User´s Manual

32

RH850 Family

Code Flash Library, Type T01

RENESAS MCU RH850 Family

Installer: RENESAS_FCL_RH850_T01_V2.xx (xx>=10)

Renesas Electronics
www.renesas.com

R01US0078ED0213
Jun 10, 2019

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.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

How to use this document

(1) Purpose and Target Readers

This manual is designed to provide the user with an understanding of the functions and characteristics of the Self-Programming Library. It is intended for users designing application systems incorporating the library. A basic knowledge of embedded systems is necessary in order to use this manual. The manual comprises an overview of the library, API description, usage notes and cautions.

Particular attention should be paid to the precautionary notes when using the manual. These notes occur within the body of the text, at the end of each section, and in the Cautions section.

The revision history summarizes the locations of revisions and additions. It does not list all revisions.

Refer to the text of the manual for details.

(2) List of Abbreviations and Acronyms

Abbreviation	Full form
API	Application Programming Interface
Code Flash	Embedded Flash where the application code or constant data is stored.
Data Flash	Embedded Flash where mainly the data of the EEPROM emulation are stored.
Dual operation	Dual operation is the capability to access flash memory during reprogramming another flash memory range. Between different Code Flash macros dual operation depends on the device implementation
ECC	Error Correction Code
Flash	Electrically erasable and programmable nonvolatile memory. Different to ROM, this type of memory can be re-programmed several times.
Flash area	Area of Flash consists of several coherent Flash blocks
Flash block	A flash block is the smallest erasable unit of the flash memory.
FCL	Code Flash Library (Code Flash access layer)
FDL	Data Flash Library (Data Flash access layer)
OPB	Option bytes used to define device behavior at startup
OTP	One Time Programmable
RAM	“Random access memory” - volatile memory with random access
ROM	“Read only memory” - nonvolatile memory. The content of that memory cannot be changed during normal operation.
Self-programming	Capability to reprogram the embedded flash without external programming tool only via control code running on the microcontroller.
Serial programming	The onboard programming mode is used to program the device with an external programmer tool.
User area (or user memory)	Code flash area that is available for usual user program. This is where most of the user code resides.

Abbreviation	Full form
User boot area (or extended user memory)	Code flash area designed for storing a bootloader application.

Table of Contents

Chapter 1 Introduction

This user manual describes the internal structure, the functionality and the application programming interface (API) of the Renesas RH850 Self-Programming Library for Code Flash (FCL) Type 01, designed for RH850 flash devices.

Note:
Do not use this library for other devices than listed in the installer readme file, as this might lead to all sorts of defective behavior including physical destruction.

The libraries are delivered in source code. Great care should be exercised when performing changes, as not intended behavior and programming faults might be the result. Also, user changes in the library non-configurable code voids any support from Renesas.

The Renesas RH850 Self-Programming Library for Code Flash Type 01 (from here on referred to as FCL) is provided for the Green Hills, IAR development environments and Renesas CubeSuite+. Due to the different compiler and assembler features, especially the assembler files differ between the environments. So, the library and application programs are distributed using an installer tool allowing selecting the appropriate environment. The IAR environment is supported for the Europe and America regions only.

The libraries are delivered together with device dependent application programs, showing the implementation of the libraries and the usage of the library functions.

This manual is based on the assumption that the device will operate in supervisor mode.

Please ensure to always use the latest release of the library in order to take advantage of improvements and potential fixes.

If you are located in Europe:
The FCL, the latest version of this user manual and other device dependent information can be downloaded from the following URL:
http://www.renesas.eu/update

If you require Flash library related support, please contact our European support team using the following mail address:
application_support.flash-eu@lm.renesas.com

If you are located in other regions:
The FCL and this user manual always recommend use of the latest version.

Note:
Please read all chapters of this user manual carefully. Much attention has been put to proper description of usage conditions and limitations. Anyhow, it can never be completely ensured that all incorrect ways of integrating the library into the user application are explicitly forbidden. So, please follow the given sequences and recommendations in this document exactly in order to make full use of the library functionality and features and in order to avoid malfunctions caused by library misuse.

1.1 Flash Infrastructure

Besides the Code Flash, many devices of the RH850 microcontroller family are equipped with a separate flash area - the Data Flash. This flash area is meant to be used exclusively for data. It cannot be used for instruction execution (code fetching).

1.1.1 Dual Operation

Common for all Flash implementations is, that during Flash modification operations (Erase/Write) a certain amount of Flash memory is not accessible for reading and/or program execution. This does not only concern the modified Flash range, but a certain part of the complete Flash system. The amount of not accessible Flash depends on the device architecture.

A standard architectural approach is the separation of the Flash into Code Flash and Data Flash. By that, it is possible to read from the Code Flash (to execute program code or read data) while Data Flash is modified, and vice versa.

To check whether dual operation is supported by a device, please refer to the device user manual.

Note:

It is not possible to modify Code Flash and Data Flash in parallel.

1.1.2 Flash Granularity

The Code Flash of RH850 device is separated into blocks of either 8KB or 32KB sizes. FCL erase operations can only be performed on complete blocks of any of the two sizes available. Some devices may not have 8KB block size and some devices may have two code flash banks. Please refer to the corresponding user manual of your device for detailed information on the number of available code flash blocks and code flash banks.

Writing of data can be done with a granularity of 256 bytes aligned with the block boundaries.

Do not read from erased Code Flash because this will generate an ECC error.

The code flash on RH850 devices is divided in two user areas:

- The user area also named user memory, is the code flash memory where the user program is programmed.
- The user boot area also named extended user memory, is designed for special application parts, e.g. a bootloader program. For some devices programming or erasure by self-programming is prohibited. Please refer to the corresponding user manual of your device for detailed information.

Chapter 2 Architecture

This chapter introduces the basic software architecture of the FCL and provides the necessary background for application designers to understand and effectively use the library. Please read this chapter carefully before moving on.

The self-programming system is built up from several hierarchical functional blocks. This user manual concentrates on the functionality and usage of the FCL. However, a short description of all involved functional blocks and their relationship is important for the general understanding of the concepts and usage of the FCL.

As depicted in Figure 1, the software architecture of the self-programming system is built up of several blocks:

- User application: This functional block represents the application (including a potential start-up program) provided by the user.
- FCL: This functional block represents the FCL that offers all the functions and operations necessary to reprogram the application in a user friendly C language interface.
- Flash hardware: This functional block represents the Flash programming hardware (the flash sequencer) controlled by the FCL.

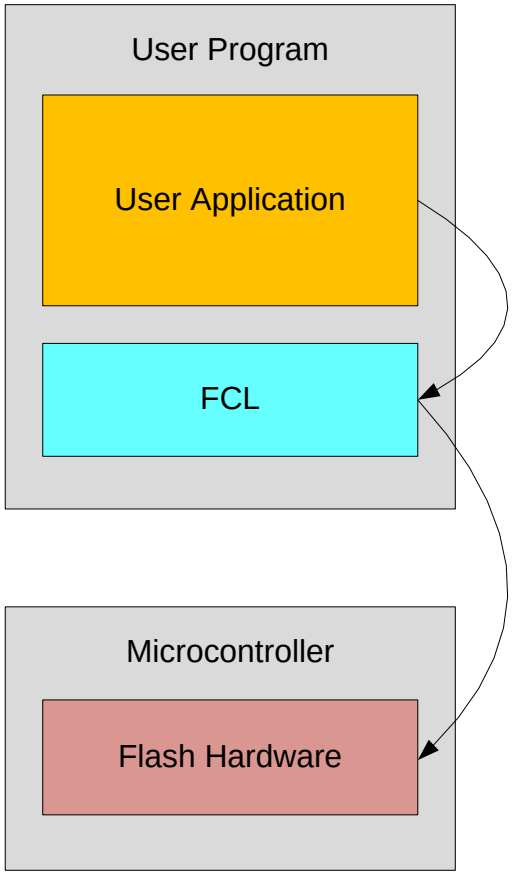

Figure 1: Symbolic representation between functional blocks

2.1 Library representation

FCL library is released as source code and is delivered as an installer package together with a sample application, build scripts and GNU Make tools, ready for compilation and operation on the dedicated device.

Please consult chapter 5.2 "File structure" for details about the delivered package.

Chapter 3 FCL Functional Specifications

3.1 Code execution in RAM

The self-programming application and the FCL are initially located in code flash. During library operation the code flash is often not accessible because hardware resources are busy with flash programming. This is why parts of the user application and library have to be copied and executed from RAM memory. This may be internal RAM, but also external RAM memory can be used if security is not a concern.

It is not necessary to copy everything to RAM. Only parts of the code that are accessed during programming have to be copied. The copied parts are not much in size, but they have to be carefully selected. Many errors arise from accessing code that was not copied or access in the code flash is made due to interrupts, exceptions, watchdog resets, etc.

To copy necessary parts into available RAM, three different methods are possible:

1. **C-startup:** The code is linked to the destination address. During system start-up, compiler specific routines are called to copy the code from a ROM image (usually in code flash) to RAM.
2. **R_FCL_CopySections:** The FCL library provides an API function – `R_FCL_CopySections` - that copies all the needed sections to a used defined address.
3. **User specific:** In case of a specific implementation, the user is responsible for the correct location of the sections.

Depending on the configured mode (Internal or User mode, see next section 3.2 “Operating modes”) the following linker sections need to be copied to RAM:

Table 1: Linker sections copied to RAM

Pre-compilation configuration	Sections copied to RAM
R_FCL_HANDLER_CALL_USER	R_FCL_CODE_RAM_USRINT R_FCL_CODE_RAM_USR R_FCL_CODE_RAM R_FCL_CODE_ROMRAM
R_FCL_HANDLER_CALL_INTERNAL	R_FCL_CODE_RAM_USRINT R_FCL_CODE_RAM_USR R_FCL_CODE_RAM

For further information regarding the linker sections please refer to chapter 5.3 “Linker sections".

3.2 Operating modes

Following two major scenarios may be considered for self-programming. These are reflected by the library modes:

User mode:

Most parts of the Self-Programming Library are executed in the internal RAM, amongst with the reprogramming control functions and other user code to be executed during self-programming. This library mode is best to use for devices with sufficient RAM. User code execution is always possible during self-programming, because a Flash operation is just initiated by the FCL function call. While the FCL returns control to the user application, the Flash operation is executed in background. The user has to poll the operation status via the status check function (see section 4.3.2.3 “R_FCL_Handler"). Interrupt routines as well as user code execution are possible, if all related functions as well as the interrupt vector table are located in RAM.

To enable this mode, the library must be configured to use the user mode (see section 4.1 “Library compile-time configuration").

The following picture illustrates an example of operation in user operation mode.

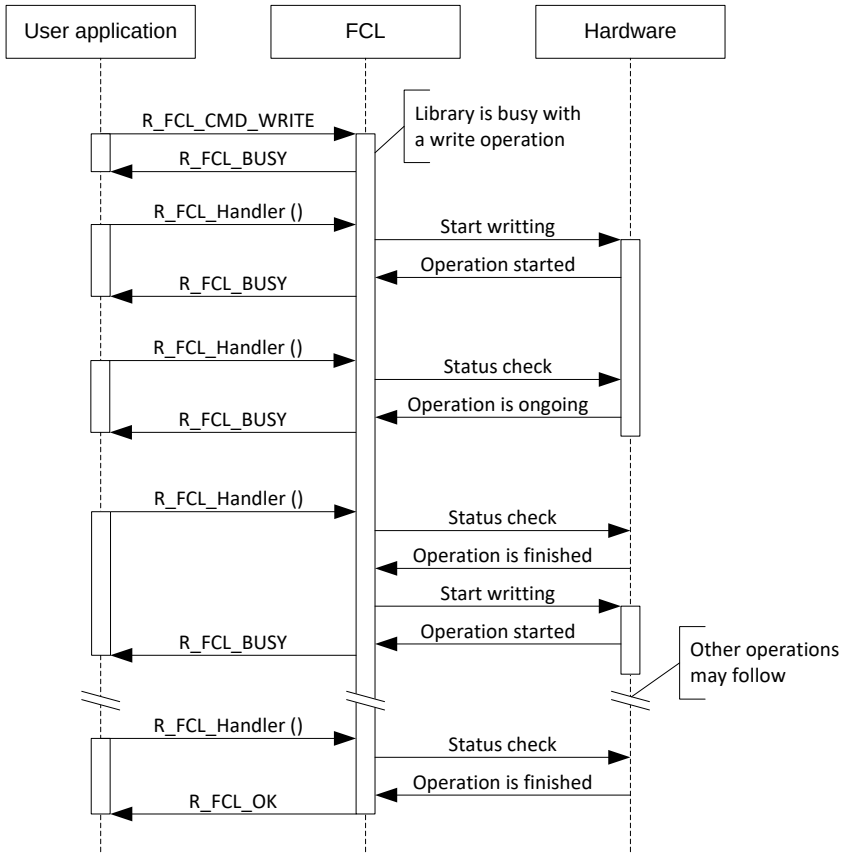

Figure 2: Asynchronous execution in User mode

The user application has to pool the library with `R_FCL_Handler` function in order to process all the smaller operations that happen in the library and the hardware.

Internal mode:

Only small parts of the library are executed in RAM, the rest is executed in the Code Flash. Normal user code execution during self-programming is impossible, because a FCL function is executed synchronously - an operation does not return until the operation is finished. Therefore only interrupts are possible during self-programming.

To enable this mode, the library must be configured to use the internal mode (see 4.1 “Library compile-time configuration”).

3.3 Request oriented operation

The FCL utilizes request-response architecture in order to initiate the commands. This means any "requester" (any tasks in the user application) has to prepare a request structure and pass it by reference to the library using via `R_FCL_Execute` function. The FCL interprets the content of the request structure, checks plausibility of the structure members and initiates the execution. The feedback is reflected to the requester via the status member (`status_enu`) of the same request structure.

The `status_enu` structure member is updated differently depending on the library operation mode. In internal mode, the `R_FCL_Execute` function will update the status with the final result when the function returns. In user mode, however, `R_FCL_Execute` will return immediately with a `R_FCL_BUSY` status (that is, if the given parameters are correct). The completion of an accepted command is done by calling `R_FCL_Handler` periodically as long as the request status remains "`R_FCL_BUSY`". In the remaining time the user application is free to perform other operations as long as they do not use Flash resources in any way.

The request structure is the central point of FCL operation as the following picture shows it:

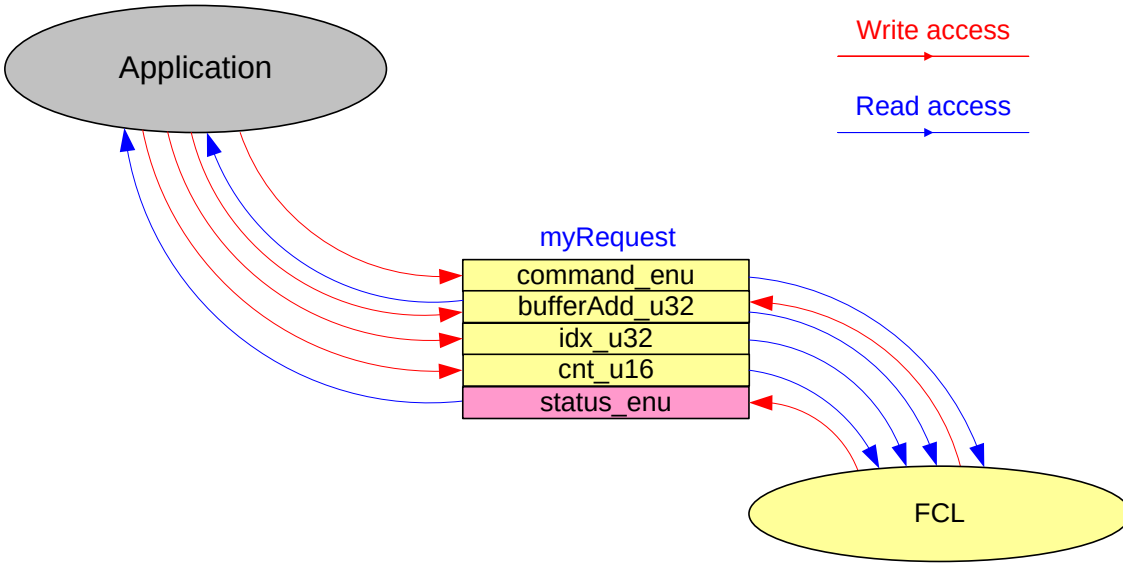

Figure 3: Usage of the request structure

3.4 Suspend / Resume mechanism

Some operations can last a long time and it is not possible to wait for them to finish. FCL has the option to suspend them and resume their execution later.

The FCL supports suspending of the `R_FCL_CMD_WRITE` and `R_FCL_CMD_ERASE` operations. After a command is suspended, Code Flash is switched on again in order to execute code from there. Furthermore, during a suspended `R_FCL_CMD_ERASE`, further `R_FCL_CMD_WRITE` operations can be executed. However, this can be done only on Flash blocks, not affected by the suspended Erase operation.

Suspend-resume facility is available only for user mode.

The picture below shows an example of how to suspend an erase command, perform a write, and then resume the erase command:

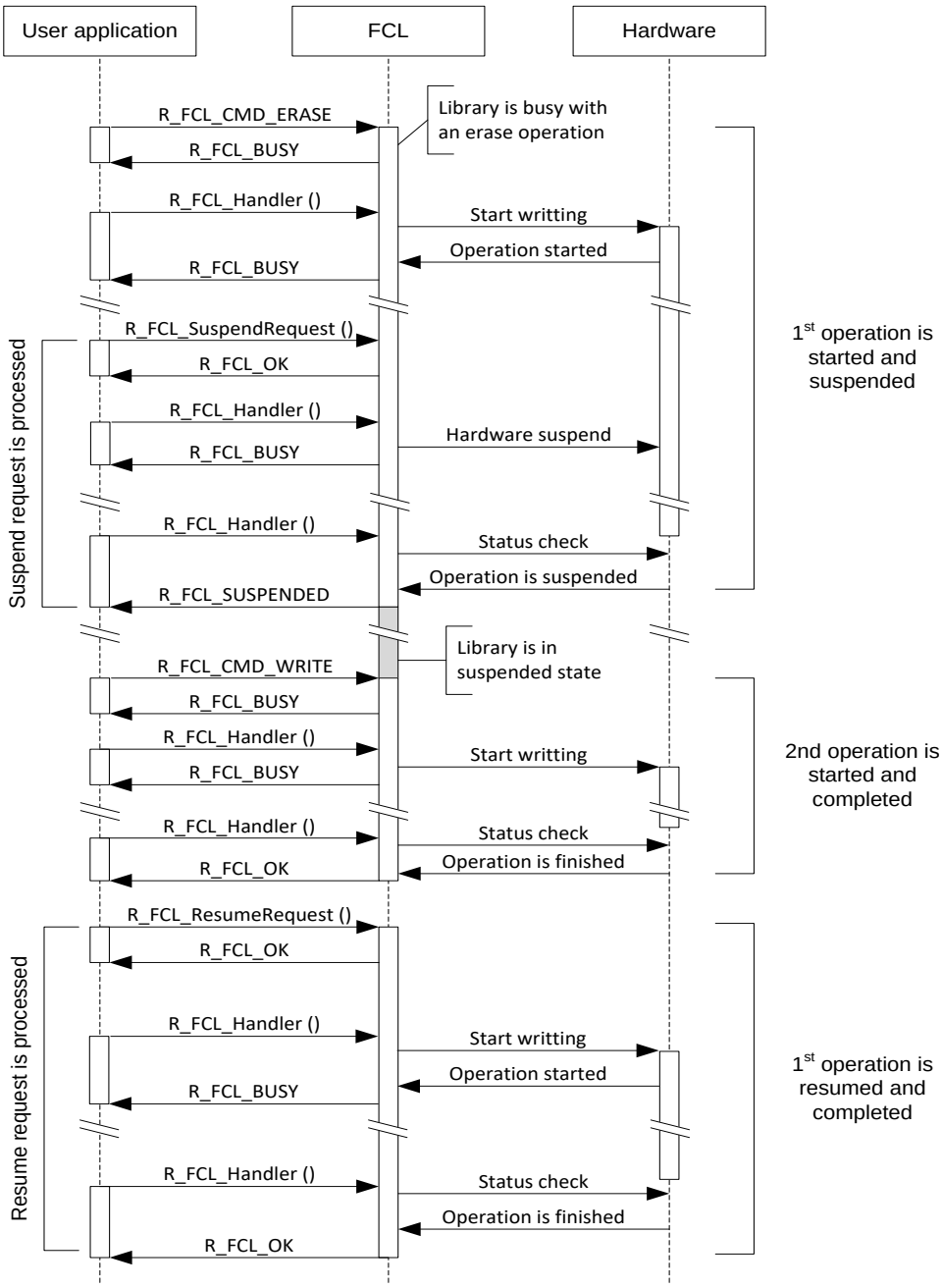

Figure 4: Example of suspend resume flow

Please note that the following sequences are not possible:

- erase ► suspend ► erase
- write ► suspend ► erase
- write ► suspend ► write
- nesting suspend, for example: erase ► suspend ► write ► suspend

When Erase processing is suspended and resumed, this is not considered as an additional erase with respect to the specified Flash erase endurance.

3.5 Cancel mechanism

The Flash Erase and Write are long lasting operations. Under certain conditions, the user application cannot wait for the end of a long lasting Flash operation. So, such operation should be cancel-able.

Cancel facility is available only for user mode.

The picture below shows an example of how to cancel an erase command:

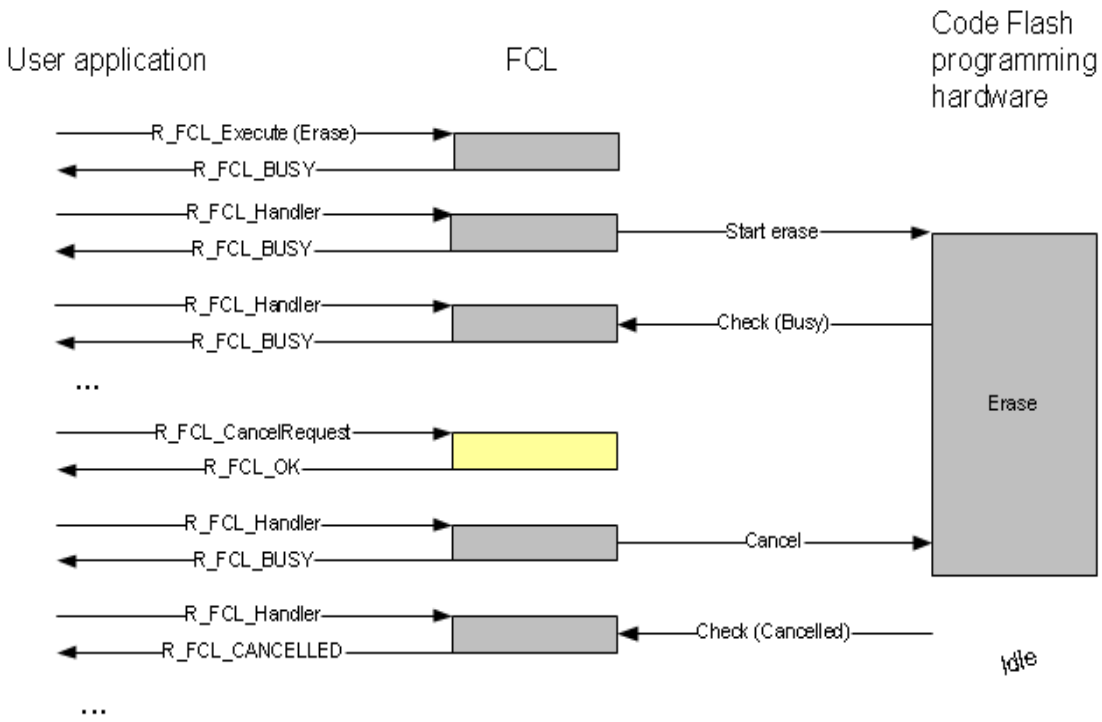

Figure 5: Cancel an Erase operation

An on-going or suspended Erase/Write operation always ends in cancelled. If an operation is already suspended and a second operation is on-going, then both operations will be cancelled if it is requested.

3.6 Loop supervision

Some FCL commands have internal polling loops to check for hardware status. These loops are time supervised by FCL software to avoid locking the CPU in an infinite loop. If a hardware error occurs the FCL will abort after a certain timeout and report `R_FCL_ERR_INTERNAL` on the current command.

The timeout depends on the latency of the polling loop but it is smaller than 800 microseconds provided that the device CPU is running at a frequency in the range permitted for FCL normal operation.

3.7 Internal error

When FCL detects abnormal behavior it will report command status `R_FCL_ERR_INTERNAL`.

The following steps can be taken to recover after `R_FCL_ERR_INTERNAL` error is encountered:

a. run the command again

b. reinitialize the library (execute `R_FCL_Init` function then run `R_FCL_CMD_PREPARE_ENV` command)

c. if b. fails again then reset the device and proceed to re-initialization

d. if c. fails again then replace the device

Please note that depending on the hardware error, Code Flash or Data Flash can remain in programming mode and in such case only remedy c. and d. can be applied.

Chapter 4 User Interface (API)

4.1 Library compile-time configuration

The pre-compilation configuration of the FCL is located in the file `fcl_cfg.h`. The user has to configure all parameters and attributes by adapting the related constant definition in that header-file. This file may also contain device or application specific defines.

The configuration file contains several option elements to configure and customize FCL. These options are described in the following table:

Table 2: Pre-compilation options

Option	Description
R_FCL_COMMAND_EXECUTION_MODE	Defines whether the status check should be performed by the library internally or by the user in order to allow execution of user code in between the status checks. Possible values are: R_FCL_HANDLER_CALL_INTERNAL R_FCL_HANDLER_CALL_USER For further description see below.
R_FCL_SUPPORT_LOCKBIT	Enables or disables the following commands: R_FCL_CMD_GET_LOCKBIT R_FCL_CMD_SET_LOCKBIT R_FCL_CMD_ENABLE_LOCKBITS R_FCL_CMD_DISABLE_LOCKBITS
R_FCL_SUPPORT_OTP	Enables or disables the following commands: R_FCL_CMD_GET_OTP R_FCL_CMD_SET_OTP
R_FCL_SUPPORT_DEVICENAME	Enables or disables the following commands: R_FCL_CMD_GET_DEVICE_NAME
R_FCL_SUPPORT_BLOCKCNT	Enables or disables the following commands: R_FCL_CMD_GET_BLOCK_CNT
R_FCL_SUPPORT_BLOCKENDADDR	Enables or disables the following commands: R_FCL_CMD_GET_BLOCK_END_ADDR
R_FCL_SUPPORT_OPB	Enables or disables the following commands: R_FCL_CMD_GET_OPB R_FCL_CMD_SET_OPB
R_FCL_SUPPORT_ID	Enables or disables the following commands: R_FCL_CMD_GET_ID R_FCL_CMD_SET_ID
R_FCL_SUPPORT_RESETVECTOR	Enables or disables the following commands: R_FCL_CMD_GET_RESET_VECTOR R_FCL_CMD_SET_RESET_VECTOR

Option	Description
R_FCL_SUPPORT_SECURITYFLAGS	Enables or disables the following commands: R_FCL_CMD_SET_READ_PROTECT_FLAG R_FCL_CMD_SET_WRITE_PROTECT_FLAG R_FCL_CMD_SET_ERASE_PROTECT_FLAG R_FCL_CMD_SET_SERIAL_PROG_DISABLED R_FCL_CMD_SET_SERIAL_ID_ENABLED
R_FCL_NO_BFA_SWITCH supported by V2.11 and later versions	During execution of command R_FCL_CMD_PREPARE_ENV, Code Flash needs to be switched off several times in order to read data from the firmware area and to prepare FCL environment. There are some devices that need less switching in order to perform this preparation. If the FCL is running on such device then R_FCL_NO_BFA_SWITCH must be defined. When the FCL is built with this option defined, do not define R_FCL_MIRROR_FCU_COPY or R_FCL_NO_FCU_COPY. This is required for the F1K, F1KM, F1KH device group but not for other devices.
R_FCL_MIRROR_FCU_COPY supported by V2.12 and later versions (for future products)	When this option is defined, the FCL does not switch between the user area and FCU firmware area. When the FCL is built with this option defined, do not define R_FCL_NO_BFA_SWITCH or R_FCL_NO_FCU_COPY.
R_FCL_NO_FCU_COPY supported by V2.12 and later versions	When this option is defined, the FCU firmware transfer function is not executed. This option is set and built for RH850/D1M1A, D1M1-V2, and D1S1. When the FCL is built with this option defined, do not define R_FCL_NO_BFA_SWITCH or R_FCL_MIRROR_FCU_COPY.
R_FCL_CFL2_START_ADDR_2048K supported by V2.12 and later versions (for future products)	This option sets the start address of code flash bank B to 0x00200000. For code flash bank B, refer to the hardware manual of the device. The target devices have code flash bank B and the start address of code flash bank B is 0x00200000.
R_FCL_CFL2_START_ADDR_4096K supported by V2.12 and later versions (for future products)	This option sets the start address of code flash bank B to 0x00400000. For code flash bank B, refer to the hardware manual of the device. The target devices have code flash bank B and the start address of code flash bank B is 0x00400000.

Pre-compilation option R_FCL_COMMAND_EXECUTION_MODE can have these values:

- R_FCL_HANDLER_CALL_INTERNAL (internal mode)

  Advantages:

  - no polling necessary
  - less RAM consumption

  Disadvantages:

  - no return to the application during self-programming
  - user code execution during self-programming is possible only by interrupts
  - it is not possible to erase the code flash area from which the command was initiated

- R_FCL_HANDLER_CALL_USER (user mode)

  Advantages:

  - less CPU load
  - it is possible to execute user code simultaneous with self-programming

  Disadvantages:

  - more RAM consumption
  - status polling is necessary

One important difference between the two modes is the possibility to reprogram entire code flash (or the area of code flash from where the user program resides). Both modes can perform a full code flash erase but in practice only user mode makes sense. This implies that in order to reprogram entire code flash, code compiled with FCL in internal mode will have to relocate itself otherwise it will hang up.

For details about modes of operation please refer to section 3.2 “Operating modes".

Note:

The pre-compilation definitions R_FCL_NO_FCU_COPY, R_FCL_MIRROR_FCU_COPY, and R_FCL_NO_BFA_SWITCH supported by V2.12 and later versions of RH850 FCL Type01 are only applicable to specific device groups.

The following table shows the correspondence between the definitions and device groups.

Pre-compilation definition	F1L/F1M/ F1H	D1L/ D1M1/D1M1H/ D1M2/D1M2H	D1M1A/ D1M1-V2/ D1S1	F1K/F1KM/ F1KH	Future products
R_FCL_NO_BFA_SWITCH	-	-	-	✓	-
R_FCL_MIRROR_FCU_COPY	-	-	-	-	✓
R_FCL_NO_FCU_COPY	-	-	✓	-	-

Note: For the device groups supported by the version of the FCL you are using, see the support.txt file that came with the FCL.

4.2 Data types

This section describes all data definitions used and offered by the FCL. In order to reduce the probability of type mismatches in the user application, please make strict usage of the provided types and avoid using standard data types instead.

Definitions are similar to those in the standard C99 `stdint.h` header, but please carefully check that there are no size or endianness mismatches if you are using other definitions in your project.

Compiling V2.13 or a later version of the RH850 FCL Type 01

When you are using V2.13 or a later version of the RH850 FCL Type 01, it is assumed that all data definitions will be compiled according to C99* or a later standard defined by the ISO.

If you select C99 or a later standard for the GHS compiler, Renesas compiler or IAR compiler you are using, the compiler provides the “stdint.h” header file containing data definitions.

If you select a standard earlier than C99 or no standard, on the other hand, data definitions in "r_typedefs.h” (equivalent to stdint.h) will be used.

*Note: The formal name of C99 is ISO/IEC 9899:1999.

Compiling V2.12 or an earlier version of the RH850 FCL Type 01

When you are using V2.12 or an earlier version of the RH850 FCL Type 01, change the directive `#include "r_typedefs.h"`, which is in the following two files, to `#include "stdint.h"`.

- r_fcl_hw_access.c
- r_fcl_user_if.c

<Example of the change>

```
#include "stdint.h"
/* #include "r_typedefs.h" */
```

Similarly, when using the attached sample files, change the directive `#include "r_typedefs.h"` to `#include "stdint.h"`.

The sample files for the RH850 FCL Type 01 are as follows.

- main.c
- fcl_descriptor.c
- fcl_user.c

4.2.1 Simple type definitions

Type definitions in cases where C99 or a later standard is selected for the compiler:

See “stdint.h” provided with the compiler.

Type definition:

Type definitions (extracted from "r_typedefs.h") in cases where a standard earlier than C99 or no standard is selected for the compiler:

```
typedef signed char         int8_t;
typedef unsigned char       uint8_t;
typedef signed short        int16_t;
typedef unsigned short      uint16_t;
typedef signed long         int32_t;
typedef unsigned long       uint32_t;
typedef unsigned char       rBool;
```

Description: These simple types are used throughout the library API. All library specific simple type definitions can be found in file `r_typedefs.h`, which is part of the library installation package.

4.2.2 r_fcl_command_t

Type definition:

```
typedef enum R_FCL_COMMAND_T
{
    R_FCL_CMD_PREPARE_ENV,
    R_FCL_CMD_ERASE,
    R_FCL_CMD_WRITE,
    R_FCL_CMD_SET_LOCKBIT,
    R_FCL_CMD_GET_LOCKBIT,
    R_FCL_CMD_ENABLE_LOCKBITS,
    R_FCL_CMD_DISABLE_LOCKBITS,
    R_FCL_CMD_SET_OTP,
    R_FCL_CMD_GET_OTP,
    R_FCL_CMD_SET_OPB,
    R_FCL_CMD_GET_OPB,
    R_FCL_CMD_SET_ID,
    R_FCL_CMD_GET_ID,
    R_FCL_CMD_SET_READ_PROTECT_FLAG,
    R_FCL_CMD_GET_READ_PROTECT_FLAG,
    R_FCL_CMD_SET_WRITE_PROTECT_FLAG,
    R_FCL_CMD_GET_WRITE_PROTECT_FLAG,
    R_FCL_CMD_SET_ERASE_PROTECT_FLAG,
    R_FCL_CMD_GET_ERASE_PROTECT_FLAG,
    R_FCL_CMD_SET_SERIAL_PROG_DISABLED,
    R_FCL_CMD_GET_SERIAL_PROG_DISABLED,
    R_FCL_CMD_SET_SERIAL_ID_ENABLED,
    R_FCL_CMD_GET_SERIAL_ID_ENABLED,
    R_FCL_CMD_SET_RESET_VECTOR,
    R_FCL_CMD_GET_RESET_VECTOR,
    R_FCL_CMD_GET_BLOCK_CNT,
    R_FCL_CMD_GET_BLOCK_END_ADDR,
    R_FCL_CMD_GET_DEVICE_NAME
} r_fcl_command_t;
```

Description: The library offers a set of Flash operations that are initiated and controlled by the library. All operations are initiated by a single call of `R_FCL_Execute` and later on controlled by the library function `R_FCL_Handler`. Managing these operations is the main purpose of the library.

Details of for the available commands can be found in section 4.4 “Library commands”.

Member / Value: Please check Table 3 "List of available commands" below.

Table 3: List of available commands

Member / Value	Description
R_FCL_CMD_PREPARE_ENV	Copies the library’s internal functions to RAM and initializes internal states and variables. Note, however, that library’s internal functions will not be copied to RAM when the RH850/F1K, F1KM, F1KH, D1M1A, D1M1-V2, or D1S1 is in use.
R_FCL_CMD_ERASE	Erases one or more Flash blocks
R_FCL_CMD_WRITE	Writes multiple of 256 bytes into the Flash
R_FCL_CMD_SET_LOCKBIT	Locks a Flash block to prevent it from further modification
R_FCL_CMD_GET_LOCKBIT	Reads out the current lock bit setting
R_FCL_CMD_ENABLE_LOCKBITS	Enables the mechanism to protect a block using lock bits
R_FCL_CMD_DISABLE_LOCKBITS	Disables the mechanism to protect a block using lock bits
R_FCL_CMD_SET_OTP	Sets one-time-programmable flag for one block
R_FCL_CMD_GET_OTP	Reads one-time-programmable flags
R_FCL_CMD_SET_OPB	Writes a new option byte setting
R_FCL_CMD_GET_OPB	Reads out the current option byte setting
R_FCL_CMD_SET_ID	Writes a new OCID, used to protect the debug and serial interfaces and enable self-programming
R_FCL_CMD_GET_ID	Reads out the current OCID setting
R_FCL_CMD_SET_READ_PROTECT_FLAG	Enables read protection
R_FCL_CMD_GET_READ_PROTECT_FLAG	Reads out the current read protection setting
R_FCL_CMD_SET_WRITE_PROTECT_FLAG	Enables write protection
R_FCL_CMD_GET_WRITE_PROTECT_FLAG	Reads out the current write protection setting
R_FCL_CMD_SET_ERASE_PROTECT_FLAG	Enables erase protection
R_FCL_CMD_GET_ERASE_PROTECT_FLAG	Reads out the current erase protection setting
R_FCL_CMD_SET_SERIAL_PROG_DISABLED	Disables serial programming
R_FCL_CMD_GET_SERIAL_PROG_DISABLED	Checks whether serial programming is disabled or not
R_FCL_CMD_SET_SERIAL_ID_ENABLED	Enables OCID check
R_FCL_CMD_GET_SERIAL_ID_ENABLED	Checks whether OCID check is disabled or not
R_FCL_CMD_SET_RESET_VECTOR	Writes a new reset vector into the device
R_FCL_CMD_GET_RESET_VECTOR	Reads out the current reset vector setting
R_FCL_CMD_GET_BLOCK_CNT	Reads out the number of Flash blocks

Member / Value	Description
R_FCL_CMD_GET_BLOCK_END_ADDR	Reads out the end address of a block
R_FCL_CMD_GET_DEVICE_NAME	Reads out the ASCII coded name of the device

4.2.3 r_fcl_status_t

Type definition:

```
typedef enum R_FCL_STATUS_T
{
    R_FCL_OK,
    R_FCL_BUSY,
    R_FCL_SUSPENDED,
    R_FCL_ERR_FLMD0,
    R_FCL_ERR_PARAMETER,
    R_FCL_ERR_PROTECTION,
    R_FCL_ERR_REJECTED,
    R_FCL_ERR_FLOW,
    R_FCL_ERR_WRITE,
    R_FCL_ERR_ERASE,
    R_FCL_ERR_COMMAND,
    R_FCL_CANCELLED,
    R_FCL_ERR_INTERNAL
} r_fcl_status_t;
```

Description: The enumeration type `r_fcl_status_t` defines the FCL status return values. The status / error codes above are returned by the library to indicate the current status of a FCL command. Other API functions for initialization, Suspend-Resume and Cancel are returning this status codes too.

The root causes and interpretation of all status and error codes depends on the executed operation or called function. The meaning of the codes for the operations is explained in the following table.

Member / Value:

Member / Value	Description
R_FCL_OK	The requested operation finished successfully
R_FCL_BUSY	The requested operation was successfully started and it is on-going
R_FCL_SUSPENDED	Current operation is suspended
R_FCL_ERR_FLMD0	The requested command was not executed due to an enabled FLMD0 hardware protection
R_FCL_ERR_PARAMETER	The requested command was not executed due to wrong input data passed in the request structure
R_FCL_ERR_PROTECTION	The requested command was not completely executed due to an enabled security feature (e.g. enabled erase protection, lock bit set, ... etc)
R_FCL_ERR_REJECTED	The requested command was not executed because another operation is on-going
R_FCL_ERR_FLOW	The current request was not executed because of incorrect initialization sequence or incorrect suspend resume sequence
R_FCL_ERR_WRITE	The requested write command encountered an error because of writing into not erased area, due to a hardware error or due to unstable FLMD0 pin
R_FCL_ERR_ERASE	The requested erase command encountered an error because of a hardware erase error or due to unstable FLMD0 pin
R_FCL_ERR_COMMAND	The requested command does not exist or it was disabled via pre-compilation options
R_FCL_CANCELLED	Current operation is cancelled
R_FCL_ERR_INTERNAL	Library or hardware internal error

4.2.4 r_fcl_request_t

Type definition:

```
typedef volatile struct R_FCL_REQUEST_T
{
    r_fcl_command_t     command_enu;
    uint32_t            bufferAdd_u32;
    uint32_t            idx_u32;
    uint16_t            cnt_u16;
    r_fcl_status_t      status_enu;
} r_fcl_request_t;
```

Description: All user operations are initiated by a central initiation function called R_FCL_Execute. All information required for the execution is passed to the FCL by the request structure. Also the error is returned by the same structure. See section 3.3 “Request oriented operation” for details about this structure.

Section 4.4 “Library commands" shows in detail how to fill this structure and how to check the results for each desired command.

Member / Value:

Member / Value	Description
command_enu	User command to execute
bufferAdd_u32	• R_FCL_CMD_WRITE: address of the write buffer of the application • All GET commands: address of the buffer for the result • SET commands: address of the buffer containing the data to set (if necessary) • All other commands: Parameter not used
idx_u32	• R_FCL_CMD_WRITE: code flash address to write (must be 256 bytes aligned) • R_FCL_CMD_ERASE: first block to be erased by the erase operation • R_FCL_SET_LOCKBIT, R_FCL_GET_LOCKBIT, R_FCL_SET_OTP, R_FCL_GET_OTP, affected block number • All other commands: Parameter not used
cnt_u16	• R_FCL_CMD_WRITE: numbers of write units to write (one write unit is 256 bytes wide) • R_FCL_CMD_ERASE: number of blocks to erase • All other commands: Parameter not used
status_enu	Library return codes (status and error codes depend on the called command)

4.2.5 r_fcl_descriptor_t

Type definition:

```
typedef struct R_FCL_DESCRIPTOR_T
{
    uint32_t    id_au32[4];
    uint32_t    addrRam_u32;
    uint16_t    frequencyCpuMHz_u16;
} r_fcl_descriptor_t;
```

Description: The run-time configuration (see chapter 5.2” File structure”) is defined in a separate data type. A variable of the data type is read during initialization phase and internal variables are set according to the configuration.

Member / Value:

Member / Value	Description
id_au32	This defines the ID used to enable self-programming. An incorrect ID setting will result in a protection error.
addrRam_u32	This defines the start address of RAM reserved for execution of the FCL.
frequencyCpuMHz_u16	This defines the CPU frequency in MHz. The value shall be rounded up to the nearest integer, e.g. for a frequency of 24.3 MHz set the value to 25. Please check the device user’s manual for limit values. The timeout supervision timing as well as the Flash hardware frequency is internally derived from the CPU frequency.

4.3 Functions

The following list provides a summary of all API functions described in this document:

R_FCL_Init

R_FCL_CopySections

R_FCL_CalcFctAddr

R_FCL_GetVersionString

R_FCL_Execute

R_FCL_Handler

R_FCL_SuspendRequest

R_FCL_ResumeRequest

R_FCL_CancelRequest

4.3.1 Initialization

4.3.1.1 R_FCL_Init

Outline: Initialization of the library

Interface: C Interface

```
r_fcl_status_t R_FCL_Init (const r_fcl_descriptor_t * descriptor_pstr)
```

Arguments: Parameters

Argument	Type	Access	Description
descriptor_pstr	r_fcl_descriptor_t	r	FCL configuration descriptor

Return value

Type	Description		
r_fcl_status_t	R_FCL_OK	meaning	operation finished successfully
		reason	no problems during execution
		remedy	nothing
	R_FCL_ERR_PARAMETER	meaning	operation stopped due to invalid parameter values
		reason	Descriptor variable is still uninitialized or undefined
		remedy	set/correct descriptor variable and repeat the command

Pre-conditions: None

Post-conditions: None

Description: This function initializes the FCL. It must be called before any execution of a FCL function. It initializes all internal variables and performs some parameter checks.

The function shall be executed from ROM.

Example:

```
/* Initialize Self-Programming Library */
r_fcl_status_t status_enu;

status_enu = R_FCL_Init (&RTConfig_enu);
/* Error treatment ... */
```

4.3.1.2 R_FCL_CopySections

Outline: Copy used linker segments from ROM to a new address in RAM

Interface: C Interface

```
r_fcl_status_t R_FCL_CopySections (void)
```

Arguments: Parameters

None

Return value

Type	Description		
r_fcl_status_t	R_FCL_OK	meaning	operation finished successfully
		reason	no problems during execution
		remedy	nothing
	R_FCL_ERR_FLOW	meaning	function execution currently not possible
		reason	wrong library handling flow (e.g. library not initialized, library operation on-going)
		remedy	correct the flow
	R_FCL_ERR_INTERNAL	meaning	function execution currently not possible
		reason	wrong RAM address given in the initialization configuration descriptor
		remedy	reinitialize the library with correct configuration descriptor

Pre-conditions: Library must be initialized (call function `R_FCL_Init`)

Post-conditions: None

Description: This function is used to copy some FCL code sections to a specified destination address in RAM. From that location code can be executed while Code Flash is not available. Please check 5.3 “Linker sections" for details about copied code sections.

The `R_FCL_CopySections` function shall be executed from ROM.

Example:

```
/* Copy FCL to internal RAM */
r_fcl_status_t status_enu;

status_enu = R_FCL_CopySections ();
/* Error treatment */
```

4.3.1.3 R_FCL_CalcFctAddr

Outline: Calculate new address after copy process

Interface: C Interface

```
uint32_t R_FCL_CalcFctAddr (uint32_t addFct_u32)
```

Arguments: Parameters

Argument	Type	Access	Description
addFct_u32	uint32_t	r	ROM address of copied function

Return value

Type	Description
uint32_t	New RAM address of function.

Pre-conditions: Library must be in initialized state (call function `R_FCL_Init`).

Post-conditions: None

Description: This function calculates the new address of a function copied from ROM to RAM. To calculate the new address of the function, the copied function must be located in one of the FCL linker segments described in chapter 5.3 “Linker sections".

The function shall be executed from ROM.

Example:

```
/* Calculate new address of user control function fctUserCtrl located in FCL
section R_FCL_CODE_RAM_USR */
uint32_t (*fpFct)( void );

fpFct = (uint32_t(*)())R_FCL_CalcFctAddr ((void *)fctUserCtrl);
```

4.3.2 Operation

4.3.2.1 R_FCL_GetVersionString

Outline: Return library version string

Interface: C Interface

```
const uint8_t *R_FCL_GetVersionString (void)
```

Arguments: Parameters

None

Return value

Type	Description
const uint8_t *	The library version is a string value in the following format: “SH850T01xxxxxYZabcD” Please check function description below for details.

Pre-conditions: The function is located in the `FCL_CODE_ROM` section. As this is not copied to RAM by `R_FCL_Copy_Sections`, the function has to be called on its linked location.

Post-conditions: None

Description: This function returns the pointer to the library version string. The version string is a zero terminated string identifying the library (same definition as the agreed version string used in former libraries). The version string is stored in the library code section.

The version string has the following structure:

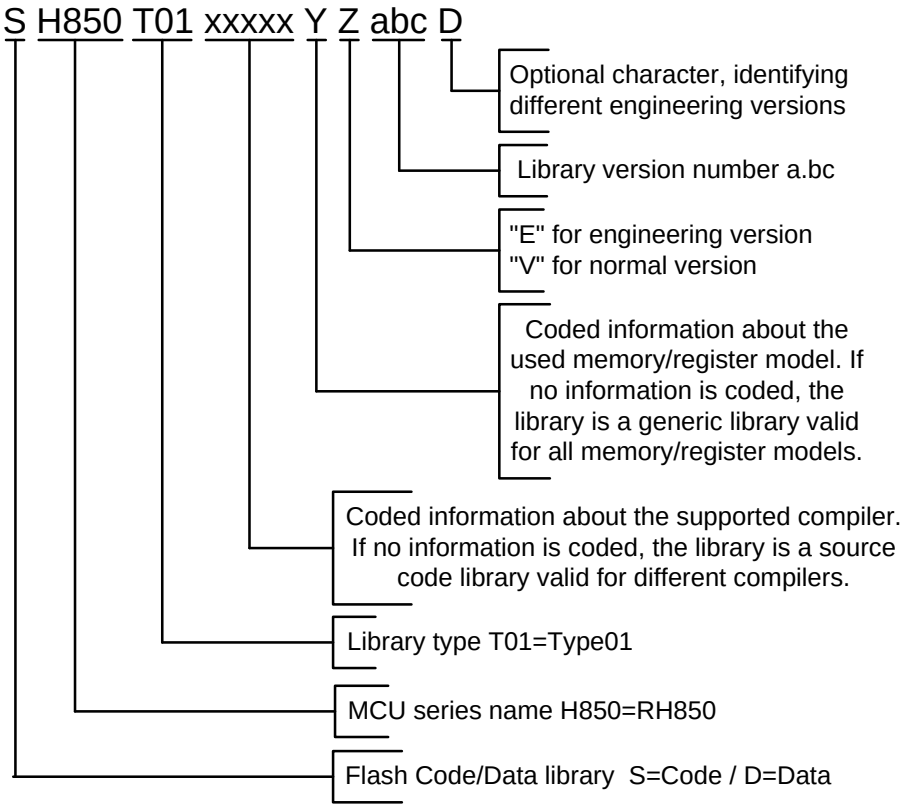

Figure 6: Version string

Example:

```
/* Read library version */
const uint8_t *version_pu08;

version_pu08 = R_FCL_GetVersionString ();
```

4.3.2.2 R_FCL_Execute

Outline: Initiate a new user command

Interface: C Interface

```
void R_FCL_Execute (r_fcl_request_t * request_pstr)
```

Arguments: Parameters

Argument	Type	Access	Description
request_pstr	r_fcl_request_t	rw	Request structure (see chapter 4.2 “Data types” for details). After function return, member status_enu contains the return value of the command initiation explained in chapter 4.2.3 “r_fcl_status_t”.

Return value

None

Pre-conditions: Initialization sequence shall be performed:

- commands that perform flash operations need to have hardware protection disabled (FLMD0 pin/register)
- library must be initialized (call function `R_FCL_Init`)
- the FCL linker segments must be copied (call `R_FCL_CopySections`)
- for commands other than `R_FCL_CMD_PREPARE_ENV`, library must be in prepared state (call `R_FCL_Execute` with `R_FCL_CMD_PREPARE_ENV` command)

Post-conditions: None

Description: The execute function initiates all Flash modification operations. The operation type and operation parameters are passed to the FCL by a request structure, the status and the result of the operation are returned to the user application also by a member of the same structure.

Different combinations of values for members of the request structure are possible. Please check Chapter 4.4 ”Library commands” for how to fill the request structure. Depending on how the library was configured this function operates in two ways:

- In Internal mode, requested command is executed synchronously
- In User mode, requested command is executed asynchronously. Except when an error has occurred, the command execution is only initiated by `R_FCL_Execute` function. Command completion must be carried out by repeatedly calling `R_FCL_Handler` until the status in the request structure is no longer `R_FCL_BUSY`.

Please note that there are important differences between the two modes regarding execution from ROM versus execution from RAM and what areas can be reprogramed. Please check Chapter 6 “Cautions” for what restrictions apply to each mode.

Depending on the used library mode the function can be executed from either ROM or RAM. Thus it is located in linker section R_FCL_CODE_ROMRAM.

Example:

```
/* Erase blocks 10, 11, 12 and 13 */
r_fcl_request_t myRequest;

myRequest.command_enu     = R_FCL_CMD_ERASE
myRequest.idx_u32         = 10
myRequest.cnt_u16         = 4

R_FCL_Execute (&myRequest);

#if R_FCL_COMMAND_EXECUTION_MODE == R_FCL_HANDLER_CALL_USER
    while (myRequest.status_enu == R_FCL_BUSY)
    {
        R_FCL_Handler ();
    }
#endif

if (R_FCL_OK != myRequest.status_enu)
{
    /* Error treatment ... */
}
```

4.3.2.3 R_FCL_Handler

Caution: Function is only available in user mode

Outline: Handles FCL command and operating processing.

Interface: C Interface

```
void R_FCL_Handler (void)
```

Arguments: Parameters

None

Return value

None

Pre-conditions: Initialization sequence shall be performed:

- commands that perform flash operations need to have hardware protection disabled (FLMD0 pin/register)
- library must be initialized (call function `R_FCL_Init`)
- the FCL linker segments must be copied (call `R_FCL_CopySections`)
- for commands other than `R_FCL_CMD_PREPARE_ENV`, library must be in prepared state (call `R_FCL_Execute` with `R_FCL_CMD_PREPARE_ENV` command)

Post-conditions: None

Description: This function handles the command processing for the FCL Flash operations. After operation initiation by `R_FCL_Execute`, this function needs to be called frequently. The function checks the operation status and updates the request structure `status_enu` variable when the operation has finished. By that, the operations end can be polled.

The function should be executed from RAM. Thus it is located in linker section `R_FCL_CODE_RAM`.

Example: see `R_FCL_Execute` (user mode)

4.3.2.4 R_FCL_SuspendRequest

Caution: Function is only available in user mode

Outline: This function requests suspending an on-going Flash Erase or Write operation (e.g. in order to be able to read the Flash).

Interface: C Interface

```
r_fcl_status_t R_FCL_SuspendRequest (void)
```

Arguments: Parameters

None

Return value

Type	Description		
r_fcl_status_t	R_FCL_OK	meaning	operation finished successfully
		reason	no problems during execution
		remedy	nothing
	R_FCL_ERR_FLOW	meaning	suspend currently not possible
		reason	wrong library handling flow (e.g. library not initialized, no operation on-going)
		remedy	correct the flow
	R_FCL_ERR_REJECTED	meaning	suspend currently not possible
		reason	the on-going command is not suspend-able (not an erase or write command)
		remedy	nothing, call suspend only for Erase and Write commands

Pre-conditions:
- A Flash Erase or Write command must be started or operating
- Another command may not be suspended already

Post-conditions:
- It is not possible to suspend an `R_FCL_CMD_ERASE` command to perform another `R_FCL_CMD_ERASE` operation.
- It is not possible to perform `R_FCL_CMD_ERASE` or `R_FCL_CMD_WRITE` if the suspended command is `R_FCL_CMD_WRITE`.

Description: The function suspends an on-going Flash Erase or Write operation. A suspend is just requested by this function. Suspend handling is done by the `R_FCL_Handler` function. Thus `R_FCL_Handler` must be executed until the Flash operation is suspended. This is reported by the request structure status return value `R_FCL_SUSPENDED`.

The function should be executed from RAM or any other save location. Thus it is located in linker section R_FCL_CODE_RAM.

Example:

```
/* Erase blocks 0, 1, 2 and 3 */
r_fcl_request_t myRequest;
r_fcl_status_t  srRes_enu;
uint32_t        i;

myRequest.command_enu     = R_FCL_CMD_ERASE;
myRequest.idx_u32         = 0;
myRequest.cnt_u16         = 4;

R_FCL_Execute (&myRequest);

/* call the handler some time */
i = 0;
while ((myRequest.status_enu == R_FCL_BUSY) && (i < 10))
{
    R_FCL_Handler ();
    i++;
}

/* Suspend request and wait until suspended */
srRes_enu = R_FCL_SuspendRequest ();
if (srRes_enu != R_FCL_OK)
{
    /* Error treatment ... */
}

while (myRequest.status_enu != R_FCL_SUSPENDED)
{
    R_FCL_Handler ();
}

/* Now the FCL is suspended and we can read the Flash ... */

/* Erase resume */
srRes_enu = R_FCL_ResumeRequest ();
if (srRes_enu != R_FCL_OK)
{
    /* Error treatment ... */
}

/* Finish the erase */
while (myRequest.status_enu == R_FCL_SUSPENDED)
{
    R_FCL_Handler ();
}
while (myRequest.status_enu == R_FCL_BUSY)
{
    R_FCL_Handler ();
}

if (myRequest.status_enu != R_FCL_OK)
{
    /* Error treatment ... */
}
```

4.3.2.5 R_FCL_ResumeRequest

Caution: Function is only available in user mode

Outline: This function requests to resume a previously suspended command.

Interface: C Interface

```
r_fcl_status_t R_FCL_ResumeRequest (void)
```

Arguments: Parameters

None

Return value

Type	Description		
r_fcl_status_t	R_FCL_OK	meaning	operation finished successfully
		reason	no problems during execution
		remedy	nothing
	R_FCL_ERR_FLOW	meaning	resume currently not possible
		reason	wrong library handling flow (e.g. library not initialized, no operation suspended, on-going data flash operation)
		remedy	correct the flow

Pre-conditions: A flash operating command must have been successfully suspended (status should be `R_FCL_SUSPENDED`).

If the content of the request structure was changed during stand-by period it must be restored.

Post-conditions: None.

Description: This function requests to resume the previous suspended FCL operation. The resume is just requested by this function. Resume handling is done by the `R_FCL_Handler` function. Thus `R_FCL_Handler` must be executed until the Flash operation is resumed. This is reported by the request structure status return value.

The function should be executed from RAM or any other save location. Thus it is located in linker section `R_FCL_CODE_RAM`.

Example: see `R_FCL_SuspendRequest`

4.3.2.6 R_FCL_CancelRequest

Caution: Function is only available in user mode

Outline: This function requests cancelling an on-going or suspended Erase or Write Flash operation.

Interface: C Interface

```
r_fcl_status_t R_FCL_CancelRequest (void)
```

Arguments: Parameters

None

Return value

Type	Description		
r_fcl_status_t	R_FCL_OK	meaning	operation finished successfully
		reason	no problems during execution
		remedy	nothing
	R_FCL_ERR_FLOW	meaning	cancel currently not possible
		reason	wrong library handling flow (e.g. library not initialized, no operation on-going or suspended, on-going data flash operation)
		remedy	correct the flow
	R_FCL_ERR_REJECTED	meaning	cancel currently not possible
		reason	the on-going command is not cancel-able (not an erase or write command)
		remedy	nothing, call cancel only for on-going or suspended Erase and Write commands

Pre-conditions:
- A Flash Erase or Write command must be started or operating or suspended
- No other cancel request accepted

Post-conditions: None

Description: The function cancels an on-going or suspended Flash Erase/Write operation. A cancel is just requested by this function. Cancel handling is done by the `R_FCL_Handler` function. Thus `R_FCL_Handler` must be executed until the Flash operation is cancelled. This is reported by the request structure status return value `R_FCL_CANCELLED`.

The function should be executed from RAM or any other save location. Thus it is located in linker section R_FCL_CODE_RAM.

Example:

```
/* Erase block 0,1,2 and 3 */
r_fcl_request_t myRequest ;
r_fcl_status_t  srRes_enu ;
uint32_t i ;

myRequest.command_enu     = R_FCL_CMD_ERASE
myRequest.idx_u32         = 0
myRequest.cnt_u16         = 4

R_FCL_Execute(&myRequest);

/* call the handler some time */
i= 0;
while ((myRequest.status_enu == R_FCL_BUSY) && (i<10))
{
    R_FCL_Handler ();
    i++;
}

/* Cancel request and wait until cancelled */
srRes_enu = R_FCL_CancelRequest () ;
if (R_FCL_OK != srRes_enu)
{
    /* Error treatment */
    ...
)

while (R_FCL_CANCELLED != myRequest.status_enu)
{
    R_FCL_Handler ();
}
```

4.4 Library commands

A short overview of the commands available for FCL and the request structure used by each command is given in the following table. Please note that commands are stripped of `R_FCL_CMD_` prefix.

Table 4: Request structure for commands

Command	bufferAdd_u32 [buffer address]	idx_u32 [index]	cnt_u16 [count]
PREPARE_ENV	X	X	X
ERASE	X	starting block number	number of blocks to erase
WRITE	address of source data – size should be a multiple of 256 bytes	destination flash address – 256 bytes aligned	number of groups of 256 bytes to write
SET_LOCKBIT	X	block number	X
GET_LOCKBIT	address of destination data – 1 word	block number	X
ENABLE_LOCKBITS	X	X	X
DISABLE_LOCKBITS	X	X	X
SET_OTP	X	block number	X
GET_OTP	address of destination data – 1 word	block number	X
SET_OPB	address of source data – 32 bytes	X	X
GET_OPB	address of destination data – 32 bytes	X	X
SET_ID	address of source data – 16 bytes	X	X
GET_ID	address of destination data – 16 bytes	X	X
SET_READ_PROTECT_FLAG	X	X	X
GET_READ_PROTECT_FLAG	address of destination data – 1 word	X	X
SET_WRITE_PROTECT_FLAG	X	X	X
GET_WRITE_PROTECT_FLAG	address of destination data – 1 word	X	X
SET_ERASE_PROTECT_FLAG	X	X	X
GET_ERASE_PROTECT_FLAG	address of destination data – 1 word	X	X
SET_SERIAL_PROG_DISABLED	X	X	X
GET_SERIAL_PROG_DISABLED	address of destination data – 1 word	X	X
SET_SERIAL_ID_ENABLED	X	X	X
GET_SERIAL_ID_ENABLED	address of destination data – 1 word	X	X
SET_RESET_VECTOR	address of source data – 16 bytes	X	X

Command	bufferAdd_u32 [buffer address]	idx_u32 [index]	cnt_u16 [count]
GET_RESET_VECTOR	address of destination data – 16 bytes	X	X
GET_BLOCK_CNT	address of destination data – 1 word	X	X
GET_BLOCK_END_ADDR	address of destination data – 1 word	block number	X
GET_DEVICE_NAME	address of destination data – 16 bytes	X	X

X – Structure member is not used

Note 1: In all cases CPU alignment requirements apply.

4.4.1 R_FCL_CMD_PREPARE_ENV

The prepare environment command is used to initialize the Flash programming hardware. This code is used during self-programming and needs to be executed outside the internal Code Flash.

The Code Flash might become inaccessible during command execution.

Table 5: Configuration of the request structure

Request structure member	Value	Description
command_enu	R_FCL_CMD_PREPARE_ENV	requested command
bufferAdd_u32	not used	
idx_u32	not used	
cnt_u16	not used	
status_enu	see next table	automatically updated by `R_FCL_Execute` and `R_FCL_Handler`

When the command is finished, the result will be updated in the request structure `status_enu` member. If the library is configured in user mode, the operation member `status_enu` during the operation is set to `R_FCL_BUSY`.

Table 6: Status returned by the operation

Status	Background and Handling	
R_FCL_OK	meaning	operation finished successfully, the library is prepared for further usage
	reason	no problems during execution
	remedy	nothing to do
R_FCL_BUSY*	meaning	operation is not finished yet
	reason	no problems during execution
	remedy	continue to call `R_FCL_Handler`
R_FCL_ERR_FLOW	meaning	current command is rejected
	reason	Library is not initialized or in a wrong state. Thus function execution is not possible.

Status	Background and Handling	
	remedy	investigate in the root cause and correct the library handling flow
R_FCL_ERR_REJECTED	meaning	current command is rejected
	reason	library is busy performing another operation
	remedy	repeat the command when the preceding command is finished
R_FCL_ERR_PARAMETER	meaning	current command is rejected or stopped
	reason	wrong settings in the descriptor detected (e.g. CPU frequency)
	remedy	correct the values in the descriptor and repeat the command
R_FCL_ERR_INTERNAL	meaning	a library internal error occurred, which could not happen in case of normal application execution
	reason	error that cannot be determined by the library, such as caused by • FCL code or data sections destruction, wrong program flow • hardware defect
	remedy	Please refer to section 3.7 ”Internal error”.
R_FCL_ERR_PROTECTION	meaning	current command is rejected
	reason	• authentication ID value in the descriptor does not match the one in the device • FLMD0 register / pin was changed to low during command operation • defect hardware
	remedy	• correct authentication ID setting in the descriptor • check FLMD0 setting and correct it • change the device
R_FCL_ERR_FLMD0	meaning	• current command is rejected
	reason	• the FLMD0 register / pin is not set correctly
	remedy	• investigate in the root cause and correct the register value or the input of the pin

*available in user mode only

4.4.2 R_FCL_CMD_ERASE

The erase command is used to erase a number of Flash blocks defined by a start block and the number of blocks. This command can be executed also on the second Code Flash Bank, if available.

The Code Flash might become inaccessible during command execution.

Table 7: Configuration of the request structure

Request structure member	Value	Description
command_enu	R_FCL_CMD_ERASE	requested command
bufferAdd_u32	not used	
idx_u32	0 ... block count of the device – 1	operation start block in user area
	0x80000000 ... 0x80000000u + block count of the user boot area - 1	operation start block in user boot area
cnt_u16	1 ... block count of the device - idx_u32	number of blocks to erase
status_enu	see next table	automatically updated by `R_FCL_Execute` and `R_FCL_Handler`

When the command is finished, the result will be updated in the request structure `status_enu` member. If the library is configured in user mode, the operation member `status_enu` during the operation is set to `R_FCL_BUSY`.

Table 8: Status returned by the operation

Status	Background and Handling	
R_FCL_OK	meaning	operation finished successfully, the defined blocks are blank now
	reason	no problems during execution
	remedy	nothing to do
R_FCL_BUSY*	meaning	operation started successfully
	reason	no problems during execution
	remedy	continue to call `R_FCL_Handler`
R_FCL_SUSPENDED*	meaning	an on-going erase operation was successfully suspended
	reason	suspend processing successfully finished
	remedy	nothing to do
R_FCL_CANCELLED*	meaning	an on-going or suspended erase operation was successfully cancelled
	reason	cancel processing successfully finished
	remedy	nothing
R_FCL_ERR_FLOW	meaning	current command is rejected
	reason	Library is not initialized or in a wrong state. Thus function execution is not possible
	remedy	investigate in the root cause and correct the library handling flow
R_FCL_ERR_FLMD0	meaning	current command is rejected

Status	Background and Handling	
	reason	the FLMD0 register / pin is not set correctly
	remedy	investigate in the root cause and correct the register value or the input of the pin
R_FCL_ERR_REJECTED	meaning	current command is rejected
	reason	library is busy performing another operation
	remedy	repeat the command when the preceding command is finished
R_FCL_ERR_PARAMETER	meaning	current command is rejected or stopped
	reason	wrong block or count value specified
	remedy	investigate in the root cause and correct the parameters
R_FCL_ERR_ERASE	meaning	affected Flash area could not be erased completely
	reason	• FCL was initialized with incorrect CPU frequency • hardware defect • FLMD0 not stable during command execution
	remedy	• Re-initialize FCL with correct frequency • a Flash block, respectively the complete Code Flash, should be considered as defect • correct FLMD0 handling and repeat the command
R_FCL_ERR_PROTECTION	meaning	current command is rejected
	reason	current security settings (security flags) prevent a flash block erase
	remedy	disable security setting and repeat the command
R_FCL_ERR_INTERNAL	meaning	a library internal error occurred, which could not happen in case of normal application execution
	reason	error that cannot be determined by the library, such as caused by • FCL code or data sections destruction, wrong program flow • hardware defect
	remedy	Please refer to section 3.7 ”Internal error”.

*available in user mode only

4.4.3 R_FCL_CMD_WRITE

The write command is used to write a number of data words (256 bytes aligned) located in the RAM into the Code Flash at the location specified by the destination address. This command can be executed also on the second Code Flash Bank, if available.

The Code Flash might become inaccessible during command execution.

Table 9: Configuration of the request structure

Request structure member	Value	Description
command_enu	R_FCL_CMD_WRITE	requested command
bufferAdd_u32		data source address
idx_u32	0 ... last Flash address - 256	write destination address in user area
	0x01000000 ... last user boot address - 256	write destination address in user boot area
cnt_u16	1 ... Flash size / 256	number of 256 bytes blocks to write
status_enu	see next table	automatically updated by `R_FCL_Execute` and `R_FCL_Handler`

When the command is finished, the result will be updated in the request structure `status_enu` member.

If the library is configured in user mode, the operation member `status_enu` during the operation is set to `R_FCL_BUSY`.

Table 10: Status returned by the operation

Status	Background and Handling	
R_FCL_OK	meaning	operation finished successfully, the defined blocks are written now
	reason	no problems during execution
	remedy	nothing to do
R_FCL_BUSY*	meaning	operation started successfully
	reason	no problems during execution
	remedy	continue to call `R_FCL_Handler`
R_FCL_SUSPENDED*	meaning	an on-going write operation was successfully suspended
	reason	suspend processing successfully finished
	remedy	nothing
R_FCL_CANCELLED*	meaning	an on-going or suspended write operation was successfully cancelled
	reason	cancel processing successfully finished
	remedy	nothing
R_FCL_ERR_FLOW	meaning	current command is rejected
	reason	library is not initialized or in a wrong state, thus function execution is not possible
	remedy	investigate in the root cause and correct the library handling flow
R_FCL_ERR_FLMD0	meaning	current command is rejected

Status	Background and Handling	
	reason	the FLMD0 register / pin is not set correctly
	remedy	investigate in the root cause and correct the register value or the input of the pin
R_FCL_ERR_REJECTED	meaning	current command is rejected
	reason	library is busy performing another operation
	remedy	repeat the command when the preceding command is finished
R_FCL_ERR_PARAMETER	meaning	current command is rejected or stopped
	reason	wrong destination address or wrong count value specified
	remedy	investigate in the root cause and correct the parameters
R_FCL_ERR_WRITE	meaning	Affected Flash area could not be written correctly
	reason	• affected Flash area to write was not completely and successfully erased • FCL was initialized with incorrect CPU frequency • hardware defect • FLMD0 not stable during command execution
	remedy	• correct application flow (erase Flash area before write operation) • Re-initialize FCL with correct frequency • a Flash block, respectively the complete Code Flash, should be considered as defect • correct FLMD0 handling and repeat the command
R_FCL_ERR_PROTECTION	meaning	current command is rejected
	reason	current security settings (security flags) prevent a modification of the code flash
	remedy	disable security setting and repeat the command
R_FCL_ERR_INTERNAL	meaning	a library internal error occurred, which could not happen in case of normal application execution
	reason	error that cannot be determined by the library, such as caused by • FCL code or data sections destruction, wrong program flow • hardware defect
	remedy	Please refer to section 3.7 ”Internal error”.

*available in user mode only

4.4.4 R_FCL_CMD_SET_LOCKBIT

The command is used to disable write and erase of a specified Flash block.

Note 1: When an `R_FCL_CMD_ERASE` command is successfully performed on a certain block, the lock bits are erased together with the content of the block.

Note 2: The effect of the lock bits can be enabled or disabled with `R_FCL_CMD_ENABLE_LOCKBITS` / `R_FCL_CMD_DISABLE_LOCKBITS` commands.

Note 3: If errors are produced during this command then the result of `R_FCL_CMD_GET_LOCKBIT` command is not reliable.

Table 11: Configuration of the request structure

Request structure member	Value	Description
command_enu	R_FCL_CMD_SET_LOCKBIT	requested command
bufferAdd_u32	not used	
idx_u32	0 ... block count of the device – 1	block number in user area
	0x80000000 ... 0x80000000u + block count of the user boot area - 1	block number in user boot area
cnt_u16	not used	
status_enu	see next table	automatically updated by `R_FCL_Execute` and `R_FCL_Handler`

When the command is finished, the result will be updated in the request structure `status_enu` member. If the library is configured in user mode, the operation member `status_enu` during the operation is set to `R_FCL_BUSY`.

Table 12: Status returned by the operation

Status	Background and Handling	
R_FCL_OK	meaning	operation finished successfully, the requested block is now locked and cannot be erased or written
	reason	no problems during execution
	remedy	nothing to do
R_FCL_BUSY*	meaning	operation started successfully
	reason	no problems during execution
	remedy	continue to call `R_FCL_Handler`
R_FCL_ERR_FLOW	meaning	current command is rejected
	reason	library is not initialized or in a wrong state, thus function execution is not possible
	remedy	investigate in the root cause and correct the library handling flow
R_FCL_ERR_FLMD0	meaning	current command is rejected
	reason	the FLMD0 register / pin is not set correctly
	remedy	investigate in the root cause and correct the register value or the input of the pin
R_FCL_ERR_WRITE	meaning	Lock bits could not be written successfully

Status	Background and Handling	
	reason	• affected Flash block was not completely and successfully erased • FCL was initialized with incorrect CPU frequency • hardware defect • FLMD0 not stable during command execution
	remedy	• correct application flow (erase Flash block before setting the lock bit) • Re-initialize FCL with correct frequency • a Flash block, respectively the complete Code Flash, should be considered as defect • correct FLMD0 handling and repeat the command
R_FCL_ERR_REJECTED	meaning	current command is rejected
	reason	library is busy performing another operation
	remedy	repeat the command when the preceding command is finished
R_FCL_ERR_PARAMETER	meaning	current command is rejected or stopped
	reason	wrong block value specified
	remedy	investigate in the root cause and correct the parameters
R_FCL_ERR_PROTECTION	meaning	current command is rejected
	reason	current security settings (security flags) prevent a modification of the lock bit
	remedy	disable security setting and repeat the command
R_FCL_ERR_INTERNAL	meaning	a library internal error occurred, which could not happen in case of normal application execution
	reason	error that cannot be determined by the library, such as caused by • FCL code or data sections destruction, wrong program flow • hardware defect
	remedy	Please refer to section 3.7 ”Internal error”.

*available in user mode only

4.4.5 R_FCL_CMD_GET_LOCKBIT

The command is used to read the protection setting of a specified Flash block. A set flag is indicated by a one, a cleared flag by a zero.

Note 1: When an `R_FCL_ERR_ERASE` command is successfully performed on a certain block, the lock bits are erased together with the content of the block.

Note 2: The effect of the lock bits can be enabled or disabled with `R_FCL_CMD_ENABLE_LOCKBITS` / `R_FCL_CMD_DISABLE_LOCKBITS` commands.

Table 13: Configuration of the request structure

Request structure member	Value	Description
command_enu	R_FCL_CMD_GET_LOCKBIT	requested command

Request structure member	Value	Description
bufferAdd_u32		address of buffer to store the return value of the “get” command (buffer contains a word)
idx_u32	0 ... block count of the device – 1	block number in user area
	0x80000000 ... 0x80000000u + block count of the user boot area - 1	block number in user boot area
cnt_u16	not used	
status_enu	see next table	automatically updated by `R_FCL_Execute` and `R_FCL_Handler`

When the command is finished, the result will be updated in the request structure `status_enu` member.
If the library is configured in user mode, the operation member `status_enu` during the operation is set to `R_FCL_BUSY`.

Table 14: Status returned by the operation

Status	Background and Handling	
R_FCL_OK	meaning	operation finished successfully, the lock bit for the requested block is available in buffer
	reason	no problems during execution
	remedy	nothing to do
R_FCL_BUSY*	meaning	operation started successfully
	reason	no problems during execution
	remedy	continue to call `R_FCL_Handler`
R_FCL_ERR_FLOW	meaning	current command is rejected
	reason	library is not initialized or in a wrong state, thus function execution is not possible
	remedy	investigate in the root cause and correct the library handling flow
R_FCL_ERR_REJECTED	meaning	current command is rejected
	reason	library is busy performing another operation
	remedy	repeat the command when the preceding command is finished
R_FCL_ERR_INTERNAL	meaning	a library internal error occurred, which could not happen in case of normal application execution
	reason	error that cannot be determined by the library, such as caused by • FCL code or data sections destruction, wrong program flow • hardware defect
	remedy	Please refer to section 3.7 ”Internal error”.
R_FCL_ERR_PARAMETER	meaning	current command is rejected or stopped
	reason	wrong block value or wrong destination buffer address was specified

Status	Background and Handling	
	remedy	investigate in the root cause and correct the parameters

*available in user mode only

4.4.6 R_FCL_CMD_ENABLE_LOCKBITS

The command is used to enable the mechanism to protect single Flash blocks by a dedicated lock bit. If lock bits mechanism is enabled and a `R_FCL_CMD_ERASE` or `R_FCL_CMD_WRITE` command is performed on a certain block that has its lock bits set, will result in `R_FCL_ERR_PROTECTION` status.

Table 15: Configuration of the request structure

Request structure member	Value	Description
command_enu	R_FCL_CMD_ENABLE_LOCKBITS	requested command
bufferAdd_u32	not used	
idx_u32	not used	
cnt_u16	not used	
status_enu	see next table	automatically updated by `R_FCL_Execute` and `R_FCL_Handler`

When the command is finished, the result will be updated in the request structure `status_enu` member.

Table 16: Status returned by the operation

Status	Background and Handling	
R_FCL_OK	meaning	operation finished successfully, the lock bit mechanism is enabled now
	reason	no problems during execution
	remedy	nothing to do
R_FCL_ERR_FLOW	meaning	library is not initialized or in a wrong state, thus function execution is not possible
	reason	investigate in the root cause and correct the library handling flow
	remedy	current command is rejected
R_FCL_ERR_REJECTED*	meaning	current command is rejected
	reason	library is busy performing another operation
	remedy	repeat the command when the preceding command is finished

*available in user mode only

4.4.7 R_FCL_CMD_DISABLE_LOCKBITS

The command is used to disable the mechanism to protect single Flash blocks by a dedicated lock bit. If lock bits mechanism is disabled `R_FCL_CMD_ERASE` or `R_FCL_CMD_WRITE` commands perform normally on a certain block that has its lock bits set. A successful `R_FCL_CMD_ERASE` operation will also erase lock bits associated with the erased blocks when lock bits mechanism is disabled.

Table 17: Configuration of the request structure

Request structure member	Value	Description
command_enu	R_FCL_CMD_DISABLE_LOCKBITS	requested command
bufferAdd_u32	not used	
idx_u32	not used	
cnt_u16	not used	
status_enu	see next table	automatically updated by `R_FCL_Execute` and `R_FCL_Handler`

When the command is finished, the result will be updated in the request structure `status_enu` member.

Table 18: Status returned by the operation

Status	Background and Handling	
R_FCL_OK	meaning	operation finished successfully, the lock bit mechanism is disabled now
	reason	no problems during execution
	remedy	nothing to do
R_FCL_ERR_FLOW	meaning	current command is rejected
	reason	library is not initialized or in a wrong state, thus function execution is not possible
	remedy	investigate in the root cause and correct the library handling flow
R_FCL_ERR_REJECTED*	meaning	any other operation is on-going
	reason	repeat the command when the preceding command is finished
	remedy	library is not initialized or in a wrong state, thus function execution is not possible

*available in user mode only

4.4.8 R_FCL_CMD_SET_OTP

The command is used to set the OTP bit which disables further modifications of the affected Flash block of the device.

Note: If errors are produced during this command then the result of `R_FCL_CMD_GET_OTP` command is not reliable.

Table 19: Configuration of the request structure

Request structure member	Value	Description
command_enu	R_FCL_CMD_SET_OTP	requested command
bufferAdd_u32	not used	
idx_u32	0 ... block count of the device – 1	block number in user area
	0x80000000 ... 0x80000000u + block count of the user boot area - 1	block number in user boot area
cnt_u16	not used	

Request structure member	Value	Description
status_enu	see next table	automatically updated by `R_FCL_Execute` and `R_FCL_Handler`

When the command is finished, the result will be updated in the request structure `status_enu` member.
If the library is configured in user mode, the operation member `status_enu` during the operation is set to `R_FCL_BUSY`.

Table 20: Status returned by the operation

Status	Background and Handling	
R_FCL_OK	meaning	operation finished successfully, the OTP bit for the requested block is now set
	reason	no problems during execution
	remedy	nothing to do
R_FCL_BUSY*	meaning	operation started successfully
	reason	no problems during execution
	remedy	continue to call `R_FCL_Handler`
R_FCL_ERR_FLOW	meaning	current command is rejected
	reason	library is not initialized or in a wrong state, thus function execution is not possible
	remedy	investigate in the root cause and correct the library handling flow
R_FCL_ERR_PARAMETER	meaning	current command is rejected or stopped
	reason	wrong block value
	remedy	investigate in the root cause and correct the parameters
R_FCL_ERR_FLMD0	meaning	current command is rejected
	reason	the FLMD0 register / pin is not set correctly
	remedy	investigate in the root cause and correct the register value or the input of the pin
R_FCL_ERR_WRITE	meaning	OTP bit could not be written correctly
	reason	• FCL was initialized with incorrect CPU frequency • hardware defect • FLMD0 not stable during command execution
	remedy	• Re-initialize FCL with correct frequency • consider the configuration Flash area, respectively the complete Flash as defect • check FLMD0 setting and repeat command
R_FCL_ERR_ERASE	meaning	affected Flash area could not be erased completely
	reason	• FCL was initialized with incorrect CPU frequency • hardware defect • FLMD0 not stable during command execution

Status	Background and Handling	
	remedy	• Re-initialize FCL with correct frequency • a Flash block, respectively the complete Code Flash, should be considered as defect • correct FLMD0 handling and repeat the command
R_FCL_ERR_REJECTED	meaning	current command is rejected
	reason	library is busy performing another operation
	remedy	repeat the command when the preceding command is finished
R_FCL_ERR_INTERNAL	meaning	a library internal error occurred, which could not happen in case of normal application execution
	reason	error that cannot be determined by the library, such as caused by • FCL code or data sections destruction, wrong program flow • hardware defect
	remedy	Please refer to section 3.7 ”Internal error”.

*available in user mode only

4.4.9 R_FCL_CMD_GET_OTP

The command is used to read the protection setting of individual Code Flash blocks.

Table 21: Configuration of the request structure

Request structure member	Value	Description
command_enu	R_FCL_CMD_GET_OTP	requested command
bufferAdd_u32		address of buffer to store the return value of the “get” command. Return value 0 (OTP not set) or 1 (OTP set)
idx_u32	0 ... block count of the device – 1	block number in user area
	0x80000000 ... 0x80000000u + block count of the user boot area - 1	block number in user boot area
cnt_u16	not used	
status_enu	see next table	automatically updated by `R_FCL_Execute` and `R_FCL_Handler`

When the command is finished, the result will be updated in the request structure `status_enu` member.

Table 22: Status returned by the operation

Status	Background and Handling	
R_FCL_OK	meaning	operation finished successfully, the OTP settings are available now in buffer
	reason	no problems during execution
	remedy	nothing to do

Status	Background and Handling	
R_FCL_ERR_FLOW	meaning	current command is rejected
	reason	library is not initialized or in a wrong state, thus function execution is not possible
	remedy	investigate in the root cause and correct the library handling flow
R_FCL_ERR_PARAMETER	meaning	current command is rejected or stopped
	reason	wrong block value or wrong destination buffer address was specified
	remedy	investigate in the root cause and correct the parameters
R_FCL_ERR_REJECTED*	meaning	current command is rejected
	reason	library is busy performing another operation
	remedy	repeat the command when the preceding command is finished

*available in user mode only

4.4.10 R_FCL_CMD_SET_OPB

The command is used to set the option bytes of the device.

Note: If errors are produced during this command then the result of `R_FCL_CMD_GET_OPB` command is not reliable.

Table 23: Configuration of the request structure

Request structure member	Value	Description
command_enu	R_FCL_CMD_SET_OPB	requested command
bufferAdd_u32		option bytes source buffer address, 32 bytes
idx_u32	not used	
cnt_u16	not used	
status_enu	see next table	automatically updated by `R_FCL_Execute` and `R_FCL_Handler`

When the command is finished, the result will be updated in the request structure `status_enu` member. If the library is configured in user mode, the operation member `status_enu` during the operation is set to `R_FCL_BUSY`.

Table 24: Status returned by the operation

Status	Background and Handling	
R_FCL_OK	meaning	operation finished successfully, the option bytes are now set
	reason	no problems during execution
	remedy	nothing to do
R_FCL_BUSY*	meaning	operation started successfully
	reason	no problems during execution
	remedy	continue to call `R_FCL_Handler`

Status	Background and Handling	
R_FCL_ERR_FLOW	meaning	current command is rejected
	reason	library is not initialized or in a wrong state, thus function execution is not possible
	remedy	investigate in the root cause and correct the library handling flow
R_FCL_ERR_PARAMETER	meaning	current command is rejected or stopped
	reason	wrong source buffer address was specified
	remedy	investigate in the root cause and correct the parameters
R_FCL_ERR_FLMD0	meaning	current command is rejected
	reason	the FLMD0 register / pin is not set correctly
	remedy	investigate in the root cause and correct the register value or the input of the pin
R_FCL_ERR_WRITE	meaning	data could not be written correctly
	reason	• FCL was initialized with incorrect CPU frequency • hardware defect • FLMD0 not stable during command execution
	remedy	• Re-initialize FCL with correct frequency • configuration Flash area, respectively the complete Flash, should be considered as defect • check FLMD0 setting and repeat command
R_FCL_ERR_ERASE	meaning	affected Flash area could not be erased completely
	reason	• FCL was initialized with incorrect CPU frequency • hardware defect • FLMD0 not stable during command execution
	remedy	• Re-initialize FCL with correct frequency • a Flash block, respectively the complete Code Flash, should be considered as defect • correct FLMD0 handling and repeat the command
R_FCL_ERR_REJECTED	meaning	current command is rejected
	reason	library is busy performing another operation
	remedy	repeat the command when the preceding command is finished
R_FCL_ERR_PROTECTION	meaning	current command is rejected
	reason	current security settings (security flags) prevent a modification of the option bytes
	remedy	disable security setting and repeat the command
R_FCL_ERR_INTERNAL	meaning	a library internal error occurred, which could not happen in case of normal application execution
	reason	error that cannot be determined by the library, such as caused by • FCL code or data sections destruction, wrong program flow • hardware defect
	remedy	Please refer to section 3.7 ”Internal error”.

*available in user mode only

4.4.11 R_FCL_CMD_GET_OPB

The command is used to read the current option bytes.

Table 25: Configuration of the request structure

Request structure member	Value	Description
command_enu	R_FCL_CMD_GET_OPB	requested command
bufferAdd_u32		address of buffer to store the return value of the “get” command (buffer contains a 32-byte value)
idx_u32	not used	
cnt_u16	not used	
status_enu	see next table	automatically updated by `R_FCL_Execute` and `R_FCL_Handler`

When the command is finished, the result will be updated in the request structure `status_enu` member.

Table 26: Status returned by the operation

Status	Background and Handling	
R_FCL_OK	meaning	operation finished successfully, the option bytes are now available in buffer
	reason	no problems during execution
	remedy	nothing to do
R_FCL_ERR_FLOW	meaning	current command is rejected
	reason	library is not initialized or in a wrong state, thus function execution is not possible
	remedy	investigate in the root cause and correct the library handling flow
R_FCL_ERR_REJECTED*	meaning	current command is rejected
	reason	library is busy performing another operation
	remedy	repeat the command when the preceding command is finished
R_FCL_ERR_PARAMETER	meaning	current command is rejected or stopped
	reason	wrong destination buffer address was specified
	remedy	investigate in the root cause and correct the parameters

*available in user mode only

4.4.12 R_FCL_CMD_SET_ID

The command is used to set the ID used for user authentication. The ID is used during self-programming as well as during serial programming.

Note: If errors are produced during this command then the result of `R_FCL_CMD_GET_ID` command is not reliable.

Table 27: Configuration of the request structure

Request structure member	Value	Description
command_enu	R_FCL_CMD_SET_ID	requested command
bufferAdd_u32		ID source buffer address, 16 bytes
idx_u32	not used	
cnt_u16	not used	
status_enu	see next table	automatically updated by `R_FCL_Execute` and `R_FCL_Handler`

When the command is finished, the result will be updated in the request structure `status_enu` member. If the library is configured in user mode, the operation member `status_enu` during the operation is set to `R_FCL_BUSY`.

Table 28: Status returned by the operation

Status	Background and Handling	
R_FCL_OK	meaning	operation finished successfully, the authentication ID is now set
	reason	no problems during execution
	remedy	nothing to do
R_FCL_BUSY*	meaning	operation started successfully
	reason	no problems during execution
	remedy	continue to call `R_FCL_Handler`
R_FCL_ERR_FLOW	meaning	current command is rejected
	reason	library is not initialized or in a wrong state, thus function execution is not possible
	remedy	investigate in the root cause and correct the library handling flow
R_FCL_ERR_PARAMETER	meaning	current command is rejected or stopped
	reason	wrong source buffer address was specified
	remedy	investigate in the root cause and correct the parameters
R_FCL_ERR_FLMD0	meaning	current command is rejected
	reason	the FLMD0 register / pin is not set correctly
	remedy	investigate in the root cause and correct the register value or the input of the pin
R_FCL_ERR_WRITE	meaning	ID could not be written correctly
	reason	• FCL was initialized with incorrect CPU frequency • hardware defect • FLMD0 not stable during command execution

Status	Background and Handling	
	remedy	• Re-initialize FCL with correct frequency • configuration Flash area, respectively the complete Flash, should be considered as defect • check FLMD0 setting and repeat command
R_FCL_ERR_ERASE	meaning	affected Flash area could not be erased completely
	reason	• FCL was initialized with incorrect CPU frequency • hardware defect • FLMD0 not stable during command execution
	remedy	• Re-initialize FCL with correct frequency • a Flash block, respectively the complete Code Flash, should be considered as defect • correct FLMD0 handling and repeat the command
R_FCL_ERR_REJECTED	meaning	current command is rejected
	reason	library is busy performing another operation
	remedy	repeat the command when the preceding command is finished
R_FCL_ERR_PROTECTION	meaning	current command is rejected
	reason	current security settings (security flags) prevent a modification of the code flash
	remedy	disable security setting and repeat the command
R_FCL_ERR_INTERNAL	meaning	a library internal error occurred, which could not happen in case of normal application execution
	reason	error that cannot be determined by the library, such as caused by • FCL code or data sections destruction, wrong program flow • hardware defect
	remedy	Please refer to section 3.7 ”Internal error”.

*available in user mode only

4.4.13 R_FCL_CMD_GET_ID

The command is used to read the current ID setting. The ID must be known to start a library command. The command can be used to prove that the ID update was executed using the correct parameters.

Table 29: Configuration of the request structure

Request structure member	Value	Description
command_enu	R_FCL_CMD_GET_ID	requested command
bufferAdd_u32		address of buffer to store the return value of the “get” command (buffer contains a 16-byte value)
idx_u32	not used	
cnt_u16	not used	
status_enu	see next table	automatically updated by `R_FCL_Execute` and `R_FCL_Handler`

When the command is finished, the result will be updated in the request structure `status_enu` member.

Table 30: Status returned by the operation

Status	Background and Handling	
R_FCL_OK	meaning	operation finished successfully, the authentication ID is now available in buffer
	reason	no problems during execution
	remedy	nothing to do
R_FCL_ERR_FLOW	meaning	current command is rejected
	reason	library is not initialized or in a wrong state, thus function execution is not possible
	remedy	investigate in the root cause and correct the library handling flow
R_FCL_ERR_PARAMETER	meaning	current command is rejected or stopped
	reason	wrong destination buffer address was specified
	remedy	investigate in the root cause and correct the parameters
R_FCL_ERR_REJECTED*	meaning	current command is rejected
	reason	library is busy performing another operation
	remedy	repeat the command when the preceding command is finished

*available in user mode only

4.4.14 R_FCL_CMD_SET_READ_PROTECT_FLAG

The command is used to enable the read protection of the device and disable the read command during serial programming.

Note: If errors are produced during this command then the result of `R_FCL_CMD_GET_READ_PROTECT_FLAG` command is not reliable.

Table 31: Configuration of the request structure

Request structure member	Value	Description
command_enu	R_FCL_CMD_SET_READ_PROTECT_FLAG	requested command
bufferAdd_u32	not used	
idx_u32	not used	
cnt_u16	not used	
status_enu	see next table	automatically updated by `R_FCL_Execute` and `R_FCL_Handler`

When the command is finished, the result will be updated in the request structure `status_enu` member.
If the library is configured in user mode, the operation member `status_enu` during the operation is set to `R_FCL_BUSY`.

Table 32: Status returned by the operation

Status	Background and Handling	
R_FCL_OK	meaning	operation finished successfully, the read protection flag is now set
	reason	no problems during execution
	remedy	nothing to do
R_FCL_BUSY*	meaning	operation started successfully
	reason	no problems during execution
	remedy	continue to call `R_FCL_Handler`
R_FCL_ERR_FLOW	meaning	current command is rejected
	reason	library is not initialized or in a wrong state, thus function execution is not possible
	remedy	investigate in the root cause and correct the library handling flow
R_FCL_ERR_FLMD0	meaning	current command is rejected
	reason	the FLMD0 register / pin is not set correctly
	remedy	investigate in the root cause and correct the register value or the FLMD0 pin input
R_FCL_ERR_WRITE	meaning	flag could not be written correctly
	reason	• FCL was initialized with incorrect CPU frequency • hardware defect • FLMD0 not stable during command execution
	remedy	• Re-initialize FCL with correct frequency • configuration Flash area, respectively the complete Flash, should be considered as defect • check FLMD0 setting and repeat command
R_FCL_ERR_ERASE	meaning	affected Flash area could not be erased completely
	reason	• FCL was initialized with incorrect CPU frequency • hardware defect • FLMD0 not stable during command execution
	remedy	• Re-initialize FCL with correct frequency • a Flash block, respectively the complete Code Flash, should be considered as defect • correct FLMD0 handling and repeat the command
R_FCL_ERR_REJECTED	meaning	current command is rejected
	reason	library is busy performing another operation
	remedy	repeat the command when the preceding command is finished
R_FCL_ERR_INTERNAL	meaning	a library internal error occurred, which could not happen in case of normal application execution
	reason	error that cannot be determined by the library, such as caused by • FCL code or data sections destruction, wrong program flow • hardware defect

Status	Background and Handling	
	remedy	Please refer to section 3.7 ”Internal error”.

*available in user mode only

4.4.15 R_FCL_CMD_GET_READ_PROTECT_FLAG

The command is used to read the current protection setting of the device. A set flag is indicated by a one, a cleared flag by a zero.

Table 33: Configuration of the request structure

Request structure member	Value	Description
command_enu	R_FCL_CMD_GET_READ_PROTECT_FLAG	requested command
bufferAdd_u32		address of buffer to store the return value of the “get” command (buffer contains a word)
idx_u32	not used	
cnt_u16	not used	
status_enu	see next table	automatically updated by `R_FCL_Execute` and `R_FCL_Handler`

When the command is finished, the result will be updated in the request structure `status_enu` member.

Table 34: Status returned by the operation

Status	Background and Handling	
R_FCL_OK	meaning	operation finished successfully, the value of read protection flag is now available in buffer
	reason	no problems during execution
	remedy	nothing to do
R_FCL_ERR_FLOW	meaning	current command is rejected
	reason	library is not initialized or in a wrong state, thus function execution is not possible
	remedy	investigate in the root cause and correct the library handling flow
R_FCL_ERR_PARAMETER	meaning	current command is rejected or stopped
	reason	wrong destination buffer address was specified
	remedy	investigate in the root cause and correct the parameters
R_FCL_ERR_REJECTED*	meaning	current command is rejected
	reason	library is busy performing another operation
	remedy	repeat the command when the preceding command is finished

*available in user mode only

4.4.16 R_FCL_CMD_SET_WRITE_PROTECT_FLAG

The command is used to enable the write protection of the device and disable the write command during serial programming.

Note: If errors are produced during this command then the result of `R_FCL_CMD_GET_WRITE_PROTECT_FLAG` command is not reliable.

Table 35: Configuration of the request structure

Request structure member	Value	Description
command_enu	R_FCL_CMD_SET_WRITE_PROTECT_FLAG	requested command
bufferAdd_u32	not used	
idx_u32	not used	
cnt_u16	not used	
status_enu	see next table	automatically updated by `R_FCL_Execute` and `R_FCL_Handler`

When the command is finished, the result will be updated in the request structure `status_enu` member. If the library is configured in user mode, the operation member `status_enu` during the operation is set to `R_FCL_BUSY`.

Table 36: Status returned by the operation

Status	Background and Handling	
R_FCL_OK	meaning	operation finished successfully, the write protection flag is now set
	reason	no problems during execution
	remedy	nothing to do
R_FCL_BUSY*	meaning	operation started successfully
	reason	no problems during execution
	remedy	continue to call `R_FCL_Handler`
R_FCL_ERR_FLOW	meaning	current command is rejected
	reason	library is not initialized or in a wrong state, thus function execution is not possible
	remedy	investigate in the root cause and correct the library handling flow
R_FCL_ERR_FLMD0	meaning	current command is rejected
	reason	the FLMD0 register / pin is not set correctly
	remedy	investigate in the root cause and correct the register value or the input of the pin
R_FCL_ERR_WRITE	meaning	flag could not be written correctly
	reason	• FCL was initialized with incorrect CPU frequency • hardware defect • FLMD0 not stable during command execution

Status	Background and Handling	
	remedy	• Re-initialize FCL with correct frequency • configuration Flash area, respectively the complete Flash, should be considered as defect • check FLMD0 setting and repeat command
R_FCL_ERR_ERASE	meaning	affected Flash area could not be erased completely
	reason	• FCL was initialized with incorrect CPU frequency • hardware defect • FLMD0 not stable during command execution
	remedy	• Re-initialize FCL with correct frequency • a Flash block, respectively the complete Code Flash, should be considered as defect • correct FLMD0 handling and repeat the command
R_FCL_ERR_REJECTED	meaning	current command is rejected
	reason	library is busy performing another operation
	remedy	repeat the command when the preceding command is finished
R_FCL_ERR_INTERNAL	meaning	a library internal error occurred, which could not happen in case of normal application execution
	reason	error that cannot be determined by the library, such as caused by • FCL code or data sections destruction, wrong program flow • hardware defect
	remedy	Please refer to section 3.7 ”Internal error”.

*available in user mode only

4.4.17 R_FCL_CMD_GET_WRITE_PROTECT_FLAG

The command is used to read the current protection setting of the device. A set flag is indicated by a one, a cleared flag by a zero.

Table 37: Configuration of the request structure

Request structure member	Value	Description
command_enu	R_FCL_CMD_GET_WRITE_PROTECT_FLAG	requested command
bufferAdd_u32		address of buffer to store the return value of the “get” command (buffer contains a word)
idx_u32	not used	
cnt_u16	not used	
status_enu	see next table	automatically updated by `R_FCL_Execute` and `R_FCL_Handler`

When the command is finished, the result will be updated in the request structure `status_enu` member.

If the library is configured in user mode, the operation member `status_enu` during the operation is set to `R_FCL_BUSY`.

Table 38: Status returned by the operation

Status	Background and Handling	
R_FCL_OK	meaning	operation finished successfully, the value of write protection flag is now available in buffer
	reason	no problems during execution
	remedy	nothing to do
R_FCL_ERR_FLOW	meaning	current command is rejected
	reason	library is not initialized or in a wrong state, thus function execution is not possible
	remedy	investigate in the root cause and correct the library handling flow
R_FCL_ERR_PARAMETER	meaning	current command is rejected or stopped
	reason	wrong destination buffer address was specified
	remedy	investigate in the root cause and correct the parameters
R_FCL_ERR_REJECTED*	meaning	current command is rejected
	reason	library is busy performing another operation
	remedy	repeat the command when the preceding command is finished

*available in user mode only

4.4.18 R_FCL_CMD_SET_ERASE_PROTECT_FLAG

The command is used to enable the erase protection of the device and disable the erase command during serial programming.

Note: If errors are produced during this command then the result of `R_FCL_CMD_GET_ERASE_PROTECT_FLAG` command is not reliable.

Table 39: Configuration of the request structure

Request structure member	Value	Description
command_enu	R_FCL_CMD_SET_ERASE_PROTECT_FLAG	requested command
bufferAdd_u32	not used	
idx_u32	not used	
cnt_u16	not used	
status_enu	see next table	automatically updated by `R_FCL_Execute` and `R_FCL_Handler`

When the command is finished, the result will be updated in the request structure `status_enu` member.
If the library is configured in user mode, the operation member `status_enu` during the operation is set to `R_FCL_BUSY`.

Table 40: Status returned by the operation

Status	Background and Handling	
R_FCL_OK	meaning	operation finished successfully, the erase protection flag is now set
	reason	no problems during execution
	remedy	nothing to do
R_FCL_BUSY*	meaning	operation started successfully
	reason	no problems during execution
	remedy	continue to call `R_FCL_Handler`
R_FCL_ERR_FLOW	meaning	current command is rejected
	reason	library is not initialized or in a wrong state, thus function execution is not possible
	remedy	investigate in the root cause and correct the library handling flow
R_FCL_ERR_FLMD0	meaning	current command is rejected
	reason	the FLMD0 register / pin is not set correctly
	remedy	investigate in the root cause and correct the register value or the input of the pin
R_FCL_ERR_WRITE	meaning	flag could not be written correctly
	reason	• FCL was initialized with incorrect CPU frequency • hardware defect • FLMD0 not stable during command execution
	remedy	• Re-initialize FCL with correct frequency • configuration Flash area, respectively the complete Flash, should be considered as defect • check FLMD0 setting and repeat command
R_FCL_ERR_ERASE	meaning	affected Flash area could not be erased completely
	reason	• FCL was initialized with incorrect CPU frequency • hardware defect • FLMD0 not stable during command execution
	remedy	• Re-initialize FCL with correct frequency • a Flash block, respectively the complete Code Flash, should be considered as defect • correct FLMD0 handling and repeat the command
R_FCL_ERR_REJECTED	meaning	current command is rejected
	reason	library is busy performing another operation
	remedy	repeat the command when the preceding command is finished
R_FCL_ERR_INTERNAL	meaning	a library internal error occurred, which could not happen in case of normal application execution
	reason	error that cannot be determined by the library, such as caused by • FCL code or data sections destruction, wrong program flow • hardware defect

Status	Background and Handling	
	remedy	Please refer to section 3.7 ”Internal error”.

*available in user mode only

4.4.19 R_FCL_CMD_GET_ERASE_PROTECT_FLAG

The command is used to read the current protection setting of the device. A set flag is indicated by a one, a cleared flag by a zero.

Table 41: Configuration of the request structure

Request structure member	Value	Description
command_enu	R_FCL_CMD_GET_ERASE_PROTECT_FLAG	requested command
bufferAdd_u32		address of buffer to store the return value of the “get” command (buffer contains a word)
idx_u32	not used	
cnt_u16	not used	
status_enu	see next table	automatically updated by `R_FCL_Execute` and `R_FCL_Handler`

When the command is finished, the result will be updated in the request structure `status_enu` member.

Table 42: Status returned by the operation

Status	Background and Handling	
R_FCL_OK	meaning	operation finished successfully, the value of erase protection flag is now available in buffer
	reason	no problems during execution
	remedy	nothing to do
R_FCL_ERR_FLOW	meaning	current command is rejected
	reason	library is not initialized or in a wrong state, thus function execution is not possible
	remedy	investigate in the root cause and correct the library handling flow
R_FCL_ERR_PARAMETER	meaning	current command is rejected or stopped
	reason	wrong destination buffer address was specified
	remedy	investigate in the root cause and correct the parameters
R_FCL_ERR_REJECTED*	meaning	current command is rejected
	reason	library is busy performing another operation
	remedy	repeat the command when the preceding command is finished

*available in user mode only

4.4.20 R_FCL_CMD_SET_SERIAL_PROG_DISABLED

The command is used to disable complete serial programming.

Note: If errors are produced during this command then the result of `R_FCL_CMD_GET_SERIAL_PROG_DISABLED` command is not reliable.

Table 43: Configuration of the request structure

Request structure member	Value	Description
command_enu	R_FCL_CMD_SET_SERIAL_PROG_DISABLED	requested command
bufferAdd_u32	not used	
idx_u32	not used	
cnt_u16	not used	
status_enu	see next table	automatically updated by `R_FCL_Execute` and `R_FCL_Handler`

When the command is finished, the result will be updated in the request structure `status_enu` member. If the library is configured in user mode, the operation member `status_enu` during the operation is set to `R_FCL_BUSY`.

Table 44: Status returned by the operation

Status	Background and Handling	
R_FCL_OK	meaning	operation finished successfully, programming via serial programming is now disabled
	reason	no problems during execution
	remedy	nothing to do
R_FCL_BUSY*	meaning	operation started successfully
	reason	no problems during execution
	remedy	continue to call `R_FCL_Handler`
R_FCL_ERR_FLOW	meaning	current command is rejected
	reason	library is not initialized or in a wrong state, thus function execution is not possible
	remedy	investigate in the root cause and correct the library handling flow
R_FCL_ERR_FLMD0	meaning	current command is rejected
	reason	the FLMD0 register / pin is not set correctly
	remedy	investigate in the root cause and correct the register value or the input of the pin
R_FCL_ERR_WRITE	meaning	flag could not be written correctly
	reason	• FCL was initialized with incorrect CPU frequency • hardware defect • FLMD0 not stable during command execution

Status	Background and Handling	
	remedy	• Re-initialize FCL with correct frequency • configuration Flash area, respectively the complete Flash, should be considered as defect • check FLMD0 setting and repeat command
R_FCL_ERR_ERASE	meaning	affected Flash area could not be erased completely
	reason	• FCL was initialized with incorrect CPU frequency • hardware defect • FLMD0 not stable during command execution
	remedy	• Re-initialize FCL with correct frequency • a Flash block, respectively the complete Code Flash, should be considered as defect • correct FLMD0 handling and repeat the command
R_FCL_ERR_REJECTED	meaning	current command is rejected
	reason	library is busy performing another operation
	remedy	repeat the command when the preceding command is finished
R_FCL_ERR_INTERNAL	meaning	a library internal error occurred, which could not happen in case of normal application execution
	reason	error that cannot be determined by the library, such as caused by • FCL code or data sections destruction, wrong program flow • hardware defect
	remedy	Please refer to section 3.7 ”Internal error”.

*available in user mode only

4.4.21 R_FCL_CMD_GET_SERIAL_PROG_DISABLED

The command is used to read the current status of the serial programming interface. A set flag is indicated by a one, a cleared flag by a zero.

Table 45: Configuration of the request structure

Request structure member	Value	Description
command_enu	R_FCL_CMD_GET_SERIAL_PROG_DISABLED	requested command
bufferAdd_u32		address of buffer to store the return value of the “get” command (buffer contains a word)
idx_u32	not used	
cnt_u16	not used	
status_enu	see next table	automatically updated by `R_FCL_Execute` and `R_FCL_Handler`

When the command is finished, the result will be updated in the request structure `status_enu` member.

Table 46: Status returned by the operation

Status	Background and Handling	
R_FCL_OK	meaning	operation finished successfully, the value of serial programming disabled flag is now available in buffer
	reason	no problems during execution
	remedy	nothing to do
R_FCL_ERR_FLOW	meaning	current command is rejected
	reason	library is not initialized or in a wrong state, thus function execution is not possible
	remedy	investigate in the root cause and correct the library handling flow
R_FCL_ERR_PARAMETER	meaning	current command is rejected or stopped
	reason	wrong destination buffer address was specified
	remedy	investigate in the root cause and correct the parameters
R_FCL_ERR_REJECTED*	meaning	current command is rejected
	reason	library is busy performing another operation
	remedy	repeat the command when the preceding command is finished

*available in user mode only

4.4.22 R_FCL_CMD_SET_SERIAL_ID_ENABLED

The command is used to enable the ID authentication mechanism on serial programming interface.

Note: If errors are produced during this command then the result of `R_FCL_CMD_GET_SERIAL_ID_ENABLED` command is not reliable.

Table 47: Configuration of the request structure

Request structure member	Value	Description
command_enu	R_FCL_CMD_SET_SERIAL_ID_ENABLED	requested command
bufferAdd_u32	not used	
idx_u32	not used	
cnt_u16	not used	
status_enu	see next table	automatically updated by `R_FCL_Execute` and `R_FCL_Handler`

When the command is finished, the result will be updated in the request structure `status_enu` member. If the library is configured in user mode, the operation member `status_enu` during the operation is set to `R_FCL_BUSY`.

Table 48: Status returned by the operation

Status	Background and Handling	
R_FCL_OK	meaning	operation finished successfully, authentication on serial interface is now required

Status	Background and Handling	
	reason	no problems during execution
	remedy	nothing to do
R_FCL_BUSY*	meaning	operation started successfully
	reason	no problems during execution
	remedy	continue to call `R_FCL_Handler`
R_FCL_ERR_FLOW	meaning	current command is rejected
	reason	library is not initialized or in a wrong state, thus function execution is not possible
	remedy	investigate in the root cause and correct the library handling flow
R_FCL_ERR_FLMD0	meaning	current command is rejected
	reason	the FLMD0 register / pin is not set correctly
	remedy	investigate in the root cause and correct the register value or the input of the pin
R_FCL_ERR_WRITE	meaning	flag could not be written correctly
	reason	• FCL was initialized with incorrect CPU frequency • hardware defect • FLMD0 not stable during command execution
	remedy	• Re-initialize FCL with correct frequency • configuration Flash area, respectively the complete Flash, should be considered as defect • check FLMD0 setting and repeat command
R_FCL_ERR_ERASE	meaning	affected Flash area could not be erased completely
	reason	• FCL was initialized with incorrect CPU frequency • hardware defect • FLMD0 not stable during command execution
	remedy	• Re-initialize FCL with correct frequency • a Flash block, respectively the complete Code Flash, should be considered as defect • correct FLMD0 handling and repeat the command
R_FCL_ERR_REJECTED	meaning	current command is rejected
	reason	library is busy performing another operation
	remedy	repeat the command when the preceding command is finished
R_FCL_ERR_INTERNAL	meaning	a library internal error occurred, which could not happen in case of normal application execution
	reason	error that cannot be determined by the library, such as caused by • FCL code or data sections destruction, wrong program flow • hardware defect
	remedy	Please refer to section 3.7 ”Internal error”.

*available in user mode only

4.4.23 R_FCL_CMD_GET_SERIAL_ID_ENABLED

The command is used to read the current setting of the status of ID authentication mechanism on serial programming interface. A set flag is indicated by a one, a cleared flag by a zero.

Table 49: Configuration of the request structure

Request structure member	Value	Description
command_enu	R_FCL_CMD_GET_SERIAL_ID_ENABLED	requested command
bufferAdd_u32		address of buffer to store the return value of the “get” command (buffer contains a word)
idx_u32	not used	
cnt_u16	not used	
status_enu	see next table	automatically updated by `R_FCL_Execute` and `R_FCL_Handler`

When the command is finished, the result will be updated in the request structure `status_enu` member.

Table 50: Status returned by the operation

Status	Background and Handling	
R_FCL_OK	meaning	operation finished successfully, the value of authentication ID on serial interface flag is now available in buffer
	reason	no problems during execution
	remedy	nothing to do
R_FCL_ERR_FLOW	meaning	current command is rejected
	reason	library is not initialized or in a wrong state, thus function execution is not possible
	remedy	investigate in the root cause and correct the library handling flow
R_FCL_ERR_PARAMETER	meaning	current command is rejected or stopped
	reason	wrong destination buffer address was specified
	remedy	investigate in the root cause and correct the parameters
R_FCL_ERR_REJECTED*	meaning	current command is rejected
	reason	library is busy performing another operation
	remedy	repeat the command when the preceding command is finished

*available in user mode only

4.4.24 R_FCL_CMD_SET_RESET_VECTOR

The command is used to set the variable reset vector of the device.

Note 1: It is not possible to change reset vector if any of the OTP flags is set.

Note 2: If errors are produced during this command then the result of `R_FCL_CMD_GET_RESET_VECTOR` command is not reliable.

Table 51: Configuration of the request structure

Request structure member	Value	Description
command_enu	R_FCL_CMD_SET_RESET_VECTOR	requested command
bufferAdd_u32		data source buffer address (16 bytes)
idx_u32	not used	
cnt_u16	not used	
status_enu	see next table	automatically updated by `R_FCL_Execute` and `R_FCL_Handler`

When the command is finished, the result will be updated in the request structure `status_enu` member. If the library is configured in user mode, the operation member `status_enu` during the operation is set to `R_FCL_BUSY`.

Table 52: Status returned by the operation

Status	Background and Handling	
R_FCL_OK	meaning	operation finished successfully, reset vector was changed successfully
	reason	no problems during execution
	remedy	nothing to do
R_FCL_BUSY*	meaning	operation started successfully
	reason	no problems during execution
	remedy	continue to call `R_FCL_Handler`
R_FCL_ERR_FLOW	meaning	current command is rejected
	reason	library is not initialized or in a wrong state, thus function execution is not possible
	remedy	investigate in the root cause and correct the library handling flow
R_FCL_ERR_FLMD0	meaning	current command is rejected
	reason	the FLMD0 register / pin is not set correctly
	remedy	investigate in the root cause and correct the register value or the input of the pin
R_FCL_ERR_WRITE	meaning	Reset vector could not be written correctly
	reason	• FCL was initialized with incorrect CPU frequency • hardware defect • FLMD0 not stable during command execution

Status	Background and Handling	
	remedy	• Re-initialize FCL with correct frequency • configuration Flash area, respectively the complete Flash, should be considered as defect • check FLMD0 setting and repeat command
R_FCL_ERR_ERASE	meaning	affected Flash area could not be erased completely
	reason	• FCL was initialized with incorrect CPU frequency • hardware defect • FLMD0 not stable during command execution
	remedy	• Re-initialize FCL with correct frequency • a Flash block, respectively the complete Code Flash, should be considered as defect • correct FLMD0 handling and repeat the command
R_FCL_ERR_REJECTED	meaning	current command is rejected
	reason	library is busy performing another operation
	remedy	repeat the command when the preceding command is finished
R_FCL_ERR_PROTECTION	meaning	current command is rejected
	reason	current security settings (security flags) prevent a modification of the code flash
	remedy	disable security setting and repeat the command
R_FCL_ERR_INTERNAL	meaning	a library internal error occurred, which could not happen in case of normal application execution
	reason	error that cannot be determined by the library, such as caused by • FCL code or data sections destruction, wrong program flow • hardware defect
	remedy	Please refer to section 3.7 ”Internal error”.

*available in user mode only

4.4.25 R_FCL_CMD_GET_RESET_VECTOR

The command is used to read the value of the variable reset vector.

Table 53: Configuration of the request structure

Request structure member	Value	Description
command_enu	R_FCL_CMD_GET_RESET_VECTOR	requested command
bufferAdd_u32		address of buffer to store the return value of the “get” command (buffer contains 16 bytes)
idx_u32	not used	
cnt_u16	not used	
status_enu	see next table	automatically updated by `R_FCL_Execute` and `R_FCL_Handler`

When the command is finished, the result will be updated in the request structure `status_enu` member.

Table 54: Status returned by the operation

Status	Background and Handling	
R_FCL_OK	meaning	operation finished successfully, the reset vector is read to the buffer
	reason	no problems during execution
	remedy	nothing to do
R_FCL_ERR_FLOW	meaning	current command is rejected
	reason	library is not initialized or in a wrong state, thus function execution is not possible
	remedy	investigate in the root cause and correct the library handling flow
R_FCL_ERR_PARAMETER	meaning	current command is rejected or stopped
	reason	wrong destination buffer address was specified
	remedy	investigate in the root cause and correct the parameters
R_FCL_ERR_REJECTED*	meaning	current command is rejected
	reason	library is busy performing another operation
	remedy	repeat the command when the preceding command is finished

*available in user mode only

4.4.26 R_FCL_CMD_GET_BLOCK_CNT

The command is used to return amount of Flash blocks of the device.

Table 55: Configuration of the request structure

Request structure member	Value	Description
command_enu	R_FCL_CMD_GET_BLOCK_CNT	requested command
bufferAdd_u32		address of buffer to store the return value of the “get” command (buffer contains a word value)
idx_u32	not used	
cnt_u16	not used	
status_enu	see next table	automatically updated by `R_FCL_Execute` and `R_FCL_Handler`

When the command is finished, the result will be updated in the request structure `status_enu` member.

Table 56: Status returned by the operation

Status	Background and Handling	
R_FCL_OK	meaning	operation finished successfully, the number of user area blocks available on the device is now available in buffer
	reason	no problems during execution
	remedy	nothing to do
R_FCL_ERR_FLOW	meaning	current command is rejected
	reason	library is not initialized or in a wrong state, thus function execution is not possible
	remedy	investigate in the root cause and correct the library handling flow
R_FCL_ERR_PARAMETER	meaning	current command is rejected or stopped
	reason	wrong destination buffer address was specified
	remedy	investigate in the root cause and correct the parameters
R_FCL_ERR_REJECTED*	meaning	current command is rejected
	reason	library is busy performing another operation
	remedy	repeat the command when the preceding command is finished

*available in user mode only

4.4.27 R_FCL_CMD_GET_BLOCK_END_ADDR

The command is used to return the end address of a specified Flash block.

Table 57: Configuration of the request structure

Request structure member	Value	Description
command_enu	R_FCL_CMD_GET_BLOCK_END_ADDR	requested command
bufferAdd_u32		address of buffer to store the return value of the “get” command (buffer contains a 32-bit value)
idx_u32	0 ... block count of the device – 1	block number in user area
	0x80000000 ... 0x80000000u + block count of the user boot area - 1	block number in user boot area
cnt_u16	not used	
status_enu	see next table	automatically updated by `R_FCL_Execute` and `R_FCL_Handler`

When the command is finished, the result will be updated in the request structure `status_enu` member.

Table 58: Status returned by the operation

Status	Background and Handling	
R_FCL_OK	meaning	operation finished successfully, the end address for requested block is now available in buffer
	reason	no problems during execution
	remedy	nothing to do
R_FCL_ERR_FLOW	meaning	current command is rejected
	reason	library is not initialized or in a wrong state, thus function execution is not possible
	remedy	investigate in the root cause and correct the library handling flow
R_FCL_ERR_PARAMETER	meaning	current command is rejected or stopped
	reason	wrong block value or wrong destination buffer address was specified
	remedy	investigate in the root cause and correct the parameters
R_FCL_ERR_REJECTED*	meaning	current command is rejected
	reason	library is busy performing another operation
	remedy	repeat the command when the preceding command is finished

*available in user mode only

4.4.28 R_FCL_CMD_GET_DEVICE_NAME

The command is used to return the name of the device.

Table 59: Configuration of the request structure

Request structure member	Value	Description
command_enu	R_FCL_CMD_GET_DEVICE_NAME	requested command
bufferAdd_u32		address of buffer to store the return value of the “get” command (buffer contains 16 bytes ASCII encoded)
idx_u32	not used	
cnt_u16	not used	
status_enu	see next table	automatically updated by `R_FCL_Execute` and `R_FCL_Handler`

When the command is finished, the result will be updated in the request structure `status_enu` member.

Table 60: Status returned by the operation

Status	Background and Handling	
R_FCL_OK	meaning	operation finished successfully, the device name is now available in buffer
	reason	no problems during execution
	remedy	nothing to do
R_FCL_ERR_FLOW	meaning	current command is rejected
	reason	library is not initialized or in a wrong state, thus function execution is not possible
	remedy	investigate in the root cause and correct the library handling flow
R_FCL_ERR_PARAMETER	meaning	current command is rejected or stopped
	reason	wrong destination buffer address was specified
	remedy	investigate in the root cause and correct the parameters
R_FCL_ERR_REJECTED*	meaning	current command is rejected
	reason	library is busy performing another operation
	remedy	repeat the command when the preceding command is finished

*available in user mode only

Chapter 5 Library Setup and Usage

This chapter contains important information about how to put the FCL into operation and how to integrate it into your application. Please read this chapter carefully—and also especially Chapter 6 “Cautions” —in order to avoid problems and malfunction of the library. Before integrating the library into your project, however, please make sure that you have read and understood how the FCL works and which basic concepts are used (see Chapter 2 “Architecture” and Chapter 3 “FCL Functional Specifications”).

5.1 Obtaining the library

The FCL is provided by means of an installer via the Renesas homepage at

http://www.renesas.eu/update

Please follow the instructions of the installer carefully. Please ensure to always work on the latest version of the library.

5.2 File structure

The library is delivered as a complete compilable sample project which contains the FCL and in addition an application sample to show the library implementation and usage in the target application.

The delivery package contains dedicated directories for the library containing the source and the header files.

5.2.1 Overview

The following picture contains the library and the application related files:

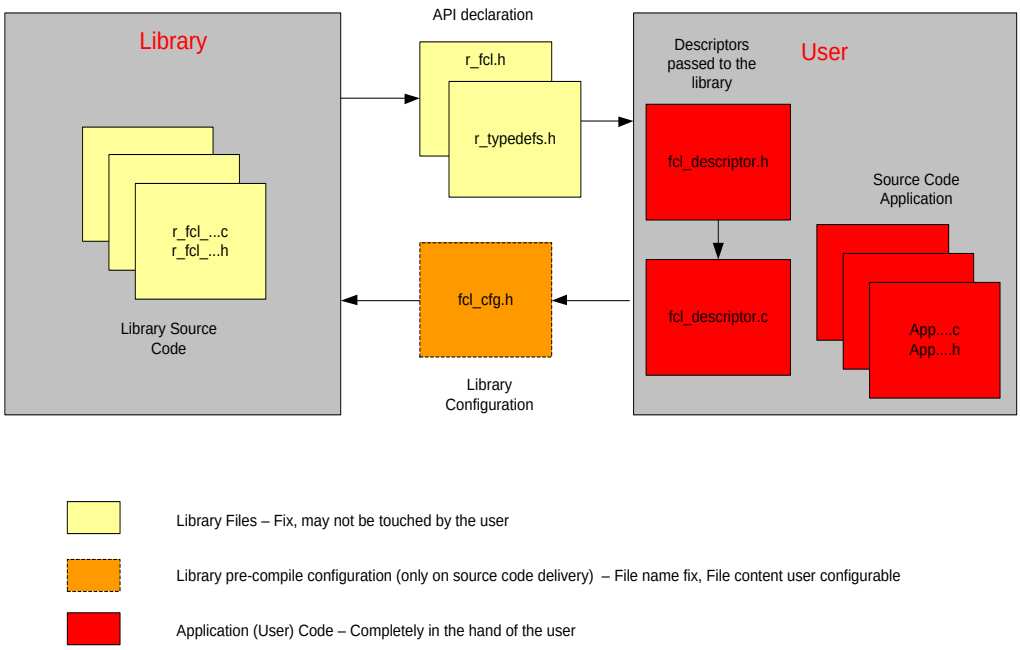

Figure 7: Library and application file structure

The library code consists of different source files, starting with `r_fcl_...` The files shall not be touched by the user.

In case of source code delivery, the library must be configured for compilation. The file `fcl_cfg.h` contains defines for that. As it is included by the library source files, the file contents may be modified by the user, but the file name shall not.

5.2.2 Filesystem structure of delivery package

The following table contains all files installed by the library installer:

- Files in red belong to the build environment, controlling the compile, link and target build process
- Files in blue belong to the sample application
- Files in green are description files only
- Files in black belong to the FCL

Table 61: File structure of the FCL

File		Description
<installation_folder>/FCL		
Release.txt		Library package release notes
support.txt		List of supported devices
<installation_folder>/FCL /<compiler>/<device_name>		
Build.bat		Batch file to build the FCL sample application
Clean.bat		Batch file to clean the FCL sample application
Makefile		Make file that controls the build and clean process
<installation_folder>/FCL /<compiler>/<device_name>/Sample(1)		
dr7f70xxxx_startup.850(6)	<for GHS compiler>	Device and compiler specific start-up code
cstart.asm	<for REC compiler>	
cstartup.s	<for IAR compiler>	
dr7f70xxxx.ld(6)	<for GHS compiler>	Compiler specific linker directives
dr7f70xxxx_pic.dir(6)	<for REC compiler>	
layout.icf lnkr7f70xxxxxafp.icf(6)	<for IAR compiler>	
dr7f70xxxx.dvf.h(6) dr7f70xxxx_irq.h(6)	<for GHS compiler>	Definitions of IO registers, interrupt and exceptions vector table, for RH850 devices. <for GHS compiler>: Use dr7f70xxxx.dvf.h(6) or. dr7f70xxxx_0.h(6), and io_macros_v2.h. <for REC compiler>: Use "boot.asm" or "vecttbl.asm".
iodefine.h boot.asm	<for REC compiler>	
ior7f70xxxxxafp.h(2) (6)	<for IAR compiler>	
main.c fcl_ctrl.c(3)		Sample application code
target.h		Initialization code for target microcontroller
fcl_prefetch.850(4)	<for GHS compiler>	Initializing of prefetch area
fcl_prefetch.s(5)	<for IAR compiler>	

File		Description
fcl_cfg.h		User defined configuration for FCL
fcl_descriptor.c		FCL descriptor used in the sample application
fcl_descriptor.h		FCL descriptor used in the sample application
fcl_user.c		library pre-initialization
fcl_user.h		
<installation_folder>/FCL /<compiler>/FCL		
r_fcl.h		FCL API definitions.
r_fcl_types.h		User interface type definitions and all error and status codes used during self-programming
<installation_folder>/FCL /<compiler>/FCL/lib		
r_typedefs.h		C types used by FCL library
r_fcl_env.h		Internal FCL definitions
r_fcl_global.h		Global variables and settings used during self-programming
r_fcl_hw_access.c r_fcl_user_if.c		FCL main source code
r_fcl_hw_access_asm.850	<for GHS compiler>	Internal library compiler specific source code
r_fcl_hw_access_asm.asm	<for REC compiler>	
r_fcl_hw_access_asm.s	<for IAR compiler>	

(1) File names are dependent on the chosen device. Shown filenames are valid for F1L devices.

(2) This file is not included in the library installer.
Please obtain this file from IAR development environment.
e.g.) C:\Program Files\IAR Systems\Embedded Workbench 7.3\rh850\inc

(3) An application control file (fcl_ctrl.c) is a file used for the sample for FCL V2.13 and later. Because this processing is executed in main.c, this file is unnecessary at FCL V2.12 or earlier.

(4) fcl_prefetch.850 is only provided for GHS compilers (linkers) that are usable with V2.12 or an earlier version of the FCL. If you are using V2.13 or a later version of the FCL, the contents of fcl_prefetch.850 are included in r_fcl_hw_access_asm.850.

(5) fcl_prefetch.s is only provided for IAR compilers (linkers) that are usable with V2.12 or an earlier version of the FCL. If you are using V2.13 or a later version of the FCL, the contents of fcl_prefetch.s are included in r_fcl_hw_access_asm.s.

(6) xxxx is a number (e.g. dr7f701007).

(7) The make.exe file which is run from the batch files that come with the RH850 code flash library (FCL) Type01 is an external tool, and requires downloading from a Web site that distributes make.exe. As stated in Release.txt, GNU Make was used to confirm operation of the sample application. If you wish to use a GNU Make environment, download make.exe from the Web site of GNU Make and install it. Execute the batch files after that.

5.3 Linker sections

The following sections are related to the FCL and need to be defined in the linker file (please see linker directive file in the sample application for an example):

- FCL data sections:

    - R_FCL_DATA - contains the variables required by FCL. It can be located either in internal or in external RAM.
- FCL code sections
    - R_FCL_CONST – contains library internal constant data
    - R_FCL_CODE_ROM – contains the code executed at the beginning of self-programming. This code is executed from the linked location. Mainly this is the initialization code.
    - R_FCL_CODE_USRINT – contains user interrupt routines that may be executed in parallel with FCL operation when code flash is unavailable.
    - R_FCL_CODE_USR – contains user code that has to be executed in parallel with FCL operation when code flash is unavailable.
    - R_FCL_CODE_RAM – contains parts of FCL code that handle flash operations and thus need to be located outside the flash area.
    - R_FCL_CODE_ROMRAM - contains the user interface. Depending on the library configuration (status check mode), code from this section will be executed in RAM (in case of status check user mode) or in Flash (in case of status check internal mode).
    - R_FCL_CODE_RAM_EX_PROT – this small section is copied to RAM where it has the purpose of avoiding ECC prefetch exceptions when CPU is executing code near the end of the previous section.
- Sample application section:
    - R_FCL_RESERVE – is an example on how to reserve RAM space for FCL sections to be copied. The following sections will be copied into this section in the sample application: R_FCL_CODE_USRINT, R_FCL_CODE_USR, R_FCL_CODE_RAM, R_FCL_CODE_ROMRAM and R_FCL_CODE_RAM_EX_PROT.
      The use of this section is not mandatory if space in RAM is reserved otherwise.

Note 1: It is not allowed to change the order of the FCL sections, or to place other sections between FCL sections, otherwise the FCL library will not work correctly. Empty spaces between sections due to alignment are allowed.

Note 2: FCL sections must be defined even if they are empty.

Note 3: Sections R_FCL_CODE_USRINT and R_FCL_CODE_USR are solely for the user code. The user shall not place code or data in any other FCL sections.

5.4 Sample application

It is very important to have theoretic background about the Code Flash and the FCL in order to successfully implement the library into the user application. Therefore it is important to read this user manual in advance. The best way, after initial reading of the user manual, will be testing the FCL application sample.

After a first compile run, it will be worth playing around with the library in the debugger. By that you will get a feeling for the source code files and the working mechanism of the library.

Note: Before compiling the sample application, the compiler path must be configured: in the sample “Makefile” set the variable COMPILER_INSTALL_DIR to point to the correct compiler directory.

Later on, the sample might be reconfigured to use the internal mode to get a feeling of the CPU load and execution time during different modes.

After this exercise it might be easier to understand and follow the recommendations and considerations of this document.

5.5 Self-programming sequence

The following flow charts represent typical FCL sequence during device operation including the API functions to be used. Error treatment is not detailed in the flow charts for simplification reasons.

5.5.1 Typical flow chart for reprogramming in user mode

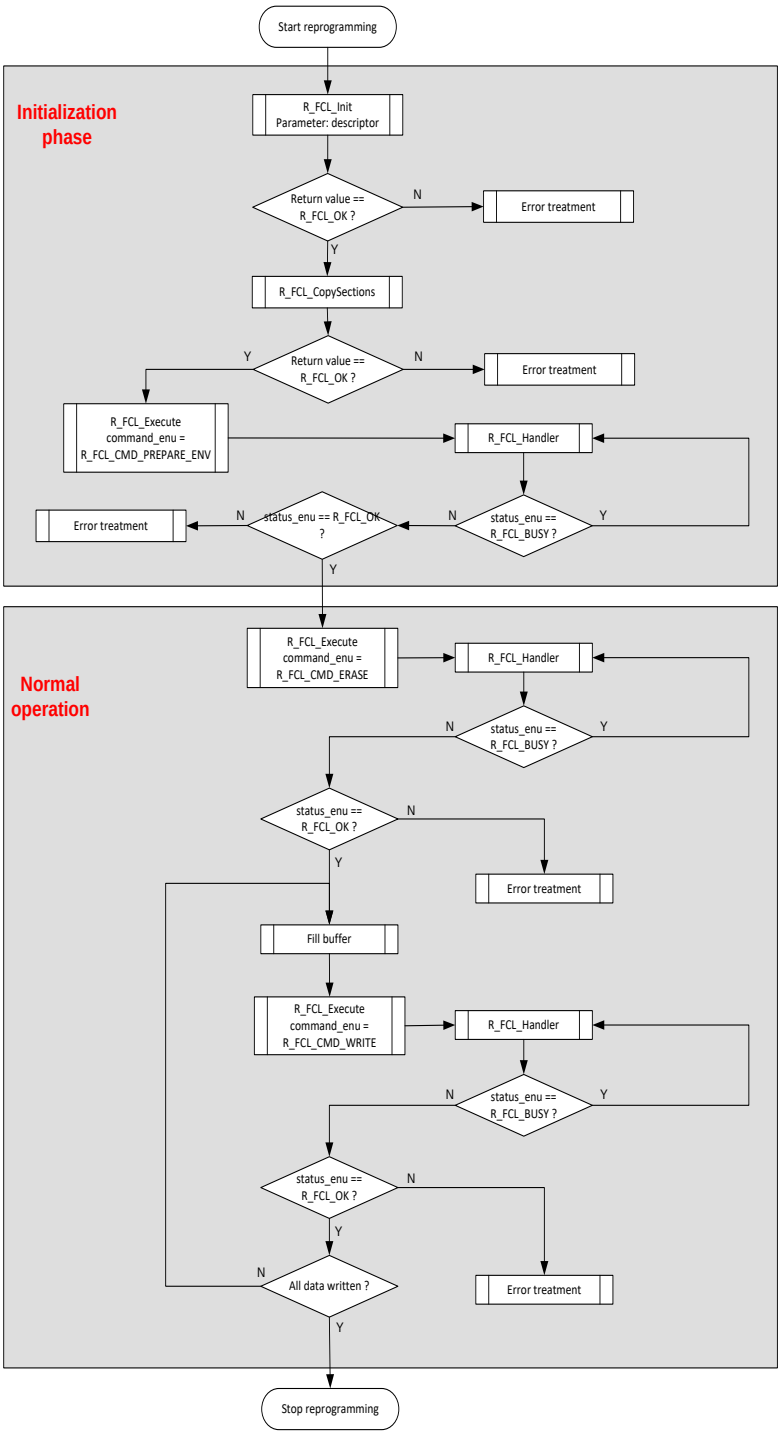

Figure 8: Typical reprogramming flow in user mode

5.5.2 Typical flow chart for reprogramming in internal mode

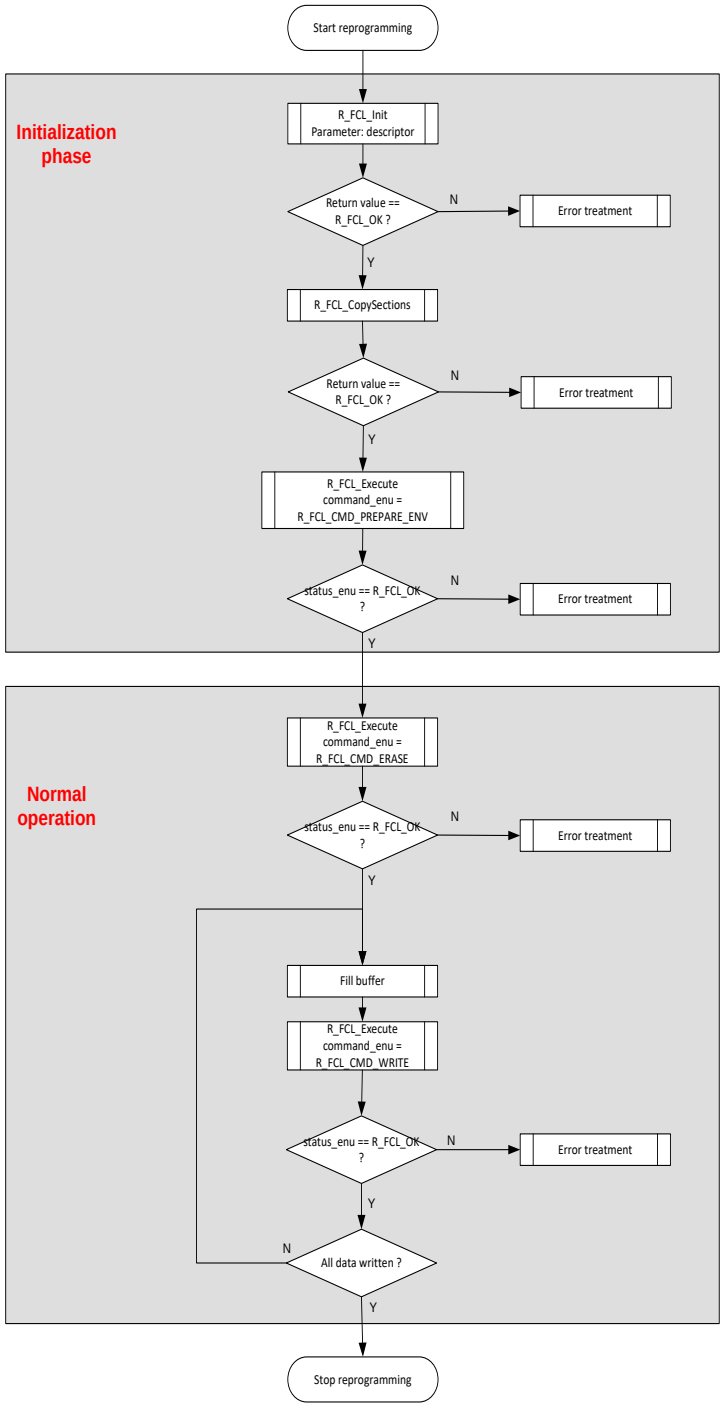

Figure 9: Typical reprogramming flow in internal mode

5.6 MISRA Compliance

The FCL code has been tested regarding MISRA™ compliance.

The used tool is the QA C™ Source Code Analyzer which tests against the MISRA C™ 2004 standard rules.

Note:

"MISRA" and “MISRA C” is a registered trademark of HORIBA MIRA Ltd, held on behalf of the MISRA Consortium. “QA C” is a registered trademark of Programming Research Ltd.

All MISRA related rules have been enabled. Remaining findings are commented in the code while the QAC checker machine is set to silent mode in the concerning code lines.

Chapter 6 Cautions

Before starting the development of an application using the FCL, please carefully read and understand the following cautions:

1. CPU operating frequency configuration:

   Do not change the CPU frequency during operation. If the frequency has to be changed, reinitialize the FCL with proper CPU frequency.

   If you are using an RH850/F1KM-S4 or RH850/F1KH-D8, the operating frequency of the flash sequencer controlled by the FCL is 1/8 of the CPU operating frequency (CPU clock at up to 240 MHz) by default.

   fPCLK = 1/8 fCPUCLK_H (with CKDIVMD= 1 and CPU operating frequency of up to 240 MHz)

   When the option byte setting of the RH850/F1KM-S4 or RH850/F1KH-D8 is CKDIVMD=0 (up to 120 MHz), however, the operating frequency of the flash sequencer controlled by the FCL must be changed to 1/4 of the CPU operating frequency.

   fPCLK = 1/4 fCPUCLK_H (with CKDIVMD=0 so max. CPU operating frequency is 120 MHz)

   If you are using V2.13 or a later version of the FCL, you can change the operating frequency of the flash sequencer to support CPU operating frequencies up to 120 MHz.

   To change the operating frequency of the flash sequencer, enable the line of source code that halves the denominator of the frequency ratio within the R_FCL_FCUFct_SetFrequency function in the “r_fcl_hw_access.c” file that comes with the FCL.

   Specifically, change “#if 0” to “#if 1” on the line following the comments after the “Changing CKDIVMD” keyword.

   <Keyword>

   ```
   /*****************************************************************
   * SAMPLE: Changing CKDIVMD
   *****************************************************************/
   ```

   ```
   #if 0
   ffcu = ( (ffcu + (g_fcl_data_str.deviceInfo_str.fDivider_u8 / 2u)) - 1u) /
       (g_fcl_data_str.deviceInfo_str.fDivider_u8 / 2u);
   #else
   ```

   

   ```
   #if 1
   ffcu = ( (ffcu + (g_fcl_data_str.deviceInfo_str.fDivider_u8 / 2u)) - 1u) /
       (g_fcl_data_str.deviceInfo_str.fDivider_u8 / 2u);
   #else
   ```

   Change “#if 0” to “#if 1” in cases where you wish to use the RH850/F1KM-S4 or RH850/F1KH-D8 with the CKDIVMD=0 (up to 120 MHz) setting.

   *V2.12 and earlier versions of the FCL do not support the CKDIVMD=0 (up to 120 MHz) setting.

2. Function re-entrancy:

   All functions are not re-entrant. So, re-entrant calls of any FCL function must be avoided.

3. Task switch, context change, synchronization between functions:

   Each function depends on global available information and is able to modify this information. In order to avoid synchronization problems, it is necessary that at any time only one FCL or FDL function is executed. So, it is not allowed to start an FCL or FDL function, then switch to another task context and execute another FCL or FDL function while the last one has not finished.

4. Code Flash access during self-programming:

   Code Flash accesses during an active self-programming environment is not possible at all. This is the case during execution of most of the self-programming commands. If access to Code Flash is performed that time, the result will almost always be code execution malfunction.

   Thus, please make sure that all code that may operate in parallel with FCL is available in RAM. That means, code is placed in relevant linker sections (refer to chapter 5.3 “Linker sections") and it is copied into the right place in RAM (refer to chapter 3.1 “Code execution in RAM”). Also the use of standard libraries shall be avoided, e.g. by appropriate compiler configuration.

   If a command ends with error `R_FCL_ERR_INTERNAL` further commands are rejected. It is not guaranteed that Code Flash becomes available after this error is produced. If Code Flash remains unavailable then FCL re-initialization is not possible and the device will have to be reset or replaced.

5. Interrupt vector assignment during self-programming:

   Even during self-programming command execution, it is possible to use the interrupt processing. However, during the command execution, the user program and interrupt vector in Flash cannot be used. Re-direction of the interrupt vector table to RAM is supported by the device. Furthermore, all interrupt code needs to be available in RAM. Please refer to the device user's manual for more information on how to change the interrupt vector table address.

6. Interrupted flash operations:

   In case of Flash modification operation (Erase / Write) interruption, the data in the affected Flash range (Flash block on Erase, Flash write unit on Write) gets undefined. Even if the read value should be as expected, the data retention of the Flash area may not be given.
   In case of operation interruption, erase and re-write the affected Flash area in order to ensure data integrity and retention.

7. Write operation:

   Before executing a write operation, please make sure the given address range is erased.

8. Watchdog timer:

   The watchdog timer does not stop during the execution of the FCL.

9. Preconditions for FCL operations:

   Before starting any FCL operation (any command except `R_FCL_CMD_PREPARE_ENV`), the user has to execute the following initialization sequence:

   - initialize library (call `R_FCL_Init` with correct FCL descriptor as parameter)
   - copy relevant sections to RAM (e.g. call `R_FCL_CopySections`)
   - prepare the library and Flash programming environment (call `R_FCL_Execute` with `R_FCL_CMD_PREPARE_ENV`)
   - set FLMD0 high

10. Dual operation:

    It is not possible to modify the Code Flash in parallel to a modification of the Data Flash or vice versa due to shared hardware resources.

11. Reusing the request command:

    It is not possible to change the content of the request structure during command operation. If request data is changed during command operation, the library will not work correctly.

12. Data alignment:

    The Flash operation destination address need to be aligned to the operation granularity (e.g. 256-byte granularity for the Write operation).

    User application buffers passed to the FCL by their address need to be 32-bit aligned.

13. Cancel suspended operation:

    If a cancel request is accepted, during an on-going write or erase operation and a previous operation is already suspended, then both operations will be cancelled.

14. Set new ID:

    A reset should be performed after executing R_FCL_CMD_SET_ID successfully, in order for the new IDs to be taken into account.

15. Nested operations:

    The following sequences of nested operations are not possible:

    - Any operation ► suspend ► suspend
    - Erase operation ► suspend ► other erase operation
    - Write operation ► suspend ► erase
    - Write operation ► suspend ► write

    It is recommended to avoid nesting as much as possible.

16. Areas to be accessed when a pre-compilation definition is specified:

    Access by the FCL is to specific areas in accord with the pre-compilation definition setting during initialization processing.

    When you execute the `R_FCL_CMD_PREPARE_ENV` command of `R_FCL_Execute`, permit reading from the following areas.

Pre-compilation definition supported by FCL V2.12 and later versions	Areas to be accessed
R_FCL_NO_BFA_SWITCH	0x01030000 to 0x0103029F
R_FCL_MIRROR_FCU_COPY	0x01030000 to 0x0103029F, 0x01037000 to 0x01037FFF
R_FCL_NO_FCU_COPY	0x00010000 to 0x0001029F
None	0x00010000 to 0x0001029F, 0x00017000 to 0x00017FFF

17. Self-Programming ID Authentication Mode for RH850/F1KM, F1KH:

    FCL can be used only when the option byte(OPBT1) setting is SIDAM=1 (Self-Programming ID Authentication is necessary).

    FCL can’t be used when OPBT1 setting is SIDAM=0(Self-Programming ID Authentication is NOT necessary).

Revision History

Chapter	Page	Description
All	*	Rev. 1.00: Initial document version
1.1.2 3.4 3.5 4.2.3 4.3.1.1 4.3.2.5 4.3.2.7 4.4 6	11 16 18 24 28 36 39 40 81	Rev. 1.10: Added information about 2 code flash bank devices Suspend/Resume only for Erase and Write commands Added cancel mechanism Added cancelled status Removed protection error Added R_FCL_ERR_REJECTED Added new interface function: R_FCL_CancelRequest Added and removed error codes for different commands Added new cautions
All All - - - - - 1 2.1 3.4 3.6,3.7 4.1 4.2.3 4.2.5 4.3.2.1 4.4.1 - 4.4.24	- - 1 2 - - 3,4 7 10 13 15 18 22 24 28 39 - 71	Rev. 2.11: Minor description corrections and wording update Document formatting update Updated title and installer name Updated Notice text Deleted Regional information page Deleted Preface page Updated How to use this document text Updated location specific text Updated 2.1 Library representation Updated 3.4 Suspend / Resume mechanism Added new subchapters 3.6 and 3.7 Updated Table 2 with option R_FCL_NO_BFA_SWITCH Updated Table for R_FCL_ERR_INTERNAL Updated CPU frequency parameter description Updated Pre-conditions of 4.3.2.1 R_FCL_GetVersionString Updated Table 6, 8, 10, 12, 14, 20, 24, 28, 32, 36, 40, 44, 48, 52 for R_FCL_BUSY, R_FCL_ERR_ERASE, R_FCL_ERR_WRITE and R_FCL_ERR_INTERNAL Pointed remedy for R_FCL_ERR_INTERNAL to 3.7 Added note about corresponding GET commands to SET commands: R_FCL_CMD_SET_LOCKBIT R_FCL_CMD_SET_OTP R_FCL_CMD_SET_OPB R_FCL_CMD_SET_ID R_FCL_CMD_SET_READ_PROTECT_FLAG R_FCL_CMD_SET_WRITE_PROTECT_FLAG R_FCL_CMD_SET_ERASE_PROTECT_FLAG R_FCL_CMD_SET_SERIAL_PROG_DISABLED R_FCL_CMD_SET_SERIAL_ID_ENABLED R_FCL_CMD_SET_RESET_VECTOR

Chapter	Page	Description
5.2.1	76	Updated the figure
5.2.2	77,78	Updated Table 61: File structure of the FCL
5.3	79	Updated Sample application section of 5.3 Linker sections
6	82,83	Updated Cautions
		Rev. 2.12:
4.1	17	Updated Table 2: Pre-Compile options.
	19	Added note on FCL V2.12 Pre-Compile configuration
4.3	20	Updated 4.3 Data Types
4.2.2	21	Updated Table 3: List of available commands.
4.4.1	40	Added R_FCL_ERR_FLMD0 to Table 6: Status returned by the operation
4.4.8 - 4.4.24	51 - 72	Added R_FCL_ERR_ERASE to Table 20, 24, 28, 32, 36, 40, 44, 48, 52
5.2.2	78,79	Updated Table 61: File structure of the FCL package
6	83,84	Updated No.1 of Chapter 6 Cautions and added No.16.
		Rev. 2.13:
4.1	18	Updated Table 2: Pre-compilation options.
	19	Updated note: The pre-compilation definitions
		Updated note for 4.1 Library compile-time configuration
4.2	20	Updated 4.2 Data types
4.2.1	21	Updated 4.2.1 Simple type definitions
5.2.2	79,80	Updated 5.2.2 Filesystem structure of delivery package
5.6	84	Updated 5.6 MISRA Compliance
6	85	Updated No.1 of Chapter 6 Cautions
	87	Added No.17 of Chapter 6 Cautions .

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

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

Renesas Electronics America Inc.
1001 Murphy Ranch Road, Milpitas, CA 95035, U.S.A.
Tel: +1-408-432-8888, Fax: +1-408-434-5351

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

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

Renesas Electronics (China) Co., Ltd.
Room 101-T01, Floor 1, Building 7, Yard No. 7, 8th Street, Shangdi, Haidian District, Beijing 100085, China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

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

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

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit No 3A-1 Level 3A Tower 8 UOA Business Park, No 1 Jalan Pengaturcara U1/51A, Seksyen U1, 40150 Shah Alam, Selangor, Malaysia
Tel: +60-3-5022-1288, Fax: +60-3-5022-1290

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HAL 2nd Stage, Indiranagar, Bangalore 560 038, India
Tel: +91-80-67208700

Renesas Electronics Korea Co., Ltd.
17F, KAMCO Yangjae Tower, 262, Gangnam-daero, Gangnam-gu, Seoul, 06265 Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5338

Colophon 6.0

RH850 Family User’s Manual: Code Flash Library Type T01

Publication Date: Rev.1.00 Nov 11, 2013
Rev.2.13 Jun 10, 2019

Published by: Renesas Electronics Corporation

Code Flash Library Type T01

Renesas Electronics Corporation

R01US0078ED0213