syntax	`T_u4_RFDReturn R_RFD_CheckPEMode (T_u2_FACI i_u2_TargetFaci, T_en_FACIMode i_en_FaciMode);`	
Sync/Async	Sync	
Reentrancy	Non-Reentrant	
Parameters (IN)	`T_u2_FACI`	Target FACI to be checked the mode of flash memory.
	`T_en_FACIMode`	The mode to check (P/E mode or read mode). It checks whether the current mode of flash memory is the mode specified by this parameter.
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	`T_u4_RFDReturn`	R_RFD_OK
		R_RFD_STS_BUSY
		R_RFD_ERR_NOFACI
Description	Check whether the code flash or Data Flash mode of the target FACI is the mode specified by the second parameter.	
Preconditions	Flash state must be READY state..	
Remarks	-	

・ Detail Specification

Function1 It checks whether the target FACI specified by the first parameter "i_u2_TargetFaci" is the mode (P/E mode or read mode) specified by the second parameter "i_en_FaciMode". This processing is performed by checking the value of FENTRYR_n register. The value can be specified as the first and second parameter is as follows.

Value for the first parameter	Meanings
R_RFD_FACI0	The operation target is FACI0
R_RFD_FACI1	The operation target is FACI1
R_RFD_FACI2	The operation target is FACI2
Value for the second parameter	**Meanings**
R_RFD_MODE_READ	Check whether data flash memory is read mode
R_RFD_MODE_DFPE	Check whether data flash memory is P/E mode
R_RFD_MODE_CFPE	Check whether code flash memory is P/E mode

For details about FACI, please refer to "3.6.1 About FACI".

Function2 When the parameter is specified a nonexistent FACI as the first parameter “i_u2_TargetFaci”, “R_RFD_ERR_NOFACI” is returned as a return value.

Function3 The second parameter “i_en_FaciMode” is assumed to be the correct value. If an illegal value is specified, the subsequent operation is indeterminate.

Function4 Before executing this API, please set the flash state to READY state. Whether the flash memory state is READY state can be checked by executing "R_RFD_GetFaciSequenceReady".

Function5 If FACI mode is the mode specified by the second parameter “i_en_FaciMode”, R_RFD_OK is returned as the return value.

Function6 If FACI mode is not the mode specified by the second parameter “i_en_FaciMode”, R_RFD_STS_BUSY is returned as the return value.

Function7 A precondition of this API is that Initialization of RFD28F (R_RFD_Init) has been executed in advance and has completed normally.

・ Return Value

R_RFD_OK		
	Macro number	: 0x00000000
	meaning	: Transition to expect mode has been done. In Detail, refer to 3.4 Error Handling List.
	treatment	: No need.
R_RFD_ERR_NOFACI		
	Macro number	: 0x00000101
	meaning	: Target FACI Error. In Detail, refer to 3.4 Error Handling List.
	treatment	: Should correct 1st parameter (FACI number) or configuration parameter (Hardware depended information).
R_RFD_STS_BUSY		
	Macro number	: 0x00000002
	meaning	: Transition to expect mode has not been done. In Detail, refer to 3.4 Error Handling List.
	treatment	: Execute "R_RFD_CheckPEMode”.

3.5.1.5 R_RFD_ForcedStopAndErrorClear

- Information

Syntax	`T_u4_RFDReturn R_RFD_ForcedStopAndErrorClear (T_u2_FACI i_u2_TargetFaci);`	
Sync/Async	Sync	
Reentrancy	Non-Reentrant	
Parameters (IN)	`T_u2_FACI`	Target FACI to be stopped operation forcibly.
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	`T_u4_RFDReturn`	R_RFD_OK
		R_RFD_ERR_INTERNAL_HW
		R_RFD_ERR_NOFACI
Description	Forcibly stop the operation for the target memory in the target FACI.	
Preconditions	FACI mode must be Code Flash P/E mode or Data Flash P/E mode.	
Remarks	-	

- Detail Specification

Function1 Forcibly stop the operation for the target memory in the target FACI specified by the parameter “i_u2_TargetFaci”. Issue the “Forced Stop Command” of FACI. The value can be specified as parameter is as follows.

Value	Meanings
R_RFD_FACI0	The operation target is FACI0
R_RFD_FACI1	The operation target is FACI1
R_RFD_FACI2	The operation target is FACI2

For details about FACI, please refer to "3.6.1 About FACI".

Function2 When the parameter is specified a nonexistent FACI as a parameter “i_u2_TargetFaci”, “R_RFD_ERR_NOFACI” is returned as a return value.

Function3 Forced termination processing is performed after reading the FACI command issue area and setting it to the command lock state intentionally.

Function4 When the process is completed normally, “R_RFD_OK” is returned as the return value.

Function5 If the processing is not completed within the time set as the timeout value, it is judged timeout and the operation to the flash memory is forcibly terminated (by reading the command issuing area, the status is changed to command lock state and the sequencer process is stopped) and “R_RFD_ERR_INTERNAL_HW” is returned as the return value. For the timeout value, refer to "2.1.8 Configurable Software".

Function6 Before executing this API, it is assumed that FACI mode is P/E mode. If executed in read mode, the flash access status shifts to the command lock status.

Function7 A precondition of this API is that Initialization of RFD28F (R_RFD_Init) has been executed in advance and has completed normally.

・ Return Value

R_RFD_OK		
	Macro number	: 0x00000000
	meaning	: Normal End. In Detail, refer to 3.4 Error Handling List.
	treatment	: No need.
R_RFD_ERR_INTERNAL_HW		
	Macro number	: 0x00001005
	meaning	: Timeout. In Detail, refer to 3.4 Error Handling List.
	treatment	: Retry from reset. If it occurs permanently, should correct configuration parameter (Time out value) or exchange device.
R_RFD_ERR_NOFACI		
	Macro number	: 0x00000101
	meaning	: Target FACI Error. In Detail, refer to 3.4 Error Handling List.
	treatment	: Should correct 1st parameter (FACI number) or configuration parameter (Hardware depended information).

3.5.1.6 R_RFD_StatusClear

・ Information

Syntax	`T_u4_RFDReturn R_RFD_StatusClear (T_u2_FACI i_u2_TargetFaci);`	
Sync/Async	Sync	
Reentrancy	Non-Reentrant	
Parameters (IN)	`T_u2_FACI`	The state of target FACI to be cleared.
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	`T_u4_RFDReturn`	R_RFD_OK
		R_RFD_ERR_NOFACI
Description	Clear the state of the target FACI.	
Preconditions	Flash state must be READY state..	
Remarks	-	

・ Detail Specification

Function1 Clear the state of the target FACI specified by the parameter “i_u2_TargetFaci”. Issue the “Status Clear Command” of FACI. The value can be specified as parameter is as follows.

Value	Meanings
R_RFD_FACI0	The operation target is FACI0
R_RFD_FACI1	The operation target is FACI1
R_RFD_FACI2	The operation target is FACI2

For details about FACI, please refer to "3.6.1 About FACI".

Function2 When the parameter is specified a nonexistent FACI as a parameter “i_u2_TargetFaci”, “R_RFD_ERR_NOFACI” is returned as a return value.

Function3 Return “R_RFD_OK” as return value other than the situation of Function2.

Function4 Before executing this API, please set the flash state to READY state. Whether the flash memory state is READY state can be checked by executing "R_RFD_GetFaciSequenceReady".

Function5 Before executing this API, it is assumed that FACI mode is P/E mode. If executed in read mode, the flash access status shifts to the command lock status.

Function6 A precondition of this API is that Initialization of RFD28F (R_RFD_Init) has been executed in advance and has completed normally.

- Return Value

R_RFD_OK		
	Macro number	: 0x00000000
	meaning	: Normal End. In Detail, refer to 3.4 Error Handling List.
	treatment	: No need.
R_RFD_ERR_NOFACI		
	Macro number	: 0x00000101
	meaning	: Target FACI Error. In Detail, refer to 3.4 Error Handling List.
	treatment	: Should correct 1st parameter (FACI number) or configuration parameter (Hardware depended information).

3.5.1.7 R_RFD_GetFaciSequenceReady

・ Information

Syntax	`T_u4_RFDReturn R_RFD_GetFaciSequenceReady (T_u2_FACI i_u2_TargetFaci);`	
Sync/Async	Sync	
Reentrancy	Non-Reentrant	
Parameters (IN)	T_u2_FACI	Target FACI to be checked whether the state is in ready state.
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	T_u4_RFDReturn	R_RFD_STS_READY
		R_RFD_STS_BUSY
		R_RFD_ERR_NOFACI
Description	Check whether the target FACI is in ready state.	
Preconditions	N/A	
Remarks	-	

・ Detail Specification

Function1 It checks whether the target FACI specified by the parameter “i_u2_TargetFaci” is in Ready state. It is judged by the state of the FRDY bit of the FSTATR_n register. The value can be specified as parameter is as follows.

Value	Meanings
R_RFD_FACI0	The operation target is FACI0
R_RFD_FACI1	The operation target is FACI1
R_RFD_FACI2	The operation target is FACI2

For details about FACI, please refer to "3.6.1 About FACI".

Function2 When the parameter is specified a nonexistent FACI as a parameter “i_u2_TargetFaci”, “R_RFD_ERR_NOFACI” is returned as a return value.

Function3 If the FRDY bit in the FSTATR_n register is valid, “R_RFD_STS_READY” is returned as the return value.

Function4 If the FRDY bit in the FSTATR_n register is invalid, “R_RFD_STS_BUSY” is returned as the return value.

Function5 A precondition of this API is that Initialization of RFD28F (R_RFD_Init) has been executed in advance and has completed normally.

- Return Value

R_RFD_STS_READY		
	Macro number	: 0x00000001
	meaning	: Ready State. In Detail, refer to 3.4 Error Handling List.
	treatment	: No need.
R_RFD_STS_BUSY		
	Macro number	: 0x00000002
	meaning	: Busy State. In Detail, refer to 3.4 Error Handling List.
	treatment	: Executes this function until return Ready State. If it is judged to be timeout, execute “R_RFD_ForcedStopAndErrorClear”.
R_RFD_ERR_NOFACI		
	Macro number	: 0x00000101
	meaning	: Target FACI Error. In Detail, refer to 3.4 Error Handling List.
	treatment	: Should correct 1st parameter (FACI number) or configuration parameter (Hardware depended information).

3.5.1.8 R_RFD_GetFaciStatus

- Information

Syntax	`T_u4_RFDReturn R_RFD_GetFaciStatus ( T_u2_FACI i_u2_TargetFaci);`	
Sync/Async	Sync	
Reentrancy	Non-Reentrant	
Parameters (IN)	`T_u2_FACI`	Target FACI to be acquired the state.
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	`T_u4_RFDReturn`	R_RFD_OK
		R_RFD_ERR_ACCESS_ENV
		R_RFD_ERR_BLOCKPROTECTION0_DATA
		R_RFD_ERR_BLOCKPROTECTION1_DATA
		R_RFD_ERR_CONFIGURATION_DATA
		R_RFD_ERR_ERASE
		R_RFD_ERR_ERASECOUNTER_DATA
		R_RFD_ERR_FHVE_PROTECT
		R_RFD_ERR_INTERNAL_DATA
		R_RFD_ERR_INTERNAL_HW
		R_RFD_ERR_NOFACI
		R_RFD_ERR_SECURITY_DATA
		R_RFD_ERR_SWITCH_DATA
		R_RFD_ERR_WRITE
		R_RFD_STS_ERASE_SUSPENDED
		R_RFD_STS_WRITE_SUSPENDED
		R_RFD_WARN_BLOCKPROTECTION0_DATA
		R_RFD_WARN_BLOCKPROTECTION1_DATA
		R_RFD_WARN_CONFIGURATION_DATA
		R_RFD_WARN_ERASECOUNTER_DATA
		R_RFD_WARN_INTERNAL_DATA
		R_RFD_WARN_SECURITY_DATA
		R_RFD_WARN_SWITCH_DATA
Description	Acquire the state of the target FACI.	
Preconditions	-	
Remarks	-	

・ Detail Specification

Function1 Acquire the state of the target FACI specified by the parameter “i_u2_TargetFaci”. It is judged by the state of the CMDLK bit in the FASTAT_n register and the FSTATR_n register. The value can be specified as parameter is as follows.

Value	Meanings
R_RFD_FACI0	The operation target is FACI0
R_RFD_FACI1	The operation target is FACI1
R_RFD_FACI2	The operation target is FACI2

For details about FACI, please refer to "3.6.1 About FACI".

Function2 When the parameter is specified a nonexistent FACI as a parameter “i_u2_TargetFaci”, “R_RFD_ERR_NOFACI” is returned as a return value.

Function3 In the following state, “R_RFD_ERR_FHVE_PROTECT” is returned as return value.

Register	Bit	Status
FASTAT_n	CMDLK	Valid
FSTATR_n	FHVEERR	Valid

Function4 In the following state, “R_RFD_ERR_INTERNAL_DATA” is returned as return value.

Register	Bit	Status
FASTAT_n	CMDLK	Valid
FSTATR_n	TBLDTCT	Valid

Function5 In the following state, “R_RFD_ERR_SECURITY_DATA” is returned as return value.

Register	Bit	Status
FASTAT_n	CMDLK	Valid
FSTATR_n	SECDTCT	Valid

Function6 In the following state and the target is block protection area0, “R_RFD_ERR_BLOCKPROTECTION0_DATA” is returned as return value, and the target is block protection area1, “R_RFD_ERR_BLOCKPROTECTION1_DATA” is returned as return value.

Register	Bit	Status
FASTAT_n	CMDLK	Valid
FSTATR_n	BPLDTCT	Valid

Function7 In the following state, “R_RFD_ERR_CONFIGURATION_DATA” is returned as return value.

Register	Bit	Status
FASTAT_n	CMDLK	Valid
FSTATR_n	CFGDTCT	Valid

Function8 In the following state, “R_RFD_ERR_SWITCH_DATA” is returned as return value.

Register	Bit	Status
FASTAT_n	CMDLK	Valid
FSTATR_n	SWTDTCT	Valid

Function9 In the following state, “R_RFD_ERR_ERASECOUNTER_DATA” is returned as return value.

Register	Bit	Status
FASTAT_n	CMDLK	Valid
FSTATR_n	ERCDTCT	Valid

Function10 In the following state, “R_RFD_ERR_ACCESS_ENV” is returned as return value.

Register	Bit	Status
FASTAT_n	CMDLK	Valid
FSTATR_n	ILGLERR	Valid

Function11 In the following state, “R_RFD_ERR_ERASE” is returned as return value.

Register	Bit	Status
FASTAT_n	CMDLK	Valid
FSTATR_n	ERSERR	Valid

Function12 In the following state, “R_RFD_ERR_WRITE” is returned as return value.

Register	Bit	Status
FASTAT_n	CMDLK	Valid
FSTATR_n	PRGERR	Valid

Function13 If the CMDLK bit of the FASTAT_n register is valid but the bit status of the FSTATR_n register is not within the above description, "R_RFD_ERR_INTERNAL_HW" is returned as return value.

Function14 In the following state, “R_RFD_WARN_SWITCH_DATA” is returned as return value. This is 1-bit error detection, but it is treated as a warning because it is corrected after error detection.

Register	Bit	Status
FASTAT_n	CMDLK	Invalid
FSTATR_n	SWTCRCT	Valid

Function15 In the following state, “R_RFD_WARN_SECURITY_DATA” is returned as return value. This is 1-bit error detection, but it is treated as a warning because it is corrected after error detection.

Register	Bit	Status
FASTAT_n	CMDLK	Invalid
FSTATR_n	SECCRCT	Valid

Function16 In the following state and the target is block protection area0, “R_RFD_WARN_BLOCKPROTECTION0_DATA” is returned as return value, and the target is block protection area1, “R_RFD_WARN_BLOCKPROTECTION1_DATA” is returned as return value. This is 1-bit error detection, but it is treated as a warning because it is corrected after error detection.

Register	Bit	Status
FASTAT_n	CMDLK	Invalid
FSTATR_n	BPLCRCT	Valid

Function17 In the following state, “R_RFD_WARN_CONFIGURATION_DATA” is returned as return value. This is 1-bit error detection, but it is treated as a warning because it is corrected after error detection.

Register	Bit	Status
FASTAT_n	CMDLK	Invalid
FSTATR_n	CFGCRCT	Valid

Function18 In the following state, “R_RFD_WARN_ERASECOUNTER_DATA” is returned as return value. This is 1-bit error detection, but it is treated as a warning because it is corrected after error detection.

Register	Bit	Status
FASTAT_n	CMDLK	Invalid
FSTATR_n	ERCCRCT	Valid

Function19 In the following state, “R_RFD_WARN_INTERNAL_DATA” is returned as return value. This is 1-bit error detection, but it is treated as a warning because it is corrected after error detection.

Register	Bit	Status
FASTAT_n	CMDLK	Invalid
FSTATR_n	TBLCRCT	Valid

Function20 In the following state, “R_RFD_STS_ERASE_SUSPENDED” is returned as return value.

Register	Bit	Status
FASTAT_n	CMDLK	Invalid
FSTATR_n	ERSSPD	Valid
	PRGSPD	Invalid

Function21 In the following state, “R_RFD_STS_WRITE_SUSPENDED” is returned as return value.

Register	Bit	Status
FASTAT_n	CMDLK	Invalid
FSTATR_n	ERSSPD	Invalid
	PRGSPD	Valid

Function22 In the following state, “R_RFD_OK” is returned as return value.

Register	Bit	Status
FASTAT_n	CMDLK	Invalid
FSTATR_n	ERSSPD	Invalid
	PRGSPD	Invalid

Function23 About the priority of error detection when the CMDLK bit is valid, Function3 is the highest priority, and decreases sequentially from Function4 onwards, and Function22 is the lowest priority.

Function24 A precondition of this API is that Initialization of RFD28F (R_RFD_Init) has been executed in advance and has completed normally.

- Return Value

R_RFD_OK	
Macro number :	0x00000000
meaning :	No Error. In Detail, refer to 3.4 Error Handling List.
treatment :	No need.
R_RFD_ERR_ACCESS_ENV	
Macro number :	0x00001001
meaning :	Illegal Access Error. In Detail, refer to 3.4 Error Handling List.
treatment :	It may be wrong to use. should review executing condition. Execute “R_RFD_StatusClear” or “R_RFD_ForcedStopAndErrorClear” and re-erase block where error occurred.
R_RFD_ERR_BLOCKPROTECTION0_DATA	
Macro number :	0x00001009
meaning :	ECC 2-bit error occurred in Block Protection Area0. In Detail, refer to 3.4 Error Handling List.
treatment :	Execute “R_RFD_StatusClear” or “R_RFD_ForcedStopAndErrorClear” and re-erase and re-write the Block Protection Area0.
R_RFD_ERR_BLOCKPROTECTION1_DATA	
Macro number :	0x0000100a
meaning :	ECC 2-bit error occurred in Block Protection Area1. In Detail, refer to 3.4 Error Handling List.
treatment :	Execute “R_RFD_StatusClear” or “R_RFD_ForcedStopAndErrorClear” and re-erase and re-write the Block Protection Area1.

R_RFD_ERR_CONFIGURATION_DATA	
Macro number	0x0000100b
meaning	ECC 2-bit error occurred in Configuration Setting Area. In Detail, refer to 3.4 Error Handling List.
treatment	Execute “R_RFD_StatusClear” or “R_RFD_ForcedStopAndErrorClear” and re-erase and re-write the Configuration Setting Area.

R_RFD_ERR_ERASE	
Macro number	0x00001003
meaning	Erase Error State. In Detail, refer to 3.4 Error Handling List.
treatment	Execute “R_RFD_StatusClear” or “R_RFD_ForcedStopAndErrorClear” and re-erase block where error occurred.

R_RFD_ERR_ERASECOUNTER_DATA	
Macro number	0x00001006
meaning	ECC 2-bit error occurred in Erase counter area. In Detail, refer to 3.4 Error Handling List.
treatment	Execute “R_RFD_StatusClear” or “R_RFD_ForcedStopAndErrorClear”.

R_RFD_ERR_FHVE_PROTECT	
Macro number	0x00001000
meaning	FHVE Protect Error. In Detail, refer to 3.4 Error Handling List.
treatment	Execute “R_RFD_StatusClear” or “R_RFD_ForcedStopAndErrorClear” and re-erase block where error occurred.

R_RFD_ERR_INTERNAL_DATA	
Macro number	0x00001002
meaning	Internal Data Error. In Detail, refer to 3.4 Error Handling List.
treatment	Retry from reset. If it occurs permanently, should re-update Flash Extra Area or exchange device.

R_RFD_ERR_INTERNAL_HW	
Macro number	0x00001005
meaning	Internal Hardware Error. In Detail, refer to 3.4 Error Handling List.
treatment	Retry from reset. If it occurs permanently, should exchange device.

R_RFD_ERR_NOFACI	
Macro number	0x00000101
meaning	Target FACI Error. In Detail, refer to 3.4 Error Handling List.
treatment	Should correct 1st parameter (FACI number) or configuration parameter (Hardware depended information).

R_RFD_ERR_SECURITY_DATA	
Macro number	0x00001008
meaning	ECC 2-bit error occurred in Security Setting Area. In Detail, refer to 3.4 Error Handling List.
treatment	Execute “R_RFD_StatusClear” or “R_RFD_ForcedStopAndErrorClear” and re-erase and re-write the Security Setting Area.

R_RFD_ERR_SWITCH_DATA	
Macro number	0x00001007
meaning	ECC 2-bit error occurred in Switch Area and TAG. In Detail, refer to 3.4 Error Handling List.
treatment	Execute “R_RFD_StatusClear” or “R_RFD_ForcedStopAndErrorClear” and re-erase and re-write the Switch Area and/or TAG area.

R_RFD_ERR_WRITE	
Macro number	0x00001004
meaning	Write Error State. In Detail, refer to 3.4 Error Handling List.
treatment	Execute “R_RFD_StatusClear” or “R_RFD_ForcedStopAndErrorClear” and re-erase block where error occurred.

R_RFD_STS_ERASE_SUSPENDED	
Macro number	0x00000010
meaning	Erasure Suspended State. In Detail, refer to 3.4 Error Handling List.
treatment	No need.

R_RFD_STS_WRITE_SUSPENDED	
Macro number	0x00000011
meaning	Programming Suspension State. In Detail, refer to 3.4 Error Handling List.
treatment	No need.

R_RFD_WARN_BLOCKPROTECTION0_DATA	
Macro number	0x00000006
meaning	ECC 1-bit error occurred in Block Protection Area0 and it was corrected. In Detail, refer to 3.4 Error Handling List.
treatment	Execute “R_RFD_StatusClear” or “R_RFD_ForcedStopAndErrorClear”. If necessary, re-erase and re-write the Block Protection Area0.

R_RFD_WARN_BLOCKPROTECTION1_DATA	
Macro number	0x00000007
meaning	ECC 1-bit error occurred in Block Protection Area1 and it was corrected. In Detail, refer to 3.4 Error Handling List.
treatment	Execute “R_RFD_StatusClear” or “R_RFD_ForcedStopAndErrorClear”. If necessary, re-erase and re-write the Block Protection Area1.

R_RFD_WARN_CONFIGURATION_DATA	
Macro number	0x00000008
meaning	ECC 1-bit error occurred in Configuration Setting Area and it was corrected. In Detail, refer to 3.4 Error Handling List.
treatment	Execute “R_RFD_StatusClear” or “R_RFD_ForcedStopAndErrorClear”. If necessary, re-erase and re-write the Configuration Setting Area.

R_RFD_WARN_ERASECOUNTER_DATA	
Macro number	0x00000009
meaning	ECC 1-bit error occurred in Erase counter area. In Detail, refer to 3.4 Error Handling List.
treatment	Execute “R_RFD_StatusClear” or “R_RFD_ForcedStopAndErrorClear”.

R_RFD_WARN_INTERNAL_DATA	
Macro number	0x0000000a
meaning	Internal Data Warning. In Detail, refer to 3.4 Error Handling List.
treatment	Execute “R_RFD_StatusClear” or “R_RFD_ForcedStopAndErrorClear” or exchange device.

R_RFD_WARN_SECURITY_DATA	
Macro number	0x00000005
meaning	ECC 1-bit error occurred in Security Setting Area and it was corrected. In Detail, refer to 3.4 Error Handling List.
treatment	Execute “R_RFD_StatusClear” or “R_RFD_ForcedStopAndErrorClear”. If necessary, re-erase and re-write the Security Setting Area.

R_RFD_WARN_SWITCH_DATA	
Macro number	0x00000004
meaning	ECC 1-bit error occurred in Switch Area and TAG and it was corrected. In Detail, refer to 3.4 Error Handling List.
treatment	Execute “R_RFD_StatusClear” or “R_RFD_ForcedStopAndErrorClear”. If necessary, re-erase and re-write the Switch Area and/or TAG area.

3.5.1.9 R_RFD_SuspendPERequest

・ Information

Syntax	`T_u4_RFDReturn R_RFD_SuspendPERequest (T_u2_FACI i_u2_TargetFaci);`	
Sync/Async	Async	
Reentrancy	Non-Reentrant	
Parameters (IN)	`T_u2_FACI`	Target FACI to be suspended the process.
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	`T_u4_RFDReturn`	R_RFD_OK
		R_RFD_STS_BUSY
		R_RFD_ERR_REJECT
		R_RFD_ERR_NOFACI
Description	Check whether suspension of processing of the target FACI is accepted.	
Preconditions	FACI mode must be Code Flash P/E mode or Data Flash P/E mode. Flash access state must not be Command Lock state.	
Remarks	-	

・ Detail Specification

Function1 Check whether suspension of processing of FACI specified by the parameter “i_u2_TargetFaci” is accepted. It is judged by the state of the SUSRDY bit of the FSTATR_n register value. The value can be specified as parameter is as follows.

Value	Meanings
R_RFD_FACI0	The operation target is FACI0
R_RFD_FACI1	The operation target is FACI1
R_RFD_FACI2	The operation target is FACI2

For details about FACI, please refer to "3.6.1 About FACI".

Function2 When the parameter is specified a nonexistent FACI as a parameter “i_u2_TargetFaci”, “R_RFD_ERR_NOFACI” is returned as a return value.

Function3 If the FRDY bit of the FSTATR_n register is valid at this point, it does not suspend processing and returns "R_RFD_OK" as a return value.

Function4 If the SUSRDY bit of the FSTATR_n register value is invalid, "R_RFD_ERR_REJECT" is returned as return value.

Function5 If the SUSRDY bit of the FSTATR_n register value is valid, issue the “P/E Suspension Command” of FACI and "R_RFD_STS_BUSY" is returned as return value.

Function6 Before executing this API, it is assumed that FACI mode is P/E mode. If executed in read mode, the flash access status shifts to the command lock status.

Function7 Before executing this API, it is assumed that flash access state is not command lock state. If it is in the command locked state, clear the command lock state of the target FACI or forcibly terminate the operation on the target FACI's flash memory beforehand.

Function8 After executing this API, there is a possibility that it is still being suspended. So, repeat issuing the "R_RFD_GetFaciSequenceReady" until the suspension process is completed. After checking the completion of the suspension process, issue the "R_RFD_GetFaciStatus" and check that there are no errors, and then proceed to the next process.

Function9 After checking whether the suspension of processing is accepted, the processing that you attempted to suspend may have been completed before issuing the "P/E Suspension Command" of FACI. In this case, "R_RFD_STS_BUSY" is returned as the return value.

Function10 A precondition of this API is that Initialization of RFD28F (R_RFD_Init) has been executed in advance and has completed normally.

- Return Value

R_RFD_OK		
	Macro number	: 0x00000000
	meaning	: Already done operation. In Detail, refer to 3.4 Error Handling List.
	treatment	: No need.
R_RFD_STS_BUSY		
	Macro number	: 0x00000002
	meaning	: Issue the “P/E Suspension Command”. In Detail, refer to 3.4 Error Handling List.
	treatment	: No need.
R_RFD_ERR_REJECT		
	Macro number	: 0x00000100
	meaning	: Reject Error. do not meet the command acceptance condition. In Detail, refer to 3.4 Error Handling List.
	treatment	: Retry after a while.
R_RFD_ERR_NOFACI		
	Macro number	: 0x00000101
	meaning	: Target FACI Error. In Detail, refer to 3.4 Error Handling List.
	treatment	: Should correct 1st parameter (FACI number) or configuration parameter (Hardware depended information).

3.5.1.10 R_RFD_ResumePERequest

R_RFD_ResumePERequest

- Information

Syntax	`T_u4_RFDReturn R_RFD_ResumePERequest (T_u2_FACI i_u2_TargetFaci);`	
Sync/Async	Async	
Reentrancy	Non-Reentrant	
Parameters (IN)	`T_u2_FACI`	Target FACI to be resumed the suspension process.
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	`T_u4_RFDReturn`	R_RFD_OK
		R_RFD_ERR_NOFACI
Description	Resume the suspension process for the target FACI.	
Preconditions	Flash state must be READY state.. Flash access state must not be Command Lock state. FACI mode must be Code Flash P/E mode or Data Flash P/E mode.	
Remarks	-	

- Detail Specification

Function1 Resume the suspension process of FACI specified by the parameter “i_u2_TargetFaci”. Issue "P/E Resumption Command" of FACI. The value can be specified as parameter is as follows.

Value	Meanings
R_RFD_FACI0	The operation target is FACI0
R_RFD_FACI1	The operation target is FACI1
R_RFD_FACI2	The operation target is FACI2

For details about FACI, please refer to "3.6.1 About FACI".

Function2 When the parameter is specified a nonexistent FACI as a parameter “i_u2_TargetFaci”, “R_RFD_ERR_NOFACI” is returned as a return value.

Function3 Issue "P/E Resume Command" of FACI and "R_RFD_OK" is returned as a return value.

Function4 Before this API execution, it is assumed that FACI processing is suspended. If this API is executed under other conditions, the flash access status shifts to the command lock status. Whether the FACI process is under suspension state can be checked by executing "R_RFD_GetFaciStatus".

Function5 Before executing this API, it is assumed that FACI mode is P/E mode. If executed in read mode, the flash access status shifts to the command lock status.

Function6 Before executing this API, it is assumed that flash access state is not command lock state. If it is in the command locked state, clear the command lock state of the target FACI or forcibly terminate the operation on the target FACI's flash memory beforehand.

Function7 Before executing this API, please set the flash memory state to READY state. Whether the flash memory state is READY state can be checked by executing "R_RFD_GetFaciSequenceReady".

Function8 After executing this API, there is a possibility that resumed processing may be in progress. So, repeat issuing the "R_RFD_GetFaciSequenceReady" until the resumption is completed. After checking the completion of the resumption, issue the "R_RFD_GetFaciStatus" and check that there are no errors, and then proceed to the next process.

Function9 A precondition of this API is that Initialization of RFD28F (R_RFD_Init) has been executed in advance and has completed normally.

・ Return Value

R_RFD_OK		
Macro number	:	0x00000000
meaning	:	Issue the "P/E Resume Command". In Detail, refer to 3.4 Error Handling List.
treatment	:	No need.

R_RFD_ERR_NOFACI		
Macro number	:	0x00000101
meaning	:	Target FACI Error. In Detail, refer to 3.4 Error Handling List.
treatment	:	Should correct 1st parameter (FACI number) or configuration parameter (Hardware depended information).

3.5.1.11 R_RFD_SetFHVE

・ Information

Syntax	`T_u4_RFDReturn R_RFD_SetFHVE (T_u2_FACI i_u2_TargetFaci, T_en_Protect i_en_ProtectMode);`	
Sync/Async	Sync	
Reentrancy	Non-Reentrant	
Parameters (IN)	`T_u2_FACI`	Target FACI to be set to FHVE register
	`T_en_Protect`	Specifying to enable or disable for writing, erasure and blank checking to the flash memory.
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	`T_u4_RFDReturn`	R_RFD_OK
		R_RFD_ERR_NOFACI
Description	Enable or disable writing, erasure and blank checking to the flash memory.	
Preconditions	Flash state must be READY state..	
Remarks	-	

・ Detail Specification

Function1 For the target FACI specified by the first parameter "i_u2_TargetFaci", set the enabled / disabled state of writing / erasure / blank checking for flash memory to the one specified by the parameter "i_en_ProtectMode".

Value for the first parameter	Meanings
R_RFD_FACI0	The operation target is FACI0
R_RFD_FACI1	The operation target is FACI1
R_RFD_FACI2	The operation target is FACI2
Value for the second parameter	**Meanings**
R_RFD_FHVE_PROTECT_ON	Programming / erasure / DataFlashBlankChecking disabled
R_RFD_FHVE_PROTECT_OFF	Programming / erasure / DataFlashBlankChecking enabled

Function2 It is realized by operating FHVE3 register and FHVE15 register. The relationship between the compatible device and the corresponding register is as follows.

Device	FACI Number	FHVE register for operating
RH850/U2A16	FACI0	FHVE3FP0, FHVE15FP0
	FACI1	FHVE3FP1, FHVE15FP1
	FACI2	FHVE3FP2, FHVE15FP2

Function3 When the parameter is specified a nonexistent FACI as the first parameter “i_u2_TargetFaci”, “R_RFD_ERR_NOFACI” is returned as a return value.

Function4 The parameter “i_en_ProtectMode” is assumed to be the correct value. If an illegal value is specified, the subsequent operation is indeterminate.

Function5 After controlling the FHVE3 (FHVE3FPn) register and FHVE15 (FHVE15FPn) register, read the value of FHVE15 (FHVE15FPn) as dummy and execute the _SYNCP function to synchronize.

Function6 Before executing this API, please set the flash state to READY state. Whether the flash memory state is READY state can be checked by executing "R_RFD_GetFaciSequenceReady".

Function7 A precondition of this API is that Initialization of RFD28F (R_RFD_Init) has been executed in advance and has completed normally.

Function8 If register setting is normally completed, R_RFD_OK is returned as the return value.

・ Return Value

R_RFD_OK	
Macro number	: 0x00000000
meaning	: Normal End. In Detail, refer to 3.4 Error Handling List.
treatment	: No need.

R_RFD_ERR_NOFACI	
Macro number	: 0x00000101
meaning	: Target FACI Error. In Detail, refer to 3.4 Error Handling List.
treatment	: Should correct 1st parameter (FACI number) or configuration parameter (Hardware depended information).

3.5.1.12 R_RFD_HOOK_EnterCriticalSection

・ Information

Syntax	`void R_RFD_HOOK_EnterCriticalSection (void)`	
Sync/Async	Sync	
Reentrancy	Non-Reentrant	
Parameters (IN)	N/A	
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	N/A	
Description	Starts the critical section.	
Preconditions	-	
Remarks	The contents of the critical section starting includes execution of an interrupt disable instruction. This API is cut out as user own part, so, it can be customized.	

・ Detail Specification

Function1 To use an interrupt-disabled section, call this function as a hook function at the beginning of the section.

Function2 The content of this hook function depends on the user application. For example, if you simply want to disable interrupts, describe the DI function (di instruction) and return from this hook function.

If you already have a function or the like to start controlling critical sections in your application, you can also call the corresponding function in this hook function.

Please describe the processing according to the application.

Function3 There is no return value.

3.5.1.13 R_RFD_HOOK_ExitCriticalSection

・ Information

Syntax	`void R_RFD_HOOK_ExitCriticalSection (void)`	
Sync/Async	Sync	
Reentrancy	Non-Reentrant	
Parameters (IN)	N/A	
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	N/A	
Description	Exits the critical section.	
Preconditions	-	
Remarks	The contents of the critical section exiting includes execution of an interrupt enable instruction. This API is cut out as user own part, so, it can be customized.	

・ Detail Specification

Function1 To use an interrupt-disabled section, call this function as a hook function at the end of the section.

Function2 The content of this hook function depends on the user application. For example, if you simply want to enable interrupts and return, describe the EI function (ei instruction) and return from this hook function.

If you already have a function or the like to finish controlling critical sections in your application, you can also call the corresponding function in this hook function.

Please describe the processing according to the application.

Function3 There is no return value.

3.5.2 Data Flash Control Component

3.5.2.1 R_RFD_DFIDAuth

・ Information

Syntax	`T_u4_RFDReturn R_RFD_DFIDAuth (T_u1* i_pu1_AuthIDData);`	
Sync/Async	Sync	
Reentrancy	Non-Reentrant	
Parameters (IN)	`T_u1*`	Pointer to Data flash ID data for authentication.
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	`T_u4_RFDReturn`	R_RFD_OK
		R_RFD_ERR_IDAUTH
Description	Authenticate the data flash ID.	
Preconditions	-	
Remarks	-	

・ Detail Specification

Function1 Perform the Data flash ID authentication. Specify the pointer storing ID data to be authenticated with the parameter and check whether it matches the data flash ID.

After ID authentication, refer to the IDST register values to check whether authentication has succeeded or failed. if successful ,refer to Function2. If unsuccessful, refer to Function3.

Function2 If authentication succeeds, “R_RFD_OK” is returned as a return value.

Function3 If authentication fails, “R_RFD_ERR_IDAUTH” is returned as a return value.

Function4 A precondition of this API is that Initialization of RFD28F (R_RFD_Init) has been executed in advance and has completed normally.

・ Return Value

`R_RFD_OK`		
`Macro number`	:	`0x00000000`
`meaning`	:	Normal End. In Detail, refer to 3.4 Error Handling List.
`treatment`	:	No need.

R_RFD_ERR_IDAUTH		
Macro number	:	0x00000103
meaning	:	Failed ID Authentication. In Detail, refer to 3.4 Error Handling List.
treatment	:	Should pass the correct Data Flash ID. Please check the ID to be authenticated. (Consider the endian too).

3.5.2.2 R_RFD_EraseDFRequest

- Information

Syntax	`void R_RFD_EraseDFRequest (T_u4_RfdAddress i_u4_Start, T_u4_RfdAddress i_u4_End);`	
Sync/Async	Async	
Reentrancy	Non-Reentrant	
Parameters (IN)	`T_u4_RfdAddress`	Start address for erasure area.
	`T_u4_RfdAddress`	End address for erasure area.
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	N/A	
Description	Erase the contents of the data flash memory.	
Preconditions	Flash state must be READY state.. Flash access state must not be Command Lock state. FACI mode must be Data Flash P/E mode.	
Remarks	-	

- Detail Specification

Function1 Erases the contents of the data flash memory from the address specified by the first parameter “i_u4_Start” to the address specified by the second parameter “i_u4_End”. Issue the “Area Erasure Command” of FACI.

Function2 It is assumed that the value of the first parameter “i_u4_Start” is smaller than the value of the second parameter “i_u4_End”. If this magnitude relation is reversed, the flash access status shifts to the command lock status.

Function3 It is assumed that the value of the first parameter “i_u4_Start” and the value of the second parameter “i_u4_End” are both within the range of the data flash memory. If a value outside the range is specified, the subsequent operation is indeterminate.

Function4 The value of the first parameter “i_u4_Start” and the value of the second parameter “i_u4_End” must specify the block boundary of the data flash memory. If the block boundary is not specified, the subsequent operation is indeterminate.

Function5 The value of the first parameter “i_u4_Start” and the value of the second parameter “i_u4_End” must be in the same Data Area. If they are not in the same Data Area, in other words, if the specified range crosses the Data Area, the subsequent operation is indeterminate. For details about Data Area, please refer to "3.6.2 About Data Area".

Function6 The FACI number to operate is judged from the address specified by the parameter "i_u4_Start" and the value specified by "R_RFD_DF_BASE_FACIn" and "R_RFD_DF_END" in the configuration file (r_rfd_config.h). If the FACI number to be operated cannot be decided, this API terminates without performing the erasing process.

Function7 Before executing this API, please set the flash state to READY state. Whether the flash memory state is READY state can be checked by executing "R_RFD_GetFaciSequenceReady" in the Common Flash Control Component.

Function8 Before executing this API, it is assumed that FACI mode is P/E mode. If executed in read mode, the flash access status shifts to the command lock status.

Function9 Before executing this API, it is assumed that flash access state is not command lock state. If it is in the command locked state, clear the command lock state of the target FACI or forcibly terminate the operation on the target FACI's flash memory beforehand.

Function10 The FACI number to operate is judged from the address specified by the first parameter "i_u4_Start". This decision is made by executing the “R_RFD_DFAddressToFaciNumber” function.

Function11 After executing this API, there is a possibility that it is still being erased. So, repeat issuing the "R_RFD_GetFaciSequenceReady" in the Common Flash Control Component until the erasure process is completed. After checking the completion of the erasure process, issue the "R_RFD_GetFaciStatus" in the Common Flash Control Component and check that there are no errors, and then proceed to the next process.

Function12 There is no return value.

Function13 Before issuing the "Area Erasure Command" of FACI, set "Erase-Suspended Mode" which is the processing mode when the P/E suspend command is issued during erasing processing. In Suspension-priority mode, set 0 to FCPSR_n register, in erase priority mode, set 1 to FCPSR_n register (setting value by the configuration).

Function14 A precondition of this API is that Initialization of RFD28F (R_RFD_Init) has been executed in advance and has completed normally.

- Return Value

  No return values.

3.5.2.3 R_RFD_WriteDFRequest

・ Information

Syntax	`void R_RFD_WriteDFRequest (T_u4_RfdAddress i_u4_Start, T_u2 i_u2_Length, T_pu4_RfdBuffer i_pu4_Data);`	
Sync/Async	Async	
Reentrancy	Non-Reentrant	
Parameters (IN)	`T_u4_RfdAddress`	Start address for writing area.
	`T_u2`	Data for writing length.
	`T_pu4_RfdBuffer`	Buffer address where write data is stored.
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	N/A	
Description	Write the data to the data flash memory.	
Preconditions	Flash state must be READY state.. Flash access state must not be Command Lock state. FACI mode must be Data Flash P/E mode.	
Remarks	-	

・ Detail Specification

Function1 Writes the data stored in the third parameter "i_pu4_Data" from the address indicated by the first parameter “i_u4_Start” by the amount of size indicated by the second parameter “i_u2_Length”.

Function2 The second parameter “i_u2_Length” can be one of the following values. If other values are specified, process is quit without doing anything.

- 4 (bytes)
- 8 (bytes)
- 16 (bytes)
- 32 (bytes)
- 64 (bytes)
- 128 (bytes)

Function3 Processing is the FACI command "Programming Command" when “4” is specified as the second parameter “i_u2_Length”, and processing is the FACI command "Multi Programming Command" when “8”, “16”, ”32”, ”64” or ”128” is specified as the second parameter “i_u2_Length”.

Function4 It is assumed that the value of the first parameter “i_u4_Start” and added the value of the second parameter “i_u2_Length” to it is within the data flash memory. If a value outside the range is specified, the subsequent operation is indeterminate.

Function5 The value of the first parameter “i_u4_Start” must be aligned with the value of the second parameter “i_u2_Length”. If it is not aligned, the subsequent operation is indeterminate.

Function6 The value of the first parameter “i_u4_Start” and added the value of the second parameter “i_u2_Length” to it must be in the same Data Area. If they are not in the same Data Area, in other words, if the written range crosses the Data Area, the subsequent operation is indeterminate. For details about Data Area, please refer to “3.6.2 About Data Area”.

Function7 The FACI number to operate is judged from the address specified by the parameter "i_u4_Start" and the value specified by "R_RFD_DF_BASE_FACIn" and "R_RFD_DF_END" in the configuration file (r_rfd_config.h). If the FACI number to be operated cannot be decided, this API terminates without performing the writing process.

Function8 Before executing this API, please set the flash state to READY state. Whether the flash state is READY state can be checked by executing "R_RFD_GetFaciSequenceReady" in the Common Flash Control Component.

Function9 Before executing this API, it is assumed that FACI mode is Data Flash P/E mode. If executed in read mode, the flash access status shifts to the command lock status.

Function10 Before executing this API, it is assumed that flash access state is not command lock state. If it is in the command locked state, clear the command lock state of the target FACI or forcibly terminate the operation on the target FACI's flash memory beforehand.

Function11 The FACI number to operate is judged from the address specified by the first parameter "i_u4_Start". This decision is made by executing the “R_RFD_DFAddressToFaciNumber” function.

Function12 After executing this API, there is a possibility that it is still being written. So, repeat issuing the "R_RFD_GetFaciSequenceReady" in the Common Flash Control Component until the writing process is completed. After checking the completion of the writing process, issue the "R_RFD_GetFaciStatus" in the Common Flash Control Component and check that there are no errors, and then proceed to the next process.

Function13 There is no return value.

Function14 A precondition of this API is that Initialization of RFD28F (R_RFD_Init) has been executed in advance and has completed normally.

- Return Value

  No return values.

3.5.2.4 R_RFD_BlankCheckDFRequest

- Information

Syntax	`void R_RFD_BlankCheckDFRequest(T_u4_RfdAddress i_u4_Start, T_u4_RfdAddress i_u4_End);`	
Sync/Async	Async	
Reentrancy	Non-Reentrant	
Parameters (IN)	`T_u4_RfdAddress`	Start address for checking free space.
	`T_u4_RfdAddress`	End address for checking free space.
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	N/A	
Description	Check whether there is free space in the data flash memory.	
Preconditions	Flash state must be READY state.. Flash access state must not be Command Lock state. FACI mode must be Data Flash P/E mode.	
Remarks	-	

- Detail Specification

Function1 It checks from the address specified by the first parameter “i_u4_Start” to the address specified by the second parameter “i_u4_End” whether there is free space in the data flash memory. Issue the “Blank Check Command” of FACI.

Function2 The searching direction of the free space is the first parameter "i_u4_Start" to the second parameter "i_u4_End" direction.

Function3 It is assumed that the value of the first parameter “i_u4_Start” is smaller than the value of the second parameter “i_u4_End”. If this magnitude relation is reversed, the subsequent operation is indeterminate.

Function4 It is assumed that the value of the first parameter “i_u4_Start” and the value of the second parameter “i_u4_End” are both within the range of the data flash memory. If a value outside the range is specified, the subsequent operation is indeterminate.

Function5 The value of the first parameter “i_u4_Start” and the value of the second parameter “i_u4_End” must specify the boundary of the writing unit. If the boundary of the writing unit is not specified, the subsequent operation is indeterminate. For the boundary of the write unit, see the description of the second parameter "*length*" in "3.5.2.3 R_RFD_WriteDFRequest".

Function6 The value of the first parameter “i_u4_Start” and the value of the second parameter “i_u4_End” must be in the same Data Area. If they are not in the same Data Area , in other words, if the specified range crosses the Data Area , the subsequent operation is indeterminate. For details about Data Area , please refer to “3.6.2 About Data Area”.

Function7 The FACI number to operate is judged from the address specified by the parameter "i_u4_Start" and the value specified by "R_RFD_DF_BASE_FACIn" and "R_RFD_DF_END" in the configuration file (r_rfd_config.h). If the FACI number to be operated cannot be decided, this API terminates without performing the checking free space process.

Function8 Before executing this API, please set the flash state to READY state. Whether the flash state is READY state can be checked by executing "R_RFD_GetFaciSequenceReady" in the Common Flash Control Component.

Function9 Before executing this API, it is assumed that FACI mode is Data Flash P/E mode. If executed in read mode, the flash access status shifts to the command lock status.

Function10 Before executing this API, it is assumed that flash access state is not command lock state. If it is in the command locked state, clear the command lock state of the target FACI or forcibly terminate the operation on the target FACI's flash memory beforehand.

Function11 The FACI number to operate is judged from the address specified by the first parameter "i_u4_Start". This decision is made by executing the “R_RFD_DFAddressToFaciNumber” function.

Function12 After executing this API, there is a possibility that it is still being blank checked. So, repeat issuing the "R_RFD_GetFaciSequenceReady" in the Common Flash Control Component until the blank checking process is completed. After checking the completion of the blank checking process, issue the "R_RFD_GetFaciStatus" in the Common Flash Control Component and check that there are no errors, and then proceed to the next process.

Function13 There is no return value.

Function14 A precondition of this API is that Initialization of RFD28F (R_RFD_Init) has been executed in advance and has completed normally.

- Return Value

  No return values.

3.5.2.5 R_RFD_GetBlankCheckResult

・ Information

Syntax	`T_u4_RFDReturn R_RFD_GetBlankCheckResult (T_u2_FACI i_u2_TargetFaci, T_u4_RfdAddress* o_pu4_NotBlankAddress);`	
Sync/Async	Sync	
Reentrancy	Non-Reentrant	
Parameters (IN)	`T_u2_FACI`	The target FACI that acquiring the result of the free space of data flash memory.
Parameters (IN/OUT)	N/A	
Parameters (OUT)	`T_u4_RfdAddress*`	Address of the data flash memory already written that was found first.
Return Value	`T_u4_RFDReturn`	R_RFD_OK
		R_RFD_STS_NOTBLANK
		R_RFD_ERR_NOFACI
Description	Acquires the result of the last executed free space checking of data flash memory immediately before.	
Preconditions	Flash state must be READY state.. Flash access state must not be Command Lock state. Perform checking the free space of Data Flash Memory immediately before.	
Remarks	-	

・ Detail Specification

Function1 Acquires the result of the last executed free space checking of data flash memory (R_RFD_BlankCheckDFRequest). The FACI number is specified as the first parameter “i_u2_TargetFaci”. The value can be specified as first parameter is as follows.

Value	Meanings
R_RFD_FACI0	The operation target is FACI0
R_RFD_FACI1	The operation target is FACI1
R_RFD_FACI2	The operation target is FACI2

For details about FACI, please refer to "3.6.1 About FACI".

Function2 When the parameter is specified a nonexistent FACI as a parameter “i_u2_TargetFaci”, “R_RFD_ERR_NOFACI” is returned as a return value.

Function3 Before executing this API, please set the flash state to READY state. Whether the flash state is READY state can be checked by executing "R_RFD_GetFaciSequenceReady" in the Common Flash Control Component.

Function4 Before executing this API, it is assumed that flash access state is not command lock state. If it is in the command locked state, clear the command lock state of the target FACI or forcibly terminate the operation on the target FACI's flash memory and retry processing from the blank check processing (R_RFD_BlankCheckDFRequest).

Function5 If the already written area is found by executed just before "R_RFD_BlankCheckDFRequest", “R_RFD_STS_NOTBLANK” is returned as the return value, and the address of the written area is stored in the second parameter.

Function6 If the already written area is not found by executed just before "R_RFD_BlankCheckDFRequest", “R_RFD_OK” is returned as the return value, and the “0x0” is stored in the second parameter.

Function7 A precondition of this API is that Initialization of RFD28F (R_RFD_Init) has been executed in advance and has completed normally.

Function8 It is assumed that the second parameter "o_pu4_NotBlankAddress" is a correct value. If an invalid value is specified, subsequent operations will be undefined.

・ Return Value

R_RFD_OK	
Macro number	: 0x00000000
meaning	: Blank In Detail, refer to 3.4 Error Handling List
treatment	: No need.

R_RFD_STS_NOTBLANK	
Macro number	: 0x00000003
meaning	: Not Blank. In Detail, refer to 3.4 Error Handling List.
treatment	: No need.

R_RFD_ERR_NOFACI	
Macro number	: 0x00000101
meaning	: Target FACI Error. In Detail, refer to 3.4 Error Handling List.
treatment	: Should correct 1st parameter (FACI number) or configuration parameter (Hardware depended information).

3.5.2.6 R_RFD_DFAddressToFaciNumber

・ Information

Syntax	`T_u2_FACI R_RFD_DFAddressToFaciNumber (T_u4_RfdAddress i_u4_Address)`	
Sync/Async	Sync	
Reentrancy	Non-Reentrant	
Parameters (IN)	`T_u4_RfdAddress`	Data flash memory address at which FACI number is to be acquired.
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	`T_u2_FACI`	R_RFD_FACI0
		R_RFD_FACI1
		R_RFD_FACI2
		R_RFD_INVALID_FACI
Description	Acquire the FACI number of the specified data flash memory address.	
Preconditions	-	
Remarks	If only the FACI0 and FACI2 are defined and the value specified by "i_u4_Address" is within the address range of FACI1, R_RFD_FACI0 is returned as the return value.	

・ Detail Specification

Function1 Acquires the FACI number of the data flash memory address specified by the parameter "i_u4_Address" and returns that value as the return value.

Function2 If the use of FACI0 is defined, and the use of FACI1 is defined and the use of FACI2 is defined, call the "rfd_fu2_CheckDFFaci0Faci1Faci2" function by specifying “i_u4_Address”.as a parameter.

Function3 If the use of FACI0 is defined, and the use of FACI1 is defined and the use of FACI2 is **NOT** defined, call the "rfd_fu2_CheckDFFaci0Faci1" function by specifying “i_u4_Address”.as a parameter.

Function4 If the use of FACI0 is defined, and the use of FACI1 is **NOT** defined and the use of FACI2 is defined, call the "rfd_fu2_CheckDFFaci0Faci2" function by specifying “i_u4_Address”.as a parameter.

Function5 If the use of FACI0 is defined, and the use of FACI1 is defined and the use of FACI2 is **NOT** defined, call the "rfd_fu2_CheckDFFaci0" function by specifying “i_u4_Address”.as a parameter.

Function6 If the use of FACI0 is **NOT** defined, and the use of FACI1 is defined and the use of FACI2 is defined, call the "rfd_fu2_CheckDFFaci1Faci2" function by specifying “i_u4_Address”.as a parameter.

Function7 If the use of FACI0 is **NOT** defined, and the use of FACI1 is defined and the use of FACI2 is **NOT** defined, call the "rfd_fu2_CheckDFFaci0Faci1" function by specifying “i_u4_Address”.as a parameter.

Function8 If the use of FACI0 is **NOT** defined, and the use of FACI1 is **NOT** defined and the use of FACI2 is defined, call the "rfd_fu2_CheckDFFaci2" function by specifying “i_u4_Address”.as a parameter.

Function9 If the use of FACI0 is **NOT** defined, and the use of FACI1 is **NOT** defined and the use of FACI2 is **NOT** defined, "R_RFD_INVALID_FACI" is returned as the return value of this API.

Function10 If only the FACI0 and FACI2 are defined and the value specified by "i_u4_Address" is within the address range of FACI1, R_RFD_FACI0 is returned as the return value. This is because the range of FACI1 is unknown without definition of FACI1, it is necessary to judge it as FACI0 (Do not specify the address in FACI1 when FACI1 is not used).

Function11 A precondition of this API is that Initialization of RFD28F (R_RFD_Init) has been executed in advance and has completed normally.

・ Return Value

R_RFD_FACI0	
Macro number	: 0x0000
meaning	: To be operated is FACI0.
treatment	: No need.
R_RFD_FACI1	
Macro number	: 0x0001
meaning	: To be operated is FACI1.
treatment	: No need.
R_RFD_FACI2	
Macro number	: 0x0002
meaning	: To be operated is FACI2.
treatment	: No need.
R_RFD_INVALID_FACI	
Macro number	: 0x00ff
meaning	: FACI to be operated is invalid.
treatment	: Should correct 1st parameter (FACI number) or configuration parameter (Hardware depended information).

3.5.3 Code Flash Control Component

3.5.3.1 R_RFD_IDAuth

・ Information

Syntax	T_u4_RFDReturn R_RFD_IDAuth (T_en_IDType i_en_IDType, T_u1* i_pu1_AuthIDData);	
Sync/Async	Sync	
Reentrancy	Non-Reentrant	
Parameters (IN)	T_en_IDType	The type of ID for authentication
	T_u1*	Pointer to ID data for authentication
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	T_u4_RFDReturn	R_RFD_OK
		R_RFD_ERR_IDAUTH
Description	Authenticate the ID.	
Preconditions	-	
Remarks	-	

・ Detail Specification

Function1 Perform the ID authentication. For the authentication target specified by the first parameter "i_en_IDType", specify the pointer storing the ID data to be authenticated specified by the second parameter "i_pu1_AuthIDData" and check whether it matches the specified authentication ID. The value can be specified as first parameter is as follows.

Value for the first parameter	Meanings
R_RFD_ID_SPID	The authentication target is Serial Programmer ID
R_RFD_ID_DFID	The authentication target is Data Flash ID
R_RFD_ID_OCDID	The authentication target is OCD ID
R_RFD_ID_CUSTIDA	The authentication target is Customer ID A
R_RFD_ID_CUSTIDB	The authentication target is Customer ID B
R_RFD_ID_CUSTIDC	The authentication target is Customer ID C
R_RFD_ID_RHSIFID	The authentication target is RHSIF ID

After ID authentication, refer to the IDST and IDST2 register values to check whether authentication has succeeded or failed. if successful ,refer to Function3. If unsuccessful, refer to Function4.

Function2 The first parameter “i_en_IDType” is assumed to be the correct value. If an illegal value is specified, “R_RFD_ERR_IDAUTH” is returned as a return value.

Function3 If authentication succeeds, “R_RFD_OK” is returned as a return value.

Function4 If authentication fails, “R_RFD_ERR_IDAUTH” is returned as a return value.

Function5 A precondition of this API is that Initialization of RFD28F (R_RFD_Init) has been executed in advance and has completed normally.

- Return Value

R_RFD_OK		
Macro number	:	0x00000000
meaning	:	Normal End. In Detail, refer to 3.4 Error Handling List.
treatment	:	No need.
R_RFD_ERR_IDAUTH		
Macro number	:	0x00000103
meaning	:	Failed ID Authentication. In Detail, refer to 3.4 Error Handling List.
treatment	:	Should pass the correct ID. Please check the ID to be authenticated. (Consider the endian too).

3.5.3.2 R_RFD_EraseCFRequest

・ Information

Syntax	`void R_RFD_EraseCFRequest (T_u4_RfdAddress i_u4_Start);`	
Sync/Async	Async	
Reentrancy	Non-Reentrant	
Parameters (IN)	`T_u4_RfdAddress`	Start address for erasure area
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	N/A	
Description	Erase the contents of the code flash memory.	
Preconditions	Flash state must be READY state. Flash access state must not be Command Lock state. FACI mode must be Code Flash P/E mode.	
Remarks	-	

・ Detail Specification

Function1 Erases the contents of the Code Flash Memory Area from the address specified by the parameter “i_u4_Start”. Issue the “Erase Command” of FACI (The Multiple Erase Command (Multi Erase Command) is valid only for data flash memory).

Since the unit of erasure is 1 block (16Kbytes or 64Kbytes), only the start address is specified as the parameter to be specified.

Function2 It is assumed that the value of the parameter “i_u4_Start” is within the range of the Code Flash Memory Area. If a value outside the range is specified, the subsequent operation is indeterminate.

Function3 The value of the parameter “i_u4_Start” must specify the block boundary of the Code Flash Memory Area. If the block boundary is not specified, the subsequent operation is indeterminate.

Function4 Before executing this API, please set the flash state to READY state. Whether the flash state is READY state can be checked by executing "R_RFD_GetFaciSequenceReady" in the Common Flash Control Component.

Function5 Before executing this API, it is assumed that FACI mode is Code Flash P/E mode. If executed in Read mode, the flash access status shifts to the command lock status.

Function6 Before executing this API, it is assumed that flash access state is not command lock state. If it is in the command locked state, clear the command lock state of the target FACI or forcibly terminate the operation on the target FACI's flash memory beforehand.

Function7 The FACI number to operate is judged from the address specified by the parameter "i_u4_Start" and the value specified by "R_RFD_CF_BASE_FACIn" and "R_RFD_CF_END" in the configuration file (r_rfd_config.h). If the FACI number to be operated cannot be decided, this API terminates without performing the erasing process.

Function8 After executing this API, there is a possibility that it is still being erased. So, repeat issuing the "R_RFD_GetFaciSequenceReady" in the Common Flash Control Component until the erasure process is completed. After checking the completion of the erasure process, issue the "R_RFD_GetFaciStatus" in the Common Flash Control Component and check that there are no errors, and then proceed to the next process.

Function9 There is no return value.

Function10 Before issuing the "Erase Command" of FACI, set "Erase-Suspended Mode" which is the processing mode when the P/E suspend command is issued during erasing processing. In Suspension-priority mode, set 0 to FCPSR_n register, in erase priority mode, set 1 to FCPSR_n register (setting value by the configuration).

Function11 A precondition of this API is that Initialization of RFD28F (R_RFD_Init) has been executed in advance and has completed normally.

- Return Value

  No return values.

3.5.3.3 R_RFD_WriteCFRequest

・ Information

Syntax	`void R_RFD_WriteCFRequest (T_u4_RfdAddress i_u4_Start, T_pu4_RfdBuffer i_pu4_Data);`	
Sync/Async	Async	
Reentrancy	Non-Reentrant	
Parameters (IN)	`T_u4_RfdAddress`	Start address for writing area
	`T_pu4_RfdBuffer`	Buffer address where write data is stored
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	N/A	
Description	Write the data to the code flash memory.	
Preconditions	Flash state must be READY state.. Flash access state must not be Command Lock state. FACI mode must be Code Flash P/E mode.	
Remarks	-	

・ Detail Specification

Function1 Writes the data stored in the second parameter "i_pu4_Data" from the address indicated by the first parameter “i_u4_Start”. Issue the “Program Command” of FACI (Since the Multiple Program Command (Multi Program Command) is valid only for data flash memory).

Function2 The unit of writing is 512 bytes.

Function3 The value of the first parameter “i_u4_Start” must be aligned with 512 bytes. If it is not aligned, the subsequent operation is indeterminate.

Function4 The value of the first parameter “i_u4_Start” and “i_u4_Start + 512bytes” must be in the range of Code Flash Memory. If it is not in the range of Code Flash Memory, the subsequent operation is indeterminate.

Function5 The value of the first parameter “i_u4_Start” and “i_u4_Start + 512bytes” must be in the same Bank. If they are not in the same Bank, in other words, if the written range crosses the Bank, the subsequent operation is indeterminate.

Function6 Before executing this API, please set the flash state to READY state. Whether the flash state is READY state can be checked by executing "R_RFD_GetFaciSequenceReady" in the Common Flash Control Component.

Function7 Before executing this API, it is assumed that FACI mode is Code Flash P/E mode. If executed in Read mode, the flash access status shifts to the command lock status.

Function8 Before executing this API, it is assumed that flash access state is not command lock state. If it is in the command locked state, clear the command lock state of the target FACI or forcibly terminate the operation on the target FACI's flash memory beforehand.

Function9 The FACI number to operate is judged from the address specified by the parameter "i_u4_Start" and the value specified by "R_RFD_CF_BASE_FACIn" and "R_RFD_CF_END" in the configuration file (r_rfd_config.h). If the FACI number to be operated cannot be decided, this API terminates without performing the writing process.

Function10 After executing this API, there is a possibility that it is still being written. So, repeat issuing the "R_RFD_GetFaciSequenceReady" in the Common Flash Control Component until the writing process is completed. After checking the completion of the writing process, issue the "R_RFD_GetFaciStatus" in the Common Flash Control Component and check that there are no errors, and then proceed to the next process.

Function11 There is no return value.

Function12 A precondition of this API is that Initialization of RFD28F (R_RFD_Init) has been executed in advance and has completed normally.

- Return Value

  No return values.

3.5.3.4 R_RFD_ErasePropertyRequest

・ Information

Syntax	`void R_RFD_ErasePropertyRequest(T_u4_RfdAddress i_u4_Start);`	
Sync/Async	Async	
Reentrancy	Non-Reentrant	
Parameters (IN)	`T_u4_RfdAddress`	Start address for erasure of one area in the hardware property area.
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	N/A	
Description	Erases one area in the hardware property area.	
Preconditions	Flash state must be READY state.. Flash access state must not be Command Lock state. FACI mode must be Data Flash P/E mode.	
Remarks	-	

・ Detail Specification

Function1 Using Command of FACI “Property Erase Command”, erases one area from the address indicated by the first parameter “i_u4_Start”. The erase size is one block (2 Kbytes).

Function2 Follow the flow of “Property Erase Command” of FACI.

Function3 The value that can specify as the parameter “i_u4_Start” is the address in the hardware property area and the address is aligned value by 2048 bytes (2Kbytes).

Function4 For the FACI number to be operated, in the case of Configuration Setting Area, Security Setting Area, Block Protection Area for FACI0, the operation target is FACI0, Block Protection Area for FACI1, the operation target is FACI 1. If the FACI number to be operated cannot be specified, this API terminates without performing the erasing process.

Function5 Before executing this API, please set the flash state to READY state. Whether the flash state is READY state can be checked by executing "R_RFD_GetFaciSequenceReady" in the Common Flash Control Component.

Function6 Before executing this API, it is assumed that FACI mode is Data Flash P/E mode (Not Code Flash P/E mode). If executed in Read mode, the flash access status shifts to the command lock status.

Function7 Before executing this API, it is assumed that flash access state is not command lock state. If it is in the command locked state, clear the command lock state of the target FACI or forcibly terminate the operation on the target FACI's flash memory beforehand.

Function8 After executing this API, there is a possibility that it is still being erased. So, repeat issuing the "R_RFD_GetFaciSequenceReady" in the Common Flash Control Component until the erasure process is completed. After checking the completion of the erasure process, issue the "R_RFD_GetFaciStatus" in the Common Flash Control Component and check that there are no errors, and then proceed to the next process.

Function9 There is no return value.

Function10 A precondition of this API is that Initialization of RFD28F (R_RFD_Init) has been executed in advance and has completed normally.

- Return Value

  No return values.

3.5.3.5 R_RFD_WritePropertyRequest

・ Information

Syntax	`void R_RFD_WritePropertyRequest (T_u4_RfdAddress i_u4_Start, T_pu4_RfdBuffer i_pu4_Data);`	
Sync/Async	Sync	
Reentrancy	Non-Reentrant	
Parameters (IN)	`T_u4_RfdAddress`	Start address for writing area
	`T_pu4_RfdBuffer`	Buffer address where write data is stored
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	N/A	
Description	Writes 32 bytes of data to the area in the hardware property area.	
Preconditions	Flash state must be READY state.. Flash access state must not be Command Lock state. FACI mode must be Data Flash P/E mode.	
Remarks	-	

・ Detail Specification

Function1 Using Command of FACI “Property Program Command”, writes the data stored in the second parameter "i_pu4_Data" from the address in the hardware property area indicated by the first parameter “i_u4_Start”.

Function2 Follow the flow of “Property Programming Command” of FACI.

Function3 The value that can specify as the parameter “i_u4_Start” is the address in the hardware property area and the address is aligned value by 32 bytes.

Function4 For the FACI number to be operated, in the case of Configuration Setting Area, Security Setting Area, Block Protection Area for FACI0, the operation target is FACI0, Block Protection Area for FACI1, the operation target is FACI 1. If the FACI number to be operated cannot be specified, this API terminates without performing the erasing process.

Function5 Before executing this API, please set the flash state to READY state. Whether the flash state is READY state can be checked by executing "R_RFD_GetFaciSequenceReady" in the Common Flash Control Component.

Function6 Before executing this API, it is assumed that FACI mode is Data Flash P/E mode (Not Code Flash P/E mode). If executed in Read mode, the flash access status shifts to the command lock status.

Function7 Before executing this API, it is assumed that flash access state is not command lock state. If it is in the command locked state, clear the command lock state of the target FACI or forcibly terminate the operation on the target FACI's flash memory beforehand.

Function8 After executing this API, there is a possibility that it is still being written. So, repeat issuing the "R_RFD_GetFaciSequenceReady" in the Common Flash Control Component until the writing process is completed. After checking the completion of the writing process, issue the "R_RFD_GetFaciStatus" in the Common Flash Control Component and check that there are no errors, and then proceed to the next process.

Function9 There is no return value.

Function10 A precondition of this API is that Initialization of RFD28F (R_RFD_Init) has been executed in advance and has completed normally.

- Return Value

  No return values.

3.5.3.6 R_RFD_EraseSwitchRequest

・ Information

Syntax	`T_u4_RFDReturn R_RFD_EraseSwitchRequest (void);`	
Sync/Async	Async	
Reentrancy	Non-Reentrant	
Parameters (IN)	N/A	
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	T_u4_RFDReturn	R_RFD_OK
		R_RFD_ERR_SWITCH_DATA
Description	Erases the contents of Switch Area on the back side (invalid side).	
Preconditions	Flash state must be READY state.. Flash access state must not be Command Lock state. FACI mode must be Data Flash P/E mode. Tag Update command must be completed normally.	
Remarks	If there is a process of reading in the data flash memory in the application before issuing this API, detects the ECC error before issuing this API. Setting of the DFECCCTL register to enable ECC Error Detection and setting of the DFERRINT register to enable detection by interrupt of ECC error must be performed by the user application.	

・ Detail Specification

Function1 Using Command of FACI “Switch Erase Command”, erases the contents of Switch Area on the back side (invalid side).

Function2 Set R_RFD_FACI0 as the FACI number to be operated.

Function3 Clears the ECC error before erasing Switch Area. ECC error clearing is carried out by setting 0x00000001 to DFERSTC (Data Flash Error Status Clear Register). This is to check whether an ECC error occurred by reading the VAPC flag in the TAG area implemented by Function3. If ECC errors occur before this processing, it is not possible to judge whether it is an ECC error when reading the VAPC flag, so ECC error will be cleared.

Function4 Read the VAPC flag in the TAG Area. This is because if the VAPC flag is not correct, there is no confirmation that the VAF flag in the same TAG Area points to the correct Switch Area. If the Switch Area is erased in that state, we will lose the confirmation as to whether the subsequent operation is correct or not.

Function5 When reading the DFERSTR register, if the “SEDF bit” or “DEDF bit” is valid, clears the ECC error and return the “R_RFD_ERR_SWITCH_DATA” as a return value. To clear the ECC error, enable the ERRCLR bit in the DFERSTC register (Data Flash Error Status Clear Register).

Function6 The value of VAPC flag in the TAG Area which is read in Function3 is not “0x5AA5A55A”, return the “R_RFD_ERR_SWITCH_DATA” as a return value.

Function7 If an interrupt occurs during processing from Function2 to Function6, and the status of the value or flag changes in the middle, it cannot judge of processing correctly. Therefore, RFD28F will make the interval between Function2 and Function6 a critical section. Starts the critical section before Function2 and exits the critical section after Function6. Please refer to "3.5.1.12 R_RFD_HOOK_EnterCriticalSection" for the start of the critical section and "3.5.1.13 R_RFD_HOOK_ExitCriticalSection" for the end of the critical section.

Function8 Using Command of FACI “Switch Erase Command” and erases the contents of Switch Area on the back side (invalid side), return the “R_RFD_OK” as a return value.

Function9 Before executing this API, please set the flash state to READY state. Whether the flash state is READY state can be checked by executing "R_RFD_GetFaciSequenceReady" in the Common Flash Control Component.

Function10 Before executing this API, it is assumed that FACI mode is Data Flash P/E mode (Not Code Flash P/E mode). If executed in Read mode, the flash access status shifts to the command lock status.

Function11 Before executing this API, it is assumed that flash access state is not command lock state. If it is in the command locked state, clear the command lock state of the target FACI or forcibly terminate the operation on the target FACI's flash memory beforehand.

Function12 After executing this API, there is a possibility that it is still being erased. So, repeat issuing the "R_RFD_GetFaciSequenceReady" in the Common Flash Control Component until the erasure process is completed. After checking the completion of the erasure process, issue the "R_RFD_GetFaciStatus" in the Common Flash Control Component and check that there are no errors, and then proceed to the next process.

Function13 A precondition of this API is that Initialization of RFD28F (R_RFD_Init) has been executed in advance and has completed normally.

・ Return Value

R_RFD_OK		
Macro number	:	0x00000000
meaning	:	Normal End. In Detail, refer to 3.4 Error Handling List.
treatment	:	No need.

R_RFD_ERR_SWITCH_DATA		
Macro number	:	0x00001007
meaning	:	ECC 2-bit error occurred in TAG Area. In Detail, refer to 3.4 Error Handling List.
treatment	:	Re-erase and re-write the TAG Area

3.5.3.7 R_RFD_WriteSwitchRequest

- Information

Syntax	`void R_RFD_WriteSwitchRequest (T_pu4_RfdBuffer i_pu4_Data);`	
Sync/Async	Async	
Reentrancy	Non-Reentrant	
Parameters (IN)	`T_pu4_RfdBuffer`	Buffer address where write data is stored
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	N/A	
Description	Writes the data to Switch Area on the back side (invalid side).	
Preconditions	Flash state must be READY state.. Flash access state must not be Command Lock state. FACI mode must be Data Flash P/E mode.	
Remarks	-	

- Detail Specification

Function1 Using Command of FACI “Switch Program Command”, writes the 32 bytes data stored in the parameter "i_pu4_Data" to the CVA (Configuration setting Area Valid Area), SVA (Security setting Area Valid Area), BVA0 (Block protection setting area Valid Area for FPSYS0) ,BVA1 (Block protection setting area Valid Area for FPSYS1) and Reserved area on the Back side (invalid side).

Function2 Set R_RFD_FACI0 as the FACI number to be operated.

Function3 The structure of the parameter "i_pu4_Data" is as follows.

Address	Information to store
i_pu4_Data[0]	CVA (Configuration setting Area Valid Area)
i_pu4_Data[1]	SVA (Security setting Area Valid Area)
i_pu4_Data[2]	BVA0 (Block protection setting area Valid Area for FPSYS0)
i_pu4_Data[3]	BVA1 (Block protection setting area Valid Area for FPSYS1)
i_pu4_Data[4]	Reserved area (0xFFFFFFFF)
i_pu4_Data[5]	Reserved area (0xFFFFFFFF)
i_pu4_Data[6]	Reserved area (0xFFFFFFFF)
i_pu4_Data[7]	Reserved area (0xFFFFFFFF)

The value to store to CVA (i_pu4_Data[0]), SVA (i_pu4_Data[1]), BVA0 (i_pu4_Data[2]), BVA1 (i_pu4_Data[3]) are one of the follows.

Case	Value to store
The case of pointing to Area0	0xA55A5AA5
The case of pointing to Area1	0x5AA5A55A

If the target device does not have BVA1, 0xFFFFFFFF is stored in BVA1 (i_pu4_Data[3]).

Also, 0xFFFFFFFF is stored in “Reserved area” (i_pu4_Data [4], i_pu4_Data [5], i_pu4_Data [6], i_pu4_Data [7]).

Function4 Before executing this API, please set the flash state to READY state. Whether the flash state is READY state can be checked by executing "R_RFD_GetFaciSequenceReady" in the Common Flash Control Component.

Function5 Before executing this API, it is assumed that FACI mode is Data Flash P/E mode (Not Code Flash P/E mode). If executed in Read mode, the flash access status shifts to the command lock status.

Function6 Before executing this API, it is assumed that flash access state is not command lock state. If it is in the command locked state, clear the command lock state of the target FACI or forcibly terminate the operation on the target FACI's flash memory beforehand.

Function7 After executing this API, there is a possibility that it is still being written. So, repeat issuing the "R_RFD_GetFaciSequenceReady" in the Common Flash Control Component until the writing process is completed. After checking the completion of the writing process, issue the "R_RFD_GetFaciStatus" in the Common Flash Control Component and check that there are no errors, and then proceed to the next process.

Function8 There is no return value.

Function9 A precondition of this API is that Initialization of RFD28F (R_RFD_Init) has been executed in advance and has completed normally.

- Return Value

  No return values.

3.5.3.8 R_RFD_EraseTagRequest

- Information

Syntax	`T_u4_RFDReturn R_RFD_EraseTagRequest (void);`	
Sync/Async	Async	
Reentrancy	Non-Reentrant	
Parameters (IN)	N/A	
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	`T_u4_RFDReturn`	R_RFD_OK
		R_RFD_ERR_SWITCH_DATA
		R_RFD_ERR_CONFIGURATION_DATA
		R_RFD_ERR_SECURITY_DATA
		R_RFD_ERR_BLOCKPROTECTION0_DATA
		R_RFD_ERR_BLOCKPROTECTION1_DATA
Description	Erases the contents of TAG area.	
Preconditions	Flash state must be READY state.. Flash access state must not be Command Lock state. FACI mode must be Data Flash P/E mode.	
Remarks	If there is a process of reading in the data flash memory in the application before issuing this API, detects the ECC error before issuing this API. Setting of the DFECCCTL register to enable ECC Error Detection and setting of the DFERRINT register to enable detection by interrupt of ECC error must be performed by the user application.	

- Detail Specification

Function1 Using Command of FACI “Tag Erase Command”, erases the contents of Tag Area.

Before this API is issued, confirms that valid values are written in the Switch Area on the back side. This is because the device may not operate if both Switch Area and TAG Area are in the erased state.

Also, check if valid values are written on the front side of Hardware Property Area after Tag Update. This is because the device may not operate if the unwritten Hardware Property Area becomes effective

Function2 Set R_RFD_FACI0 as the FACI number to be operated.

Function3 Calls the internal checking function to confirm which described in Function1. If the return value of internal checking function is R_RFD_OK, processing of Function4 is executed subsequently, and if it is not, return this value as the return value and exit this API.

Function4 Issues “Tag Erase Command” of FACI, return “R_RFD_OK” as a return value.

Function5 Before executing this API, please set the flash state to READY state. Whether the flash state is READY state can be checked by executing "R_RFD_GetFaciSequenceReady" in the Common Flash Control Component. If Block Protection Area for FPSYS1 exists (when the value of R_RFD_BLOCK_PROTECTION_AREA1 in the configuration is R_RFD_TRUE), it is necessary to set both the FACI0 and FACI1 flash state to READY state.

Function6 Before executing this API, it is assumed that FACI mode is Data Flash P/E mode (Not Code Flash P/E mode). If executed in Read mode, the flash access status shifts to the command lock status. If Block Protection Area for FPSYS1 exists (when the value of R_RFD_BLOCK_PROTECTION_AREA1 in the configuration is R_RFD_TRUE), both the FACI0 and FACI1 modes must be set to the Data Flash P/E mode.

Function7 Before executing this API, it is assumed that flash access state is not command lock state. If it is in the command locked state, clear the command lock state of the target FACI or forcibly terminate the operation on the target FACI's flash memory beforehand.

Function8 After executing this API, there is a possibility that it is still being erased. So, repeat issuing the "R_RFD_GetFaciSequenceReady" in the Common Flash Control Component until the erasure process is completed. After checking the completion of the erasure process, issue the "R_RFD_GetFaciStatus" in the Common Flash Control Component and check that there are no errors, and then proceed to the next process.

Function9 A precondition of this API is that Initialization of RFD28F (R_RFD_Init) has been executed in advance and has completed normally.

・ Return Value

R_RFD_OK		
Macro number	:	0x00000000
meaning	:	Normal End. In Detail, refer to 3.4 Error Handling List.
treatment	:	No need.

R_RFD_ERR_SWITCH_DATA		
Macro number	:	0x00001007
meaning	:	ECC 2-bit error occurred in Switch Area. In Detail, refer to 3.4 Error Handling List.
treatment	:	Re-erase and re-write the Switch Area.

R_RFD_ERR_CONFIGURATION_DATA	
Macro number	0x0000100b
meaning	ECC 2-bit error occurred in Configuration Setting Area. In Detail, refer to 3.4 Error Handling List.
treatment	Re-erase and re-write the Configuration Setting Area.

R_RFD_ERR_SECURITY_DATA	
Macro number	0x00001008
meaning	ECC 2-bit error occurred in Security Setting Area. In Detail, refer to 3.4 Error Handling List.
treatment	Re-erase and re-write the Security Setting Area.

R_RFD_ERR_BLOCKPROTECTION0_DATA	
Macro number	0x00001009
meaning	ECC 2-bit error occurred in Block Protection Area0. In Detail, refer to 3.4 Error Handling List.
treatment	Re-erase and re-write the Block Protection Area0.

R_RFD_ERR_BLOCKPROTECTION1_DATA	
Macro number	0x0000100a
meaning	ECC 2-bit error occurred in Block Protection Area1. In Detail, refer to 3.4 Error Handling List.
treatment	Re-erase and re-write the Block Protection Area1.

3.5.3.9 R_RFD_UpdateTagRequest

- Information

Syntax	`T_u4_RFDReturn R_RFD_UpdateTagRequest (void)`	
Sync/Async	Async	
Reentrancy	Non-Reentrant	
Parameters (IN)	N/A	
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	T_u4_RFDReturn	R_RFD_OK
		R_RFD_ERR_SWITCH_DATA
		R_RFD_ERR_CONFIGURATION_DATA
		R_RFD_ERR_SECURITY_DATA
		R_RFD_ERR_BLOCKPROTECTION0_DATA
		R_RFD_ERR_BLOCKPROTECTION1_DATA
Description	Updates the contents of TAG area.	
Preconditions	Flash state must be READY state.. Flash access state must not be Command Lock state. FACI mode must be Data Flash P/E mode.	
Remarks	If there is a process of reading in the data flash memory in the application before issuing this API, detects the ECC error before issuing this API. Setting of the DFECCCTL register to enable ECC Error Detection and setting of the DFERRINT register to enable detection by interrupt of ECC error must be performed by the user application.	

- Detail Specification

Function1 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).

Before this API is issued, confirms that valid values are written in the Switch Area. This is because the device may not operate if both Switch Area and TAG Area are in the erased state.

Also, check if valid values are written on the front side of Hardware Property Area after Tag Update. This is because the device may not operate if the unwritten Hardware Property Area becomes effective

Function2 Set R_RFD_FACI0 as the FACI number to be operated.

Function3 Calls the internal checking function to confirm which described in Function1. If the return value of internal checking function is R_RFD_OK, processing of Function4 is executed subsequently, and if it is not, return this value as the return value and exit this API.

Function4 Issues “Tag Update Command” of FACI, return “R_RFD_OK” as a return value.

Function5 Before executing this API, please set the flash state to READY state. Whether the flash state is READY state can be checked by executing "R_RFD_GetFaciSequenceReady" in the Common Flash Control Component. If Block Protection Area for FPSYS1 exists (when the value of R_RFD_BLOCK_PROTECTION_AREA1 in the configuration is R_RFD_TRUE), it is necessary to set both the FACI0 and FACI1 flash state to READY state.

Function6 Before executing this API, it is assumed that FACI mode is Data Flash P/E mode (Not Code Flash P/E mode). If executed in Read mode, the flash access status shifts to the command lock status. If Block Protection Area for FPSYS1 exists (when the value of R_RFD_BLOCK_PROTECTION_AREA1 in the configuration is R_RFD_TRUE), both the FACI0 and FACI1 modes must be set to the Data Flash P/E mode.

Function7 Before executing this API, it is assumed that flash access state is not command lock state. If it is in the command locked state, clear the command lock state of the target FACI or forcibly terminate the operation on the target FACI's flash memory beforehand.

Function8 After executing this API, there is a possibility that it is still being written. So, repeat issuing the "R_RFD_GetFaciSequenceReady" in the Common Flash Control Component until the writing process is completed. After checking the completion of the writing process, issue the "R_RFD_GetFaciStatus" in the Common Flash Control Component and check that there are no errors, and then proceed to the next process.

Function9 A precondition of this API is that Initialization of RFD28F (R_RFD_Init) has been executed in advance and has completed normally.

・ Return Value

R_RFD_OK	
Macro number	: 0x00000000
meaning	: Normal End. In Detail, refer to 3.4 Error Handling List.
treatment	: No need.

R_RFD_ERR_SWITCH_DATA	
Macro number	: 0x00001007
meaning	: ECC 2-bit error occurred in Switch Area. In Detail, refer to 3.4 Error Handling List.
treatment	: Re-erase and re-write the Switch Area.

R_RFD_ERR_CONFIGURATION_DATA	
Macro number	0x0000100b
meaning	ECC 2-bit error occurred in Configuration Setting Area. In Detail, refer to 3.4 Error Handling List.
treatment	Re-erase and re-write the Configuration Setting Area.

R_RFD_ERR_SECURITY_DATA	
Macro number	0x00001008
meaning	ECC 2-bit error occurred in Security Setting Area. In Detail, refer to 3.4 Error Handling List.
treatment	Re-erase and re-write the Security Setting Area.

R_RFD_ERR_BLOCKPROTECTION0_DATA	
Macro number	0x00001009
meaning	ECC 2-bit error occurred in Block Protection Area0. In Detail, refer to 3.4 Error Handling List.
treatment	Re-erase and re-write the Block Protection Area0.

R_RFD_ERR_BLOCKPROTECTION1_DATA	
Macro number	0x0000100a
meaning	ECC 2-bit error occurred in Block Protection Area1. In Detail, refer to 3.4 Error Handling List.
treatment	Re-erase and re-write the Block Protection Area1.

3.5.3.10 R_RFD_EraseExtendedDataRequest

・ Information

Syntax	`void R_RFD_EraseExtendedDataRequest (T_u4_RfdAddress i_u4_Start)`	
Sync/Async	Async	
Reentrancy	Non-Reentrant	
Parameters (IN)	`T_u4_RfdAddress`	Start address for erasure of Extended Data Area
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	N/A	
Description	Erase the contents of the Extended Data Area.	
Preconditions	Flash state must be READY state.. Flash access state must not be Command Lock state. FACI mode must be Data Flash P/E mode.	
Remarks	-	

・ Detail Specification

Function1 Erases the contents of the Extended Data Area from the address specified by the parameter “i_u4_Start”. Issue the “Erase Command” of FACI.

Function2 It is assumed that the value of the parameter “i_u4_Start” is the start address of the Extended Data Area. If other values are specified, the subsequent operation is indeterminate.

Function3 The value of the parameter “i_u4_Start” must specify the block boundary of the Extended Data Area. If the block boundary is not specified, the subsequent operation is indeterminate.

Function4 Set R_RFD_FACI0 as the FACI number to be operated.

Function5 Before executing this API, please set the flash state to READY state. Whether the flash state is READY state can be checked by executing "R_RFD_GetFaciSequenceReady" in the Common Flash Control Component.

Function6 Before executing this API, it is assumed that FACI mode is Data Flash P/E mode (Not Code Flash P/E mode). If executed in Read mode, the flash access status shifts to the command lock status.

Function7 Before executing this API, it is assumed that flash access state is not command lock state. If it is in the command locked state, clear the command lock state of the target FACI or forcibly terminate the operation on the target FACI's flash memory beforehand.

Function8 After executing this API, there is a possibility that it is still being erased. So, repeat issuing the "R_RFD_GetFaciSequenceReady" in the Common Flash Control Component until the erasure process is completed. After checking the completion of the erasure process, issue the "R_RFD_GetFaciStatus" in the Common Flash Control Component and check that there are no errors, and then proceed to the next process.

Function9 There is no return value.

Function10 Before issuing the "Erase Command" of FACI, set "Erase-Suspended Mode" which is the processing mode when the P/E suspend command is issued during erasing processing. In Suspension-priority mode, set 0 to FCPSR_0 register, in erase priority mode, set 1 to FCPSR_0 register (setting value by the configuration).

Function11 A precondition of this API is that Initialization of RFD28F (R_RFD_Init) has been executed in advance and has completed normally.

- Return Value

  No return values.

3.5.3.11 R_RFD_WriteExtendedDataRequest

・ Information

Syntax	`void R_RFD_WriteExtendedDataRequest (T_u4_RfdAddress i_u4_Start, T_u2 i_u2_Length, T_pu4_RfdBuffer i_pu4_Data);`	
Sync/Async	Sync	
Reentrancy	Non-Reentrant	
Parameters (IN)	T_u4_RfdAddress	Start address for writing area
	T_u2	Data for writing length
	T_pu4_RfdBuffer	Buffer address where write data is stored
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	N/A	
Description	Write the data to the Extended Data Area.	
Preconditions	Flash state must be READY state.. Flash access state must not be Command Lock state. FACI mode must be Data Flash P/E mode.	
Remarks	-	

・ Detail Specification

Function1 Writes the data stored in the third parameter "i_pu4_Data" from the address indicated by the first parameter “i_u4_Start” by the amount of size indicated by the second parameter “i_u2_Length”.

Function2 The second parameter “i_u2_Length” can be one of the following values. If other values are specified, process is quit without doing anything.

- 4（bytes）
- 8（bytes）
- 16（bytes）
- 32（bytes）
- 64（bytes）
- 128（bytes）

Function3 Processing is the FACI command "Program Command" when “4” is specified as the second parameter “i_u2_Length”, and processing is the FACI command "Multi Program Command" when “8”, “16”,“32”,“64” or ”128” is specified as the second parameter “i_u2_Length”.

Function4 It is assumed that the value of the first parameter “i_u4_Start” and added the value of the second parameter “i_u2_Length” to it is within the data flash memory. If a value outside the range is specified, the subsequent operation is indeterminate.

Function5 The value of the first parameter “i_u4_Start” must be aligned with the value of the second parameter “i_u2_Length”. If it is not aligned, the subsequent operation is indeterminate.

Function6 Set R_RFD_FACI0 as the FACI number to be operated.

Function7 Before executing this API, please set the flash state to READY state. Whether the flash state is READY state can be checked by executing "R_RFD_GetFaciSequenceReady" in the Common Flash Control Component.

Function8 Before executing this API, it is assumed that FACI mode is Data Flash P/E mode (Not Code Flash P/E mode). If executed in Read mode, the flash access status shifts to the command lock status.

Function9 Before executing this API, it is assumed that flash access state is not command lock state. If it is in the command locked state, clear the command lock state of the target FACI or forcibly terminate the operation on the target FACI's flash memory beforehand.

Function10 After executing this API, there is a possibility that it is still being written. So, repeat issuing the "R_RFD_GetFaciSequenceReady" in the Common Flash Control Component until the writing process is completed. After checking the completion of the writing process, issue the "R_RFD_GetFaciStatus" in the Common Flash Control Component and check that there are no errors, and then proceed to the next process.

Function11 There is no return value.

Function12 A precondition of this API is that Initialization of RFD28F (R_RFD_Init) has been executed in advance and has completed normally.

- Return Value

  No return values.

3.5.3.12 R_RFD_CFAddressToFaciNumber

・ Information

Syntax	`T_u2_FACI R_RFD_CFAddressToFaciNumber (T_u4_RfdAddress i_u4_Address)`	
Sync/Async	Sync	
Reentrancy	Non-Reentrant	
Parameters (IN)	`T_u4_RfdAddress`	Code flash memory address at which FACI number is to be acquired
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	`T_u2_FACI`	R_RFD_FACI0
		R_RFD_FACI1
		R_RFD_FACI2
		R_RFD_INVALID_FACI
Description	Acquire the FACI number of the specified code flash memory address.	
Preconditions	-	
Remarks	If only the FACI0 and FACI2 are defined and the value specified by "i_u4_Address" is within the address range of FACI1, R_RFD_FACI0 is returned as the return value.	

・ Detail Specification

Function1 Acquires the FACI number of the code flash memory address specified by the parameter "i_u4_Address" and returns that value as the return value.

Function2 If the use of FACI0 is defined, and the use of FACI1 is defined and the use of FACI2 is defined, call the "rfd_fu2_CheckCFFaci0Faci1Faci2" function by specifying “i_u4_Address”.as a parameter.

Function3 If the use of FACI0 is defined, and the use of FACI1 is defined and the use of FACI2 is **NOT** defined, call the "rfd_fu2_CheckCFFaci0Faci1" function by specifying “i_u4_Address”.as a parameter.

Function4 If the use of FACI0 is defined, and the use of FACI1 is **NOT** defined and the use of FACI2 is defined, call the "rfd_fu2_CheckCFFaci0Faci2" function by specifying “i_u4_Address”.as a parameter.

Function5 If the use of FACI0 is defined, and the use of FACI1 is defined and the use of FACI2 is **NOT** defined, call the "rfd_fu2_CheckCFFaci0" function by specifying “i_u4_Address”.as a parameter.

Function6 If the use of FACI0 is **NOT** defined, and the use of FACI1 is defined and the use of FACI2 is defined, call the "rfd_fu2_CheckCFFaci1Faci2" function by specifying “i_u4_Address”.as a parameter.

Function7 If the use of FACI0 is **NOT** defined, and the use of FACI1 is defined and the use of FACI2 is **NOT** defined, call the "rfd_fu2_CheckCFFaci0Faci1" function by specifying “i_u4_Address”.as a parameter.

Function8 If the use of FACI0 is **NOT** defined, and the use of FACI1 is **NOT** defined and the use of FACI2 is defined, call the "rfd_fu2_CheckCFFaci2" function by specifying “i_u4_Address”.as a parameter.

Function9 If the use of FACI0 is **NOT** defined, and the use of FACI1 is **NOT** defined and the use of FACI2 is **NOT** defined, "R_RFD_INVALID_FACI" is returned as the return value of this API.

Function10 If only the FACI0 and FACI2 are defined and the value specified by "i_u4_Address" is within the address range of FACI1, R_RFD_FACI0 is returned as the return value. This is because the range of FACI1 is unknown without definition of FACI1, it is necessary to judge it as FACI0 (Do not specify the address in FACI1 when FACI1 is not used).

Function11 A precondition of this API is that Initialization of RFD28F (R_RFD_Init) has been executed in advance and has completed normally.

・ Return Value

R_RFD_FACI0	
Macro number	: 0x0000
meaning	: To be operated is FACI0.
treatment	: No need.
R_RFD_FACI1	
Macro number	: 0x0001
meaning	: To be operated is FACI1.
treatment	: No need.
R_RFD_FACI2	
Macro number	: 0x0002
meaning	: To be operated is FACI2.
treatment	: No need.
R_RFD_INVALID_FACI	
Macro number	: 0x00ff
meaning	: FACI to be operated is invalid.
treatment	: Should correct 1st parameter (FACI number) or configuration parameter (Hardware depended information).

3.5.3.13 R_RFD_PropertyAddressToFaciNumber

- Information

Syntax	`T_u2_FACI R_RFD_PropertyAddressToFaciNumber(T_u4_RfdAddress i_u4_Address)`	
ASIL Level	ASIL-D	
Status	New	
Sync/Async	Sync	
Reentrancy	Non-Reentrant	
Parameters (IN)	`T_u4_RfdAddress`	Hardware Property Area address at which FACI number is to be acquired
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	`T_u2_FACI`	R_RFD_FACI0
		R_RFD_FACI1
		R_RFD_INVALID_FACI
Description	Acquire the FACI number of the specified Hardware Property Area address.	
Preconditions	-	
Remarks		

- Detail Specification

Function1 Acquires the FACI number of the Hardware Property Area address specified by the parameter "i_u4_Address" and returns that value as the return value.

Function2 If the parameter "i_u4_Address" is an address in the Configuration Setting Area and if the use of FACI0 is defined, R_RFD_FACI0 is returned as the return value.

Function3 If the parameter "i_u4_Address" is an address in the Configuration Setting Area and if the use of FACI0 is **NOT** defined, R_RFD_INVALID_FACI is returned as the return value.

Function4 If the parameter "i_u4_Address" is an address in the Security Setting Area and if the use of FACI0 is defined, R_RFD_FACI0 is returned as the return value.

Function5 If the parameter "i_u4_Address" is an address in the Security Setting Area and if the use of FACI0 is **NOT** defined, R_RFD_INVALID_FACI is returned as the return value.

Function6 If the parameter "i_u4_Address" is an address in the Block Protection Area0 and if the use of FACI0 is defined, R_RFD_FACI0 is returned as the return value.

Function7 If the parameter "i_u4_Address" is an address in the Block Protection Area0 and if the use of FACI0 is **NOT** defined, R_RFD_INVALID_FACI is returned as the return value.

Function8 If the parameter "i_u4_Address" is an address in the Block Protection Area1 and if the use of FACI1 is defined, R_RFD_FACI1 is returned as the return value.

Function9 If the parameter "i_u4_Address" is an address in the Block Protection Area1 and if the use of FACI1 is **NOT** defined, R_RFD_INVALID_FACI is returned as the return value.

Function10 If the parameter "i_u4_Address" is not an address in Configuration Setting Area, Security Setting Area, Block Protection Area0, or Block Protection Area1, R_RFD_INVALID_FACI is returned as a return value.

Function11 A precondition of this API is that Initialization of RFD28F (R_RFD_Init) has been executed in advance and has completed normally.

・ Return Value

R_RFD_FACI0		
Macro number	:	0x0000
meaning	:	To be operated is FACI0.
treatment	:	No need.
R_RFD_FACI1		
Macro number	:	0x0001
meaning	:	To be operated is FACI1.
treatment	:	No need.
R_RFD_INVALID_FACI		
Macro number	:	0x00ff
meaning	:	FACI to be operated is invalid.
treatment	:	Should correct 1st parameter (FACI number) or configuration parameter (Hardware depended information).

3.6 Note

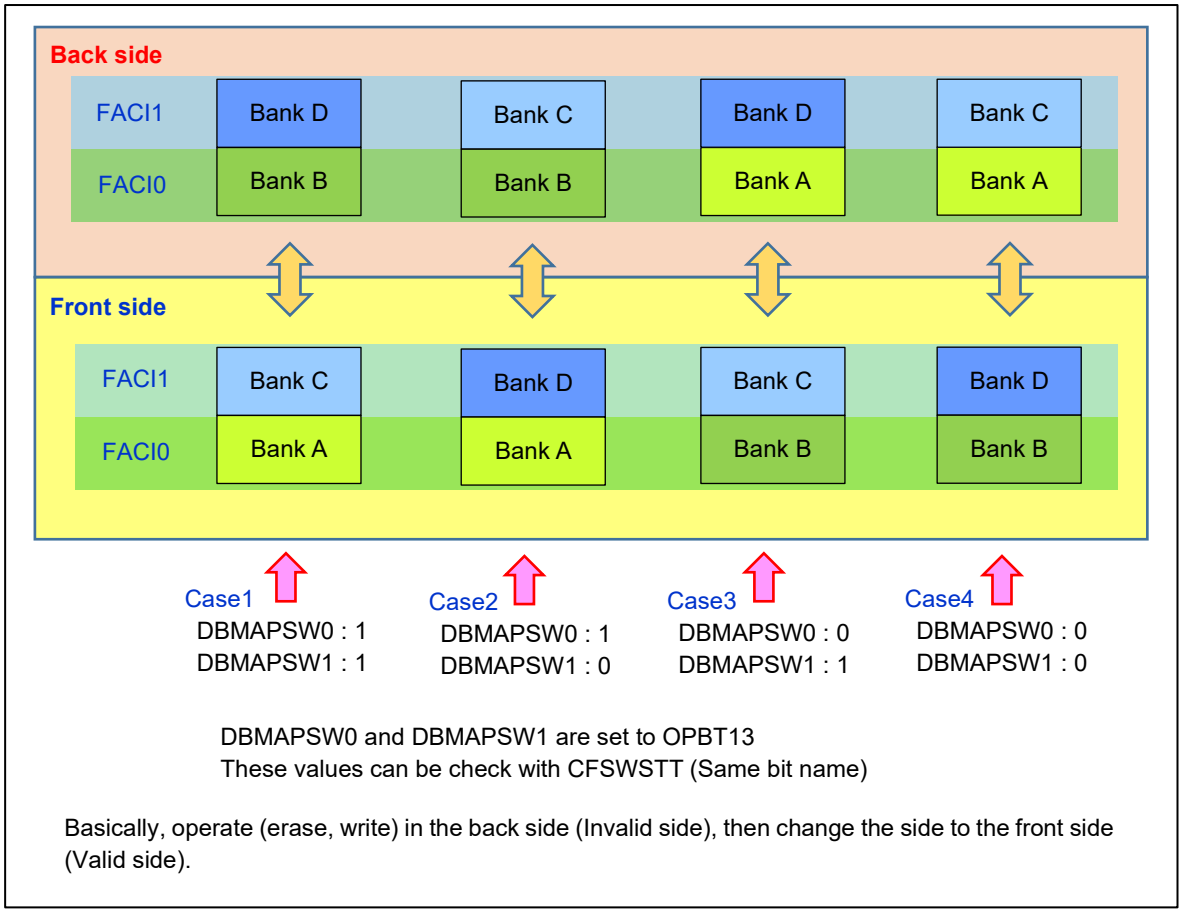
3.6.1 About FACI

RFD28F performs flash memory operation using FACI. There is 1 FACI or there are multiple FACIs for one MCU. In the case of RH850/U2A16, there are three FACI and they are named FACI0, FACI1 and FACI2.

- Code Flash Memory

Map Mode	Bank Name	FACI in charge of operation
Single map mode	Bank A Bank B	FACI0
	Bank C Bank D	FACI1
Double map mode	Front side : Bank A Back side : Bank B or Front side : Bank B Back side : Bank A	FACI0 (Recommended that operate only back side)
	Front side : Bank C Back side : Bank D or Front side : Bank D Back side : Bank C	FACI1 (Recommended that operate only back side)

Figure 3-2 Relationship between Front side, Back side and FACI in Double map mode



- Data Flash Memory

Data area Name	FACI in charge of operation
Data area 0	FACI0
Data area 1	FACI1
Data area 2	FACI2

In the case of RH850/U2A16 and when operating code flash memory, the FACI in charge will change depending on the map mode to be used. When operating data flash memory, FACI0 is in charge of data area 0, FACI1 is in charge of data area 1, and FACI 2 is in charge of data area 2.

In other words, it is necessary to perform flash memory operation for the correct FACI. In the example shown in, when writing processing to the data flash memory for 0xff200000 is suspended, suspension processing must be performed on FACI0. Also, the examples shown in Figure 3-3 and Figure 3-4 are for RH850/U2A16.

Figure 3-3 Related Flash Memory Structure for RH850/U2A16 (Single map mode)

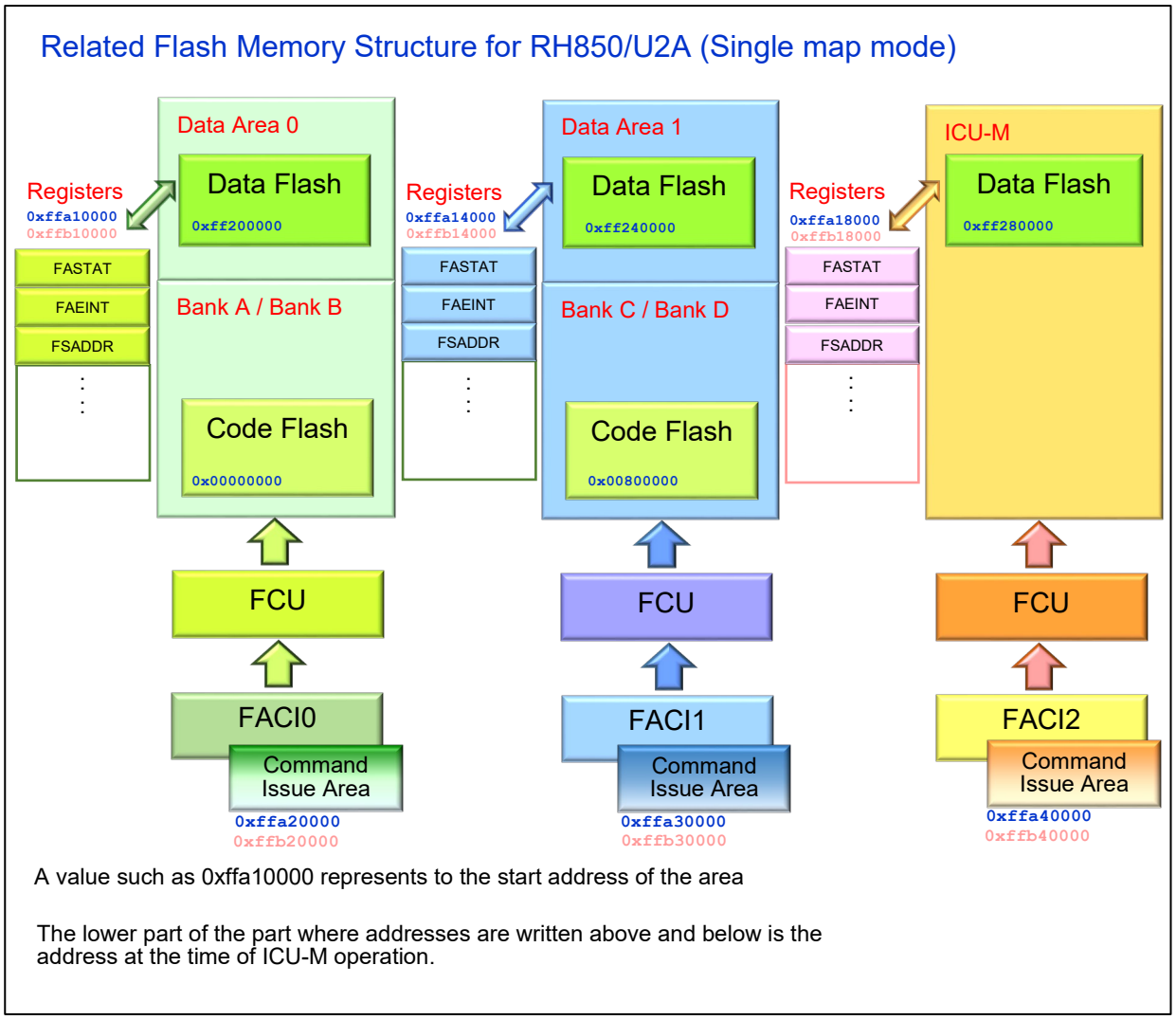
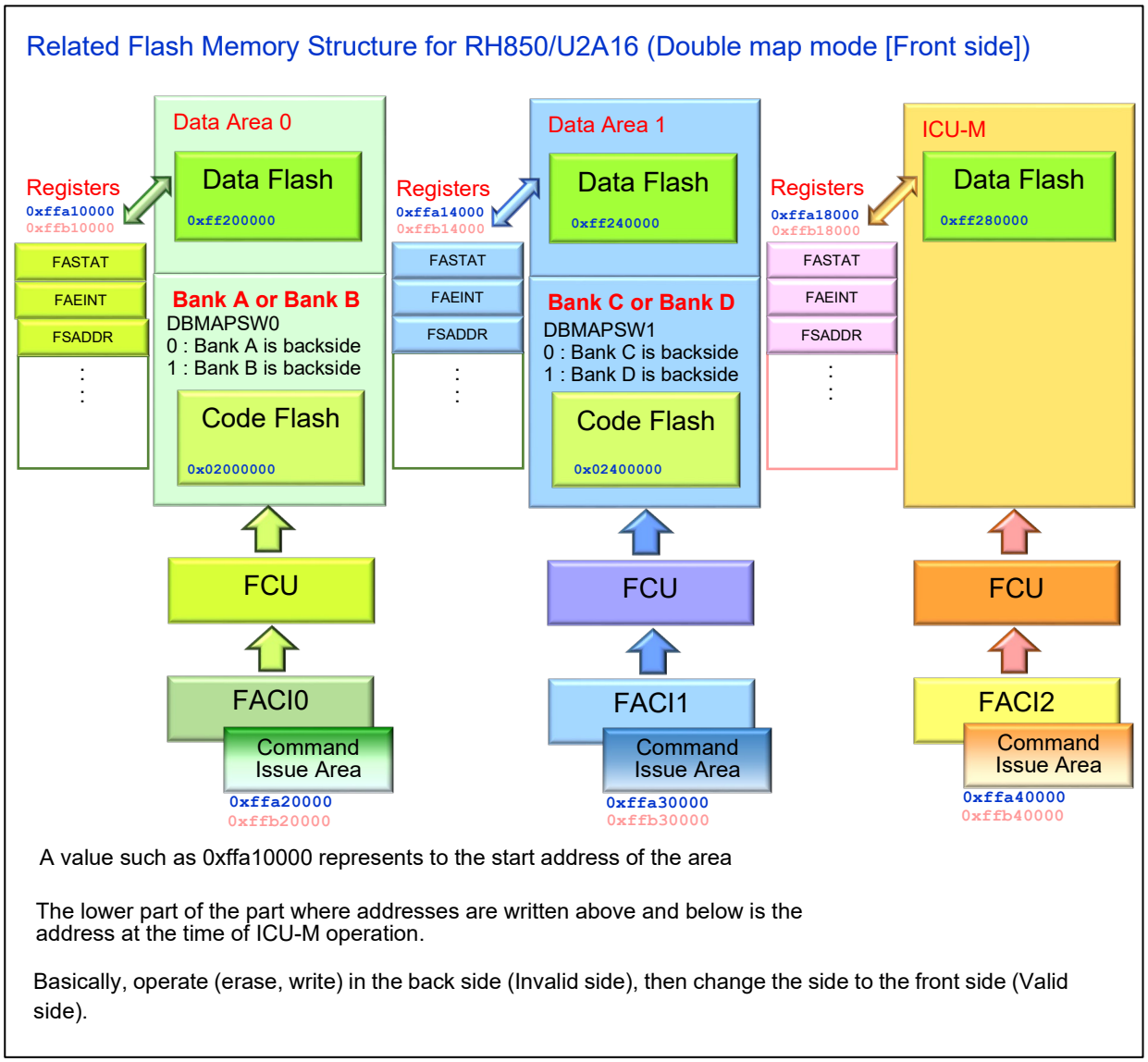


Figure 3-4 Related Flash Memory Structure for RH850/U2A16 (Double map mode)



A value such as 0xffa10000 represents to the start address of the area

The lower part of the part where addresses are written above and below is the address at the time of ICU-M operation.

Basically, operate (erase, write) in the back side (Invalid side), then change the side to the front side (Valid side).

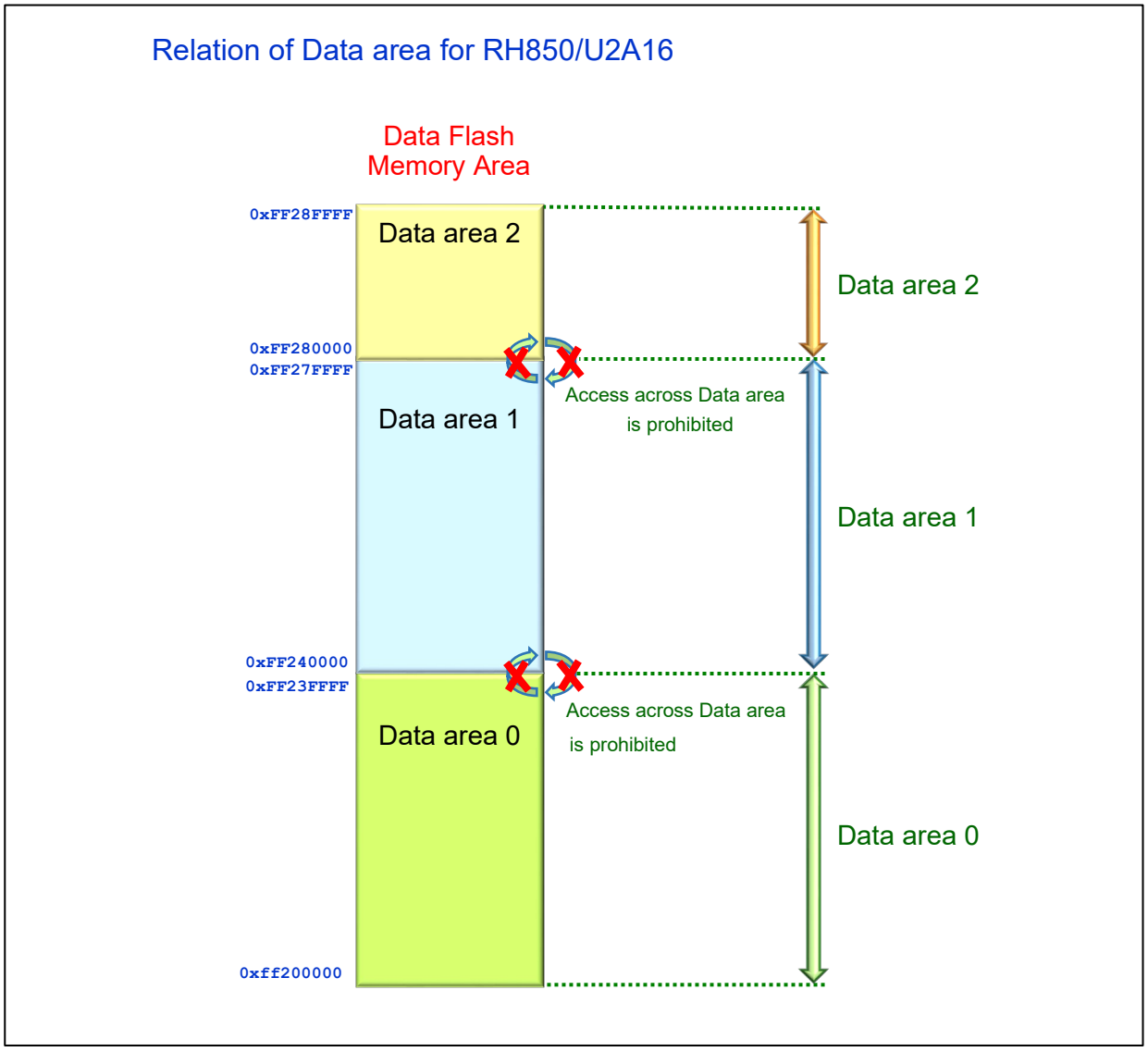
3.6.2 About Data Area

The data flash memory of RH850/U2A16 consists of multiple “Data Areas”. In case of RH850/U2A16, it consists of three Data areas, “Data area 0”, “Data area 1” and “Data area 2.

When writing, erasing or checking blank area of data flash memory, it cannot be performed between the Data areas. So please be careful. The image is shown in Figure 3-5.

For details on the Data area structure, refer to the user's manual of the hardware.

Figure 3-5 Relation of Data area for RH850/U2A16



3.6.3 About Bank

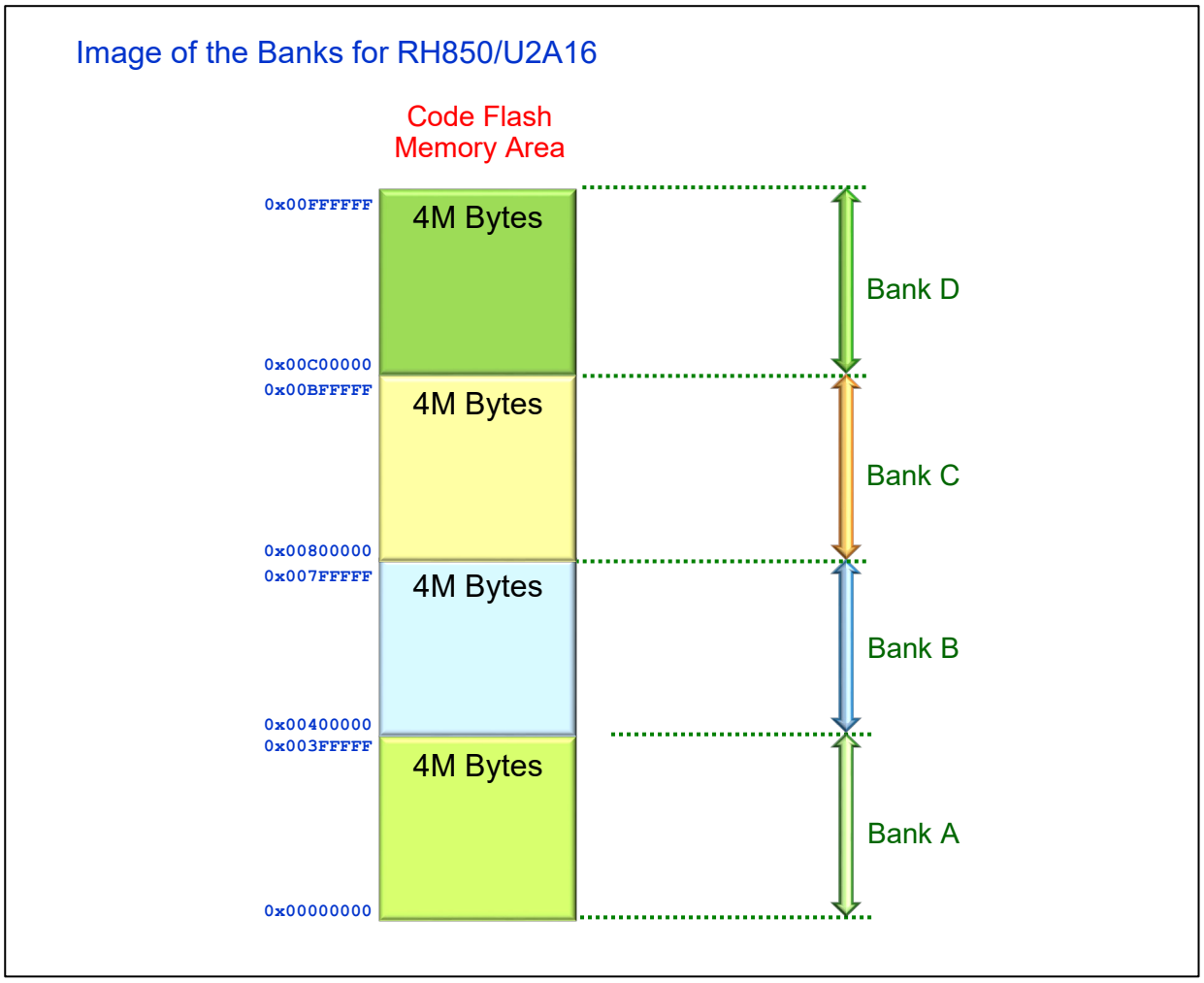
The code flash memory of RH850/U2A16 consists of multiple banks. RH850/U2A16 has BankA to BankD.

Also, each bank have multiple blocks.

The image for RH850/U2A16 is shown in Figure 3-6.

For details on the Bank structure, refer to the user's manual of the hardware.

Figure 3-6 Image of Banks for RH850/U2A16



3.6.4 About Hardware Property Area and its rewriting

There is an area called Hardware Property Area in the RH850/U2A16. There are the following three types of rewritable areas in this area, and settings that greatly affect the fundamental operation of the microcomputer, including those related to security such as option bytes and Data Flash ID, are allocated.

- Configuration Setting Area
- Security Setting Area
- Block Protection Area

However, as the Block Protection Area in the above areas, there are two areas for FPSYS0 and FPSYS1 (Block Protection Area for FPSYS0 and Block Protection Area for FPSYS1). Therefore, there are essentially four types of areas. But some devices do not have for FPSYS1.

The four types of areas allocated in the Hardware Property Area and the settings included in each area

- Configuration Setting Area
  - OTP Setting of Configuration Setting Area
  - User Programmable Option Byte
  - Reset Vector for PEn
  - OPBTn
- Security Setting Area
  - OTP Setting of Security Setting Area
  - Security OPBT (S_OPBTn)
  - OCD ID
  - Serial Programmer ID
  - Customer ID A
  - DataFlash ID
  - RHSIF ID
  - Customer ID B
  - Customer ID C

- Block Protection Area for FPSYS0
  - OTP Setting of User area 0
  - OTP Setting of User area 1
  - OTP Setting of Extended Data Area
  - Customer ID Protection & Erase Counter Setting for user area0 & user boot area0
  - Customer ID Protection & Erase Counter Setting for user area1 & user boot area1
  - Customer ID Protection Setting for Extended data area
- Block Protection Area for FPSYS1
  - OTP Setting of User area 2
  - OTP Setting of User area 3
  - Customer ID Protection & Erase Counter Setting for user area2 & user boot area2
  - Customer ID Protection & Erase Counter Setting for user area3 & user boot area3

In the above setting, there is a setting where the microcomputer never goes to operate again if wrong set values are written. However, even during updating processing correctly, processing may be interrupted due to disturbance. Especially when updating by OTA (Over the Air), the network may be interrupted and updating cannot be continued in some cases.

Therefore, in the RH850/U2A16, when updating the Hardware Property Area, we prepare a mechanism that it is possible to update safely and surely by adopting a mechanism that can improve the robustness to disturbance (robustness) and easily restore state (rollback) before update (although it does not guarantee complete robustness). This is the function below.

- We prepare each of the three types of areas "Configuration Setting Area", "Security Setting Area" and "Block Protection Area" on two each (duplicate), and make one of them operate as a valid side (as following)

  - Configuration Setting Area0／Configuration Setting Area1
  - Security Setting Area0／Security Setting Area1
  - Block Protection Area0 for FPSYS0／Block Protection Area1 for FPSYS0
  - Block Protection Area0 for FPSYS1／Block Protection Area1 for FPSYS1

- In each of the areas, information on which one of the valid sides is stored in "Switch Area". We also have two Switch Area (Switch Area 0 / Switch Area 1)

- In the Switch Area, information on which one of the valid sides is stored in "Tag Area"

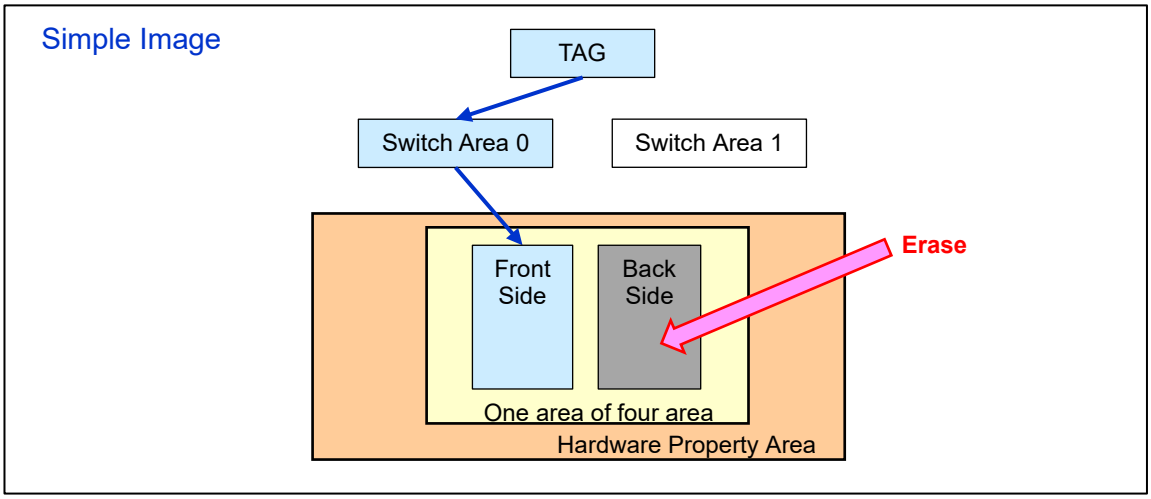
- An update progress flag is prepared for each area

When updating the settings in these areas, it cannot be updated with "R_RFD_EraseCFRequest" and "R_RFD_WriteCFRequest" used when updating the ordinary code flash memory. To use the function explained above, update using the following API.

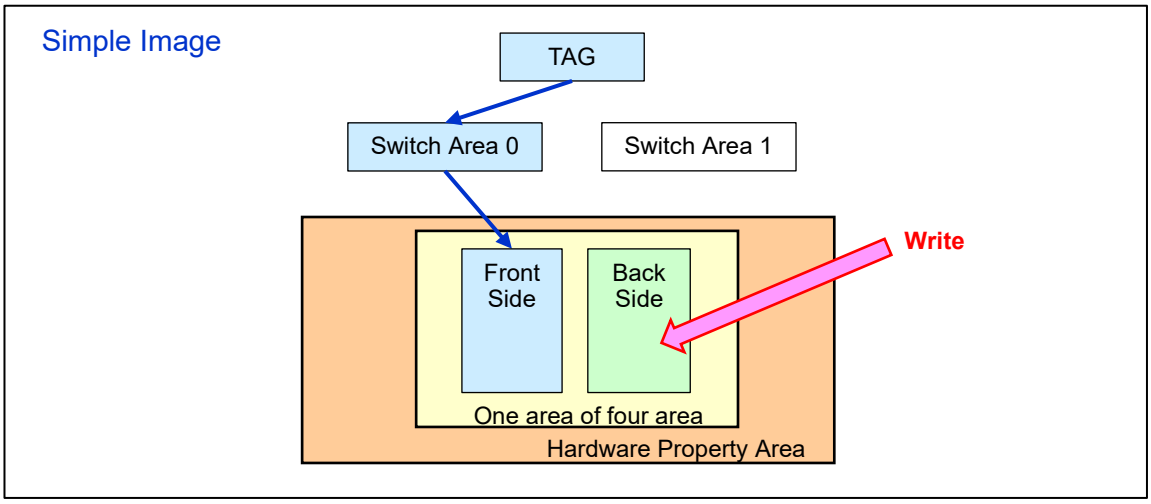
- R_RFD_ErasePropertyRequest
- R_RFD_WritePropertyRequest
- R_RFD_EraseSwitchRequest
- R_RFD_WriteSwitchRequest
- R_RFD_EraseTagRequest
- R_RFD_UpdateTagRequest

The summary of updating is as follows.

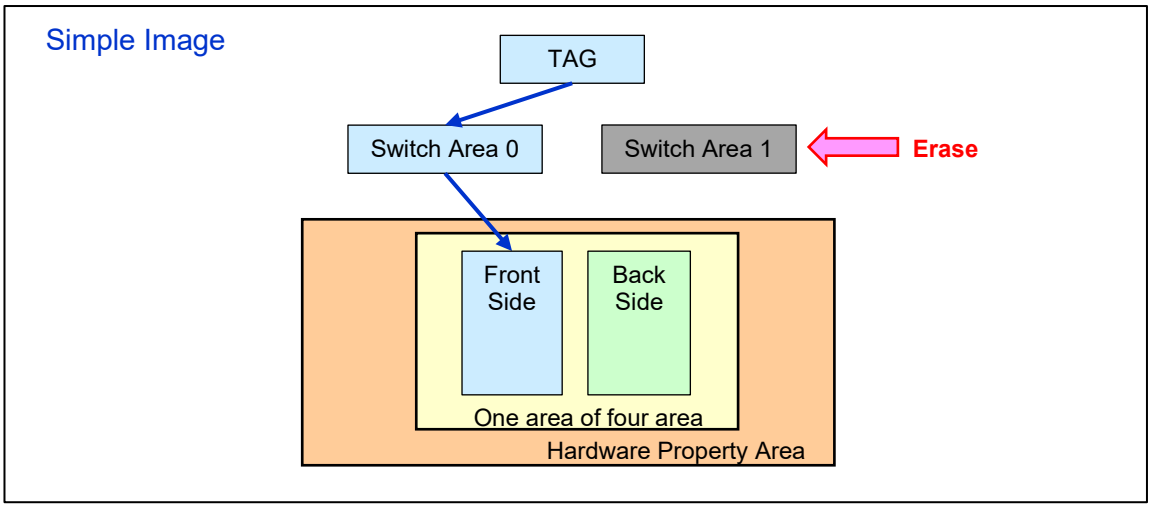
1. Erases the "Back side (now invalid side)" of the area where the setting to be updated is present among the four areas in the Hardware Property Area by issuing "R_RFD_ErasePropertyRequest"
   "Front side (now valid side)" is currently active (currently in operation), so cannot be erased. Updates "Back side (now invalid side)" and exchanges the back side to the front side after updating, so changes to activate. Activation is done by setting Switch Area.
   The reason why all areas including the setting are erased even with updating one setting is because the erase unit is large as the device specification and it is not possible to erase only the setting we want to update.



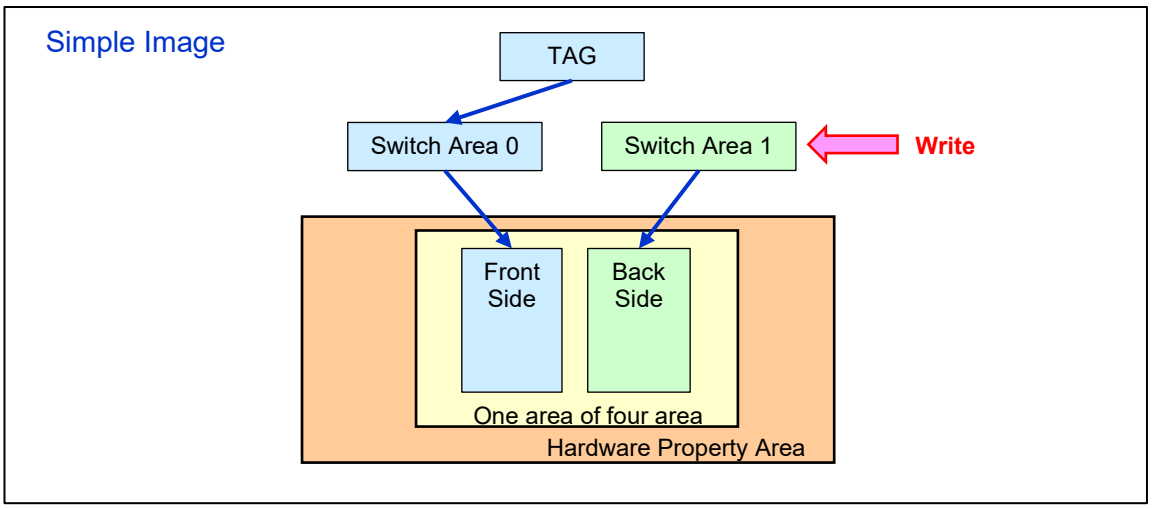
2. Writes data to the erased area by issuing R_RFD_WritePropertyRequest (writes to back side (now invalid side)). Since writing can only be done in units of 32 bytes, writing is performed for all erased settings, including settings that we want to update. For data not to be updated, copy and use the data on the front side (now valid side).



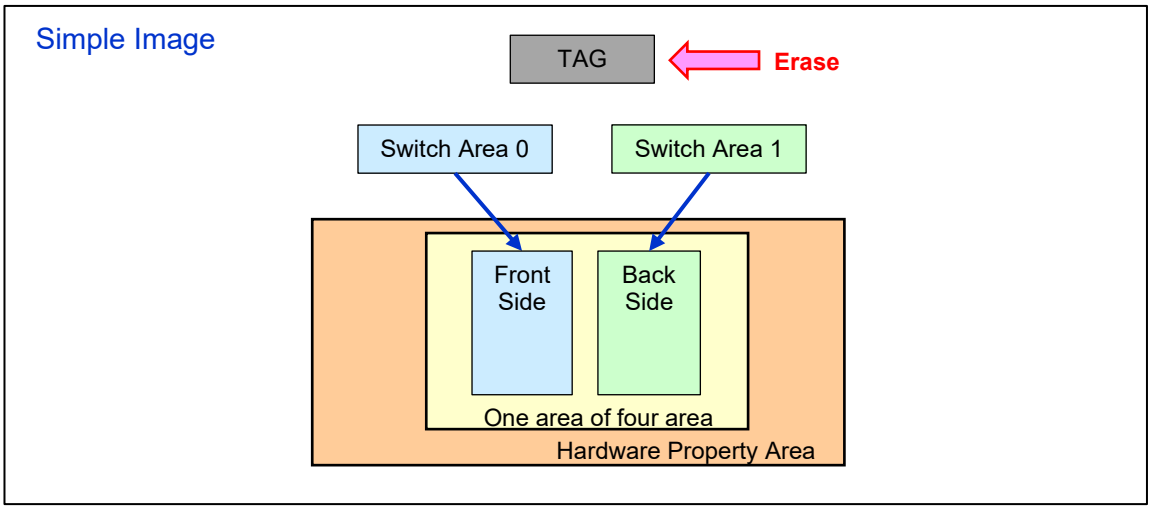
3. To update the Switch Area pointing to the valid area of each of the four areas in the Hardware Property Area, erases the back side (now invalid side) of Switch Area by issuing "R_RFD_EraseSwitchRequest". "Front side (now valid side)" is currently active (currently in operation), so cannot be erased. Updates "Back side (now invalid side)" and exchanges the back side to the front side after updating, so changes to activate. Activation is done by setting Tag Area.



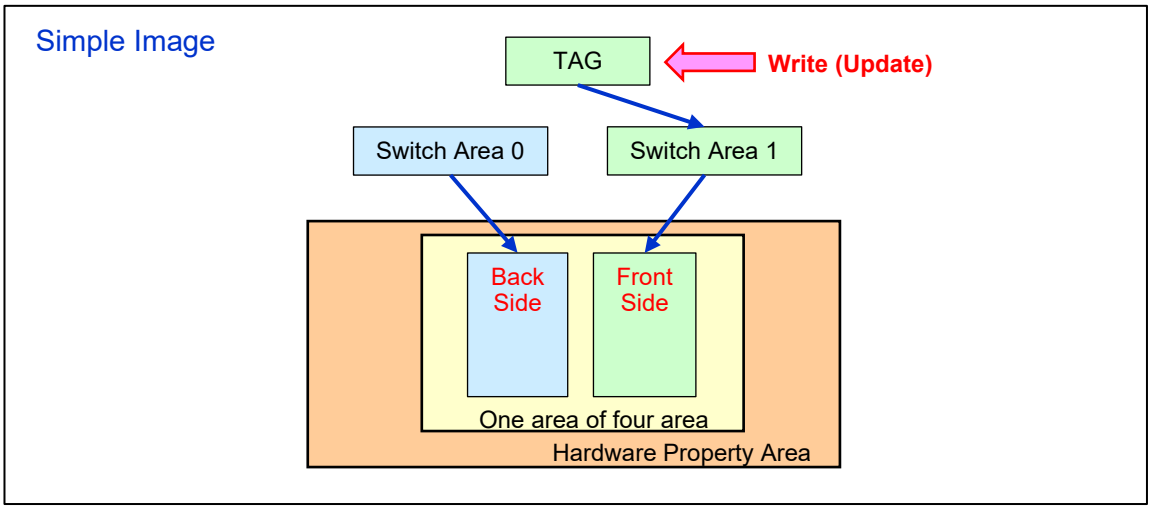
4. Writes data to the erased Switch Area by issuing R_RFD_WriteSwitchRequest (writes to back side (now invalid side)). The content to be written is the value that is different from the current value (value on the front side) for the area to be changed from the back side to the front side when switching the Switch Area, and if the front side is kept as it is, the content to be written the current value (Value on the front side).
Since the value is either "0xA55A5AA5" or "0x5AA5A55A", if the current value (value on the front side) is "0xA55A5AA5", the different value is "0x5AA5A55A". If the current value (value on the front side) Is “0x5AA5A55A", the different value is "0xA55A5AA5".



5. To update the Switch Area from the Back side to the Front side, erases the Tag Area pointing to the valid Switch Area by issuing R_RFD_EraseTagRequest.



6. Subsequently issuing R_RFD_UpdateTagRequest changes the Back side of the Switch Area to Front side and the area updated in “2” becomes valid.



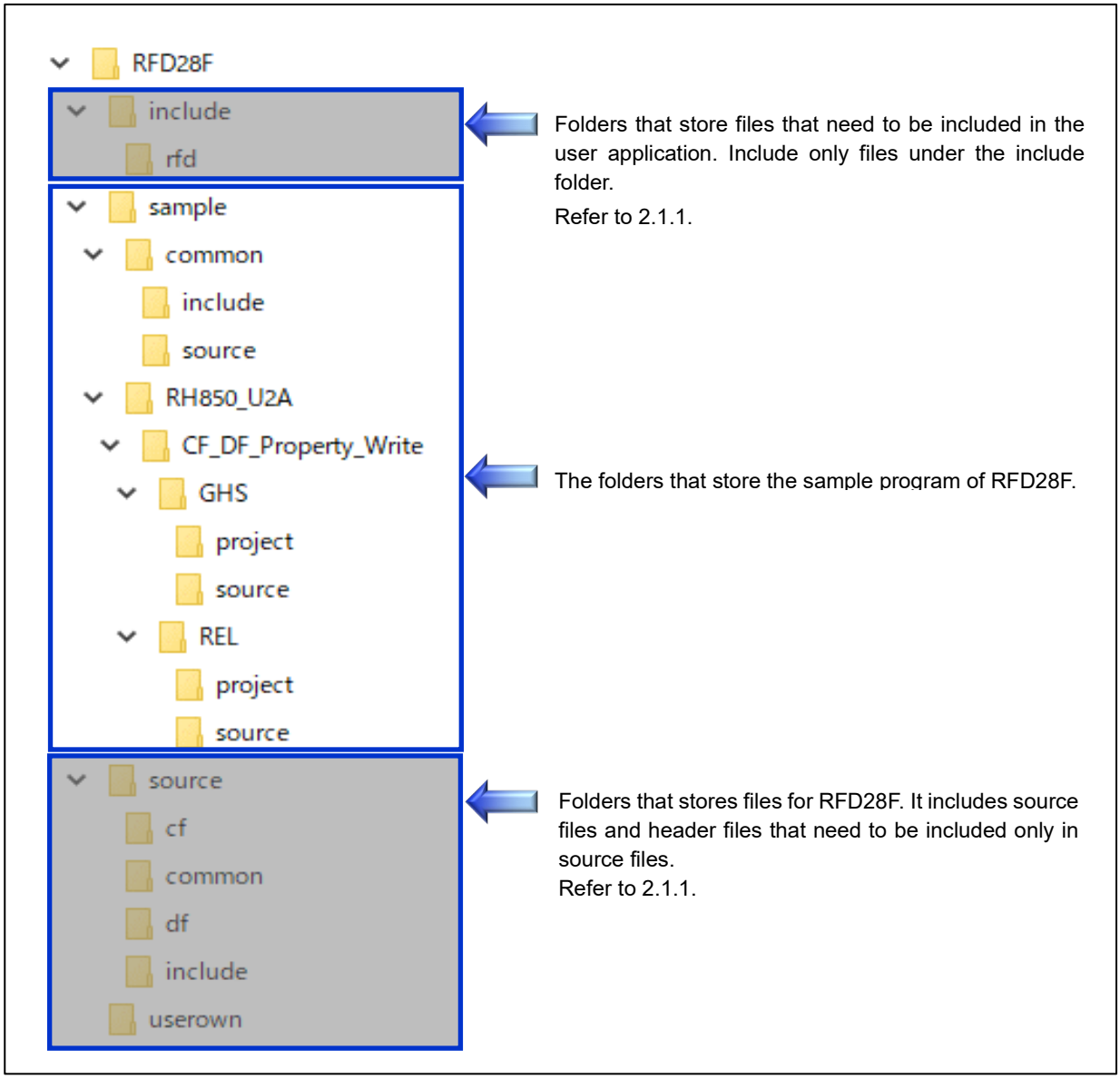
4 Sample Program

Explains the attached sample program.

4.1 File Structure

4.1.1 Folder Structure

Figure 4-1 Folder Structure



4.1.2 Source File List

Source files to allocated in “RFD28F/sample/common/source”

No	Source File Name	Summary
1	sample_control_codeflash.c	The file that describes a sample program that controls code flash with RFD28F. The sample of operations to the code flash is contained using the function defined in sample_subfunction_codeflash.c.
2	sample_control_common.c	The file that describes a sample program for common function that controls code flash, data flash and property area with RFD28F. The sample of operations to the flash is contained using the function defined in sample_subfunction_common.c.
3	sample_control_dataflash.c	The file that describes a sample program that controls data flash with RFD28F. The sample of operations to the data flash is contained using the function defined in sample_subfunction_dataflash.c.
4	sample_control_property.c	The file that describes a sample program that controls Hardware Property with RFD28F. The sample of operations to the Hardware Property is contained using the function defined in sample_subfunction_property.c.
5	sample_subfunction_codeflash.c	The file that define functions that operate code flash is using API of RFD28F. Implementing functions such as "data writing" and "erasing data" by combining API of RFD28F.
6	sample_subfunction_common.c	The file that define common functions that operate code flash, data flash and Hardware Property are using API of RFD28F. Implementing functions such as "data writing" and "erasing data" by combining API of RFD28F.
7	sample_subfunction_dataflash.c	The file that defines a functions that operate data flash is using API of RFD28F. Implementing functions such as "data writing" and "erasing data" by combining API of RFD28F.
8	sample_subfunction_property.c	The file that defines a functions that operate Hardware Property are using API of RFD28F. Implementing functions such as "data writing" and "erasing data" by combining API of RFD28F.

Source files to allocated in “RFD28F/sample/RH850_U2A/CF_DF_Property_Write/REL/source”. These files are for RH850/U2A16.

No	Source File Name	Summary
1	boot.asm	The file that described the boot process after reset
2	clock_gearup.c	The file that described the clock gear up sequence
3	cstart_pe0.asm	The file that described the process for PE0. This process is called from boot process.
4	cstart_pe1.asm	The file that described the process for PE1. This process is called from boot process.
5	cstart_pe2.asm	The file that described the process for PE2. This process is called from boot process.
6	cstart_pe3.asm	The file that described the process for PE3. This process is called from boot process.
7	initsct.c	The file that described memory initialization processing for GHS. It also copies data with initial value. It is called from boot process for each PE.
8	main_pe0.c	The file that described main function for PE0.
9	main_pe1.c	The file that described main function for PE1.
10	main_pe2.c	The file that described main function for PE2.
11	main_pe3.c	The file that described main function for PE3.
12	prefetch.asm	The file that contained for securing the prefetch area
13	vecttbl.asm	The file that contained vector table. Since only the reset is used in the sample, only the reset is enabled.

Source files to allocated in “RFD28F/sample/RH850_U2A/CF_DF_Property_Write/GHS/source”. These files are for RH850/U2A16.

No	Source File Name	Summary
1	boot.850	The file that described the boot process after reset
2	clock_gearup.c	The file that described the clock gear up sequence
3	cstart_pe0.850	The file that described the process for PE0. This process is called from boot process.
4	cstart_pe1.850	The file that described the process for PE1. This process is called from boot process.
5	cstart_pe2.850	The file that described the process for PE2. This process is called from boot process.
6	cstart_pe3.850	The file that described the process for PE3. This process is called from boot process.
7	initsct.c	The file that described memory initialization processing for GHS. It also copies data with initial value. It is called from boot process for each PE.
8	main_pe0.c	The file that described main function for PE0.
9	main_pe1.c	The file that described main function for PE1.
10	main_pe2.c	The file that described main function for PE2.
11	main_pe3.c	The file that described main function for PE3.
12	prefetch.850	The file that contained for securing the prefetch area
13	vecttbl.850	The file that contained vector table. Since only the reset is used in the sample, only the reset is enabled.
14	dr7f702300.ld	This file is Linker directive file. It describes necessary sections and their allocation information.

4.1.3 Header File List

Include files to allocated in “RFD28F/sample/common/include”

No	Header File Name	Summary
1	sample_codeflash_control.h	The file that contained macros and prototype declarations for functions used in sample program for code flash.
2	sample_common_control.h	The file that contained macros and prototype declarations for functions used in common in sample program.
3	sample_dataflash_control.h	The file that contained macros and prototype declarations for functions used in sample program for data flash.
4	sample_property_control.h	The file that contained macros and prototype declarations for functions used in sample program for Hardware Property.
5	sample_flash_control.h	The file that includes the include files 1 to 4 above.

4.2 Steps in Constructing the Sample Programs

4.2.1 For GHS sample

There is a project file (gpj file) in the following folder.

- RFD28F/sample/RH850_U2A/CF_DF_Property_Write/GHS/project

This project file is loaded by GHS builder and can be built.

Since RFD28F is provided as a source file, it is not be built only RFD28F, but will be built at the same time as the sample program (user application).

4.2.2 For REL sample

There is a project file (mtpj file) in the following folder.

- RFD28F/sample/RH850_U2A/CF_DF_Property_Write/REL/project

This project file is loaded by CS+ and can be built.

Since RFD28F is provided as a source file, it is not be built only RFD28F, but will be built at the same time as the sample program (user application).

4.3 Specification of the functions of the Sample Programs

The function specifications of the sample program are described below. The following functions are defined in "sample_function.c".

4.3.1 For Common Control

4.3.1.1 Sample_SuspendPE

- Information

Syntax	`T_u4_RFDReturn Sample_SuspendPE (T_u2_FACI i_u2_TargetFACI);`	
ASIL Level	-	
Status	New	
Sync/Async	Async	
Reentrancy	Non-Reentrant	
Parameters (IN)	T_u2_FACI	Target FACI to be suspended the process
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	T_u4_RFDReturn	R_RFD_OK
		R_RFD_ERR_ACCESS_ENV
		R_RFD_ERR_ERASE
		R_RFD_ERR_FHVE_PROTECT
		R_RFD_ERR_INTERNAL_DATA
		R_RFD_ERR_INTERNAL_HW
		R_RFD_ERR_NOFACI
		R_RFD_ERR_REJECT
		R_RFD_ERR_WRITE
		R_RFD_STS_BUSY
		R_RFD_STS_ERASE_SUSPENDED
		R_RFD_STS_WRITE_SUSPENDED
Description	Suspend the target FACI processing.	
Preconditions	-	
Remarks	If the status is command lock at execution of this function, command lock is not released within this function.	

・ Detail Specification

Function1 Suspends the processing of the target FACI specified by the first parameter "i_u2_TargetFACI".

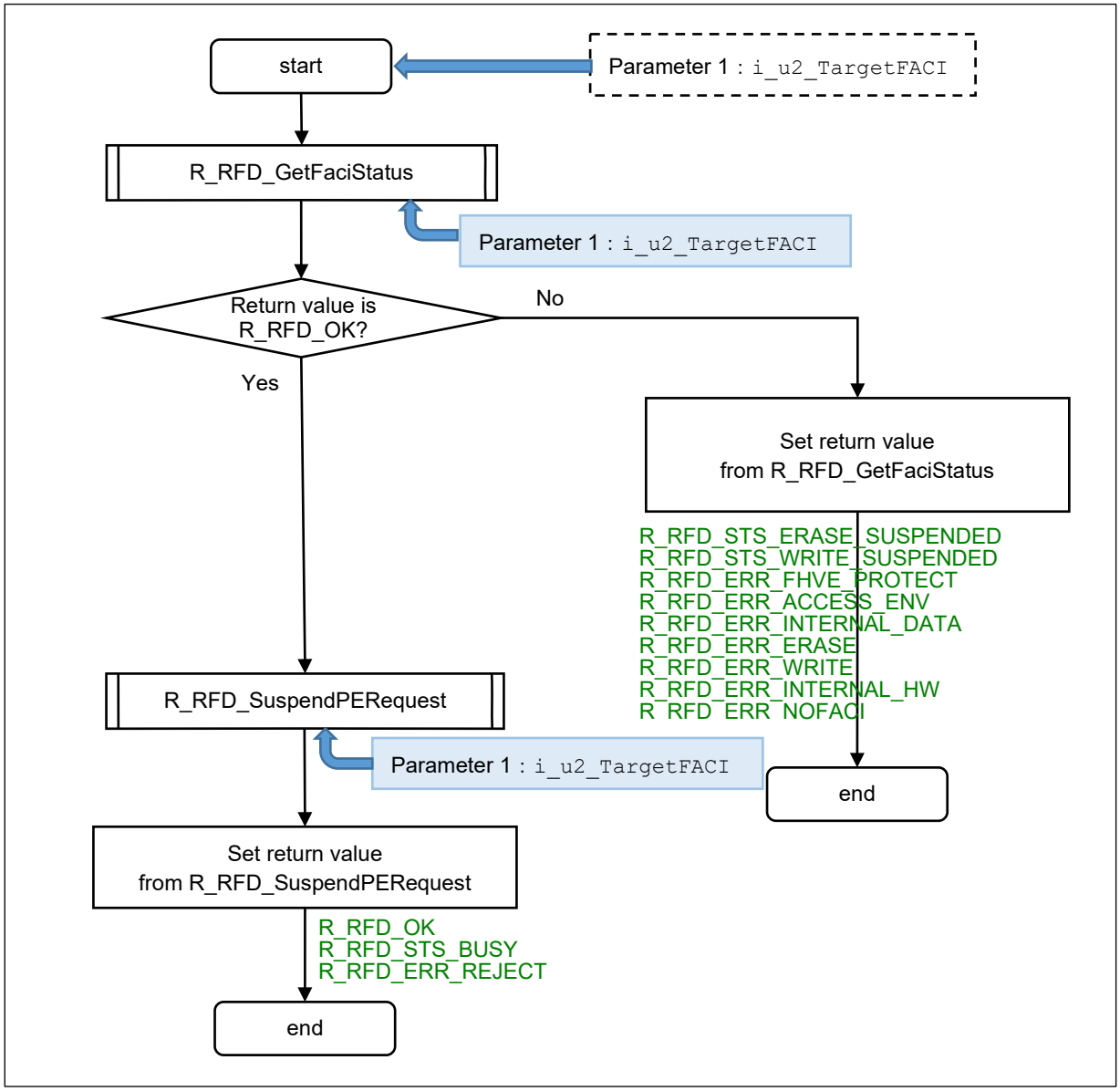
Function2 Execute “R_RFD_GetFaciStatus” to check whether it is in command lock state.

Currently, it checks whether the specified FACI number exists, and if not, it returns “R_RFD_ERR_NOFACI” as a return value.

If the status is command lock, this function ends with the return value of R_RFD_GetFaciStatus. The command lock status is not released in this function.

Function3 Execute “R_RFD_SuspendPERequest” and this function terminates with return value of “R_RFD_SuspendPERequest”

Figure 4-2 Flow of “Sample_SuspendPE”



4.3.1.2 Sample_ResumePE

・ Information

Syntax	`T_u4_RFDReturn Sample_ResumePE (T_u2_FACI i_u2_TargetFACI);`	
ASIL Level	-	
Status	New	
Sync/Async	Async	
Reentrancy	Non-Reentrant	
Parameters (IN)	`T_u2_FACI`	Target FACI to be resumed the process
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	`T_u4_RFDReturn`	R_RFD_OK
		R_RFD_ERR_NOFACI
		R_RFD_ERR_NO_SUSPENSION
Description	Resume the suspension process for the target FACI.	
Preconditions	-	
Remarks	-	

・ Detail Specification

Function1 Resume the suspension process of FACI specified by the parameter “i_u2_TargetFACI”.

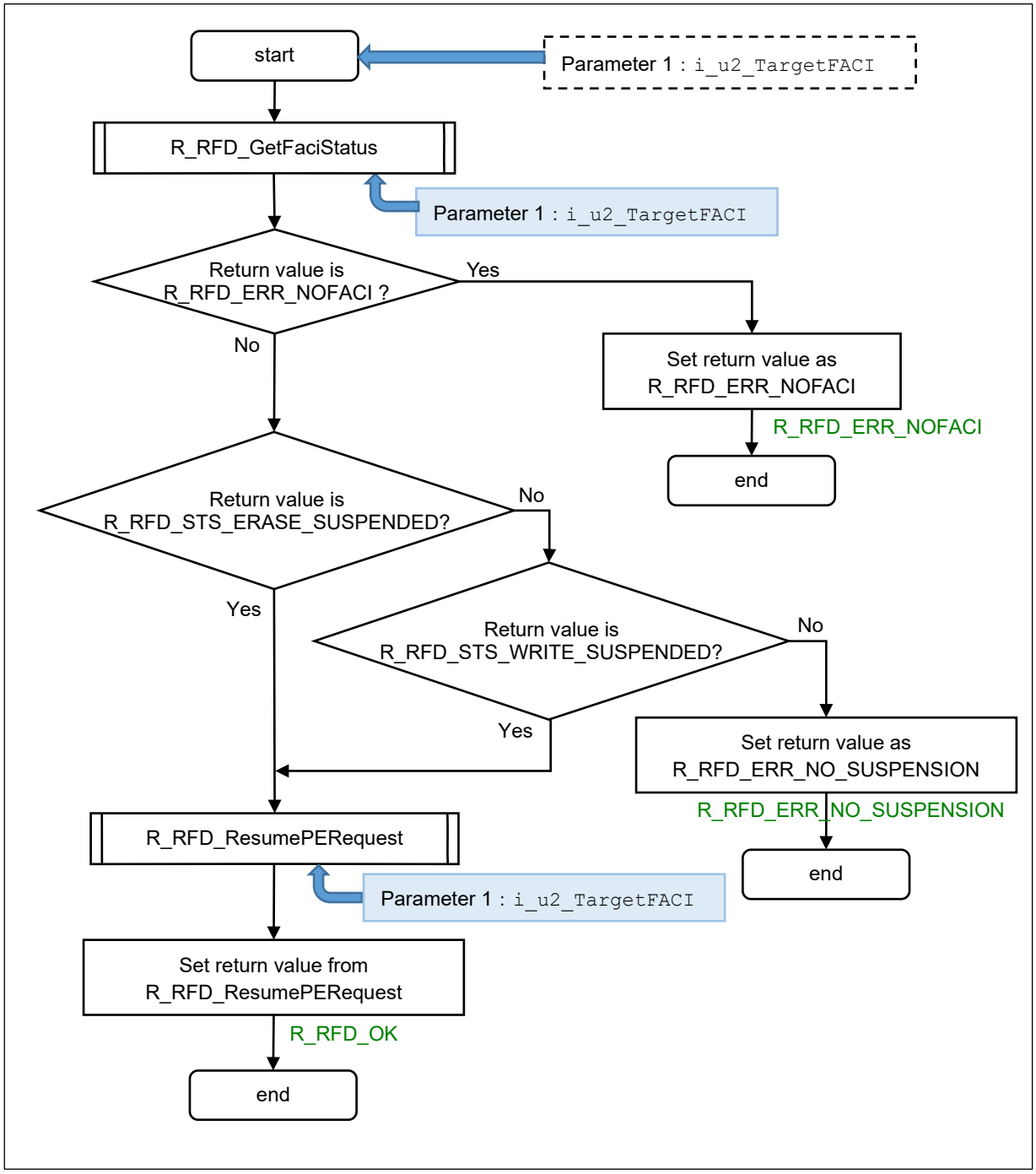
Function2 Execute “R_RFD_GetFaciStatus” to check if the process is suspended.

At this time, it checks whether the specified FACI number exists, and if not, it returns “R_RFD_ERR_NOFACI” as a return value.

If the return value of executing “R_RFD_GetFaciStatus” is not "R_RFD_STS_ERASE_SUSPENDED" or "R_RFD_STS_WRITE_SUSPENDED", terminate the function as the return value "R_RFD_ERR_NO_SUSPENSION".

Function3 Execute “R_RFD_ResumePERequest” and resume the suspension process.

Figure 4-3 Flow of "Sample_ResumePE”



4.3.1.3 Sample_CheckFaciStatus

- Information

Syntax	`T_u4_RFDReturn Sample_CheckFaciStatus (T_u2_FACI i_u2_TargetFACI);`	
ASIL Level	-	
Status	New	
Sync/Async	Sync	
Reentrancy	Non-Reentrant	
Parameters (IN)	`T_u2_FACI`	Target FACI to be controlled
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	`T_u4_RFDReturn`	R_RFD_OK
		R_RFD_ERR_ACCESS_ENV
		R_RFD_ERR_BLOCKPROTECTION0_DATA
		R_RFD_ERR_BLOCKPROTECTION1_DATA
		R_RFD_ERR_CONFIGURATION_DATA
		R_RFD_ERR_ERASE
		R_RFD_ERR_ERASECOUNTER_DATA
		R_RFD_ERR_FHVE_PROTECT
		R_RFD_ERR_INTERNAL_DATA
		R_RFD_ERR_INTERNAL_HW
		R_RFD_ERR_NOFACI
		R_RFD_ERR_SECURITY_DATA
		R_RFD_ERR_SWITCH_DATA
		R_RFD_ERR_WRITE
		R_RFD_STS_ERASE_SUSPENDED
		R_RFD_STS_WRITE_SUSPENDED
		R_RFD_WARN_BLOCKPROTECTION0_DATA
		R_RFD_WARN_BLOCKPROTECTION1_DATA
		R_RFD_WARN_CONFIGURATION_DATA
		R_RFD_WARN_ERASECOUNTER_DATA
		R_RFD_WARN_INTERNAL_DATA
		R_RFD_WARN_SECURITY_DATA
		R_RFD_WARN_SWITCH_DATA
Description	Check the FACI sequencer status for the target FACI.	
Preconditions	-	
Remarks	-	

・ Detail Specification

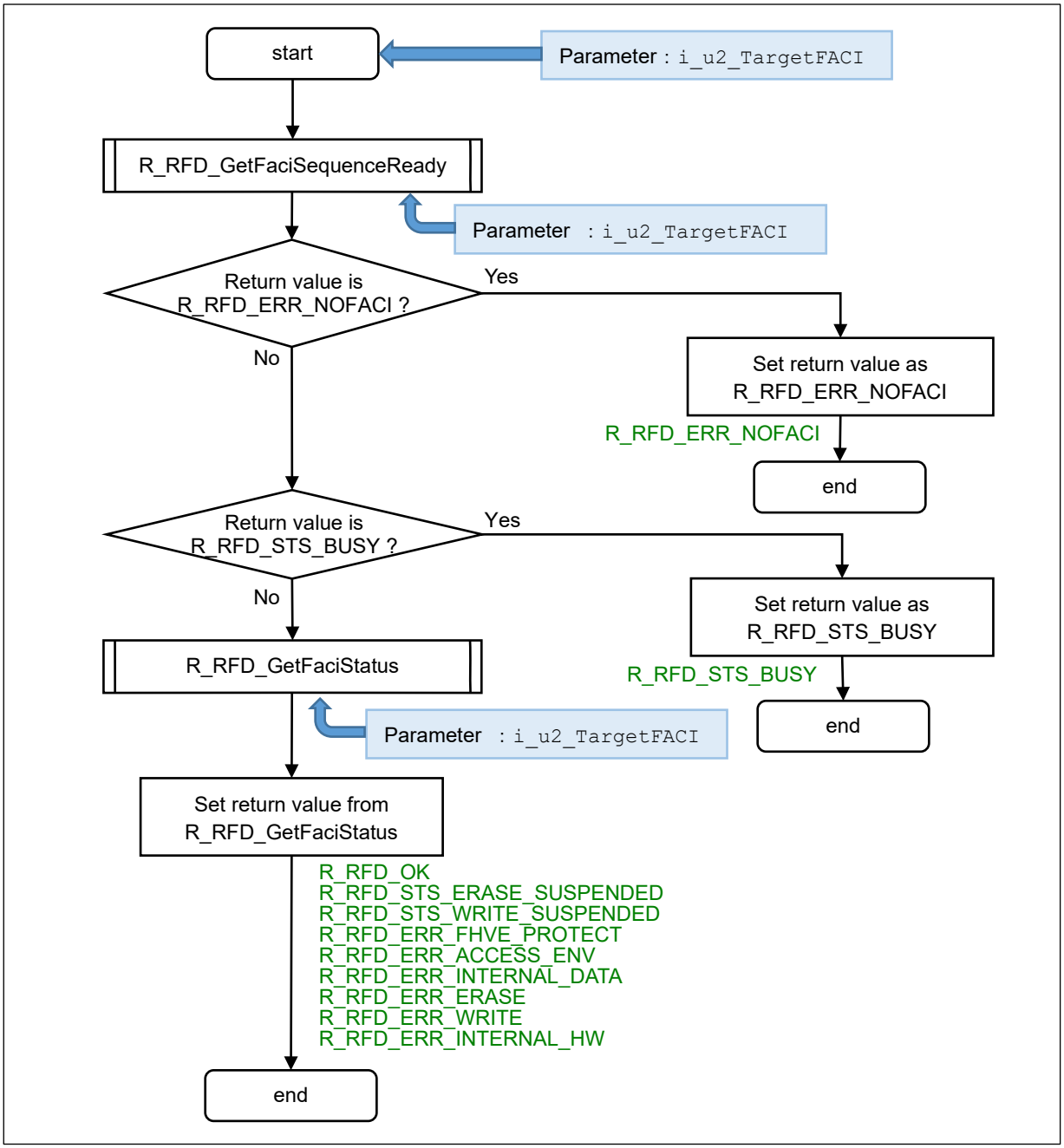
Function1 Execute “R_RFD_GetFaciSequenceReady”.

At this time, it checks whether the specified FACI number exists, and if not, it returns “R_RFD_ERR_NOFACI” as a return value.

If the return value is "R_RFD_STS_BUSY" , this function is terminated with this return value.

Function2 Execute “R_RFD_GetFaciStatus” and this function terminated with the return value of “R_RFD_GetFaciStatus”.

Figure 4-4 Flow of “Sample_CheckFaciStatus”


start
Parameter : i_u2_TargetFACI
R_RFD_GetFaciSequenceReady
Parameter : i_u2_TargetFACI
Return value is R_RFD_ERR_NOFACI ?
Yes
No
Set return value as R_RFD_ERR_NOFACI
R_RFD_ERR_NOFACI
end
Return value is R_RFD_STS_BUSY ?
Yes
No
Set return value as R_RFD_STS_BUSY
R_RFD_STS_BUSY
end
R_RFD_GetFaciStatus
Parameter : i_u2_TargetFACI
Set return value from R_RFD_GetFaciStatus
R_RFD_OK
R_RFD_STS_ERASE_SUSPENDED
R_RFD_STS_WRITE_SUSPENDED
R_RFD_ERR_FHVE_PROTECT
R_RFD_ERR_ACCESS_ENV
R_RFD_ERR_INTERNAL_DATA
R_RFD_ERR_ERASE
R_RFD_ERR_WRITE
R_RFD_ERR_INTERNAL_HW
end

4.3.1.4 Sample_ReleaseCommandLock

・ Information

Syntax	`T_u4_RFDReturn Sample_ReleaseCommandLock (T_u2_FACI i_u2_TargetFACI);`	
ASIL Level	-	
Status	New	
Sync/Async	Sync	
Reentrancy	Non-Reentrant	
Parameters (IN)	T_u2_FACI	Target FACI to be released the command lock state
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	T_u4_RFDReturn	R_RFD_OK
		R_RFD_ERR_INTERNAL_HW
		R_RFD_ERR_NOFACI
Description	Release the command lock state for the target FACI.	
Preconditions	-	
Remarks	-	

・ Detail Specification

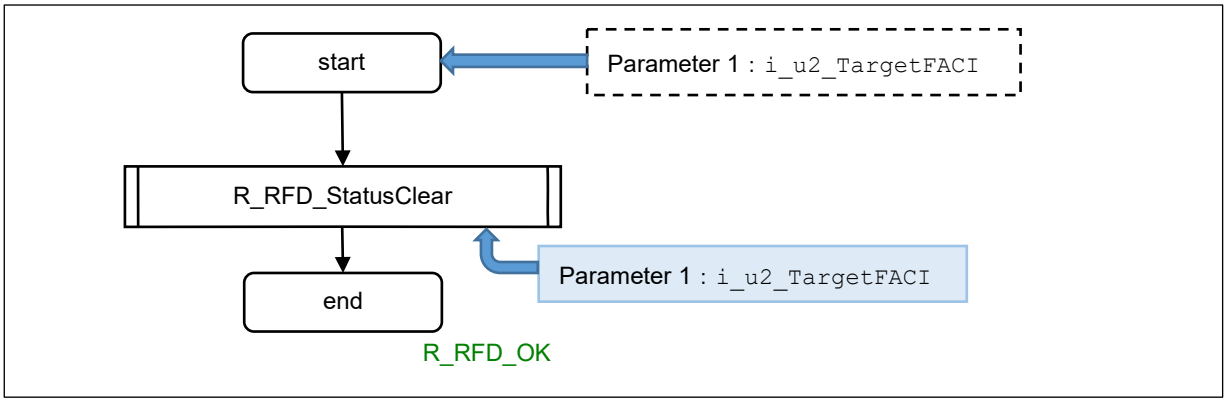
Function1 Release the command lock state in the target FACI specified by the first parameter “i_u2_TargetFACI”.

Function2 In the case of RH850/U2A16, execute " R_RFD_StatusClear" and terminate its return value as the return value of this function.

If the return value of R_RFD_GetFaciStatus is " R_RFD_ERR_FHVE_PROTECT ", execute "R_RFD_ForcedStopAndErrorClear" and terminate its return value as the return value of this function.

If the return value of R_RFD_GetFaciStatus is not " R_RFD_ERR_FHVE_PROTECT ", execute " R_RFD_StatusClear" and terminate its return value as the return value of this function.

Figure 4-5 Flow of “Sample_ReleaseCommandLock”



4.3.2 For Data Flash Control

4.3.2.1 Sample_EraseDF

- Information

Syntax	`T_u4_RFDReturn Sample_EraseDF (T_u4_RfdAddress i_u4_Start, T_u4_RfdAddress i_u4_End);`	
ASIL Level	-	
Status	New	
Sync/Async	Async	
Reentrancy	Non-Reentrant	
Parameters (IN)	`T_u4_RfdAddress`	Start address for erasure area
	`T_u4_RfdAddress`	End address for erasure area
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	`T_u4_RFDReturn`	R_RFD_OK
		R_RFD_STS_BUSY
		R_RFD_STS_ERASE_SUSPENDED
		R_RFD_STS_WRITE_SUSPENDED
		R_RFD_ERR_INTERNAL_HW
		R_RFD_ERR_NOFACI
Description	Erases the contents of the data flash memory.	
Preconditions	FACI mode must be Data Flash P/E mode.	
Remarks	-	

・ Detail Specification

Function1 Erases the contents of the data flash memory from the address specified by the first parameter “i_u4_Start” to the address specified by the second parameter “i_u4_End”.

Function2 Check whether the flash is in READY state or command lock state (Sample_CheckFaciStatus).

The FACI Number specified by parameter is executed by executing R_RFD_DFAddressToFaciNumber.

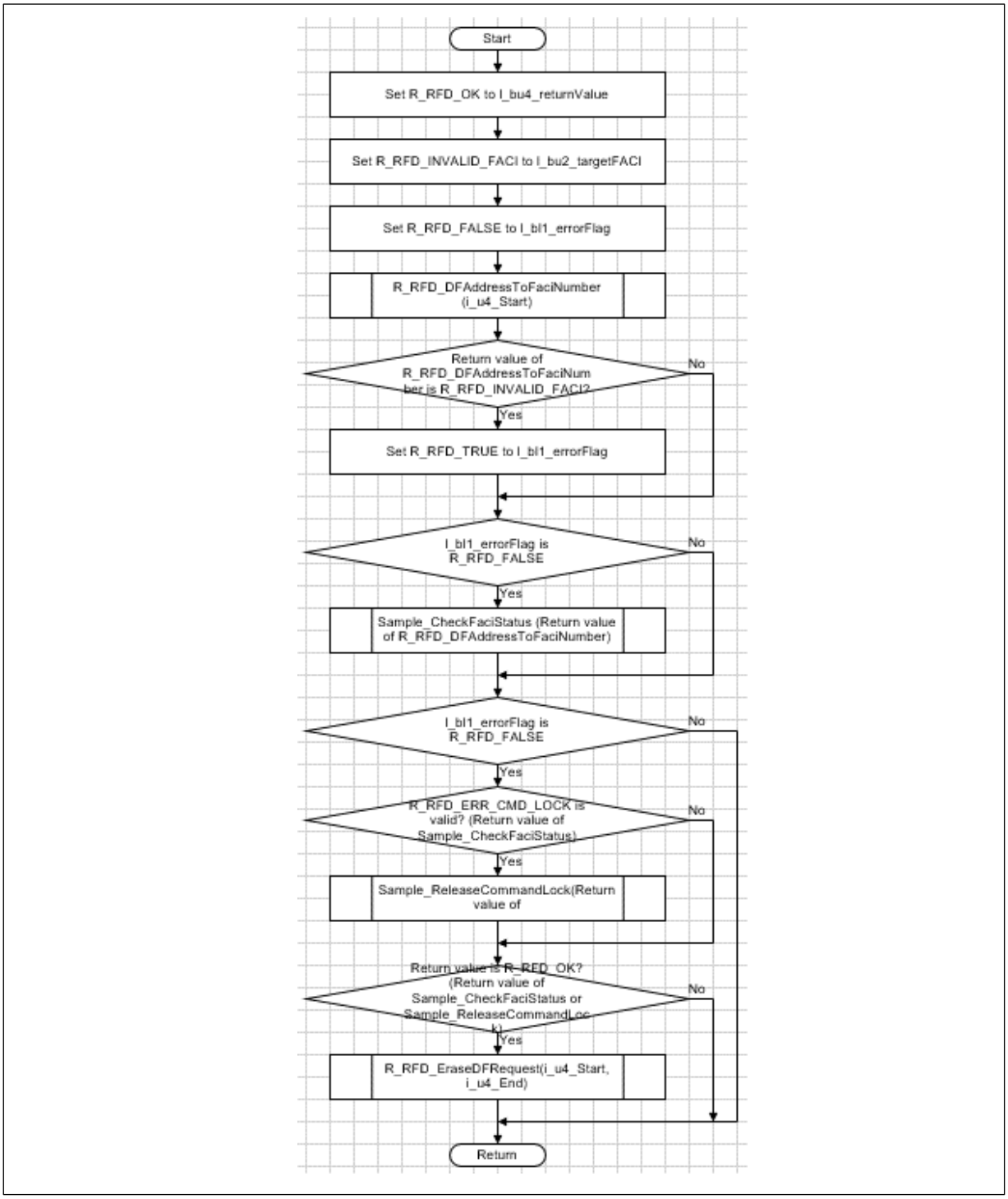
At this time, it checks whether the specified FACI number exists, and if not, it returns “R_RFD_ERR_NOFACI” as a return value.

If the return value of " Sample_CheckFaciStatus " is "R_RFD_STS_ERASE_SUSPENDED" or "R_RFD_STS_WRITE_SUSPENDED" or "R_RFD_STS_BUSY", it ends as a return value. If the return value is "R_RFD_OK", continue to execute Function3.

If the return value of “Sample_CheckFaciStatus” is not "R_RFD_STS_ERASE_SUSPENDED" or "R_RFD_STS_WRITE_SUSPENDED" or "R_RFD_STS_BUSY" or "R_RFD_OK" (In a state that represents a command lock), execute "Sample_ReleaseCommandLock". If the return value of "Sample_ReleaseCommandLock" is other than "R_RFD_OK", this function is terminated with the return value of “Sample_ReleaseCommandLock”.

Function3 Execute “R_RFD_EraseDFRequest” and proceed erasing process.

Figure 4-6 Flow of “Sample_EraseDF”



4.3.2.2 Sample_WriteDF

- Information

Syntax	`T_u4_RFDReturn Sample_WriteDF (T_u4_RfdAddress i_u4_Start, T_u2 i_u2_Length, T_pu4_RfdBuffer i_pu4_Data);`	
ASIL Level	-	
Status	New	
Sync/Async	Async	
Reentrancy	Non-Reentrant	
Parameters (IN)	`T_u4_RfdAddress`	Start address for writing area
	`T_u2`	Data for writing length
	`T_pu4_RfdBuffer`	Buffer address where write data is stored
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	`T_u4_RFDReturn`	R_RFD_OK
		R_RFD_STS_BUSY
		R_RFD_STS_ERASE_SUSPENDED
		R_RFD_STS_WRITE_SUSPENDED
		R_RFD_ERR_INTERNAL_HW
		R_RFD_ERR_NOFACI
Description	Writes the data to the data flash memory.	
Preconditions	FACI mode must be Data Flash P/E mode.	
Remarks	-	

・ Detail Specification

Function1 Writes the data stored in the third parameter "i_pu4_Data" from the address indicated by the first parameter “i_u4_Start” by the amount of size indicated by the second parameter “i_u2_Length”.

Function2 Check whether the flash is in READY state or command lock state (Sample_CheckFaciStatus).

The FACI Number specified by parameter is executed by executing R_RFD_DFAddressToFaciNumber.

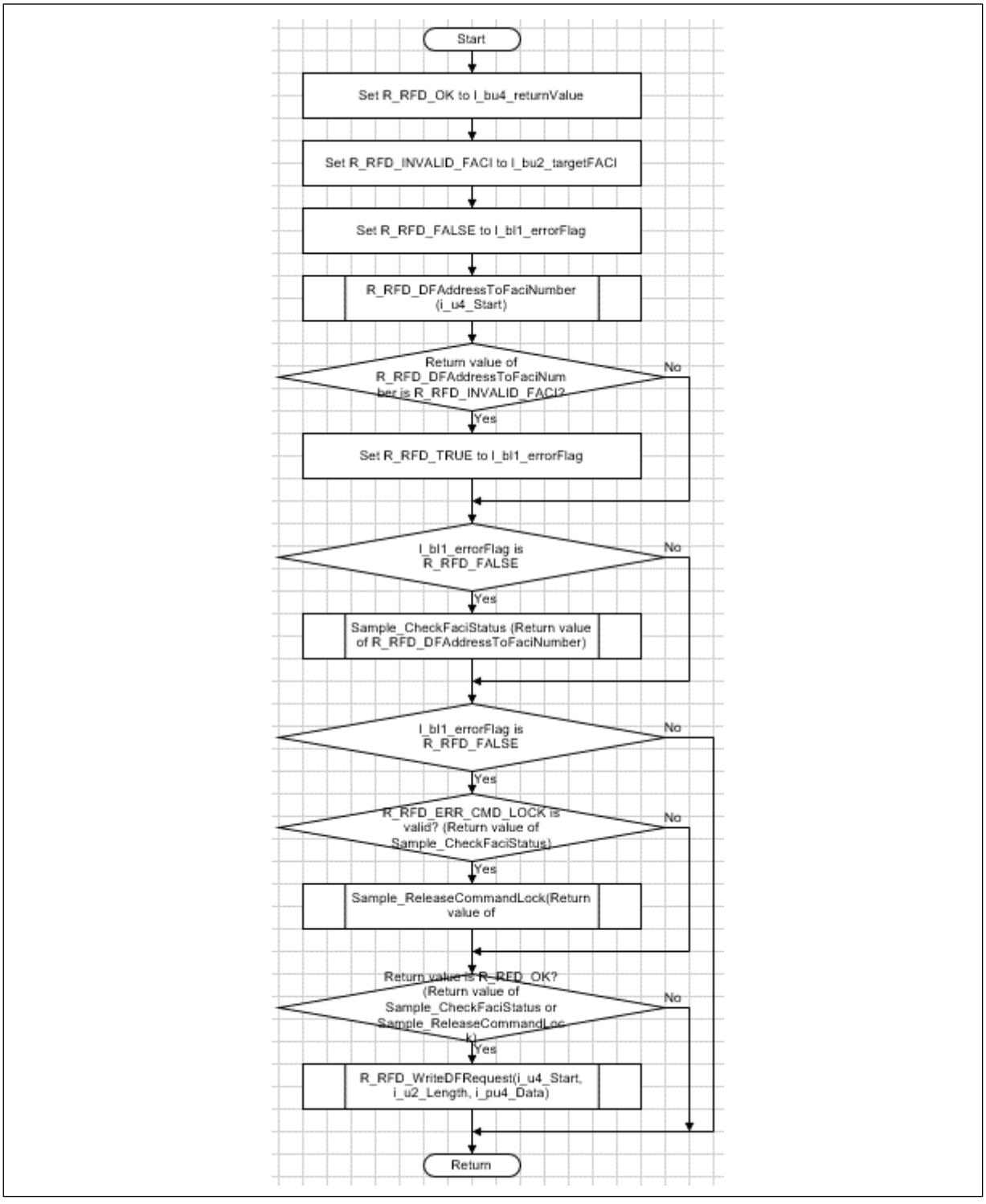
At this time, it checks whether the specified FACI number exists, and if not, it returns “R_RFD_ERR_NOFACI” as a return value.

If the return value of " Sample_CheckFaciStatus " is "R_RFD_STS_ERASE_SUSPENDED" or "R_RFD_STS_WRITE_SUSPENDED" or "R_RFD_STS_BUSY", it ends as a return value. If the return value is "R_RFD_OK", continue to execute Function3.

If the return value of “Sample_CheckFaciStatus” is not "R_RFD_STS_ERASE_SUSPENDED" or "R_RFD_STS_WRITE_SUSPENDED" or "R_RFD_STS_BUSY" or "R_RFD_OK" (In a state that represents a command lock), execute "Sample_ReleaseCommandLock". If the return value of "Sample_ReleaseCommandLock" is other than "R_RFD_OK", this function is terminated with the return value of “Sample_ReleaseCommandLock”.

Function3 Execute “R_RFD_WriteDFRequest” and proceed writing process.

Figure 4-7 Flow of “Sample_WriteDF”



4.3.2.3 Sample_BlankCheckDF

- Information

Syntax	`T_u4_RFDReturn Sample_BlankCheckDF(T_u4_RfdAddress i_u4_Start, T_u4_RfdAddress i_u4_End);`	
ASIL Level	-	
Status	New	
Sync/Async	Async	
Reentrancy	Non-Reentrant	
Parameters (IN)	`T_u4_RfdAddress`	Start address for erasure area
	`T_u4_RfdAddress`	End address for erasure area
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	`T_u4_RFDReturn`	R_RFD_OK
		R_RFD_STS_BUSY
		R_RFD_STS_ERASE_SUSPENDED
		R_RFD_STS_WRITE_SUSPENDED
		R_RFD_ERR_INTERNAL_HW
		R_RFD_ERR_NOFACI
Description	It checks whether there is free space in the data flash memory.	
Preconditions	FACI mode must be Data Flash P/E mode.	
Remarks	-	

・ Detail Specification

Function1 It checks from the address specified by the first parameter “i_u4_Start” to the address specified by the second parameter “i_u4_End” whether there is free space in the data flash memory.

Function2 Check whether the flash is in READY state or command lock state (Sample_CheckFaciStatus).

The FACI Number specified by parameter is executed by executing R_RFD_DFAddressToFaciNumber.

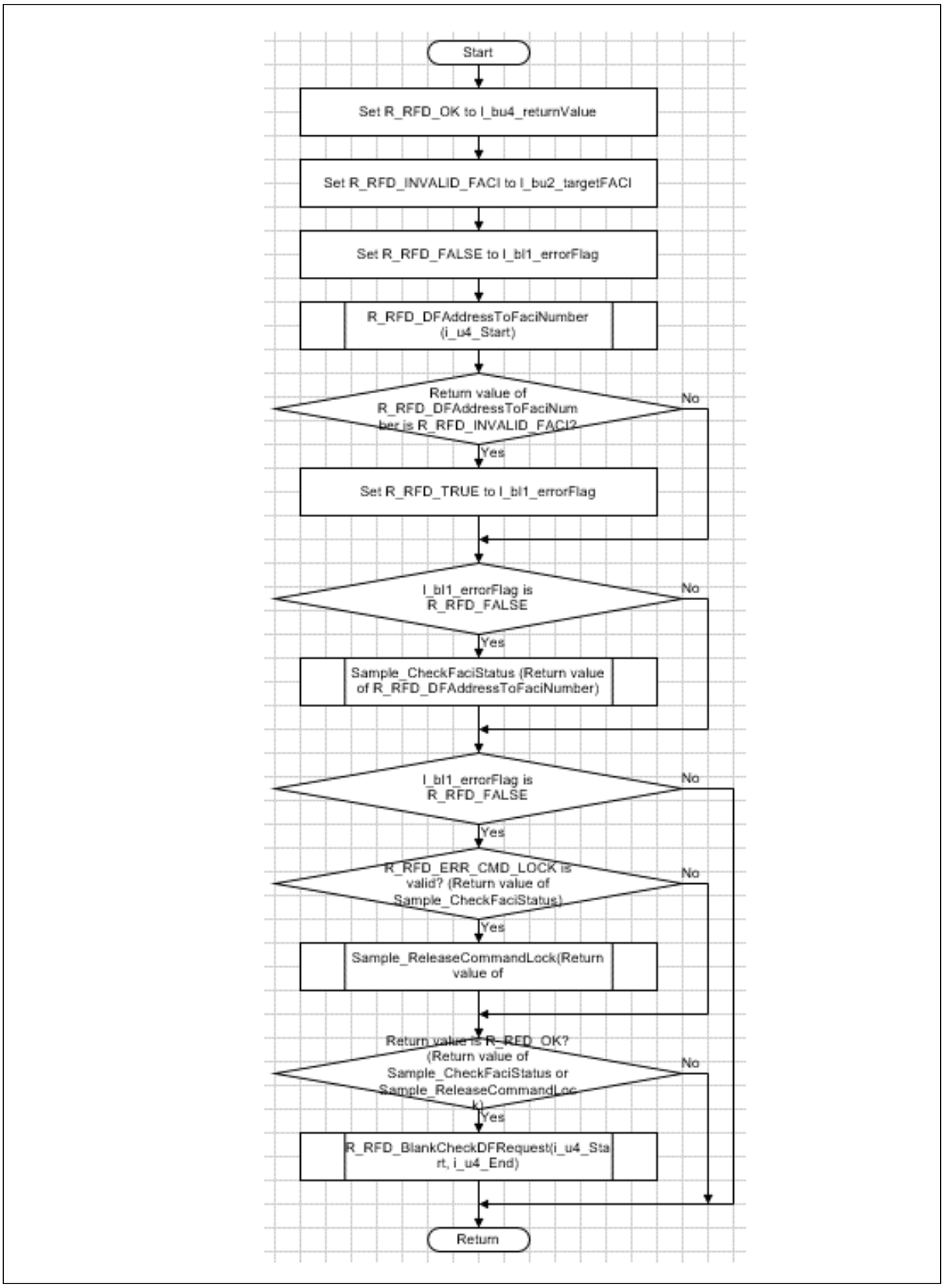
At this time, it checks whether the specified FACI number exists, and if not, it returns “R_RFD_ERR_NOFACI” as a return value.

If the return value of " Sample_CheckFaciStatus " is "R_RFD_STS_ERASE_SUSPENDED" or "R_RFD_STS_WRITE_SUSPENDED" or "R_RFD_STS_BUSY", it ends as a return value. If the return value is "R_RFD_OK", continue to execute Function3.

If the return value of “Sample_CheckFaciStatus” is not "R_RFD_STS_ERASE_SUSPENDED" or "R_RFD_STS_WRITE_SUSPENDED" or "R_RFD_STS_BUSY" or "R_RFD_OK" (In a state that represents a command lock), execute "Sample_ReleaseCommandLock". If the return value of "Sample_ReleaseCommandLock" is other than "R_RFD_OK", this function is terminated with the return value of “Sample_ReleaseCommandLock”.

Function3 Execute “R_RFD_BlankCheckDFRequest” and check blank area.

Figure 4-8 Flow of “Sample_BlankCheckDF”



4.3.3 For Code Flash Control

4.3.3.1 Sample_EraseCF

- Information

Syntax	`T_u4_RFDReturn Sample_EraseCF (T_u4_RfdAddress i_u4_Start);`	
ASIL Level	-	
Status	New	
Sync/Async	Async	
Reentrancy	Non-Reentrant	
Parameters (IN)	T_u4_RfdAddress	Start address for erasure area
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	T_u4_RFDReturn	R_RFD_OK
		R_RFD_STS_BUSY
		R_RFD_STS_ERASE_SUSPENDED
		R_RFD_STS_WRITE_SUSPENDED
		R_RFD_ERR_INTERNAL_HW
		R_RFD_ERR_NOFACI
Description	Erases the contents of the code flash memory.	
Preconditions	FACI mode must be Code Flash P/E mode.	
Remarks	-	

・ Detail Specification

Function1 The contents of the code flash memory are erased one block from the address specified by the first parameter "i_u4_Start".

Function2 Check whether the flash is in READY state or command lock state (Sample_CheckFaciStatus).

The FACI Number specified by parameter is executed by executing R_RFD_CFAddressToFaciNumber.

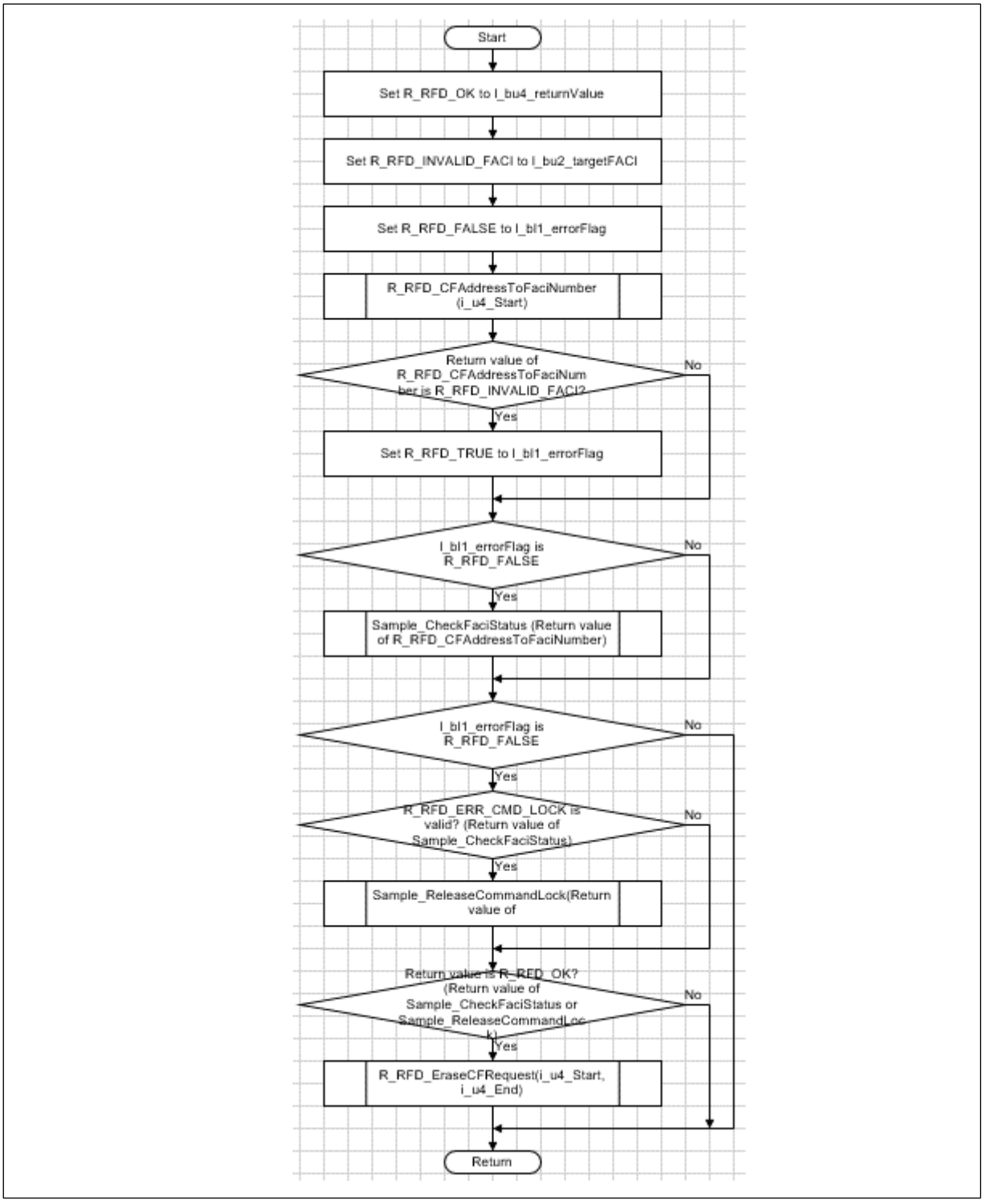
At this time, it checks whether the specified FACI number exists, and if not, it returns “R_RFD_ERR_NOFACI” as a return value.

If the return value of " Sample_CheckFaciStatus " is "R_RFD_STS_ERASE_SUSPENDED" or "R_RFD_STS_WRITE_SUSPENDED" or "R_RFD_STS_BUSY", it ends as a return value. If the return value is "R_RFD_OK", continue to execute Function3.

If the return value of “Sample_CheckFaciStatus” is not "R_RFD_STS_ERASE_SUSPENDED" or "R_RFD_STS_WRITE_SUSPENDED" or "R_RFD_STS_BUSY" or "R_RFD_OK" (In a state that represents a command lock), execute "Sample_ReleaseCommandLock". If the return value of "Sample_ReleaseCommandLock" is other than "R_RFD_OK", this function is terminated with the return value of “Sample_ReleaseCommandLock”.

Function3 Execute “R_RFD_EraseCFRequest” and proceed erasing process.

Figure 4-9 Flow of “Sample_EraseCF”



4.3.3.2 Sample_WriteCF

- Information

Syntax	`T_u4_RFDReturn Sample_WriteCF (T_u4_RfdAddress i_u4_Start, T_u2 i_u2_Length, T_pu4_RfdBuffer i_pu4_Data);`	
ASIL Level	-	
Status	New	
Sync/Async	Async	
Reentrancy	Non-Reentrant	
Parameters (IN)	`T_u4_RfdAddress`	Start address for writing area
	`T_u2`	Data for writing length
	`T_pu4_RfdBuffer`	Buffer address where write data is stored
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	`T_u4_RFDReturn`	R_RFD_OK
		R_RFD_STS_BUSY
		R_RFD_STS_ERASE_SUSPENDED
		R_RFD_STS_WRITE_SUSPENDED
		R_RFD_ERR_INTERNAL_HW
		R_RFD_ERR_NOFACI
Description	Writes the data to the code flash memory.	
Preconditions	FACI mode must be Code Flash P/E mode.	
Remarks	-	

・ Detail Specification

Function1 Writes the data to the code flash memory stored in the third parameter "i_pu4_Data" from the address indicated by the first parameter “i_u4_Start” by the amount of size indicated by the second parameter “i_u2_Length”.

Function2 Check whether the flash is in READY state or command lock state (Sample_CheckFaciStatus).

The FACI Number specified by parameter is executed by executing R_RFD_CFAddressToFaciNumber.

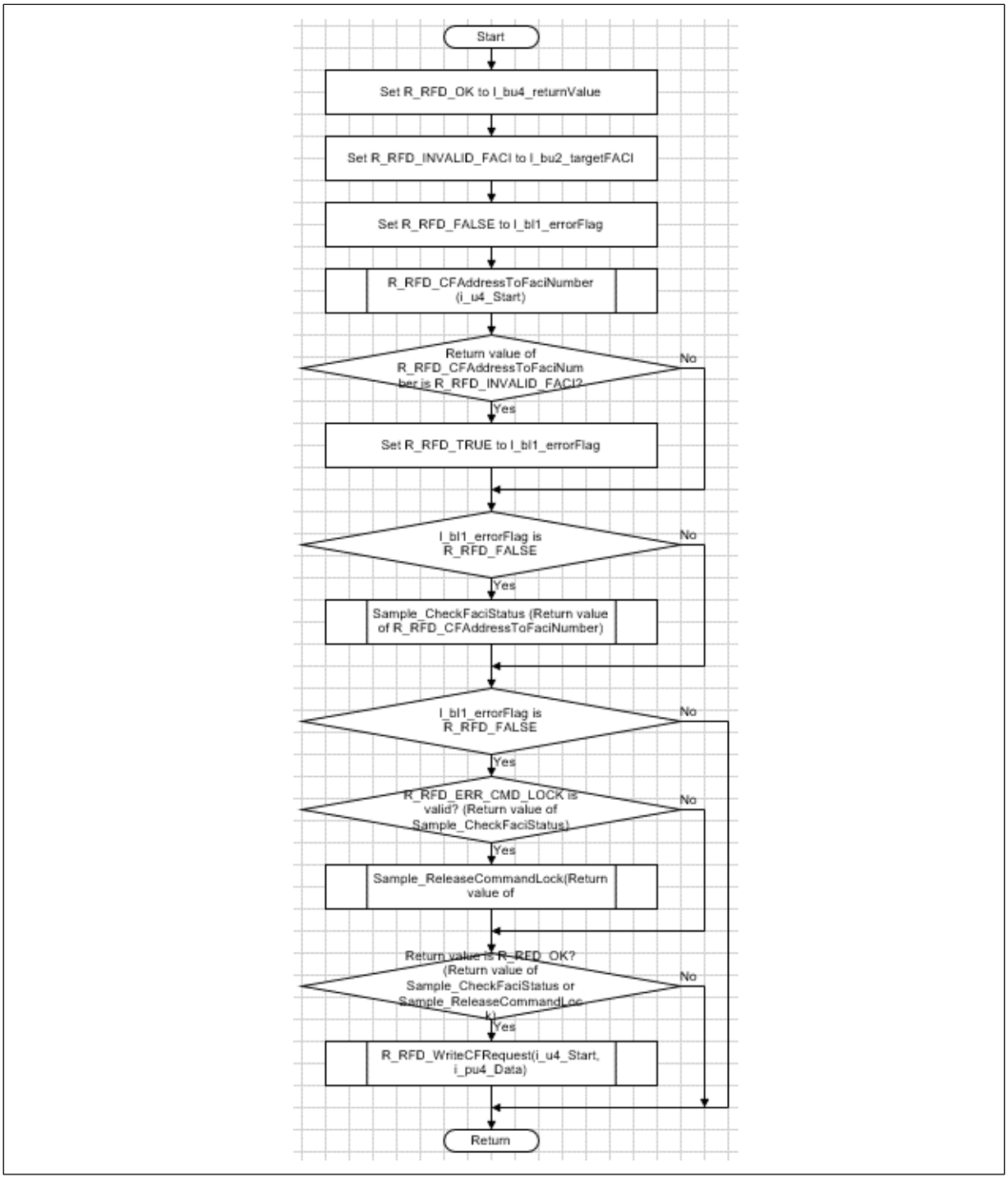
At this time, it checks whether the specified FACI number exists, and if not, it returns “R_RFD_ERR_NOFACI” as a return value.

If the return value of " Sample_CheckFaciStatus " is "R_RFD_STS_ERASE_SUSPENDED" or "R_RFD_STS_WRITE_SUSPENDED" or "R_RFD_STS_BUSY", it ends as a return value. If the return value is "R_RFD_OK", continue to execute Function3.

If the return value of “Sample_CheckFaciStatus” is not "R_RFD_STS_ERASE_SUSPENDED" or "R_RFD_STS_WRITE_SUSPENDED" or "R_RFD_STS_BUSY" or "R_RFD_OK" (In a state that represents a command lock), execute "Sample_ReleaseCommandLock". If the return value of "Sample_ReleaseCommandLock" is other than "R_RFD_OK", this function is terminated with the return value of “Sample_ReleaseCommandLock”.

Function3 Execute “R_RFD_WriteCFRequest” and proceed writing process.

Figure 4-10 Flow of “Sample_WriteCF”



4.3.4 For Hardware Property Control

4.3.4.1 Sample_EraseConfigurationSettingArea

- Information

Syntax	`T_u4_RFDReturn Sample_EraseConfigurationSettingArea (T_u4_RfdAddress i_u4_Start);`	
ASIL Level	-	
Status	New	
Sync/Async	Async	
Reentrancy	Non-Reentrant	
Parameters (IN)	`T_u4_RfdAddress`	Start address for Configuration Setting Area
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	`T_u4_RFDReturn`	R_RFD_OK
		R_RFD_STS_BUSY
		R_RFD_STS_ERASE_SUSPENDED
		R_RFD_STS_WRITE_SUSPENDED
		R_RFD_ERR_INTERNAL_HW
		R_RFD_ERR_NOFACI
		R_RFD_ERR_REJECT
Description	Erases the contents of the Configuration Setting Area within Hardware Property Area.	
Preconditions	FACI mode must be Data Flash P/E mode.	
Remarks	-	

・ Detail Specification

Function1 Erases the contents of Configuration Setting Area in the Hardware Property Area. The first parameter "i_u4_Start" is assumed to be the start address of Configuration Setting Area.

Function2 Check whether the flash is in READY state or command lock state (Sample_CheckFaciStatus).

The FACI Number specified by parameter is executed by executing R_RFD_PropertyAddressToFaciNumber.

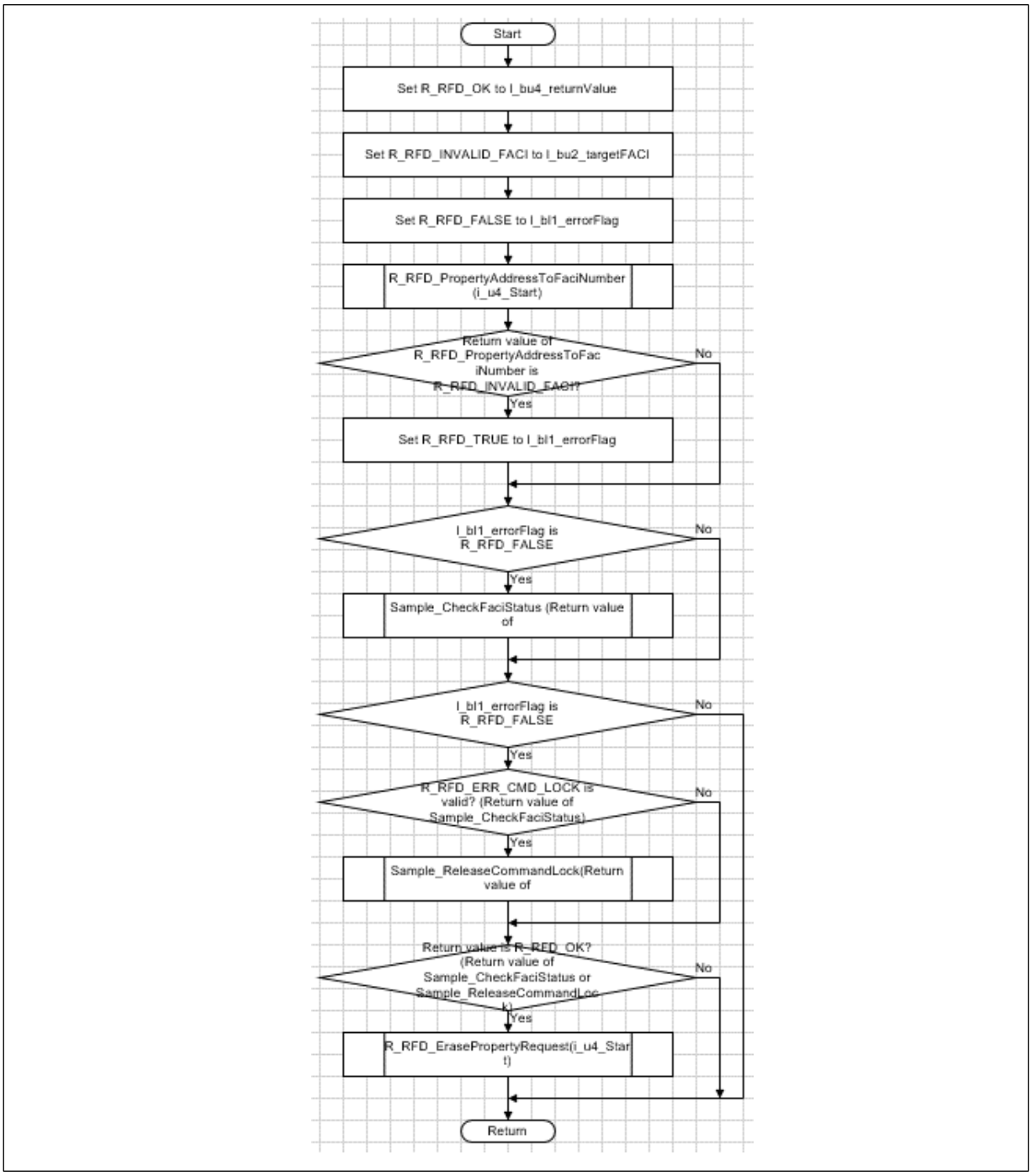
At this time, it checks whether the specified FACI number exists, and if not, it returns “R_RFD_ERR_NOFACI” as a return value.

If the return value of " Sample_CheckFaciStatus " is "R_RFD_STS_ERASE_SUSPENDED" or "R_RFD_STS_WRITE_SUSPENDED" or "R_RFD_STS_BUSY", it ends as a return value. If the return value is "R_RFD_OK", continue to execute Function3.

If the return value of “Sample_CheckFaciStatus” is not "R_RFD_STS_ERASE_SUSPENDED" or "R_RFD_STS_WRITE_SUSPENDED" or "R_RFD_STS_BUSY" or "R_RFD_OK" (In a state that represents a command lock), execute "Sample_ReleaseCommandLock". If the return value of "Sample_ReleaseCommandLock" is other than "R_RFD_OK", this function is terminated with the return value of “Sample_ReleaseCommandLock”.

Function3 Execute “R_RFD_ErasePropertyRequest” and proceed erasing process.

Figure 4-11 Flow of “Sample_EraseConfigurationSettingArea”



4.3.4.2 Sample_WriteConfigurationSettingArea

- Information

Syntax	`T_u4_RFDReturn Sample_WriteConfigurationSettingArea (T_u4_RfdAddress i_u4_Start, T_pu4_RfdBuffer i_pu4_Data);`	
ASIL Level	-	
Status	New	
Sync/Async	Async	
Reentrancy	Non-Reentrant	
Parameters (IN)	`T_u4_RfdAddress`	Start address for Configuration Setting Area
	`T_pu4_RfdBuffer`	Buffer address where write data is stored
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	`T_u4_RFDReturn`	R_RFD_OK
		R_RFD_STS_BUSY
		R_RFD_STS_ERASE_SUSPENDED
		R_RFD_STS_WRITE_SUSPENDED
		R_RFD_ERR_INTERNAL_HW
		R_RFD_ERR_NOFACI
Description	Writes the contents of the Configuration Setting Area within Hardware Property Area.	
Preconditions	FACI mode must be Data Flash P/E mode.	
Remarks	-	

・ Detail Specification

Function1 The 32 bytes of data stored in the second parameter "i_pu4_Data" are written to the address of the Configuration Setting Area indicated by the first parameter "i_u4_Start".

Function2 Check whether the flash is in READY state or command lock state (Sample_CheckFaciStatus).

The FACI Number specified by parameter is executed by executing R_RFD_PropertyAddressToFaciNumber.

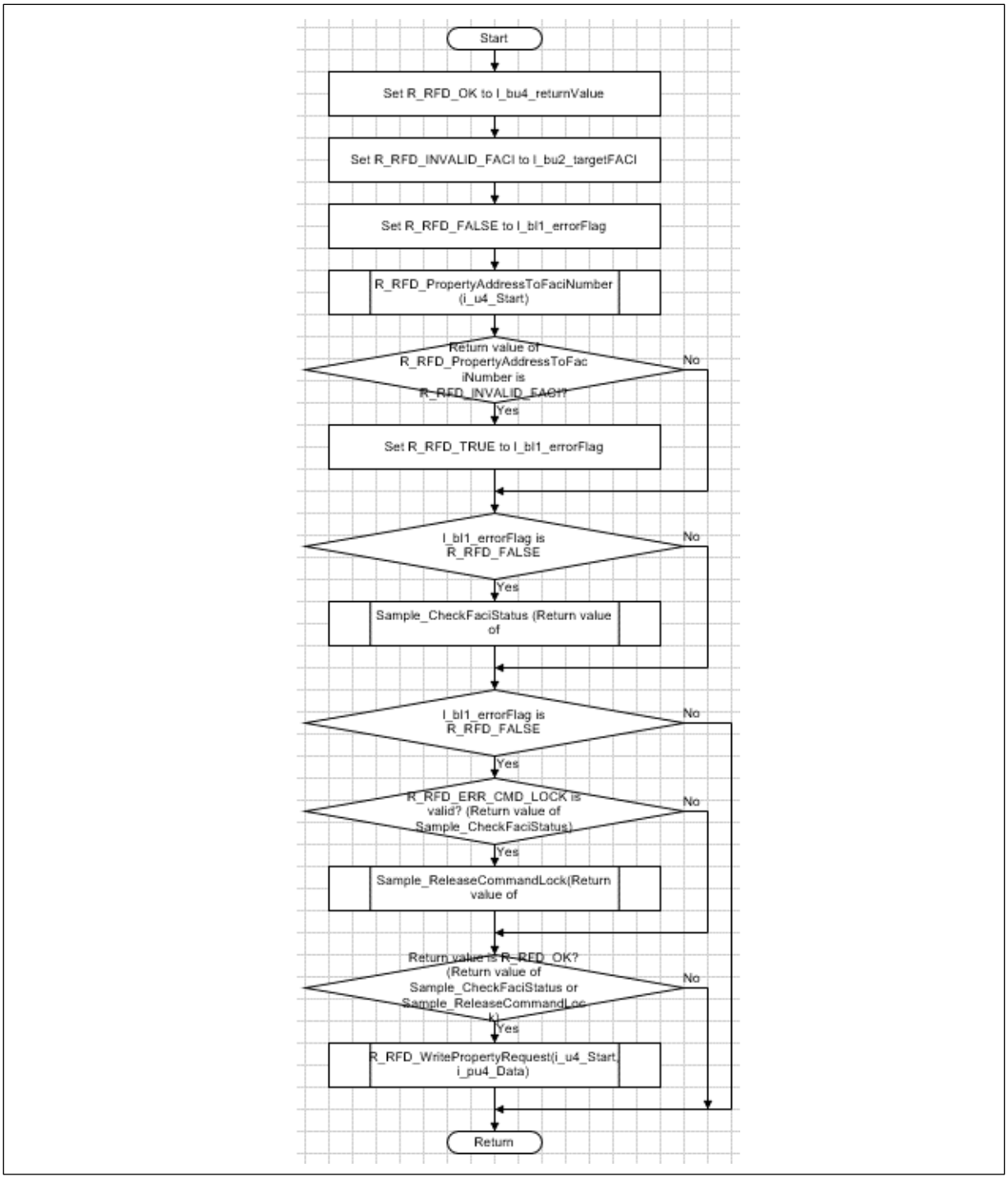
At this time, it checks whether the specified FACI number exists, and if not, it returns “R_RFD_ERR_NOFACI” as a return value.

If the return value of " Sample_CheckFaciStatus " is "R_RFD_STS_ERASE_SUSPENDED" or "R_RFD_STS_WRITE_SUSPENDED" or "R_RFD_STS_BUSY", it ends as a return value. If the return value is "R_RFD_OK", continue to execute Function3.

If the return value of “Sample_CheckFaciStatus” is not "R_RFD_STS_ERASE_SUSPENDED" or "R_RFD_STS_WRITE_SUSPENDED" or "R_RFD_STS_BUSY" or "R_RFD_OK" (In a state that represents a command lock), execute "Sample_ReleaseCommandLock". If the return value of "Sample_ReleaseCommandLock" is other than "R_RFD_OK", this function is terminated with the return value of “Sample_ReleaseCommandLock”.

Function3 Execute “R_RFD_WritePropertyRequest” and proceed writing process.

Figure 4-12 Flow of “Sample_WriteConfigurationSettingArea”



4.3.4.3 Sample_EraseSwitchArea

- Information

Syntax	`T_u4_RFDReturn Sample_EraseSwitchArea(void);`	
ASIL Level	-	
Status	New	
Sync/Async	Async	
Reentrancy	Non-Reentrant	
Parameters (IN)	N/A	
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	`T_u4_RFDReturn`	R_RFD_OK
		R_RFD_STS_BUSY
		R_RFD_STS_ERASE_SUSPENDED
		R_RFD_STS_WRITE_SUSPENDED
		R_RFD_ERR_INTERNAL_HW
		R_RFD_ERR_NOFACI
		R_RFD_ERR_SWITCH_DATA
Description	Erases the contents of the Switch Area within Hardware Property Area.	
Preconditions	FACI mode must be Data Flash P/E mode.	
Remarks	-	

・ Detail Specification

Function1 Erases the contents of Switch Area in the Hardware Property Area.

Function2 Check whether the flash is in READY state or command lock state (Sample_CheckFaciStatus).

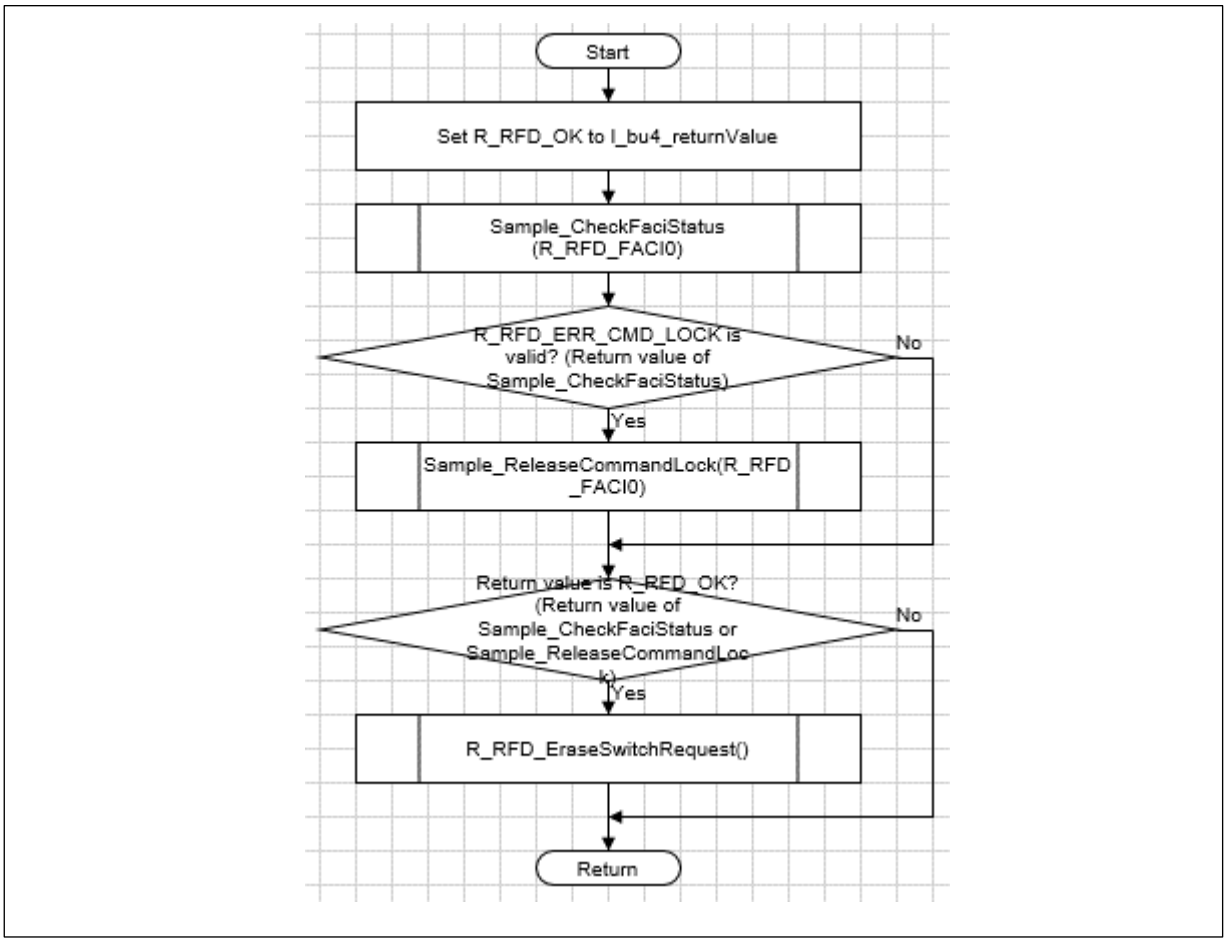
The FACI number specified in the parameter is fixed to FACI0 (R_RFD_FACI0).

If the return value of " Sample_CheckFaciStatus " is "R_RFD_STS_ERASE_SUSPENDED" or "R_RFD_STS_WRITE_SUSPENDED" or "R_RFD_STS_BUSY", it ends as a return value. If the return value is "R_RFD_OK", continue to execute Function3.

If the return value of “Sample_CheckFaciStatus” is not "R_RFD_STS_ERASE_SUSPENDED" or "R_RFD_STS_WRITE_SUSPENDED" or "R_RFD_STS_BUSY" or "R_RFD_OK" (In a state that represents a command lock), execute "Sample_ReleaseCommandLock". If the return value of "Sample_ReleaseCommandLock" is other than "R_RFD_OK", this function is terminated with the return value of “Sample_ReleaseCommandLock”.

Function3 Execute “R_RFD_EraseSwitchRequest” and proceed erasing process.

Figure 4-13 Flow of “Sample_EraseSwitchArea”



4.3.4.4 Sample_WriteSwitchArea

- Information

Syntax	`T_u4_RFDReturn Sample_WriteSwitchArea (T_pu4_RfdBuffer i_pu4_Data);`	
ASIL Level	-	
Status	New	
Sync/Async	Async	
Reentrancy	Non-Reentrant	
Parameters (IN)	`T_u4_RfdAddress`	Buffer address where write data is stored
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	`T_u4_RFDReturn`	R_RFD_OK
		R_RFD_STS_BUSY
		R_RFD_STS_ERASE_SUSPENDED
		R_RFD_STS_WRITE_SUSPENDED
		R_RFD_ERR_INTERNAL_HW
		R_RFD_ERR_NOFACI
Description	Writes the contents of the Switch Area within Hardware Property Area.	
Preconditions	FACI mode must be Data Flash P/E mode.	
Remarks	-	

・ Detail Specification

Function1 Writes the data stored in the third parameter "i_pu4_Data" from the address indicated by the first parameter “i_u4_Start” by the amount of size indicated by the second parameter “i_u2_Length”.

Function2 Check whether the flash is in READY state or command lock state (Sample_CheckFaciStatus).

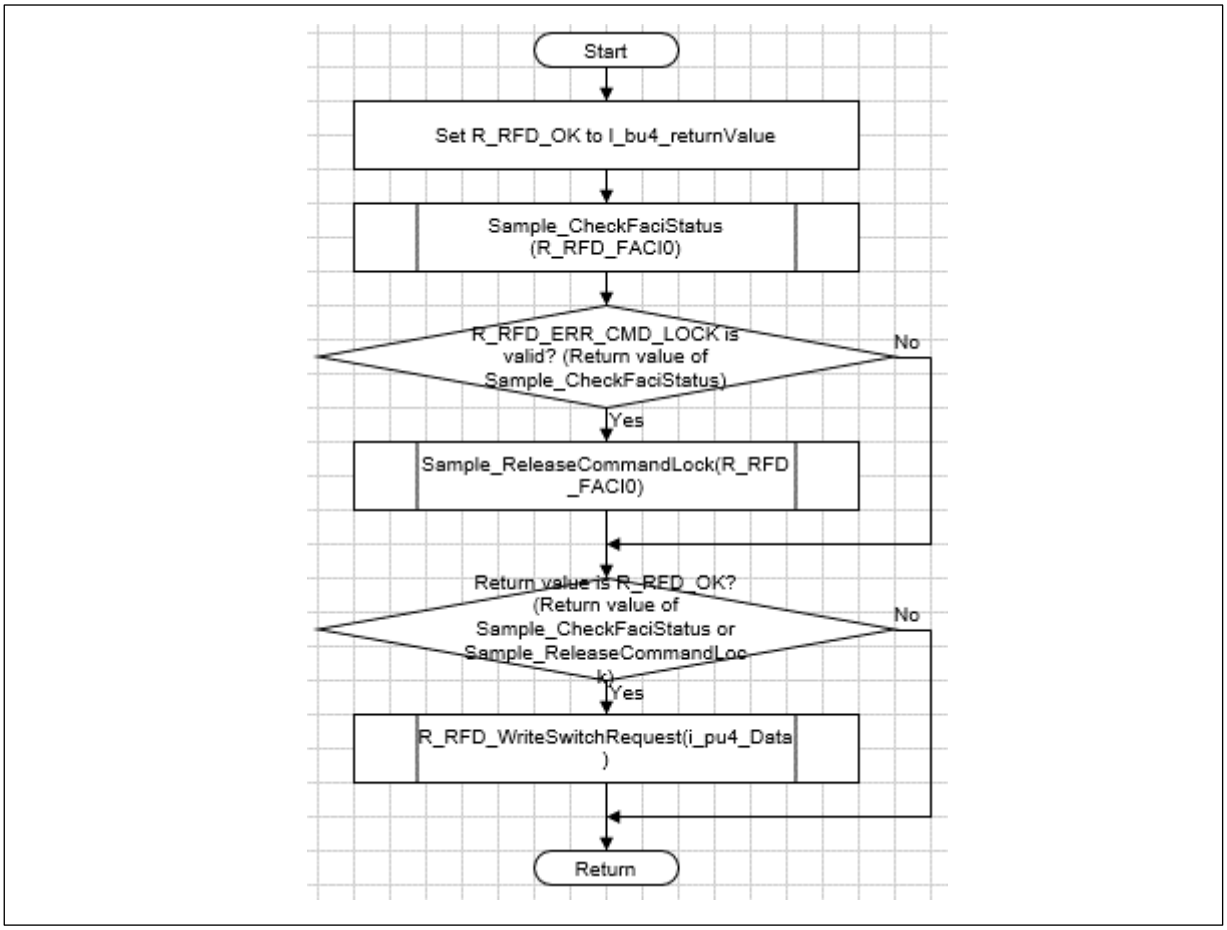
The FACI number specified in the parameter is fixed to FACI0 (R_RFD_FACI0).

If the return value of " Sample_CheckFaciStatus " is "R_RFD_STS_ERASE_SUSPENDED" or "R_RFD_STS_WRITE_SUSPENDED" or "R_RFD_STS_BUSY", it ends as a return value. If the return value is "R_RFD_OK", continue to execute Function3.

If the return value of “Sample_CheckFaciStatus” is not "R_RFD_STS_ERASE_SUSPENDED" or "R_RFD_STS_WRITE_SUSPENDED" or "R_RFD_STS_BUSY" or "R_RFD_OK" (In a state that represents a command lock), execute "Sample_ReleaseCommandLock". If the return value of "Sample_ReleaseCommandLock" is other than "R_RFD_OK", this function is terminated with the return value of “Sample_ReleaseCommandLock”.

Function3 Execute “R_RFD_WriteSwitchRequest” and proceed writing process.

Figure 4-14 Flow of “Sample_WriteSwitchArea”



4.3.4.5 Sample_EraseTagArea

- Information

Syntax	`T_u4_RFDReturn Sample_EraseTagArea(void);`	
ASIL Level	-	
Status	New	
Sync/Async	Async	
Reentrancy	Non-Reentrant	
Parameters (IN)	N/A	
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	T_u4_RFDReturn	R_RFD_OK
		R_RFD_STS_BUSY
		R_RFD_STS_ERASE_SUSPENDED
		R_RFD_STS_WRITE_SUSPENDED
		R_RFD_ERR_SWITCH_DATA
		R_RFD_ERR_CONFIGURATION_DATA
		R_RFD_ERR_SECURITY_DATA
		R_RFD_ERR_BLOCKPROTECTION0_DATA
		R_RFD_ERR_BLOCKPROTECTION1_DATA
Description	Erases the contents of the Tag Area within Hardware Property Area.	
Preconditions	FACI mode must be Data Flash P/E mode.	
Remarks	-	

・ Detail Specification

Function1 Erases the contents of Tag Area in the Hardware Property Area.

Function2 Check whether the flash is in READY state or command lock state (Sample_CheckFaciStatus).

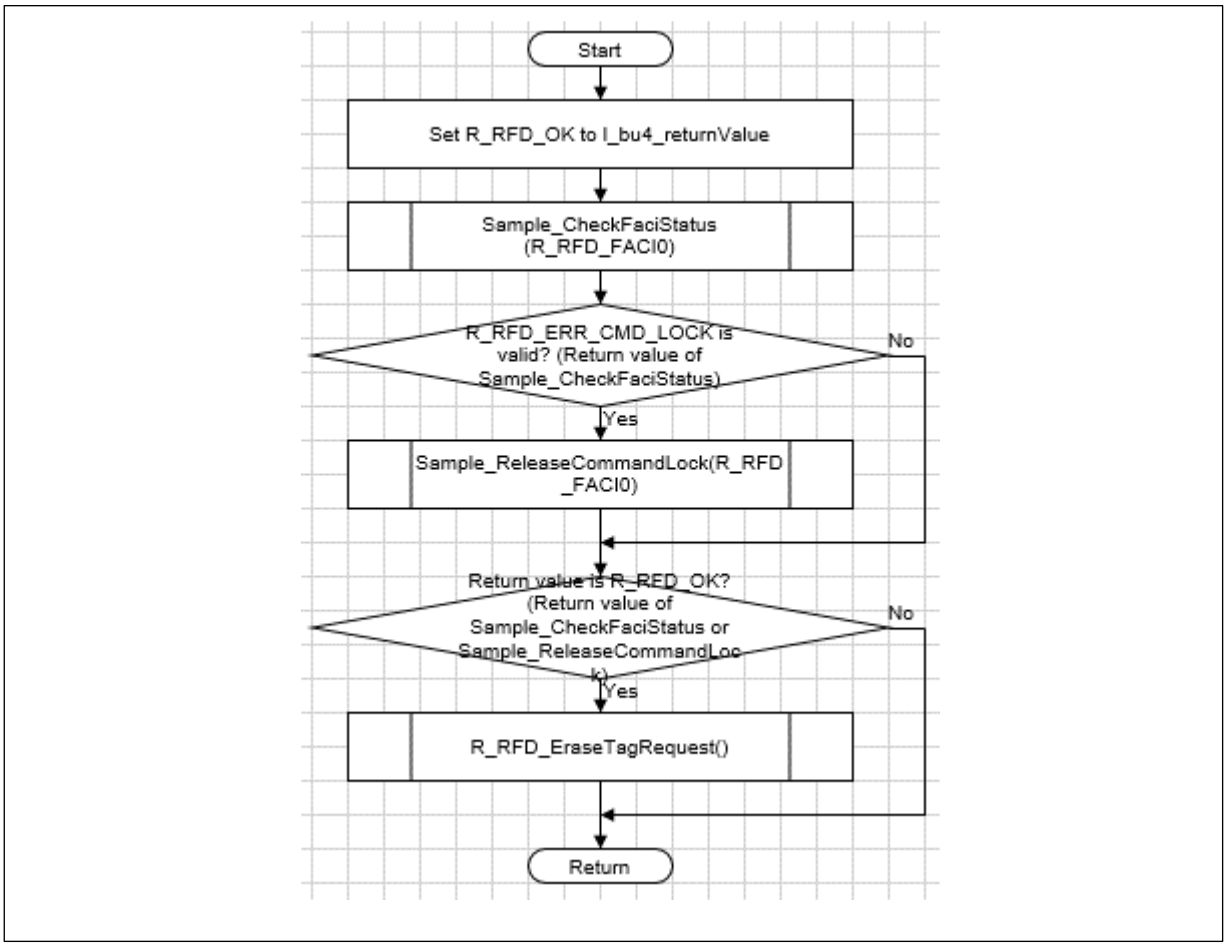
The FACI number specified in the parameter is fixed to FACI0 (R_RFD_FACI0).

If the return value of " Sample_CheckFaciStatus " is "R_RFD_STS_ERASE_SUSPENDED" or "R_RFD_STS_WRITE_SUSPENDED" or "R_RFD_STS_BUSY", it ends as a return value. If the return value is "R_RFD_OK", continue to execute Function3.

If the return value of “Sample_CheckFaciStatus” is not "R_RFD_STS_ERASE_SUSPENDED" or "R_RFD_STS_WRITE_SUSPENDED" or "R_RFD_STS_BUSY" or "R_RFD_OK" (In a state that represents a command lock), execute "Sample_ReleaseCommandLock". If the return value of "Sample_ReleaseCommandLock" is other than "R_RFD_OK", this function is terminated with the return value of “Sample_ReleaseCommandLock”.

Function3 Execute “R_RFD_EraseTagRequest” and proceed erasing process.

Figure 4-15 Flow of “Sample_EraseTagArea”



4.3.4.6 Sample_UpdateTagArea

- Information

Syntax	`T_u4_RFDReturn Sample_UpdateTagArea(void)`	
ASIL Level	-	
Status	New	
Sync/Async	Async	
Reentrancy	Non-Reentrant	
Parameters (IN)	N/A	
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	T_u4_RFDReturn	R_RFD_OK
		R_RFD_STS_BUSY
		R_RFD_STS_ERASE_SUSPENDED
		R_RFD_STS_WRITE_SUSPENDED
		R_RFD_ERR_SWITCH_DATA
		R_RFD_ERR_CONFIGURATION_DATA
		R_RFD_ERR_SECURITY_DATA
		R_RFD_ERR_BLOCKPROTECTION0_DATA
		R_RFD_ERR_BLOCKPROTECTION1_DATA
Description	Updates the Tag Area within Hardware Property Area.	
Preconditions	FACI mode must be Data Flash P/E mode.	
Remarks	-	

・ Detail Specification

Function1 Updates the Tag Area of Hardware Property Area.

Function2 Check whether the flash is in READY state or command lock state (Sample_CheckFaciStatus).

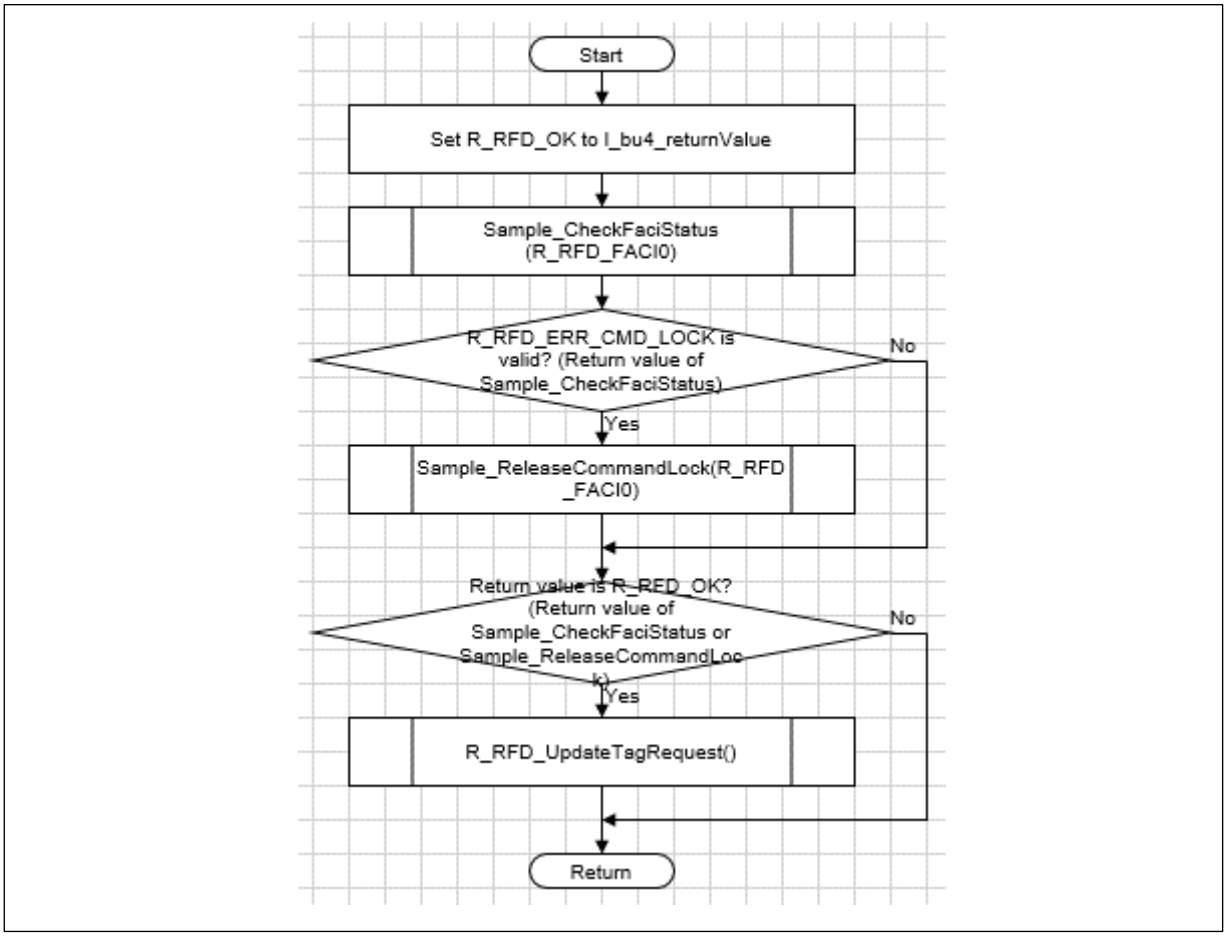
The FACI number specified in the parameter is fixed to FACI0 (R_RFD_FACI0).

If the return value of " Sample_CheckFaciStatus " is "R_RFD_STS_ERASE_SUSPENDED" or "R_RFD_STS_WRITE_SUSPENDED" or "R_RFD_STS_BUSY", it ends as a return value. If the return value is "R_RFD_OK", continue to execute Function3.

If the return value of “Sample_CheckFaciStatus” is not "R_RFD_STS_ERASE_SUSPENDED" or "R_RFD_STS_WRITE_SUSPENDED" or "R_RFD_STS_BUSY" or "R_RFD_OK" (In a state that represents a command lock), execute "Sample_ReleaseCommandLock". If the return value of "Sample_ReleaseCommandLock" is other than "R_RFD_OK", this function is terminated with the return value of “Sample_ReleaseCommandLock”.

Function3 Execute “R_RFD_UpdateTagRequest” and proceed updating process.

Figure 4-16 Flow of “Sample_UpdateTagArea”



4.4 Operation of the Sample Programs

The following describes the operation of the sample program. The relationship between function names and recorded file names is as follows.

Function name	File name
main	main_pe0.c
Sample_DataFlashControl	sample_control_dataflash.c
Sample_SuspendResume	sample_control_common.c
Sample_CodeFlashControl	sample_control_codeflash.c
Sample_PropertyAreaControl	sample_control_property.c

4.4.1 main

The process of the function are as follows.

[Common process].

1. Execute R_RFD_Init
   - Initialize the RFD28F
2. Change the FHVE protect to OFF

[For Data Flash Control].

Execute the following processing 3 to 6 when R_RFD_CONTROL_TARGET_DATAFLASH of configuration is set to R_RFD_ENABLE

3. Create data to be written
4. Execute R_RFD_DFIDAuth to authenticate the Data Flash ID
5. Call Sample_DataFlashControl function
6. Store the result of Sample_DataFlashControl function to return value
   if R_RFD_OK is returned, TRUE is as return value, and if other than R_RFD_OK is returned, FALSE is as return value, use these values as the return value of this function.

[Code Flash Control]

Execute the following processing 7 to 11 when R_RFD_CONTROL_TARGET_CODEFLASH of configuration is set to R_RFD_ENABLE

7. Set the mode to permit code flash P/E mode shifting to the FLMD control register
8. Execute R_RFD_IDAuth to authenticate the Customer ID A, Customer ID B, Customer ID C.
9. Call Sample_CodeFlashControl function
10. Store the result of Sample_CodeFlashControl function to return value
    if R_RFD_OK is returned, TRUE is as return value, and if other than R_RFD_OK is returned, FALSE is as return value, use these values as the return value of this function.
11. Set the mode to disable code flash P/E mode shifting to the FLMD control register

[Property Area Control]

Execute the following processing 12 to 13 when R_RFD_CONTROL_TARGET_CODEFLASH of configuration is set to R_RFD_ENABLE

12. Call Sample_PropertyAreaControl function
13. Store the result of Sample_PropertyAreaControl function to return value
    if R_RFD_OK is returned, TRUE is as return value, and if other than R_RFD_OK is returned, FALSE is as return value, use these values as the return value of this function.

[Common process].

14. Change the FHVE protect to ON
15. Terminate main function

4.4.2 Sample_DataFlashControl

The process of the function are as follows.

1. Shift the corresponding FACI to data flash P/E mode (R_RFD_ShiftToPEMode)
   - Confirm whether you have switched to P/E mode (R_RFD_CheckPEMode)
2. Erase the contents of the data flash memory (Sample_EraseDF)
   - Erase suspension processing and resumption processing are executed immediately after erasure start (Sample_SuspendResume)
   - Confirm whether the erase process is completed (Sample_CheckFaciStatus)
3. Check whether there is blank in the data flash memory (Sample_BlankCheckDF)
   - Check whether blank check is completed (Sample_CheckFaciStatus)
   - Acquire free space confirmation result (R_RFD_GetBlankCheckResult)
4. Write the data to the data flash memory (Sample_WriteDF)
   - Check whether writing process is completed (Sample_CheckFaciStatus)
5. Check the blank of the written block and check that there is no free space (Sample_BlankCheckDF)
6. Shift to read mode for data flash corresponding FACI (R_RFD_ShiftToReadMode)

4.4.3 Sample_SuspendResume

The process of the function are as follows.

1. Suspend the target FACI processing (Sample_SuspendPE)
2. Shift to read mode (R_RFD_ShiftToReadMode)
3. Processing during suspension status (There is no processing in this sample)
4. Shift to P/E mode (R_RFD_ShiftToPEMode)
5. Resume the suspension process for the target FACI (Sample_ResumePE)

4.4.4 Sample_CodeFlashControl

The process of the function are as follows.

1. Shift the corresponding FACI to code flash P/E mode (R_RFD_ShiftToPEMode)
   - Confirm whether you have switched to P/E mode (R_RFD_CheckPEMode)
2. Erase the contents of the code flash memory (Sample_EraseCF)
   - Erase suspension processing and resumption processing are executed immediately after erasure start (Sample_SuspendResume)
   - Confirm whether the erase process is completed (Sample_CheckFaciStatus)
3. Write the data to the code flash memory (Sample_WriteCF)
   - Check whether writing process is completed (Sample_CheckFaciStatus)
4. Shift to read mode for code flash corresponding FACI (R_RFD_ShiftToReadMode)
   - If the command is locked, release the command lock state (Sample_ReleaseCommandLock)

4.4.5 Sample_PropertyAreaControl

The process of the function are as follows. 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.

1. Read the all contents from Configuration Setting Area to temporary (Read the value of offset 0x200-0x21F, 0x300-0x47F from the top of Configuration Setting Area).

   (In fact, for the element of Configuration Setting Area that you want to rewrite here, write the value after rewriting to the corresponding part in the temporary. In the sample, the value is not changed and the read value is written as it is.)

2. Shift the corresponding FACI to data flash P/E mode (R_RFD_ShiftToPEMode)
   - Confirm whether you have switched to P/E mode (R_RFD_CheckPEMode)
3. Erase the contents of the Configuration Setting Area (Sample_EraseConfigurationSettingArea)
4. Write the data to the Write the data to the Configuration Setting Area (Sample_WriteConfigurationSettingArea)
   - Check whether writing process is completed (Sample_CheckFaciStatus)
5. Read the values of CFGVA, SECVA, BPVA0 and BPVA1 of the FSWASTAT register to obtain the area indicating the current status.
6. Erase the contents of the Switch Area (Sample_EraseSwitchArea)
7. Write data to Switch Area (Sample_WriteSwitchArea). Set the following values for the parameter i_pu4_Data at this time

i_pu4_Data[0]	If the CFGVA value in the FSWASTAT register is 0, set 0x5AA5A55A, if it is 1, set 0xA55A5AA5.
i_pu4_Data[1]	If the SECVA value in the FSWASTAT register is 0, set 0xA55A5AA5, if it is 1, set 0x5AA5A55A.
i_pu4_Data[2]	If the BPVA0 value in the FSWASTAT register is 0, set 0xA55A5AA5, if it is 1, set 0x5AA5A55A.
i_pu4_Data[3]	If the BPVA1 value in the FSWASTAT register is 0, set 0xA55A5AA5, if it is 1, set 0x5AA5A55A.
i_pu4_Data[4]	0xFFFFFFFF
i_pu4_Data[5]	0xFFFFFFFF
i_pu4_Data[6]	0xFFFFFFFF
i_pu4_Data[7]	0xFFFFFFFF

8. Erase the contents of the Tag Area (Sample_EraseTagArea)
9. Update the Tag Area (Sample_UpdateTagArea)
10. Shift to read mode for data flash corresponding FACI (R_RFD_ShiftToReadMode)
    - If the command is locked, release the command lock state (Sample_ReleaseCommandLock)

5 Note for RFD28F

5.1 Common to all Components

5.1.1 Verification after programming

RFD28F doesn't confirm by read verify after programming, please check by user application as necessary

5.1.2 Cancel while executing command

If the RFD28F command is executed and operation to the flash memory starts, it should be canceled using R_RFD_ForcedStopAndErrorClear or R_RFD_Init.

5.1.3 Erasure and programming when suspension process

When suspension process, erasure and programming condition have restrictions. Please refer to User’s manual of the hardware about detail.

5.1.4 Termination/Suspension of processing

If the programming or erasing process to the Data Flash Memory or Code Flash Memory is terminate or suspend, the programming data or erasing data is unfixed value. In that case, avoid instruction fetch or data read for the relevant location.

5.1.5 Restriction on access to the flash memory

Flash memory access during program/erase has restriction. For details, refer to the section "Flash Memory" in the hardware user's manual. In order to perform access with accessible combinations, it shall be needed to arbitrate the access timing appropriately.

5.1.6 Initialization before any flash commands execution

RFD28F initialization API (R_RFD_Init) shall be executed before executing the RFD28F APIs.

5.1.7 Frequency setting

The correct frequency shall be set to parameter of R_RFD_init. Refer to User's manual of the hardware about suitably frequency setting of flash operating.

5.1.8 Processing for internal errors

If an internal error occurs, the RFD28F function cannot be executed in that state. It should be needed to take measures such as re-initializing the RFD28F.

5.1.9 Reserved area of flash memory

Depending on the target hardware, a range of Flash Memory may be reserved for special use. the user application cannot operate on that part. For the range of reserved flash memory, refer to the "Flash Memory" chapter in the hardware User's manual.

5.1.10 Reentrancy of RFD28F function

The user application shall not make reentrant call to RFD28F function. Especially when calling the API or the RFD28F from the interrupt handler, be careful.

5.1.11 Nested call of RFD28F functions (task switch, context change)

The user application shall not nest call RFD28F function to avoid synchronization problem of the flash memory operation. In other words, execute the RFD28F function in one task, and shall not execute the RFD28F function further in the state where execution of the RFD28F function is not completed.

5.1.12 Transition to power save mode

The user application does not transfer to the power save mode while executing the RFD28F function. When the flash memory operation is no longer in the BUSY state, you can switch to the power save mode.

5.1.13 Timeout of FACI sequencer operating

RFD28F doesn't check FACI sequencer operating time. If doesn't completed FACI sequencer operating beyond the Hardware specification, please call the R_RFD_ForcedStopAndErrorClear and terminate FACI sequencer operating.

5.1.14 Instance of RFD28F

One RFD28F instance handles at least one FACI. In other words, an instance of RFD28F can exist for each assignment FACI.

5.1.15 Synchronization of FACIs between Security Module (ICU-M)

It is possible to synchronize between PE and ICU-M. For the processing contents and processing procedure, refer to the chapter of "Flash Memory" in the hardware user's manual (Security).

5.1.16 About data retention

For data retention period, refer to ”Electrical Characteristics" in the Hardware user's manual.

5.1.17 About reset during programming/erasing

In the case of an external reset during programming and erasure, it shall be waited for at least width of reset pulse more than the min value by an external reset once the operating voltage is within the range stipulated in the electrical characteristics after assertion of the reset signal before releasing the device from the reset state. Refer to ”Electrical Characteristics" in the Hardware user's manual.

5.1.18 Return value of the initialization function of the RFD28F

If timeout occurs while executing the initialization function of RFD28F, the error "R_RFD_ERR_INTERNAL_HW" is returned. When this error occurs, initialization has not been performed successfully, and the RFD28F functions cannot be used reliably. The user application shall re-execute R_RFD_Init again. If the error still occurs, the hardware may have a failure.

5.1.19 Forced termination return value of RFD28F

If timeout occurs during the forced termination processing of RFD28F, the error "R_RFD_ERR_INTERNAL_HW" will be returned. If this error occurs, the function has not been completed successfully, and the RFD28F functions cannot be used reliably thereafter. Re-execute R_RFD_Init again. If the error still occurs, the hardware may have a failure.

5.1.20 FACI operation in user applications using RFD28F

The user applications that use RFD28F, shall not directly operate on the FACI, except for the status check of Switch

Area, status check of FACI reset transfer and error handling. The user application shall ensure exclusive control if it is required to operate directly on FACI.

5.1.21 Should not write the code that cannot be continued, such as infinite loop.

The user application should not write the code that cannot be continued, such as infinite loop.

5.2 For Data Flash Component

5.2.1 Reading the unprogrammed area of the data flash memory

If an unprogrammed area is read, an ECC error may occur. If the user wants to eliminate the occurrence of this ECC error, the user application shall check whether there is free space in the relevant area (blank check) in order to verify that it is not programmed. For details, refer to the section on RFD28F User's Manual "API Reference".

5.2.2 No function is provided to read Data Flash Memory contents

RFD28F is not involved in reading the Data Flash Memory contents.

5.2.3 About the execution area of the Data Flash Memory operation

The programs that programming, erasure, checking free space for the Data Flash Memory can be executed from the Code Flash Memory area. Execution of programs from the RAM is also possible, but from the security point of view it is safer to execute from the Code Flash Memory area.

5.2.4 About continuity of address area-controlled Data Flash

Be sure to set the address of the data flash to be controlled by RFD28F in continuous area.

5.2.5 FACI to be controlled and its definition

Make sure that the FACI to be operated by RFD28F is continuous. For example, when FACI0 and FACI2 are controlled, RFD28F interprets that the operation target Data Flash of FACI1 operates with FACI0.

5.2.6 Confirmation of completion of checking blanks in the Data Flash Memory

The user application shall confirm that checking blanks in the Data Flash Memory has been completed by issuing the "R_RFD_GetFaciSequenceReady" on the user application side. Continue to issue the "R_RFD_GetFaciSequenceReady" until the return value of the "R_RFD_GetFaciSequenceReady" is READY state. After that, the user application shall check OK state using “R_RFD_GetFaciStatus”. After successful completion, the user application shall issue "R_RFD_GetBlankCheckResult" and obtains the result.

5.3 For Code Flash Component

5.3.1 Reading the unprogrammed area of the code flash memory

If an unprogrammed area is read, an ECC error may occur. If the user wants to eliminate the occurrence of this ECC error, the user application shall check whether there is free space in the relevant area (blank check) in order to verify that it is not programmed. Since the blank check function for code flash memory is not provided by RFD28F, the blank check should be executed on the user application side. For details on ECC errors, see the section "Reading Flash Memory" in the hardware user's manual, for details on blank checking, see the section "Blank Checking Flash Memory" in the hardware user's manual.

5.3.2 Confirm erasure of the Extended Data Area

If an unprogrammed area is read, an ECC error may occur. If the user wants to eliminate the occurrence of this ECC error, the user application shall check whether there is free space in the the relevant area (blank check) in order to verify that it is not programmed. Since the blank check function for extended data area is not provided by RFD28F, the blank check shall be executed on the user application side. For details, refer to the section "Reading Flash Memory" in the hardware user's manual.

5.3.3 Confirm erasure of the hardware property

If an unprogrammed area is read, an ECC error may occur. If the user wants to eliminate the occurrence of this ECC error, the user application shall check whether there is free space in the relevant area (blank check) in order to verify that it is not programmed. Since the blank check function for hardware property is not provided by RFD28F, the blank check should be executed on the user application side. For details, refer to the section "Reading Flash Memory" in the hardware user's manual.

5.3.4 ECC error when reading Code Flash Memory contents

RFD28F is not involved in reading the code flash memory contents. Therefore, if an ECC error occurs during reading, it should be handled by the user application.

5.3.5 ECC error when reading Extended Data Area contents

RFD28F is not involved in reading the code flash memory contents. Therefore, if an ECC error occurs during reading, it should be handled by the user application.

5.3.6 ECC error when reading Hardware Property Area contents

RFD28F is not involved in reading the Hardware Property Area contents. However, R_RFD_EraseSwitchRequest, R_RFD_EraseTagRequest and R_RFD_UpdateTagRequest does read each flag before executing flash sequencer.

5.3.7 FACI to be controlled and its definition

Make sure that the FACI to be operated by RFD28F is continuous. For example, when FACI0 and FACI2 are controlled, RFD28F interprets that the operation target Code Flash of FACI1 operates with FACI0.

5.3.8 Erasure Code Flash Memory when OTA is not performed (normal case)

Erase the area to be written.

5.3.9 Erasure Code Flash Memory when OTA is performed

- In single map mode
  Erases the area that is not currently running.
- In double map mode
  Erases back side (invalid side).

Erasure the contents of Code flash memory at Driving cycle is performed by "Erasure Code Flash Memory when OTA is performed".

5.3.10 Writing Code Flash Memory when OTA is not performed (normal case)

Write after erasing the area to be written.

5.3.11 Writing Code Flash Memory when OTA is performed

- In single map mode
  Writes the area that is not currently running. After writing, change the execution area with GCFU.
- In double map mode
  In front side, writes the BankB or BankD area.
  In back side, writes the BankA or BankC area.

After writing, switch between execution mode by switching between mode A and mode B. Rewrite the contents of Code flash memory at Driving cycle is performed by "Writing Code Flash Memory when OTA is performed".

5.3.12 Protection for programming/erasing of the Code Flash Memory

When writing to the Code Flash Memory is prohibited by the FPMON register, it controls so as not to shift to P/E mode. Refer to the "FLMDCNT" section in the hardware user's manual for information on how to release write protection. FLMDCNT register is not controlled by RFD28F. Shall be controlled it on the user application side.

5.3.13 No function is provided to read Code Flash Memory contents

RFD28F is not involved in reading the Code Flash Memory contents.

5.3.14 No function is provided to read contents from the extended data area

RFD28F is not involved in reading the Extended Data Area contents.

5.3.15 No function is provided to read Hardware Property contents

RFD28F is not involved in reading the Hardware Property Area contents. However, R_RFD_EraseSwitchRequest, R_RFD_EraseTagRequest and R_RFD_UpdateTagRequest read each flag before executing flash sequencer

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	2020. 6.11	-	Newly issued

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 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.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; 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.
6. 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.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. 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.
8. 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.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall 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.
10. 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.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(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.4.0-1 November 2017)

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/.

RFD28F Renesas Flash Driver User's Manual

[For Free Package] For RH850/U2A16

Publication Date: Ver.1.00E Jun 11, 2020

Published by: Renesas Electronics Corporation

Renesas Flash Driver
RFD28F

Renesas Electronics Corporation