User Manual

32

Self-Programming Library for Code Flash

Type T01, European Release

RENESAS 32-Bit MCU
RH Family / RH850 Series

Installer:
RENESAS_FCL_RH850_T01V2.xx

Renesas Electronics
www.renesas.com

R01US0078ED0110
02.09.2014

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of 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. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. 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 of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may 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.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics.

8. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti- crime systems; safety equipment; and medical equipment not specifically designed for life support.

   “Specific”: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.

9. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
10. Although Renesas Electronics endeavours to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, 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, please evaluate the safety of the final products or system manufactured by you.
11. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
12. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
13. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

    Note 1 “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its majority- owned subsidiaries.

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

Regional information

Some information contained in this document may vary from country to country. Before using any Renesas Electronics product in your application, please contact the Renesas Electronics office in your country to obtain a list of authorized representatives and distributors. They will verify:

- Device availability
- Ordering information
- Product release schedule
- Availability of related technical literature
- Development environment specifications (for example, specifications for third-party tools and components, host computers, power plugs, AC supply voltages, and so forth)
- Network requirements

In addition, trademarks, registered trademarks, export restrictions, and other legal issues may also vary from country to country.

Visit

http://www.renesas.com

to get in contact with your regional representatives and distributors.

Preface

Readers This manual is intended for users who want to understand the functions of the concerned libraries.

Purpose This manual presents the software manual for the concerned libraries.

Note Additional remark or tip

Caution Item deserving extra attention

Numeric notation

Binary:	xxxx or xxxB
Decimal:	xxxx
Hexadecimal	xxxxH or 0x xxxx

Numeric prefix Representing powers of 2 (address space, memory capacity):

K (kilo)	210 = 1024
M (mega):	220 = 1024^2 = 1,048,576
G (giga):	230 = 1024^3 = 1,073,741,824

Register X, x = don’t care

Diagrams Block diagrams do not necessarily show the exact software flow but the functional structure. Timing diagrams are for functional explanation purposes only, without any relevance to the real hardware implementation.

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
Firmware	Firmware is a piece of software that is located in a hidden area of the device, handling the interfacing to the flash.
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.
Flash Macro	A certain number of Flash blocks are grouped together in a Flash macro.
FCL	Code Flash Library (Code Flash access layer)
FDL	Data Flash Library (Data Flash access layer)
FW	Firmware
IWDT	Internal watchdog timer
OPB	Option Bytes used to define device behaviour at startup
OTP	One Time Programmable
NVM	Non volatile memory. All memories that hold the value, even when the power is cut off. E.g. Flash memory, EEPROM, MRAM...
NSIS	Package installer system from Nullsoft

Abbreviation	Full form
RAM	“Random access memory” - volatile memory with random access
REE	Renesas Electronics Europe GmbH
REL, ROS	Renesas Electronics Japan
ROM	“Read only memory” - nonvolatile memory. The content of that memory cannot be changed during normal operation.
RV40F	Name of the Flash technology used in RH850 devices.
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.
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 based on the RV40F flash technology.

Note:
Do not use this library for devices based on other Flash technologies than RV40F, as this might lead to all sorts of defective behaviour including physical destruction.

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

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

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

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

This manual is based on the assumption that the device will operate in supervisor mode. For information on other modes, refer to the user's manual for the target device.

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

The flash technology which is utilized in RH850 devices is called RV40F. 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 RV40F 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.

Reading an erased RV40F code flash byte will return the value 0xFF.

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. It consists of a number of code flash blocks numbered starting from 0x0. The write address for this blocks starts at 0x0 and increments in steps of 256.
- The user boot area also named extended user memory, is designed for special application parts, e.g. a bootloader program. The blocks in this area are numbered starting with 0x80000000. The write address for this area starts at 0x01000000 and increments in steps of 256. 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 (Sequencer), controlled by the FCL.

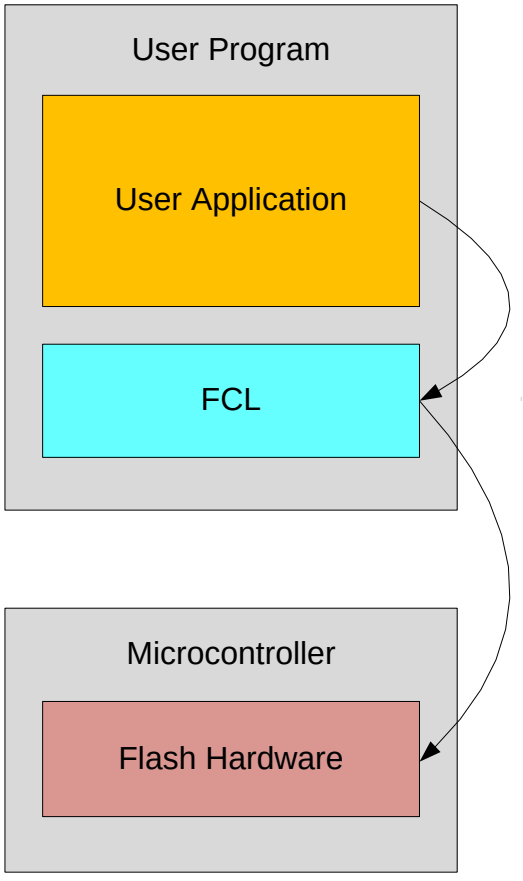

Figure 1: Symbolic representation between functional blocks

2.1 Library presentation

FCL library can be released in two forms:

- Source code, comes inside an NSIS installer package together with a sample application, build scripts and GNU Make tools, ready for compilation and flashing.
- Precompiled objects for specific compilers

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-compiled 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 5.4 "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 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.

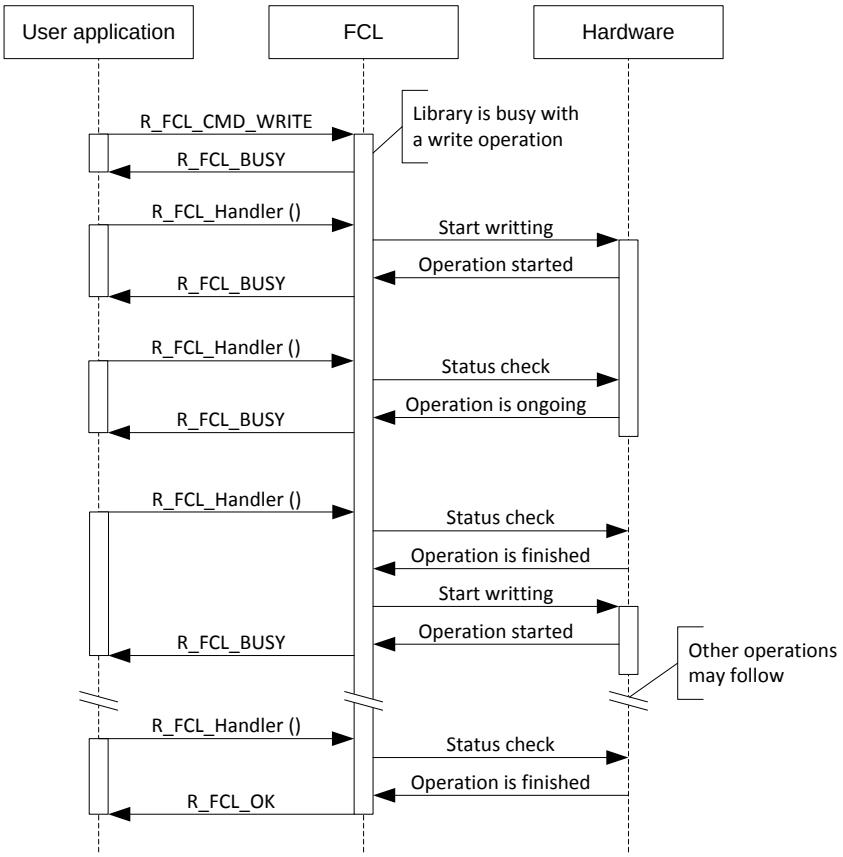

Figure 2: Asynchronous execution in User mode

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 Pre-compile 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:

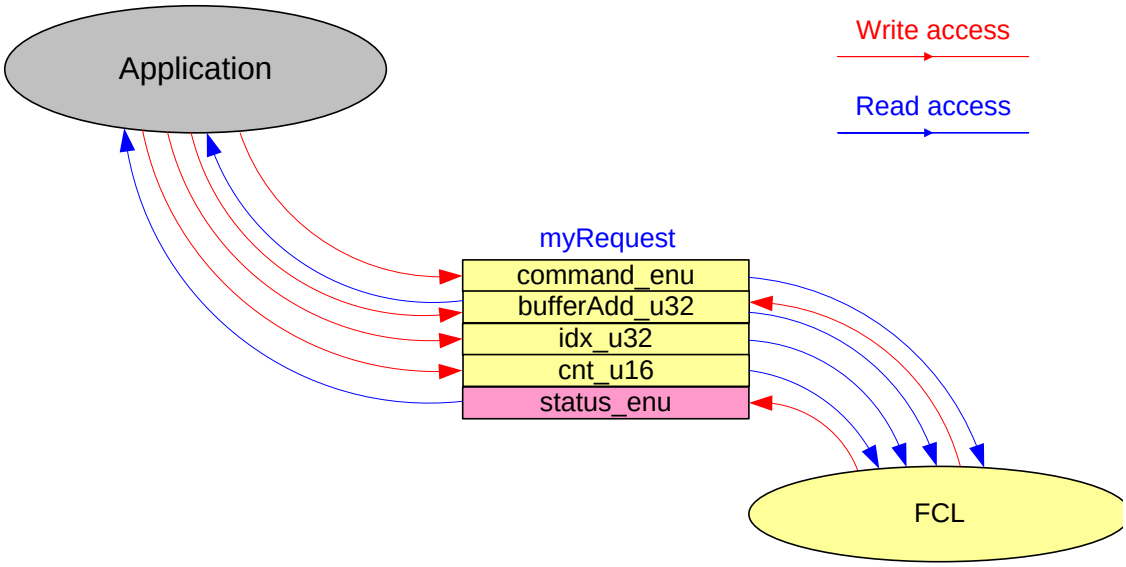

Figure 3: Usage of the request structure

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 suspend option can be used only for the `R_FCL_CMD_WRITE` and `R_FCL_CMD_ERASE` operations because they can last a long time. After a command is suspended, another one can be started or the user application can perform other operations unrelated to FCL.

Suspend-resume facility is available only for user mode.

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

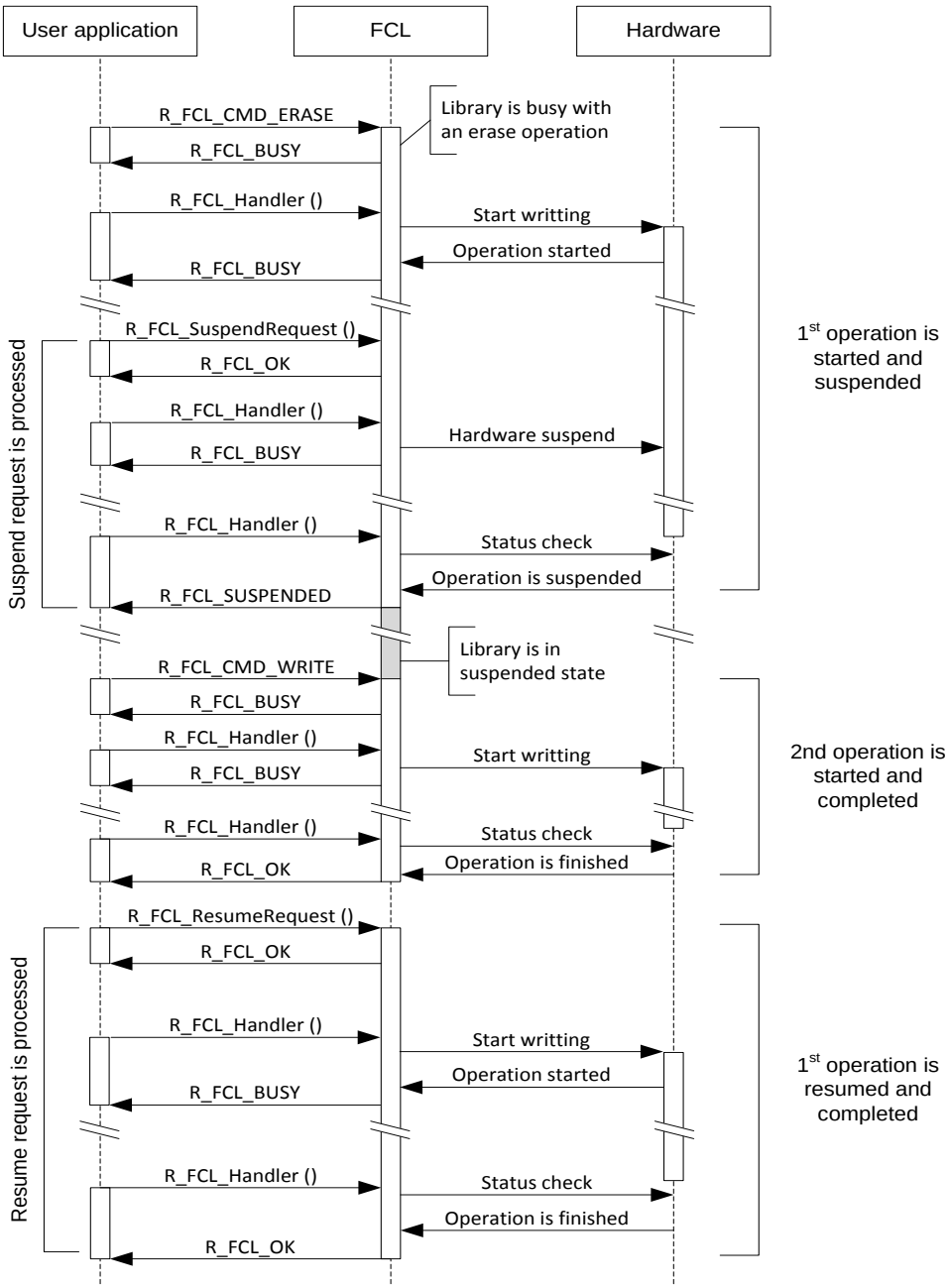

Figure 4: Example of suspend resume flow

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 bellow shows an example of how to cancel an erase command:

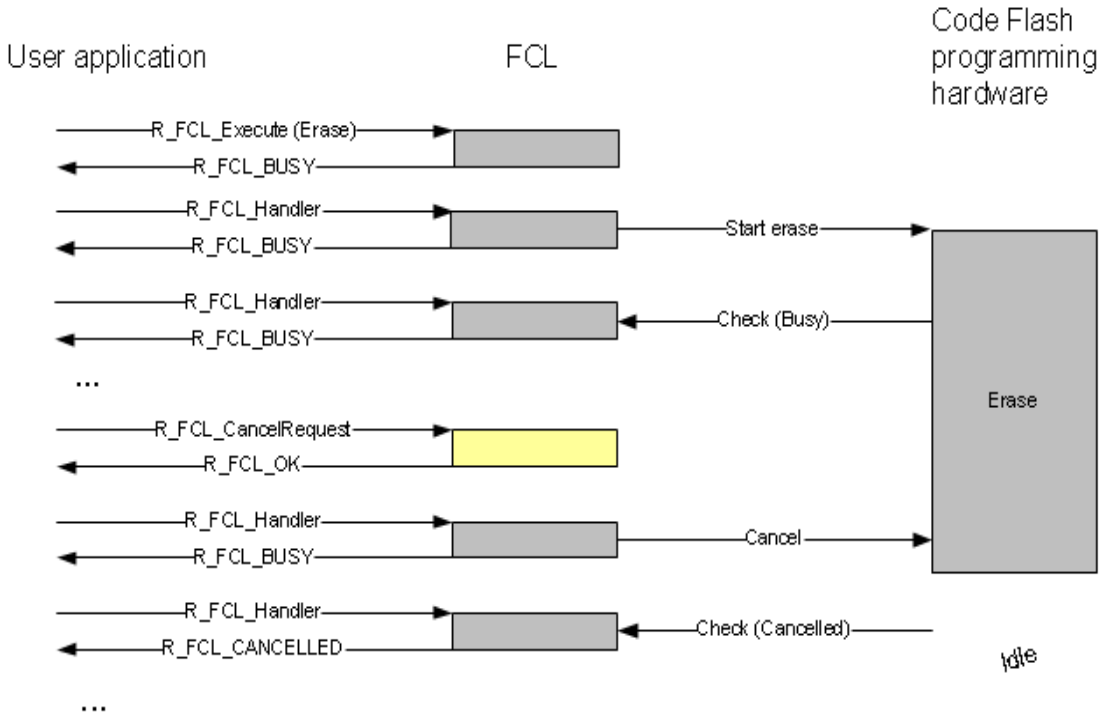

Figure 5: Cancel an Erase operation

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 operation will be cancelled if it is requested.

Chapter 4 User Interface (API)

4.1 Library compile-time configuration

The pre-compile 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: Precompile options

Option	Description

R_FCL_STATUS_CHECK	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	Enable or disable 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	Enable or disable following commands:
	`R_FCL_CMD_GET_OTP`
	`R_FCL_CMD_SET_OTP`

R_FCL_SUPPORT_DEVICENAME	Enable or disable following commands: `R_FCL_CMD_GET_DEVICE_NAME`
R_FCL_SUPPORT_BLOCKCNT	Enable or disable following commands: `R_FCL_CMD_GET_BLOCK_CNT`
R_FCL_SUPPORT_BLOCKENDADDR	Enable or disable following commands: `R_FCL_CMD_GET_BLOCK_END_ADDR`

R_FCL_SUPPORT_OPB	Enable or disable following commands:
	`R_FCL_CMD_GET_OPB`
	`R_FCL_CMD_SET_OPB`

R_FCL_SUPPORT_ID	Enable or disable following commands:
	`R_FCL_CMD_GET_ID`
	`R_FCL_CMD_SET_ID`

R_FCL_SUPPORT_RESETVECTOR	Enable or disable following commands:
	`R_FCL_CMD_GET_RESET_VECTOR`
	`R_FCL_CMD_SET_RESET_VECTOR`

Option	Description
R_FCL_SUPPORT_SECURITYFLAGS	Enable or disable 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

Precompile option R_FCL_STATUS_CHECK 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 pooling 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 crash.

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

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 with standard C99 stdint.h header but please review carefully that there are no size or endianess mismatches if you use other definitions in your project.

4.2.1 Simple type definitions

Type definition:

```
typedef unsigned char       uint8_t;
typedef unsigned short      uint16_t;
typedef unsigned long       uint32_t;
```

Description: These simple types are used throughout the complete 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" bellow.

Table 3: List of available commands

Member / Value	Description
R_FCL_CMD_PREPARE_ENV	Copy library internal functions to RAM and initialize internal states and variables
R_FCL_CMD_ERASE	Erase one or more Flash blocks
R_FCL_CMD_WRITE	Write multiple of 256 bytes into the Flash
R_FCL_CMD_SET_LOCKBIT	Lock a Flash block to prevent it from further modification
R_FCL_CMD_GET_LOCKBIT	Read out the current lock bit setting
R_FCL_CMD_ENABLE_LOCKBITS	Enable the mechanism to protect a block using lockbits
R_FCL_CMD_DISABLE_LOCKBITS	Disable the mechanism to protect a block using lockbits
R_FCL_CMD_SET_OTP	Set one-time-programmable flag for one block
R_FCL_CMD_GET_OTP	Read one-time-programmable flags
R_FCL_CMD_SET_OPB	Write a new Option Byte setting
R_FCL_CMD_GET_OPB	Read out the current Option Byte setting
R_FCL_CMD_SET_ID	Write a new OCID, used to protect the debug and serial interfaces and enable Self-Programming
R_FCL_CMD_GET_ID	Read out the current OCID setting
R_FCL_CMD_SET_READ_PROTECT_FLAG	Enables read protection
R_FCL_CMD_GET_READ_PROTECT_FLAG	Read out the current read protection setting
R_FCL_CMD_SET_WRITE_PROTECT_FLAG	Enables write protection
R_FCL_CMD_GET_WRITE_PROTECT_FLAG	Read out the current write protection setting
R_FCL_CMD_SET_ERASE_PROTECT_FLAG	Enables erase protection
R_FCL_CMD_GET_ERASE_PROTECT_FLAG	Read 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	Write a new Reset Vector into the device
R_FCL_CMD_GET_RESET_VECTOR	Read out the current Reset Vector setting
R_FCL_CMD_GET_BLOCK_CNT	Read out the number of Flash blocks
R_FCL_CMD_GET_BLOCK_END_ADDR	Read out the end address of a block
R_FCL_CMD_GET_DEVICE_NAME	Read 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 or due to a hardware error (e.g. tired memory cell)
R_FCL_ERR_ERASE	The requested erase command encountered an error because of a hardware erase error
R_FCL_ERR_COMMAND	The requested command does not exist or it was disabled via precompiled options
R_FCL_CANCELLED	Current operation is cancelled
R_FCL_ERR_INTERNAL	Library 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, “Run-time configuration”) 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 (for details see chapter 5.2.3, “Configuration elements”).
addrRam_u32	This defines the start address of RAM reserved for execution of the FCL.
frequencyCpuMHz_u16	This defines the Flash hardware frequency in MHz (for details see chapter 5.2.3, “Configuration elements”)

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 perform 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 3.4.1, “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 bellow for details.

Pre-conditions: Function must be called only from ROM flash. Function must not be called after erasing `R_FCL_CONST` section.

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:

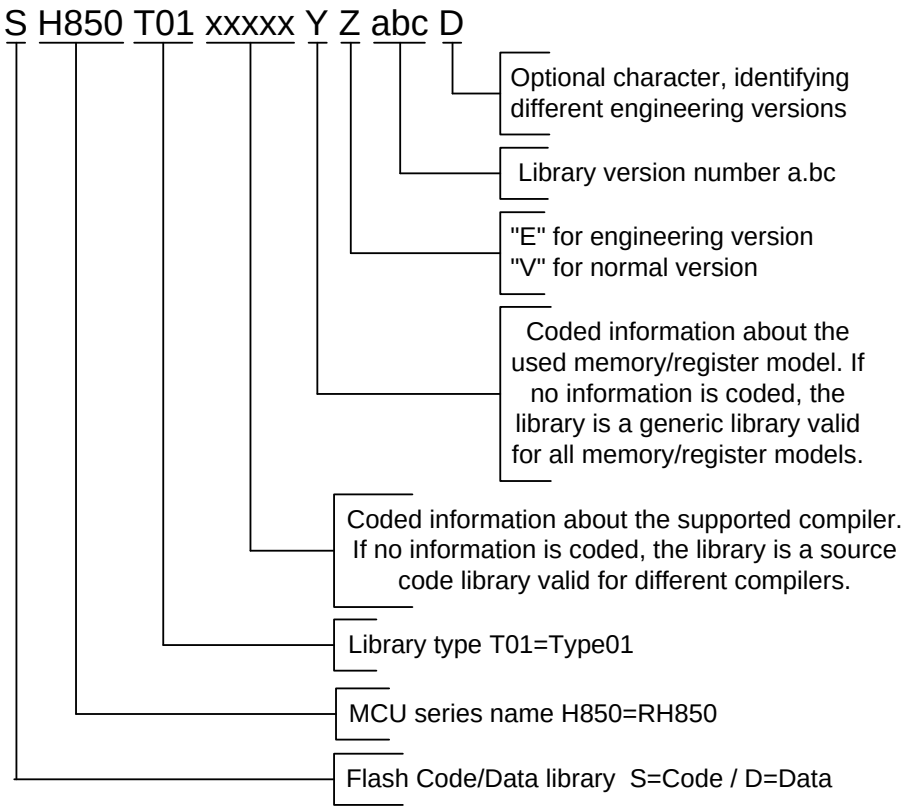

Figure 6: Version string

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 5.3, “Data types” for details After function return, member status_enu contains the return value of the command initiation explained in chapter 5.4, “Command / Operations”

: 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

Command	bufferAdd_u32 [buffer address]	idx_u32 [index]	cnt_u16 [count]
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
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 copy used firmware code to the RAM. This code is used during Self-Programming and need 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

Request structure member	Value	Description
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	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_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 (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	application bug (e.g. program run-away, destroyed program counter) or hardware problem
	remedy	refrain from further Flash operations and investigate in the root cause
R_FCL_ERR_PROTECTION	meaning	current command is rejected

Status	Background and Handling	
	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

*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	nothing to do
R_FCL_SUSPENDED*	meaning	an on-going erase operation was successfully suspended
	reason	suspend processing successfully finished
	remedy	nothing to do

Status	Background and Handling	
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
	reason	the FLMD0 register / pin is not set correct
	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	at least one bit within the specified block is not erased completely
	reason	hardware defect or FLMD0 changed to low value during command execution
	remedy	Check FLMD0 setting. If the setting is correct, a Flash block, respectively the complete Code Flash, should be considered as defect.
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	application bug (e.g. program run-away, destroyed program counter) or hardware problem
	remedy	refrain from further Flash operations and investigate in the root cause

*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	nothing to do
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
	reason	the FLMD0 register / pin is not set correct
	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

Status	Background and Handling	
	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	at least one data bit could not be written correctly
	reason	• the application tried to write on not erased addresses • at least one bit of the area to write was not completely erased (not blank) • at least one bit of the written area could not be completely written • FLMD0 changed to low value during command execution
	remedy	• correct the flow • a Flash block, respectively the complete Code Flash, should be considered as defect • check FLMD0 setting and repeat 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	application bug (e.g. program run-away, destroyed program counter) or hardware problem
	remedy	refrain from further Flash operations and investigate in the root cause

*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 lockbits are erased together with the content of the block.

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

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

Request structure member	Value	Description
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	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_FLMD0	meaning	current command is rejected
	reason	the FLMD0 register / pin is not set correct
	remedy	investigate in the root cause and correct the register value or the input of the pin
R_FCL_ERR_WRITE	meaning	at least one data bit could not be written correctly
	reason	• at least one bit of the written area could not be completely written • FLMD0 changed to low value during command execution
	remedy	• a Flash block, respectively the complete Code Flash, should be considered as defect • check FLMD0 setting and repeat 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

Status	Background and Handling	
	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	application bug (e.g. program run-away, destroyed program counter) or hardware problem
	remedy	refrain from further Flash operations and investigate in the root cause

*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 lockbits are erased together with the content of the block.

Note 2: The effect of the lockbits 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
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 lockbit for the requested block is available in buffer
	reason	no problems during execution
	remedy	nothing to do
R_FCL_BUSY*	meaning	operation started successfully

Status	Background and Handling	
	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_INTERNAL	meaning	a library internal error occurred, which could not happen in case of normal application execution
	reason	application bug (e.g. program run-away, destroyed program counter) or hardware problem
	remedy	refrain from further Flash operations and investigate in the root cause
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

*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 lockbit. If lockbits mechanism is enabled and a `R_FCL_CMD_ERASE` or `R_FCL_CMD_WRITE` command is performed on a certain block that has its lockbits 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 lockbit mechanism is enabled now

Status	Background and Handling	
	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 lockbit. If lockbits mechanism is disabled `R_FCL_CMD_ERASE` or `R_FCL_CMD_WRITE` commands perform normally on a certain block that has its lockbits set. A successful `R_FCL_CMD_ERASE` operation will also erase lockbits associated with the erased blocks when lockbits 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 lockbit 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

Status	Background and Handling	
	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 one-time-programmable flag which disables further modifications of the Flash of the device.

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	
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	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
	remedy	investigate in the root cause and correct the parameters

Status	Background and Handling	
R_FCL_ERR_FLMD0	meaning	current command is rejected
	reason	the FLMD0 register / pin is not set correct
	remedy	investigate in the root cause and correct the register value or the input of the pin
R_FCL_ERR_WRITE	meaning	at least one data bit could not be written correctly
	reason	• at least one bit of the written area could not be completely written • FLMD0 changed to low value during command execution
	remedy	• configuration Flash area, respectively the complete Flash, may be defect • check FLMD0 setting and repeat 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	application bug (e.g. program run-away, destroyed program counter) or hardware problem
	remedy	refrain from further Flash operations and investigate in the root cause

*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 (buffer is 1 word size but only the LSB should be considered)
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
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.

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

Status	Background and Handling	
R_FCL_BUSY*	meaning	operation started successfully
	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 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 correct
	remedy	investigate in the root cause and correct the register value or the input of the pin
R_FCL_ERR_WRITE	meaning	at least one data bit could not be written correctly
	reason	• at least one bit of the written area could not be completely written • FLMD0 changed to low value during command execution
	remedy	• configuration Flash area, respectively the complete Flash, may be defect • check FLMD0 setting and repeat 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	application bug (e.g. program run-away, destroyed program counter) or hardware problem
	remedy	refrain from further Flash operations and investigate in the root cause

*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 bytes 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.

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	

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 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	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 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 correct
	remedy	investigate in the root cause and correct the register value or the input of the pin
R_FCL_ERR_WRITE	meaning	at least one data bit could not be written correctly
	reason	• at least one bit of the written area could not be completely written • FLMD0 changed to low value during command execution
	remedy	• configuration Flash area, respectively the complete Flash, may be defect • check FLMD0 setting and repeat 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

Status	Background and Handling	
	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	application bug (e.g. program run-away, destroyed program counter) or hardware problem
	remedy	refrain from further Flash operations and investigate in the root cause

*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

Status	Background and Handling	
	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.

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	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_FLMD0	meaning	current command is rejected
	reason	the FLMD0 register / pin is not set correct
	remedy	investigate in the root cause and correct the register value or the FLMD0 pin input
R_FCL_ERR_WRITE	meaning	at least one data bit could not be written correctly

Status	Background and Handling	
	reason	• at least one bit of the written area could not be completely written • FLMD0 changed to low value during command execution
	remedy	• configuration Flash area, respectively the complete Flash, may be defect • check FLMD0 setting and repeat 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	application bug (e.g. program run-away, destroyed program counter) or hardware problem
	remedy	refrain from further Flash operations and investigate in the root cause

*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

Status	Background and Handling	
	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.

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	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

Status	Background and Handling	
	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 correct
	remedy	investigate in the root cause and correct the register value or the input of the pin
R_FCL_ERR_WRITE	meaning	at least one data bit could not be written correctly
	reason	• at least one bit of the written area could not be completely written • FLMD0 changed to low value during command execution
	remedy	• configuration Flash area, respectively the complete Flash, may be defect • check FLMD0 setting and repeat 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	application bug (e.g. program run-away, destroyed program counter) or hardware problem
	remedy	refrain from further Flash operations and investigate in the root cause

*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.

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

Status	Background and Handling	
	remedy	nothing to do
R_FCL_BUSY*	meaning	operation started successfully
	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_FLMD0	meaning	current command is rejected
	reason	the FLMD0 register / pin is not set correct
	remedy	investigate in the root cause and correct the register value or the input of the pin
R_FCL_ERR_WRITE	meaning	at least one data bit could not be written correctly
	reason	• at least one bit of the written area could not be completely written • FLMD0 changed to low value during command execution
	remedy	• configuration Flash area, respectively the complete Flash, may be defect • check FLMD0 setting and repeat 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	application bug (e.g. program run-away, destroyed program counter) or hardware problem
	remedy	refrain from further Flash operations and investigate in the root cause

*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)

Request structure member	Value	Description
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.

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	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_FLMD0	meaning	current command is rejected
	reason	the FLMD0 register / pin is not set correct
	remedy	investigate in the root cause and correct the register value or the input of the pin
R_FCL_ERR_WRITE	meaning	at least one data bit could not be written correctly
	reason	• at least one bit of the written area could not be completely written • FLMD0 changed to low value during command execution
	remedy	• configuration Flash area, respectively the complete Flash, may be defect • check FLMD0 setting and repeat 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	application bug (e.g. program run-away, destroyed program counter) or hardware problem
	remedy	refrain from further Flash operations and investigate in the root cause

*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.

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	

Request structure member	Value	Description
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
	reason	no problems during execution
	remedy	nothing to do
R_FCL_BUSY*	meaning	operation started successfully
	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_FLMD0	meaning	current command is rejected
	reason	the FLMD0 register / pin is not set correct
	remedy	investigate in the root cause and correct the register value or the input of the pin
R_FCL_ERR_WRITE	meaning	at least one data bit could not be written correctly
	reason	• at least one bit of the written area could not be completely written • FLMD0 changed to low value during command execution
	remedy	• configuration Flash area, respectively the complete Flash, may be defect • check FLMD0 setting and repeat 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	application bug (e.g. program run-away, destroyed program counter) or hardware problem

Status	Background and Handling	
	remedy	refrain from further Flash operations and investigate in the root cause

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

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	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_FLMD0	meaning	current command is rejected
	reason	the FLMD0 register / pin is not set correct
	remedy	investigate in the root cause and correct the register value or the input of the pin
R_FCL_ERR_WRITE	meaning	at least one data bit could not be written correctly
	reason	• at least one bit of the written area could not be completely written • FLMD0 changed to low value during command execution
	remedy	• configuration Flash area, respectively the complete Flash, may be defect • check FLMD0 setting and repeat command

Status	Background and Handling	
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	application bug (e.g. program run-away, destroyed program counter) or hardware problem
	remedy	refrain from further Flash operations and investigate in the root cause

*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 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

Status	Background and Handling	
	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

Status	Background and Handling	
	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 misbehaviour 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 and Chapter 3).

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:

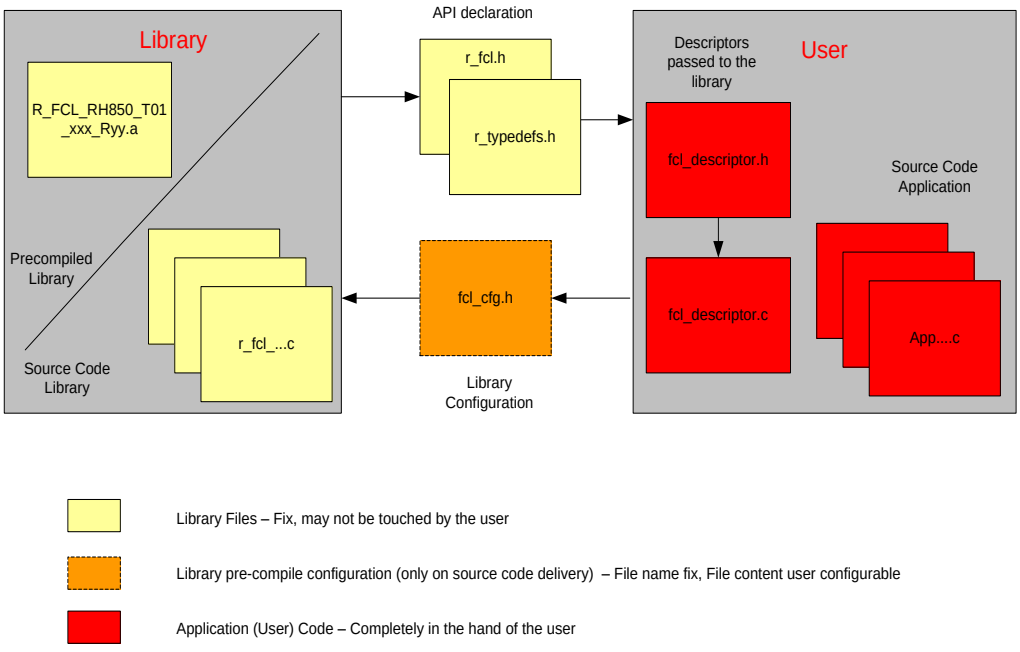

Figure 7: Library and application file structure

The library code consists of different source files, starting with `r_fcl_...` The files shall not be touched by the user, independently, if the library is distributed as source code or pre-compiled.

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>	
Release.txt	Library package release notes.
<installation_folder>/Make	
GNUPublicLicense.txt	GNU Make utility license file
Readme.txt	Extra information for source code of GNU Make.
libiconv2.dll	Minimal installation of GNU Make utility.
libintl3.dll	
make.exe	
setup.exe	GNU Make installer package.
<installation_folder>/<device_name>/<compiler>	
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>/<device_name>/<compiler>/Sample	
dr7f701035_0.h dr7f701035_irq.h	Device specific header files.
dr7f701035_startup.850	Device and compiler specific start-up code.
dr7f701035.ld	Compiler specific linker directives.
fcl_descriptor.c	FCL descriptor used in the sample application.
fcl_descriptor.h	FCL descriptor used in the sample application.
icu_feret.h	Definitions for device Intelligent Cryptographic Unit.
io_macros_v2.h	Definitions of IO macros for RH850 devices
main.c	Sample application code
target.h	Initialization code for target microcontroller.

File	Description
<installation_folder>/<device_name>/<compiler>/Sample/FCL	
fcl_cfg.h	User defined configuration for FCL.
r_fcl.h	FCL API definitions.
<installation_folder>/<device_name>/<compiler>/Sample/FCL/lib	
r_fcl_hw_access.c r_fcl_user_if.c	FCL main source code.
r_fcl_env.h	Internal FCL definitions.
r_fcl_global.h	Global variables and settings used during self-programming
r_fcl_types.h	User interface type definitions and all error and status codes used during self-programming
r_typedefs.h	C types used by FCL library.
r_fcl_hw_access_asm.850 for GHS	Internal library compiler specific source code.

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.
  - R_FCL_RESERVE – is a RAM reserved space. The following sections will be copied into this section: R_FCL_CODE_USRINT, R_FCL_CODE_USR, R_FCL_CODE_RAM, R_FCL_CODE_ROMRAM and R_FCL_CODE_RAM_EX_PROT.

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

Note 2: 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

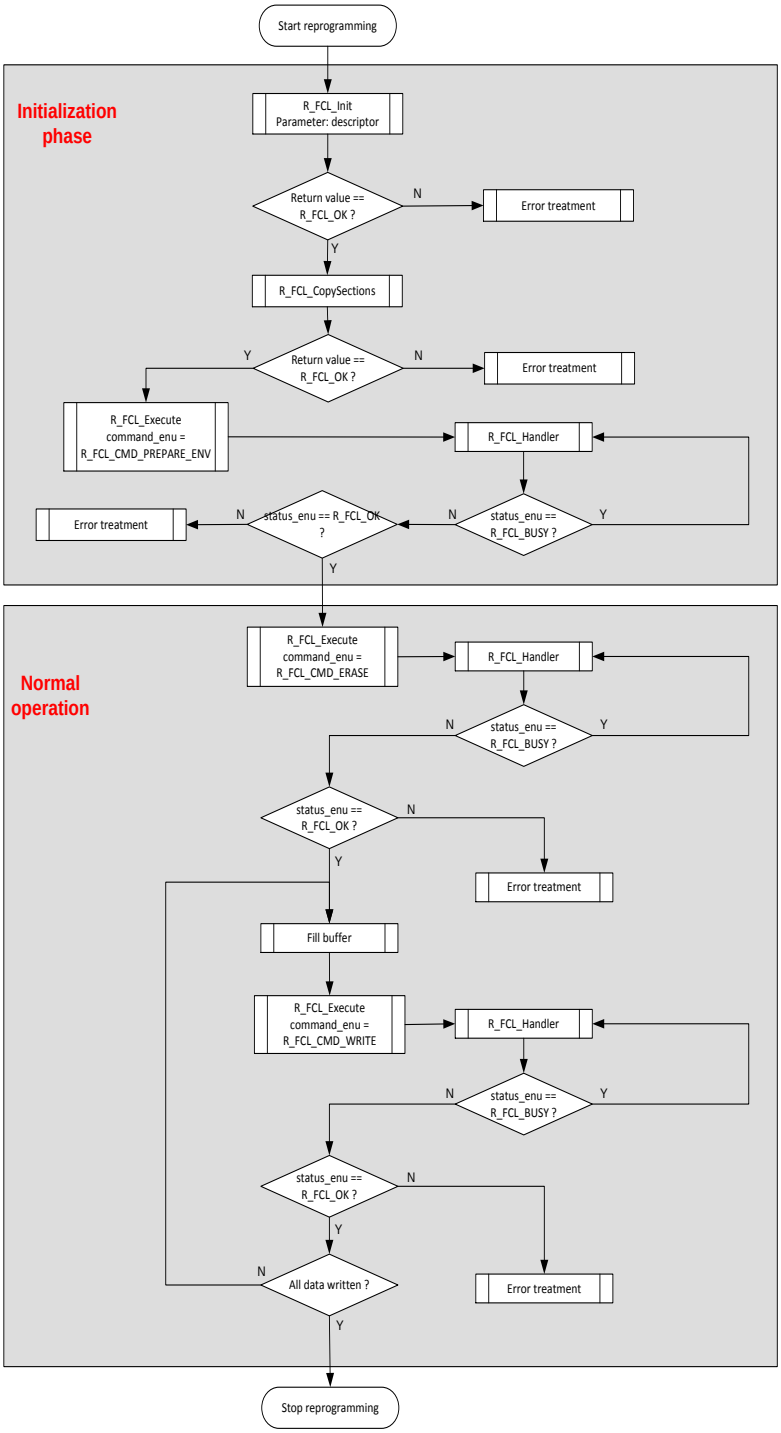

Figure 8: Typical reprogramming flow in user mode

5.5.2 Typical flow chart for reprogramming in internal mode

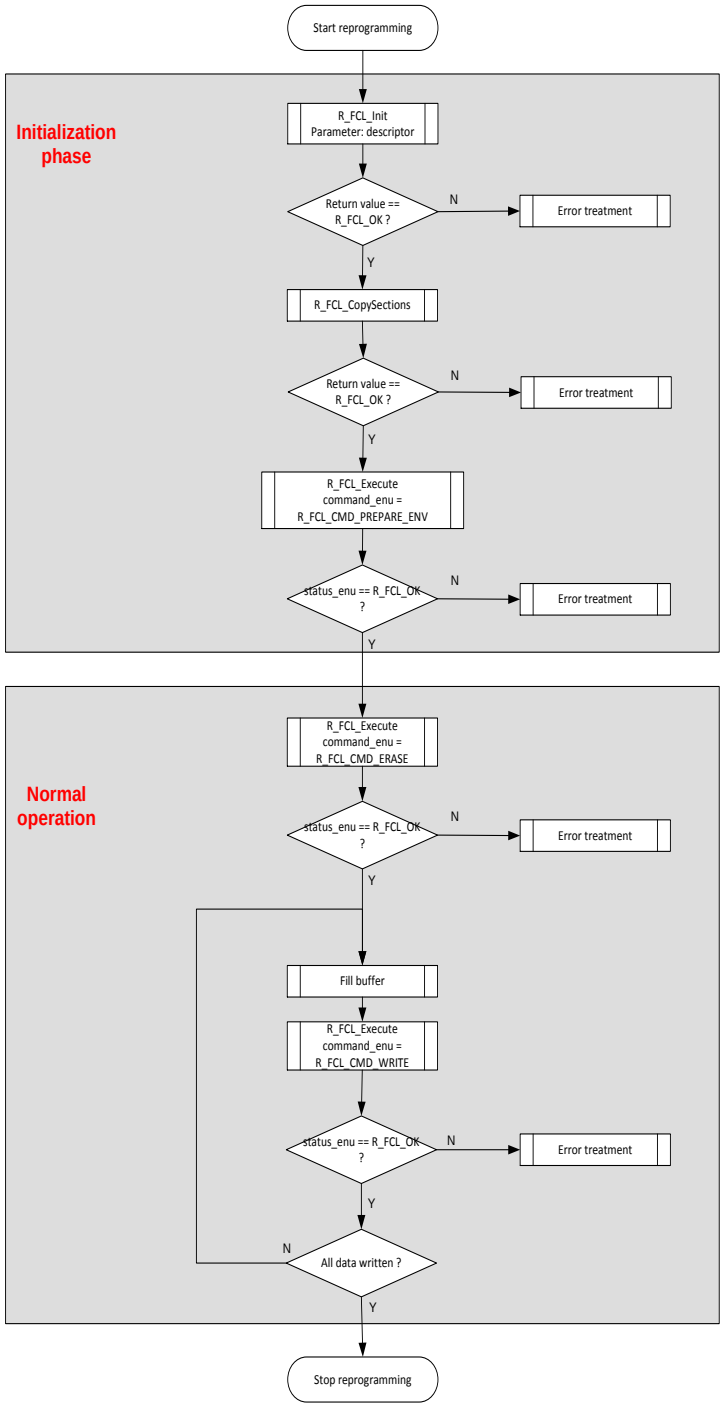

Figure 9: Typical reprogramming flow in internal mode

5.6 MISRA Compliance

The FCL code has been tested regarding MISRA compliance.

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

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 FLC with proper CPU frequency.

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. If such access to code flash is performed the result will almost always be a crash.

   Many compilation and run-time errors arise due to access in code flash during FCL operation. Because code flash is not available during operation of most of the FCL commands, 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 5.4 “Linker sections”) and it is copied into the right place in RAM (refer to chapter 3.1 “Code execution in RAM”).

5. Interrupt Vector Assignment during Self-Programming:

   The generation of exceptions or interrupts during Self-Programming may lead to the fetching of vectors from the ROM. To prevent access to the ROM area due to the generation of interrupts, set the interrupt table register (INTBP) and exception vector (EBASE) of the CPU so that the destination for the fetching of interrupt vector and interrupt routines are located in RAM.

   If non-maskable interrupts are used it is advisable to break long lasting FCL operations into smaller operations in such a way that the interrupt conditions can be checked and/or treated before they appear (e.g. to check if watchdog has to be refreshed or check for low voltage supply.)

6. Interrupted flash operations:

   In case of Flash modification operation (Erase / Write) interruption, the electrical conditions of the affected Flash range (Flash block on erase, Flash write unit on Write) get undefined. It is impossible to give a statement on the read value after the interruption. Furthermore, the resulting read value is not reliable; the electrical margin for the specified data retention may not be given. In such case, erase and re-write the affected Flash block(s) 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 (call `R_FCL_CopySections`)

- prepare the library (call R_FCL_Execute with R_FCL_CMD_PREPARE_ENV).

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 crash.

12. Data alignment:

    When specifying an operation, destination address has to be aligned with Code Flash unit of granularity.

    RH850 devices may also add alignment restrictions for data types larger than 8 bits. Please consult device hardware manual for details.

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.

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

Self-Programming Library for Code Flash

Renesas Electronics Corporation

R01US0078ED0110