User's Manual

RZ/T2M Group

Encoder I/F Configuration Library User's Manual

RENESAS MCU
RZ Family / RZ/T Series

Renesas Electronics
www.renesas.com

Rev.4.00 Feb 2026

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

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

(Rev.5.0-1 October 2020)

Corporate Headquarters

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

www.renesas.com

Trademarks

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

Contact information

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

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

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)
   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on
   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state
   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins
   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals
   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin
   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses
   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products
   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

How to Use This Manual

1. Purpose and Target Readers

This manual is for users to understand the functions and usage of the library software “Encoder I/F Configuration Library”. It is intended for users who design application systems using this library software. To use this manual, you need a basic knowledge of programming languages and microcomputers.

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 Usage Notes 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.

Table of Contents

Corporate Headquarters 2

Contact information 2

Trademarks 2

1. Overview 1
1.1 Summary 1
1.2 Function 1
1.3 Software Configuration 2

2. Operating Environment 3

3. File Configuration 4

4. EC-Lib API Specification 5
4.1 List of API Functions 5
4.2 Data Type Used 5
4.3 Structure Used 5
4.4 Definition of Values (Macro) 5
4.5 Definition of Values (Enumerated Types) 5
4.6 Error code 6

5. API Reference 7
5.1 How to Read the API Reference 7
5.2 R_ECL_Initialize 8
5.3 R_ECL_Terminate 9
5.4 R_ECL_ConfigurePin 10
5.5 R_ECL_Configure 11
5.6 R_ECL_UnConfigure 13
5.7 R_ECL_Start 14
5.8 R_ECL_Stop 15
5.9 R_ECL_GetVersion 16

6. Resources 17
6.1 H/W 17
6.2 OS 17
6.3 Memory 17

7. Flowchart 18

1. Overview

1.1 Summary

This document explains the API functions used to control the encoder interface configuration library (hereinafter referred to as the "EC-Lib")

1.2 Function

The EC-Lib is a software library for configuring, activating, and stopping the encoder interface mounted on the RZ/T2M.

1.3 Software Configuration

Figure 1.1 shows the software configuration of the EC-Lib.

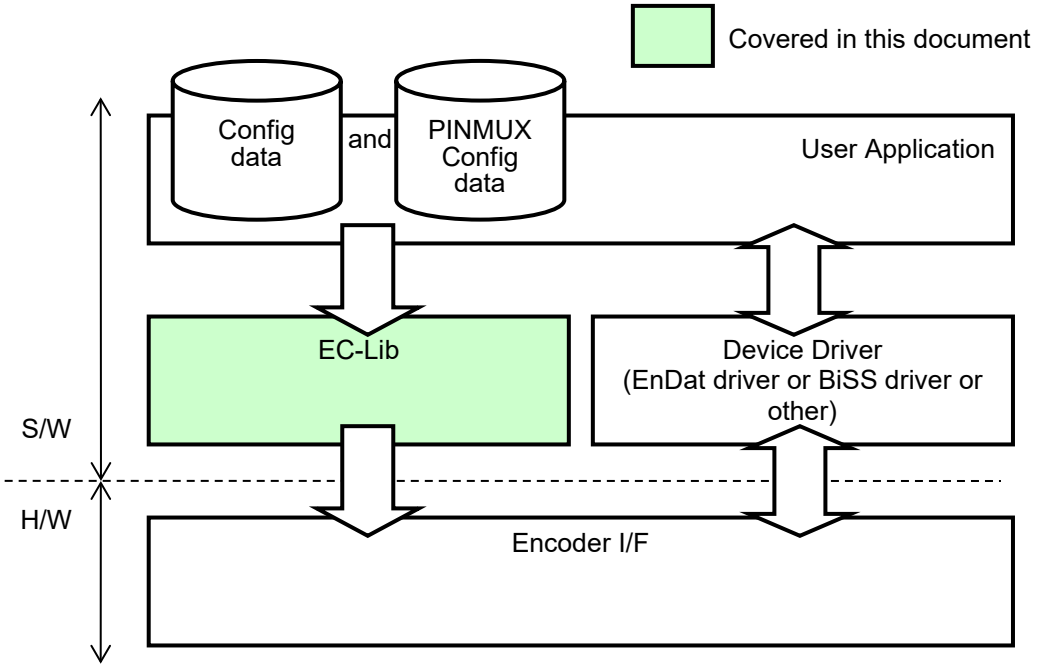


Figure 1.1 Software Configuration (EC-Lib and the Related Modules)

2. Operating Environment

The library explained in this application note operates in the following environment

Table 2.1 Operating Environment

Item	Description	
Microcomputer	RZ/T2M (Cortex®-R52)	
Operating frequency	800 MHz	
Operating voltage	1.1 V (Core) / 1.8 V (PLL, etc.) / 3.3 V (I/O)	
Development environment	For IAR	IAR Embedded Workbench® for Arm®
	For GCC	RENESAS e^2 studio GNU Arm Embedded Toolchain

3. File Configuration

Figure 3.1 shows the file configuration of the EC-Lib. The header file is the same in IAR and GCC. The library files for IAR and GCC are distinguished by their names.

```
Lib
    ecl
        r_ecl_rz2_if.h          EC-Lib header file
        r_ecl_rzt2_iar.a        EC-Lib library file (IAR version)
        r_ecl_rzt2_gcc.a        EC-Lib library file (GCC version)
```

Figure 3.1 File Configuration

4. EC-Lib API Specification

4.1 List of API Functions

Table 4.1 lists the control API functions.

Table 4.1 EC-Lib APIs

API Name	Description	Page
R_ECL_Initialize	Initialization of the Encoder I/F	8
R_ECL_Terminate	Termination of the Encoder I/F	9
R_ECL_ConfigurePin	Configuration of the PINMUX	10
R_ECL_Configure	Configuration of the Encoder I/F	11
R_ECL_UnConfigure	Unconfiguration of the Encoder I/F	13
R_ECL_Start	Starting operation of the Encoder I/F	14
R_ECL_Stop	Stopping operation of the Encoder I/F	15
R_ECL_GetVersion	Acquisition of the EC-Lib version number.	16

4.2 Data Type Used

The EC-Lib API does not define a data type.

Both of the IAR version and GCC version use uint8_t, int32_t and uint32_t defined in the standard libraries.

4.3 Structure Used

Structures are not used in the EC-Lib API.

4.4 Definition of Values (Macro)

The macro-defined values used in the EC-Lib API are indicated in the descriptions of the individual functions where they are used. See Section 4.6, Error Code for the error code macros.

4.5 Definition of Values (Enumerated Types)

Enumerated types are not used in the EC-Lib API.

4.6 Error code

In the EC-Lib API, 0 is returned for normal termination and negative values are returned for errors. Table 4.2 lists the error codes.

Table 4.2 Error Codes

Macro Name	Value	Meaning
R_ECL_SUCCESS	0	Normal termination
R_ECL_ERR_ARG	−1	Argument error
R_ECL_ERR_FORMAT	−2	Invalid format
R_ECL_ERR_BUSY	−5	Busy
R_ECL_ERR_STATUS	−9	Status error

5. API Reference

5.1 How to Read the API Reference

API Name	Category
Function summary	Synchronous/Asynchronous

Format	This indicates the format for calling the function. Be sure to include the header file indicated as #include "header file", which is the standard header file needed to execute functions of this API. I, O, and IO represent input data, output data, and input-output data, respectively.
Returned Value	This indicates the value returned by the functions. The conditions for return and other descriptions of the value are indicated following a colon after the value.
Description	This describes the specification of the API.
Note	Points to note are given here, if any.
Usage Example	This gives an example of usage of the function, if any.

5.2 R_ECL_Initialize

R_ECL_Initialize

EC-Lib API

Encoder I/F Initialization

Synchronous function

Format

```
#include "r_ecl_rzt2_if.h"
int32_t R_ECL_Initialize(void)
```

Returned Value

R_ECL_SUCCESS	:	Success
R_ECL_ERR_STATUS	:	Failure (Encoder I/F is already initialized)

Description

This function makes initial settings for the Encoder I/F.
This function initializes the internal variables of the Encoder I/F so that the Encoder I/F can be used.
It also returns the Encoder I/F from low power mode to start clocking and initialize the hardware.

Note

Be sure to call this function before using the Encoder I/F.

Usage Example

```
#include "r_ecl_rzt2_if.h"
int32_t result;

result = R_ECL_Initialize ();
if (result == R_ECL_SUCCESS)
{
   /* normal operation */
}
else
{
   /* error handling */
}
```

5.3 R_ECL_Terminate

R_ECL_Terminate

EC-Lib API

Encoder I/F Termination

Synchronous function

Format

```
#include "r_ecl_rzt2_if.h"
int32_t R_ECL_Terminate (void)
```

Returned value

R_ECL_SUCCESS	:	Success
R_ECL_ERR_STATUS	:	Failure (Encoder I/F disabled, Configuration data is loaded in the Encoder I/F)

Descriptrion

This function stops the clock supply to the Encoder I/F and shifts the Encoder I/F to the low power consumption mode.
After executing this function, the Encoder I/F cannot be used until the R_ECL_Initialize function is called again.

Note

When the configuration data is loaded in the Encoder I/F, unload the configuration data with the R_ECL_Unconfigure function before using this function.

Usage Example

```
#include "r_ecl_rzt2_if.h"
int32_t result;

result = R_ECL_Terminate ();
if (result == R_ECL_SUCCESS)
{
   /* normal operation */
}
else
{
   /* error handling */
}
```

5.4 R_ECL_ConfigurePin

R_ECL_ConfigurePin

EC-Lib API

PINMUX Configuration

Synchronous function

Format

```
#include "r_ecl_rzt2_if.h"
int32_t R_ECL_ConfigurePin(const void *const pconfig1)
```

pconfig1	I	Start address of PINMUX configuration data Designate data aligned on an eight-byte boundary.

Returned value

R_ECL_SUCCESS	:	Success
R_ECL_ERR_ARG	:	Failure (The argument pconfig1 is null)
R_ECL_ERR_FORMAT	:	Failure (The format of the configuration data is invalid)
R_ECL_ERR_BUSY	:	Failure (The specified channel is configured for Encoder I/F)
R_ECL_ERR_STATUS	:	Failure (Encoder I/F is disabled)

Description

This function loads the PINMUX configuration data for the Encoder I/F.
If you use this function again after configuring PINMUX once, the PINMUX settings will be overwritten with the newly specified configuration data.

Note

This function cannot be used if the Encoder I/F is configured. If the Encoder I/F has already been configured, use the R_ECL_UnConfigure function to unload the configuration data from the Encoder I/F before using this function.

If the return value is R_ECL_FORMAT, check that the address specified by the argument pconfig1 is the correct configuration data address.

If the configuration data specified by the argument pconfig1 exists in the cache of the LSI and mismatch with the contents of the physical memory, it cannot be loaded correctly. In that case, clean the cache before calling this API function. Alternatively, you need to take action such as placing the configuration data in a non-cached area.

Usage Example

```
#include "r_ecl_rzt2_if.h"
extern const uint32_t pinmux_config[];
int32_t result;

result = R_ECL_Configure(pinmux_config);
if (result == R_ECL_SUCCESS)
{
   /* normal operation */
}
else
{
   /* error operation */
}
```

5.5 R_ECL_Configure

R_ECL_Configure

EC-Lib API

Encoder I/F configuration

Synchronous function

Format

```
#include "r_ecl_rzt2_if.h"
int32_t R_ECL_Configure(const uint8_t id, const void *const pconfig2)
```

id	I	The ID number of the encoder interface For channel 0, R_ECL_CH_0 (defined in r_ecl_rzt2_if.h) For channel 1, R_ECL_CH_1 (defined in r_ecl_rzt2_if.h) For configuration data that uses two channels, set R_ECL_CH_0. You cannot select multiple channels.
pconfig2	I	The initial address of the configuration data Designate data aligned on a thirty-two-byte boundary.

Returned value

R_ECL_SUCCESS	:	Success
R_ECL_ERR_ARG	:	Failure (Invalid value for argument id. Multiple channels are set by the argument id. Argument pconfig2 is NULL. R_ECL_CH_1 is specified for the argument id when loading configuration data that uses two channels.)
R_ECL_ERR_FORMAT	:	Failure (The configuration data format is invalid)
R_ECL_ERR_BUSY	:	Failure (The specified channel is configured)
R_ECL_ERR_STATUS	:	Failure (Encoder I/F is disabled)

Description

This function loads the configuration data into the Encoder I/F
In this function, set only one of channel 0 and channel 1 in the argument id.
This function sets the Encoder I/F according to the configuration data specified by the argument pconfig.

Note

After executing this function, the Encoder I/F will be stopped. Start the Encoder I/F with the R_ECL_Start function before using the Encoder I/F.

Be sure to initialize the Encoder I/F in the order of R_ECL_Initialize function and R_ECL_ConfigurePin function before executing this function.

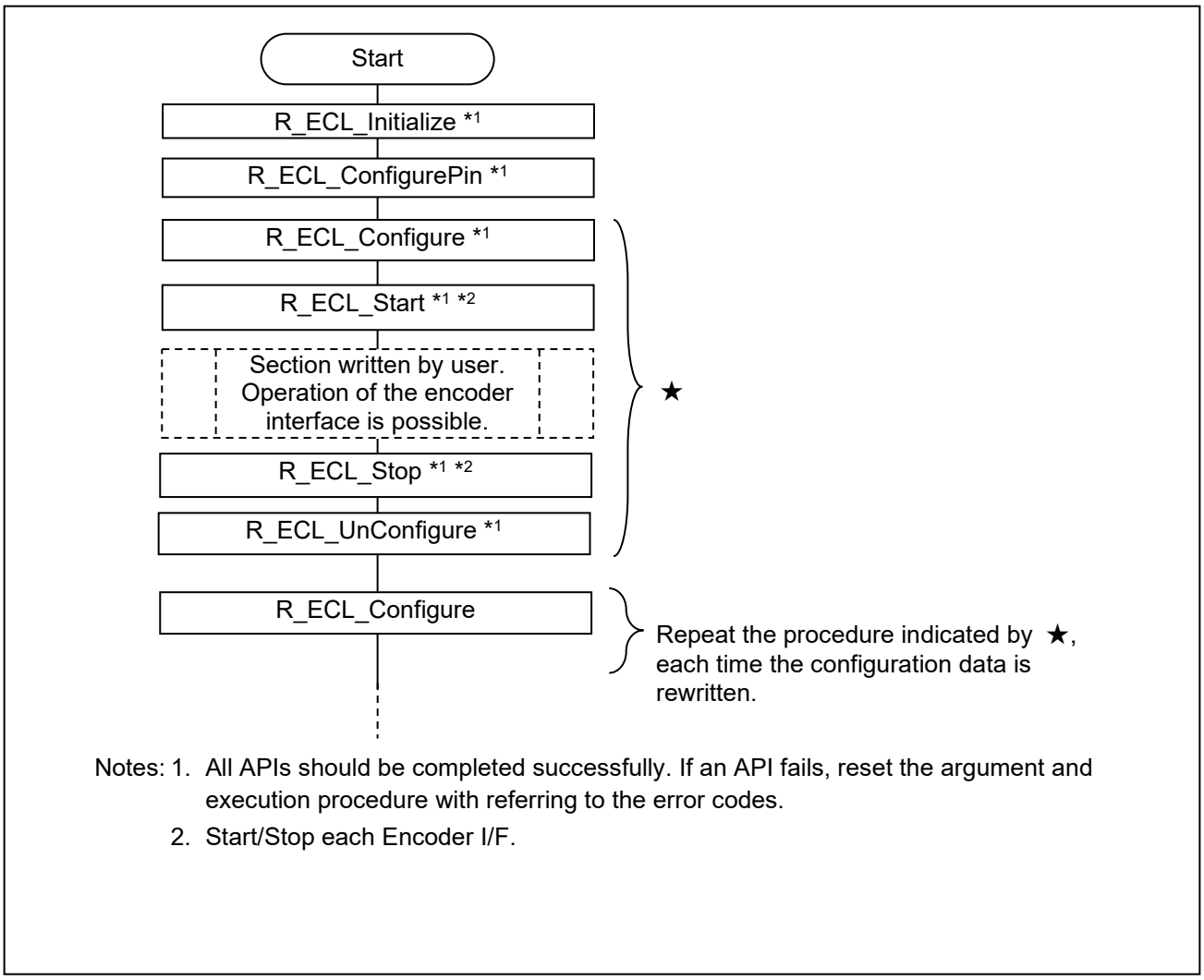
If the return value is R_ECL_FORMAT, check that the address specified by the argument pconfig2 is the correct configuration data address.

The configuration data specified by the argument pconfig2 exists in the cashe of the LSI and mismatch with the contents of the physical memory, it cannot be loaded correctly. In that case, clean the cache before calling this API function. Alternatively, you need to take action such as placing the configuration data in the non-cached area.

Usage Example

```
#include "r_ecl_rzt2_if.h"
extern const uint32_t config_data[];
int32_t result;

result = R_ECL_Configure(R_ECL_CH_0,config_data);
if (result == R_ECL_SUCCESS)
{
    /* normal operation */
}
else
{
    /* error operation */
}
```



Notes: 1. All APIs should be completed successfully. If an API fails, reset the argument and execution procedure with referring to the error codes.

2. Start/Stop each Encoder I/F.

Figure 5.1 Rewriting procedure for configuration data

5.6 R_ECL_UnConfigure

R_ECL_UnConfigure

EC-Lib API

Unconfiguring Encoder I/F

Synchronous function

Format

```
#include "r_ecl_rzt2_if.h"
int32_t R_ECL_UnConfigure(const uint8_t id)
```

id	I	The ID number of the encoder interface For channel 0 R_ECL_CH_0(defined by r_ecl_rzt2_if.h) For channel 1 R_ECL_CH_1(defined by r_ecl_rzt2_if.h) For configuration data that uses two channels, set R_ECL_CH_0. Multiple channels can be selected. To unload channels 1 and 2 at the same time, set logical-OR of R_ECL_CH_0 and R_ECL_CH_1.

Return value

R_ECL_SUCCESS	:	Success
R_ECL_ERR_ARG	:	Failure (Invalid value for argument id, Specify R_ECL_CH_1 for the argument id when unloading configuration data that uses two channels)
R_ECL_ERR_STATUS	:	Failure (Configuration data is not loaded on the specified channel)

Description

This function unloads the configujration data of each channel of Encoder I/F.
If the Encoder I/F is running, stop the Encoder I/F before unloading.

Note

If the configuration data is already unloaded from one or both of channels 0 and 1, and this function is used with channels 0 and 1 specified in the argument id, an R_ECL_ERR_STATUS error is returned.

Usage Example

```
#include "r_ecl_rzt2_if.h"
int32_t result;

result = R_ECL_UnConfigure(R_ECL_CH_0);
if (result == R_ECL_SUCCESS)
{
   /* normal operation */
}
else
{
   /* error operation */
}
```

5.7 R_ECL_Start

R_ECL_Start

EC-Lib API

Activates the Encoder I/F

Synchronous function

Format

```
#include "r_ecl_rzt2_if.h"
int32_t R_ECL_Start(const uint8_t id, const uint32_t freq)
```

id	I	The ID number of the encoder interface R_ECL_CH_0 for the channel 0 (defined with r_ecl_rzt2_if.h) R_ECL_CH_1 for the channel 1 (defined with r_ecl_rzt2_if.h) For configuration data that uses two channels, set R_ECL_CH_0 Multiple channels can be selected. When starting channels 1 and 2 at the same time, set logical-OR of R_ECL_CH_0 and R_ECL_CH_1.
freq	I	Set the operating frequency in kHz.In the case of 10MHz, it will be 10000. The operating frequency range that can be set with the function is 1562 (1.562MHz) to 50000 (50MHz). Set the argument within this range and does not exceed the maximum operating frequency of the configuration data.

Return value

R_ECL_SUCCESS	:	Success
R_ECL_ERR_ARG	:	Failure (Invalid argument id, specify R_ECL_CH_1 for argument id when starting Encoder I/F loaded with configuration data using two channels, argument freq is invalid.)
R_ECL_ERR_STATUS	:	Failure (Configujration data is not loaded in the channel specified by the argument id, the specified channel has been started.)

Description

This function activates each channel of Encoder I/F.

Note

Be sure to execute this function before using the Encoder I/F.

Be sure to initialize the Encoder I / F in the order of R_ECL_Initialize function, R_ECL_ConfigurePin function, and R_ECL_Configure function before executing this function.

If channel 0, 1 or both channels are already running and both of the channel 0 and 1 are specified for the argument of this function, an R_ECL_ERR_STATUS error is returned.

Usage Example

```
#include "r_ecl_rzt2_if.h"
#include "r_endat_rzt2_dat.h"

int32_t result;

result = R_ECL_Start(R_ECL_CH_0, R_ENDAT_FREQ);
if (result == R_ECL_SUCCESS)
{
   /* normal operation */
}
else
{
   /* error operation */
}
```

5.8 R_ECL_Stop

R_ECL_Stop

EC-Lib API

Stops the Encoder I/F

Synchronous function

Format

```
#include "r_ecl_rzt2_if.h"
int32_t R_ECL_Stop(const int32_t id)
```

id	I	The ID number of the encoder interface. R_ECL_CH_0 for the channel 0 (Defined with r_ecl_rzt2_if.h) R_ECL_CH_1 for the channel 1 (Defined with r_ecl_rzt2_if.h) For configuration data that uses two channels, set R_ECL_CH_0. Multiple channels can be selected. To stop channels 1 and 2 at the same time, specify R_ECL_CH_0 and R_ECL_CH_1 by OR.

Return value

R_ECL_SUCCESS	:	Success
R_ECL_ERR_ARG	:	Failure (Specify R_ECL_CH_1 for the argument id when stopping the Encoder I/F that loaded the config data that uses two channels and the argument id is invalid.)
R_ECL_ERR_STATUS	:	Failure (Configuration data is not loaded in the channel specified by the argument id, and the specified channel is stopped.)

Description

This function stops the operation of Encoder I/F.

Note

If channel 0, 1 or both channels are already stopped and both of the channel 0 and 1 are specified for the argument of this function, an R_ECL_ERR_STATUS error is returned.

Usage Example

```
#include "r_ecl_rzt2_if.h"
int32_t result;

result = R_ECL_Stop(R_ECL_CH_0);
if (result == R_ECL_SUCCESS)
{
   /* normal operation */
}
else
{
   /* error operation */
}
```

5.9 R_ECL_GetVersion

R_ECL_GetVersion

EC-Lib API

Acquires the EC-Lib version number

Synchronous function

Format

```
#include "r_ecl_rzt2_if.h"
uint32_t R_ECL_GetVersion(void)
```

Returned value

Version :

b31 – b24	b23 – b16	b15 – b8	b7 – b0
Reserved	Major part of the version Number	Minor part of the version number	Build number

Bits 7～0: Stores the build number.

Bits 15～8: Stores the minor version.

Bits 23～16: Stores the major version.

Bits 31～24: Reserved area. 0 is stored.

Example)
If the return value is 0x00010002, Ver.1.02
If the return value is 0x00020305, Ver.2.35

Description

This function acquires the version number of the Encoder I/F configuration library.

Usage Example

```
uint32_t ver;

uint8_t major_ver;
uint8_t minor_ver;
uint8_t build_no;

ver = R_ECL_GetVersion();

major_ver = (uint8_t)((ver >> 16) & 0xFF);
minor_ver = (uint8_t)((ver >> 8) & 0xFF);
build_no = (uint8_t)(ver & 0xFF);

printf("EC-Lib Ver.%d.%d.%d.", major_ver, minor_ver, build_no);
```

6. Resources

6.1 H/W

The EC-Lib does not occupy hardware resources.

6.2 OS

The EC-Lib does not use an OS.

6.3 Memory

Table 6.1 lists the section names and approximate sizes of the memory areas used by the EC-Lib For the exact memory sizes, refer to the release note.

Table 6.1 Memory Resources

Category	Section Name	Size (approx.)
Code	.text *1	5.5 kBytes *2
RO Data	.rodata *1	0.1 kBytes or less
RW Data	.data *1	0.3 kBytes *2
ZI Data	.bss *1	0.1 kBytes or less
R_ECL_Initialize function stack	—	0.1 kBytes or less
R_ECL_ConfigurePin function stack	—	0.1 kBytes or less
R_ECL_Configure function stack	—	0.2 kBytes
R_ECL_UnConfigure function stack	—	0.1 kBytes
R_ECL_Start function stack	—	0.1 kBytes
R_ECL_Stop function stack	—	0.1 kBytes or less
R_ECL_Terminate function stack	—	0.1 kBytes or less
R_ECL_GetVersion function stack	—	0

Notes: 1. No specific section names are attached to memory resources.
2. The size depends on the version.

7. Flowchart

Figure 7.1 shows the flowchart as an example of controlling the EC-Lib.

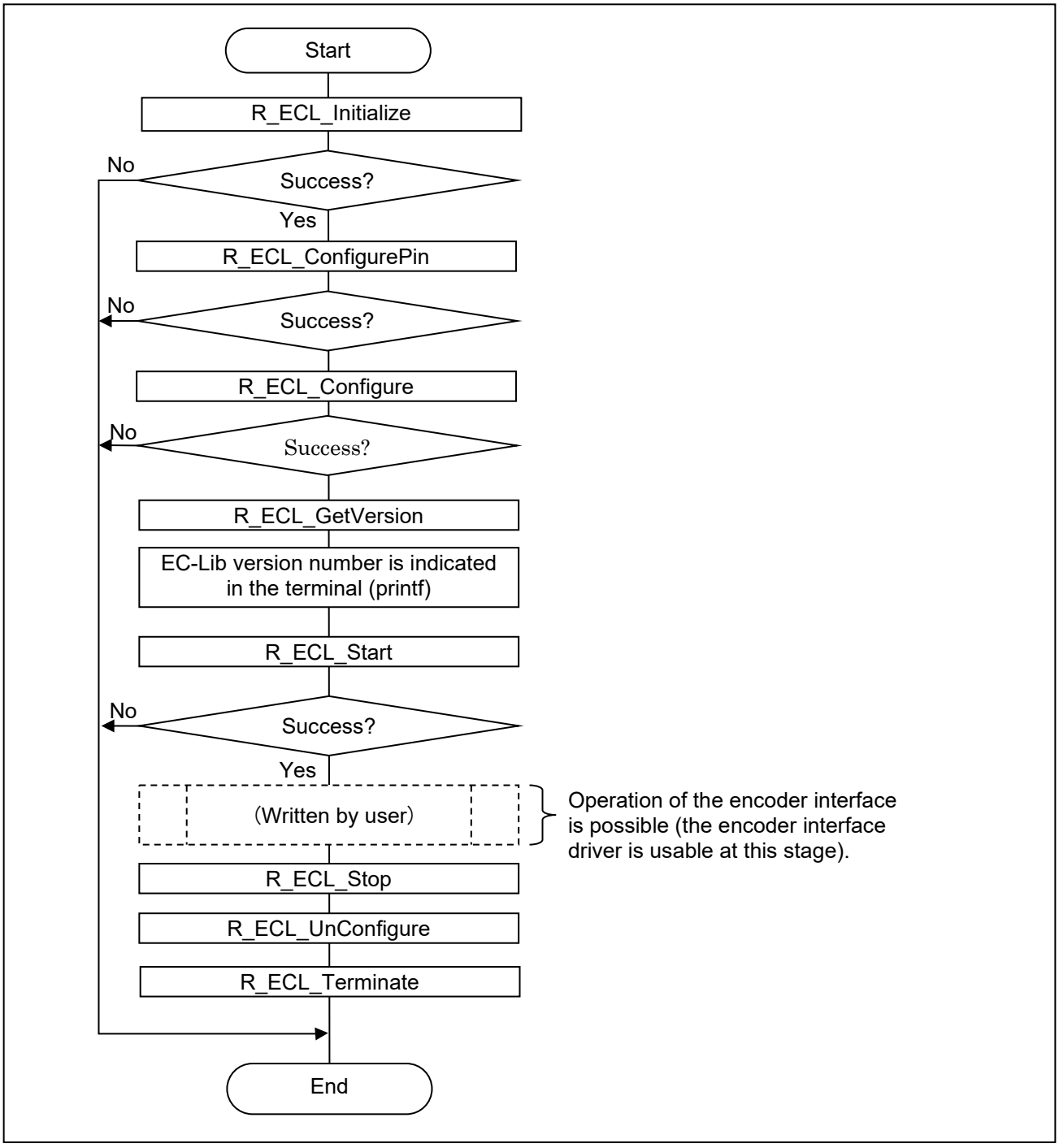


Figure 7.1 Example Flowchart of Controlling the EC-Lib

Revision History	RZ/T2M Group Encoder I/F Configuration Library User's Manual

Rev.	Date	Description	
		Page	Summary
1.00	Apr.08.22	—	First Edition issued
2.00	Jun 07.24	3	Description of the operating environment is updated.
3.00	Oct 10.25	3	Description of the operating environment and trademarks are updated.
		5	Update description about location of integer type definition.
		10	Add description about pconfig1 data alignment.
		15	Add note description for the R_ECL_Stop function,
4.00	Feb 27.26	10, 11	Update descriptions about pconfig1 and pconfig2 data alignment.
		17	Revise not to use specific section names. Update size information.

RZ/T2M Group
Encoder I/F Configuration Library User's Manual

Publication Date: Rev.4.00 Feb.27.26

Published by: Renesas Electronics Corporation

RZ/T2M Group

Renesas Electronics Corporation

R01US0563EJ0400