User's Manual

RL78 Family

DALI-2 Input Device Library

User's Manual: Light Sensor (304)

16-bit single chip microprocessor

Renesas Electronics
www.renesas.com

Rev.1.00 Oct.2023

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 intended for users who want to develop Input Device for DALI systems with RL78 microcontrollers.
Basic knowledge of electrical circuits, logic circuits, and microcomputers is required to use this manual.
This manual is broadly categorized and consists of product overview, specifications, and usage instructions.

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.

The following documents apply to the DALI Library. Make sure to refer to the latest versions of these documents. The newest versions of the documents listed may be obtained from the Renesas Electronics Web site.

Document Type	Description	Document Title	Document No.
User's Manual: Hardware	Hardware specifications (pin layout, memory map, peripheral function specifications, electrical characteristics, timing) and operation description * Refer to the application note for the usage of peripheral functions.	RL78/G23 User's Manual: Hardware	R01UH0896EJ0100
User's Manual: Software	Description of CPU instruction set	RL78 Family User's Manual: Software	R01US0015EJ0220
Application note	How to use peripheral functions, application examples Reference programs How to create programs in C language	The information is available on the Renesas Electronics website.	
Renesas Technical Update	Breaking news on product specifications, documents, etc.		

2.Explanation of abbreviations

Abbreviation	English name	Remarks
DALI	Digital Addressable Lighting Interface	International Standard for Lighting Control
NVM	Non-Volatile Memory	Non-volatile memory

Table of Contents

1.DALI304 Library Overview

1.1 Overview of library features

This library is an extension library exclusively for the DALI103i library, which is provided as a library for Input Device in DALI communication.
Refer to the DALI103i Library User's Manual for the specifications of the DALI103i library.
This library implements the hardware-independent part of the specification defined in IEC62386-304ed1.0 (hereinafter referred to as DALI304). Please use it when you want to implement an Input Device with an Instance Type 4 (Light Sensor) Instance.

Table 1-1 Processing range

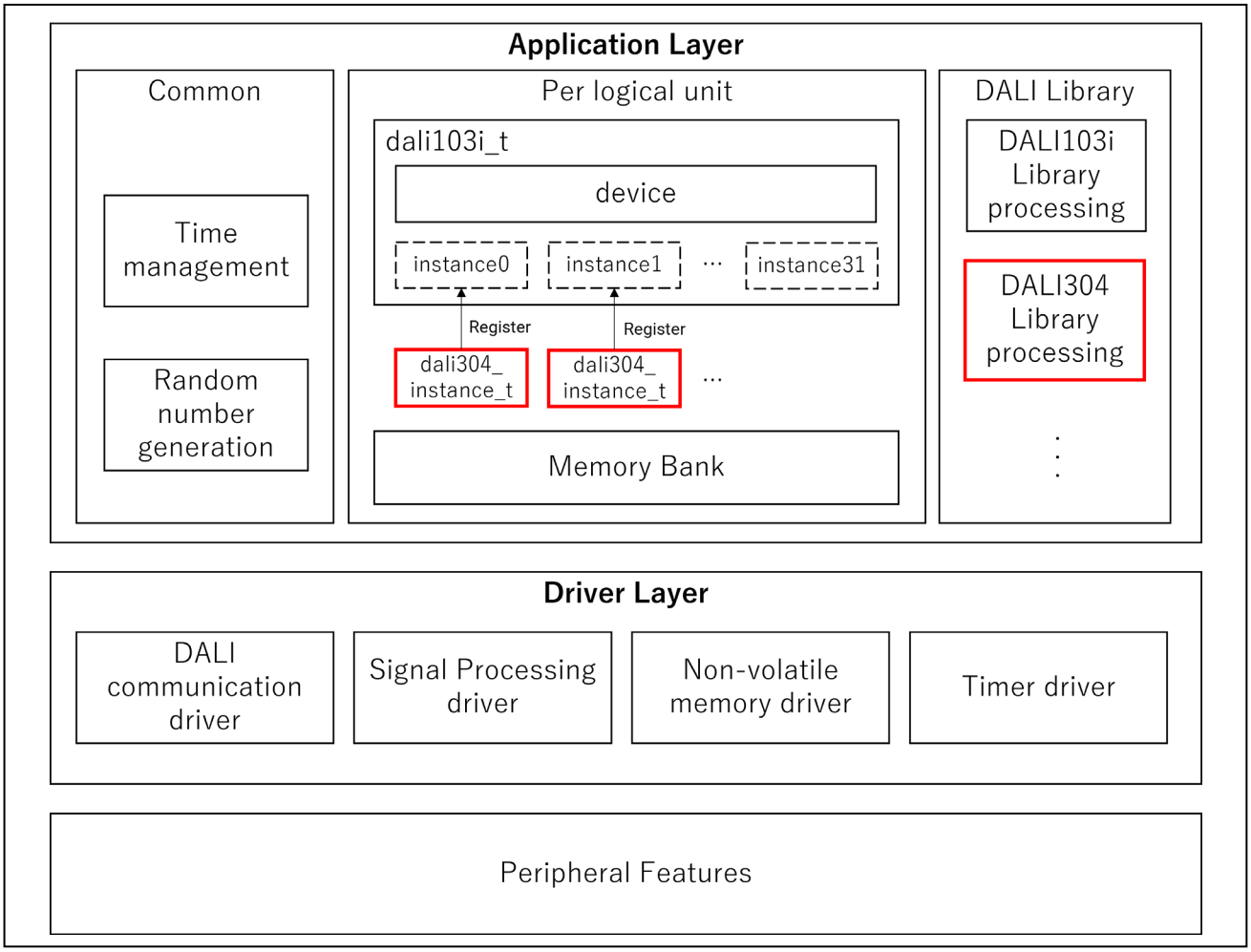
User creation processing	Library processing
・Light Sensor Input control ・Light Sensor Abnormality detection	・Received 24-bit Forward Frame processing ・Transmitted Backward Frame issuance ・Transmitted Event Message Frame issuance ・Timing control ・DALI variable manipulation

This library provides Instance Type 4 instances that can be registered to logical units defined using the DALI103i library.

1.2 Software configuration

The software configuration of the Input Device when using this library is shown below.
The part surrounded by the red line is this library. This library is assumed to be extended to the DALI103i library.

Figure 1-1 Input Device software configuration diagram



1.3 Supported standard

The standards and compiler environments supported by this library are as follows.

Table 1-2 Supported standard and library name

Supported standard	Compiler	Library name
IEC62386-304 Edition 1.0	Renesas CC-RL V1.11.00	r_dali_304_cc_gen2_v1_00.lib
	IAR C/C++ Compiler for Renesas RL78 V4.21.4	r_dali_304_iar_gen2_v1_00.a

1.4 File list

The list of files provided by this library is described below.

Table 1-3 File list

File name	Description
r_dali_304_cc_gen2_v1_00.lib r_dali_304_iar_gen2_v1_00.a	CC-RL version library file IAR version library file
r_dali304_api.h	Library header file
r_dali304_ivar.h	Definition header file for instance variable module
r_dali304_common.h	Definition header files used in multiple modules

1.5 Resource

The library resources required by this library are listed below.
The resources that do not depend on the implementation of the Input Device are listed in Table 1-4 Library resource (Fixed), and the resources that depend on the implementation of the Input Device are listed in Table 1-5 Library resource (Variable).

Table 1-4 Library resource (Fixed)

Compiler	Item		Size
CC-RL	Library resource	ROM size	3,394 [byte]
		RAM size	0 [byte]
	Maximum stack size		98 [byte] (R_DALI304_SetInputSignal)
IAR	Library resource	ROM size	3,775 [byte]
		RAM size	0 [byte]
	Maximum stack size		122 [byte] (R_DALI304_SetInputSignal)

Table 1-5 Library resource (Variable)

Compiler	Item	RAM Size
CC-RL	dali304_instance_t	214 [byte / instance]
IAR	dali304_instance_t	214 [byte / instance]

1.6 Development environment

The environment when developing this library is described below.

Table 1-6 Library development environment

Compiler	Item	Description
CC-RL	Integrated development environment	e2studio V2022-04
	C compiler	Renesas CC-RL V1.11.00
	CPU core	RL78-S2/S3 core
	Optimization level	Priority to size
	Language standard	GNU ISO C99
IAR	Integrated development environment	IAR Embedded Workbench for Renesas RL78 V4.21.4
	C compiler	IAR C/C++ Compiler For Renesas RL78 V4.21.4
	CPU core	RL78-S3 core
	Optimization level	Priority to size
	Language standard	GNU ISO C99

1.7 Notes

1. The API functions in this library are prohibited from being called by the interrupt handler in the user application.
2. Ensure that the loop processing of the program containing this library can run for a maximum of less than 1 ms. Under an environment where loop processing runs for more than 1ms, it will not meet the DALI standard specifications.
3. The dali304_instance_t type structure is a reference-only structure.

2.Programming environment

This section describes the hardware and software environments required for users to perform Input Device operation using this library.
Note that only the requirements that are necessary in addition to those in the DALI103i library are described.

2.1 Hardware requirement

2.1.1 Light Sensor

The instance type 4 instance must apply a Light Sensor (an input device that provides illuminance level information to the lighting control system through light intensity sensing) as a signal processor.

2.1.2 Failure detection mechanism

An instance of instance type 4 must detect operational anomalies, store the status in an internally stored variable, and respond to queries from the Application Controller. Therefore, a hardware-based anomaly detection mechanism is required. In addition to physical sensor failure detection, up to four types of manufacturer-specific anomaly detection can be defined. These error detection methods are manufacturer-dependent.
This feature is optional.

2.2 Software requirement

2.2.1 DALI304 Instance module definition

The DALI standard allows for the implementation of 1 to 32 instances (signal processors) per Input Device, depending on the number required. This library provides a structure (dali304_instance_t) that contains the parameters necessary to configure an instance of instance type 4. The variables of type dali304_instance_t are called DALI304 instance modules.
Define the required number of DALI304 instance modules and register them in the DALI103i module.

2.2.2 Light Sensor Driver

Implement a driver to get the illuminance level from the light sensor that serves as the input device.

2.2.3 Failure notification

When a failure occurs or is resolved by the failure detection mechanism described in the hardware requirements, call the following API function.
This feature is optional.

- Occurred failure related to instance of instance type 4
  R_DALI304_AddInstanceByteError function
- Resolves failure related to instance of instance type 4
  R_DALI304_RemoveInstanceByteError function

3.DALI304 library feature

The features of this library are described below.

3.1 Definition of data types and return values

The data types provided by this library are described below.

Table 3-1 List of data types

Type	Description
dali304_instance_t	DALI304 instance module type

The definition macros provided by this library are described below.

Table 3-2 List of instance error

Macro name	Macro value	Description
DALI304_ERRBYTE _PHYSICAL_SENSOR_FAILURE	0x01	physical sensor failure bit
DALI304_ERRBYTE _MANUFACTURER_SPECIFIC_ERROR_1	0x10	manufacturer specific error 1bit
DALI304_ERRBYTE _MANUFACTURER_SPECIFIC_ERROR_2	0x20	manufacturer specific error 2 bit
DALI304_ERRBYTE _MANUFACTURER_SPECIFIC_ERROR_3	0x40	manufacturer specific error 3 bit
DALI304_ERRBYTE _MANUFACTURER_SPECIFIC_ERROR_4	0x80	manufacturer specific error 4 bit

The return values provided by this library are listed below.

Table 3-3 List of return values (dali304_return_t)

Definition	Return value	Description
DALI304_RETURN_OK	0	Normal end
DALI304_RETURN_ERR	-1	Error end

3.2 List of structures

The structures provided by this library are described below.

Definition of instance NVM type structure (dali304_instance_nvm_t)

```
typedef struct
{
    dali103i_instance_nvm_t base;
    dali304_ivar_nvm_t add;
} dali304_instance_nvm_t;
```

Definition of instance default type structure (dali304_instance_default_t)

```
typedef struct
{
    uint8_t resolution;
} dali304_instance_default_t;
```

3.3 List of API Functions

The API functions of this library are described below.

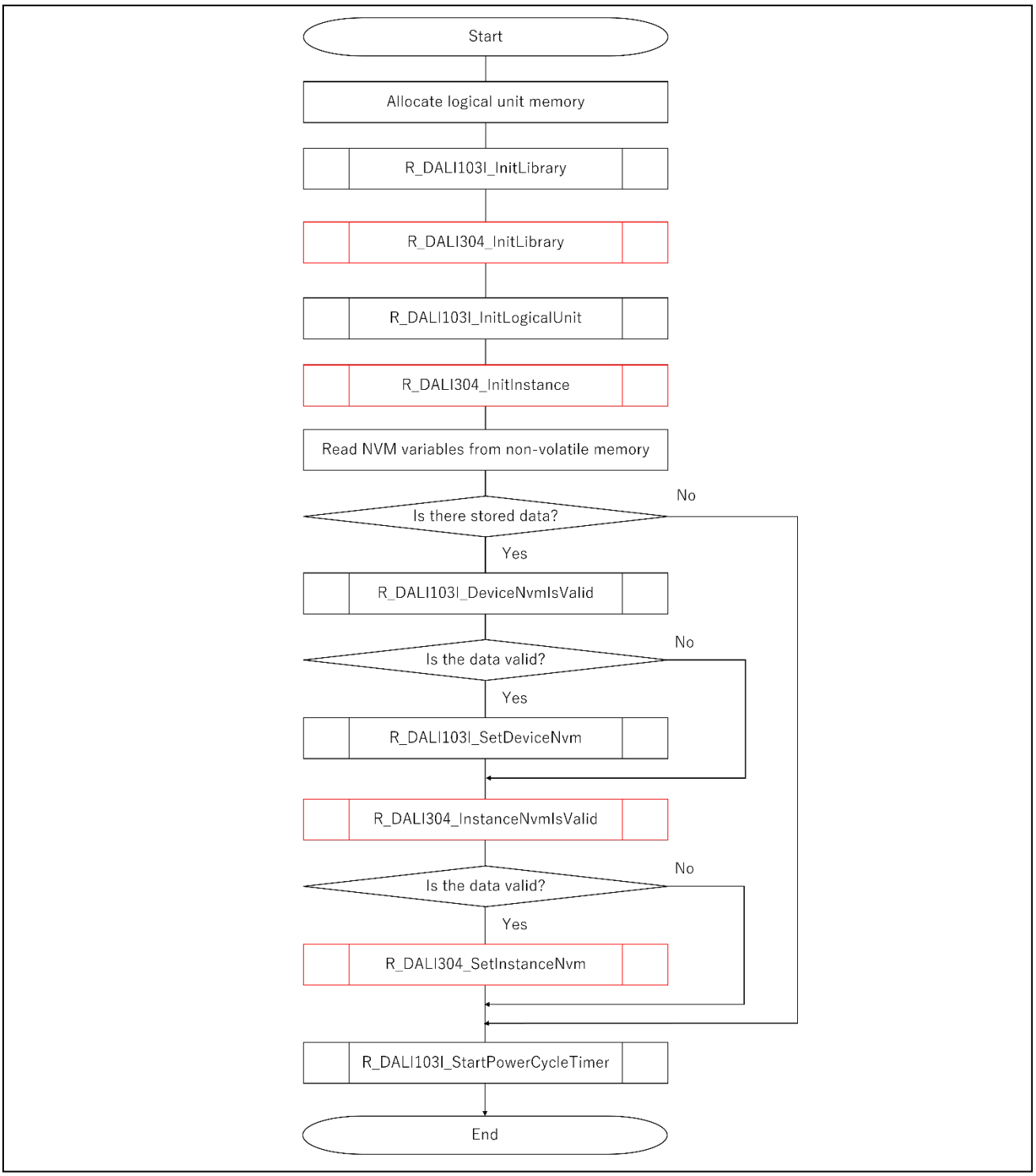
Table 3-4 List of API functions

Function name	Description
R_DALI304_InitLibrary	Initialize the DALI304 library
R_DALI304_InitInstance	initialize instance
R_DALI304_InstanceNvmIsValid	Check within the valid range of instance NVM variable values
R_DALI304_SetInstanceNvm	Set the instance NVM variable value
R_DALI304_GetInstanceNvm	Get instance NVM Variable Value
R_DALI304_InstanceNvmIsChanged	Check for instance NVM variable value change
R_DALI304_InstanceIsActive	Check the status of instanceActive
R_DALI304_EncodeToInputSignal	Encode acquired sensor values to Input Signal values
R_DALI304_SetInputSignal	Set input signal
R_DALI304_GetInputNotification	Get input notification events
R_DALI304_AddInstanceErrorByte	Add instance error
R_DALI304_RemoveInstanceErrorByte	Remove instance error
R_DALI304_GetInstanceErrorByte	Get instanceErrorByte variable value
R_DALI304_GetLibraryVersion	Get the value of the detectionRange variable

3.4 Schematic flowchart

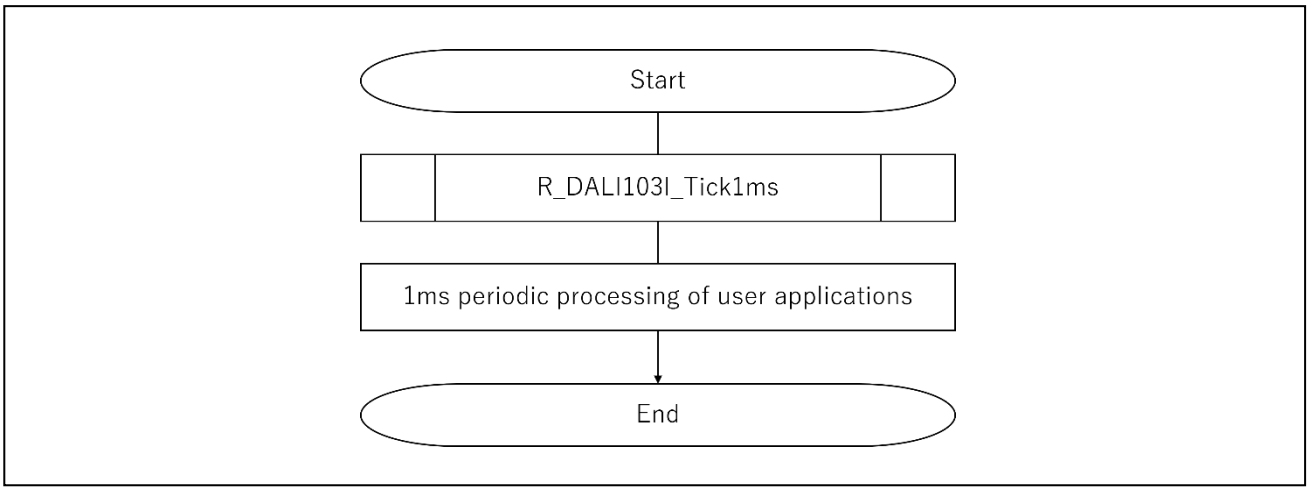
3.4.1 Initialization

The initialization flow is described below. The functions circled in red are those provided by this library.



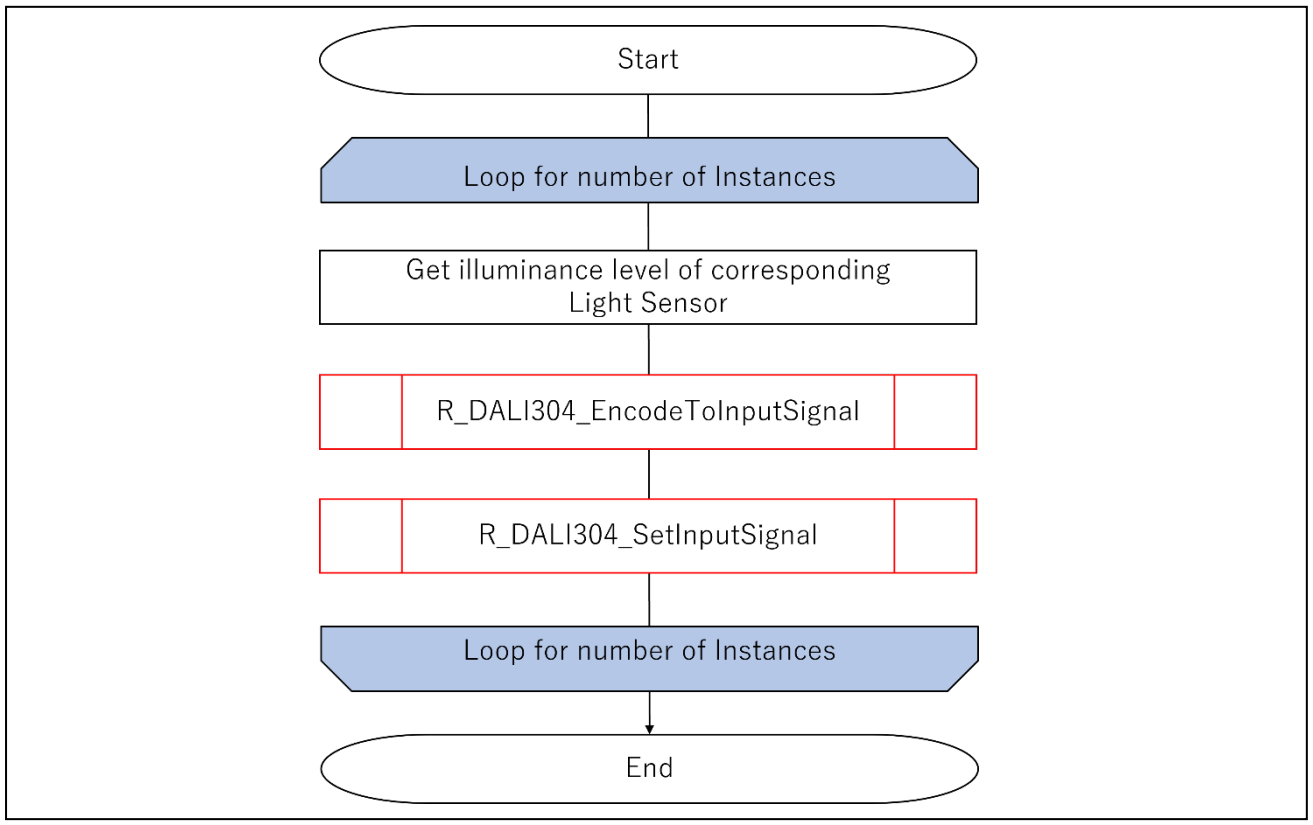
3.4.2 1ms periodic processing

This section describes the flow of 1ms periodic processing. This process should be processed at 1ms intervals.



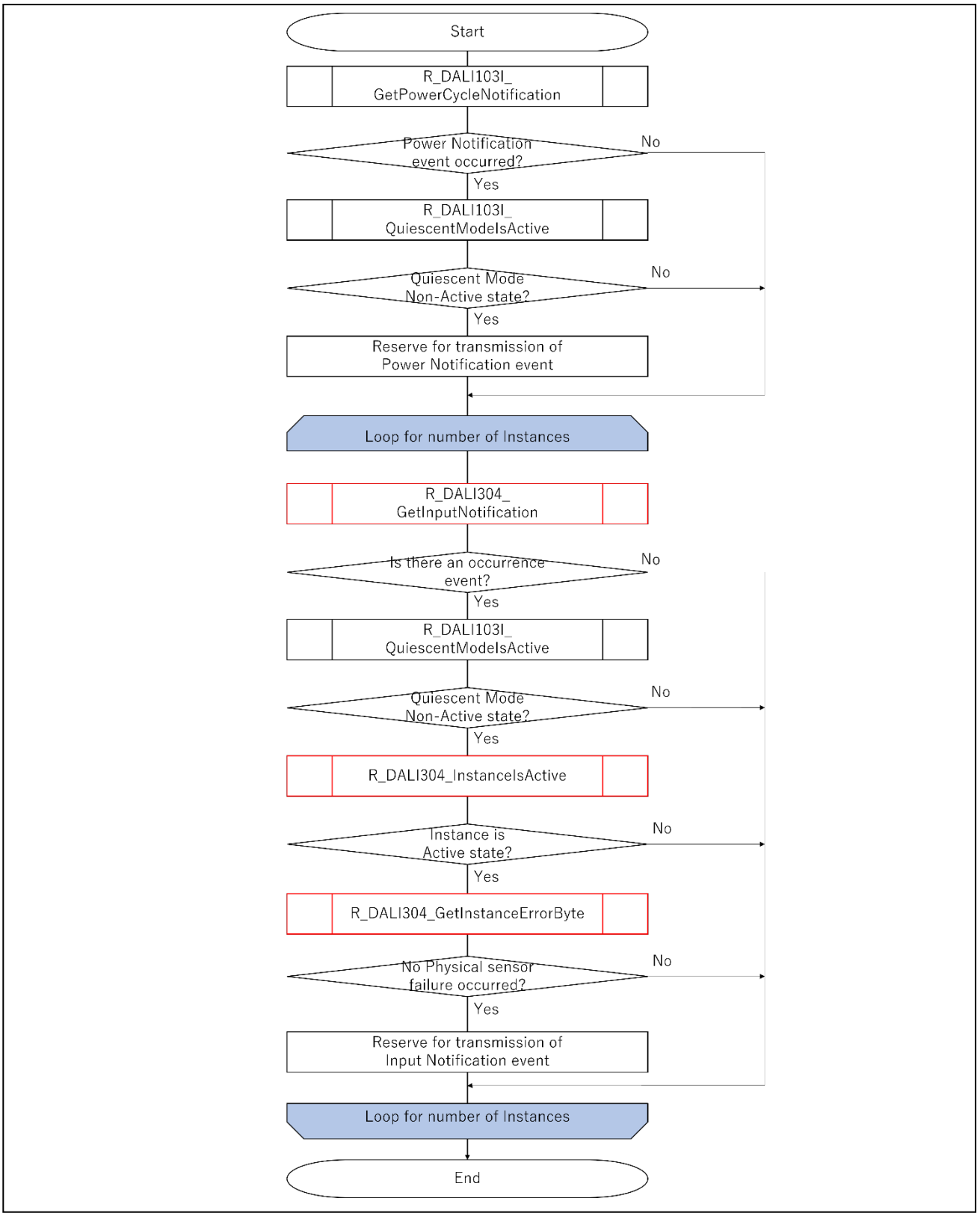
3.4.3 Input Signal update processing

The flow of Input Signal update is described below. The functions circled in red are those provided by this library.



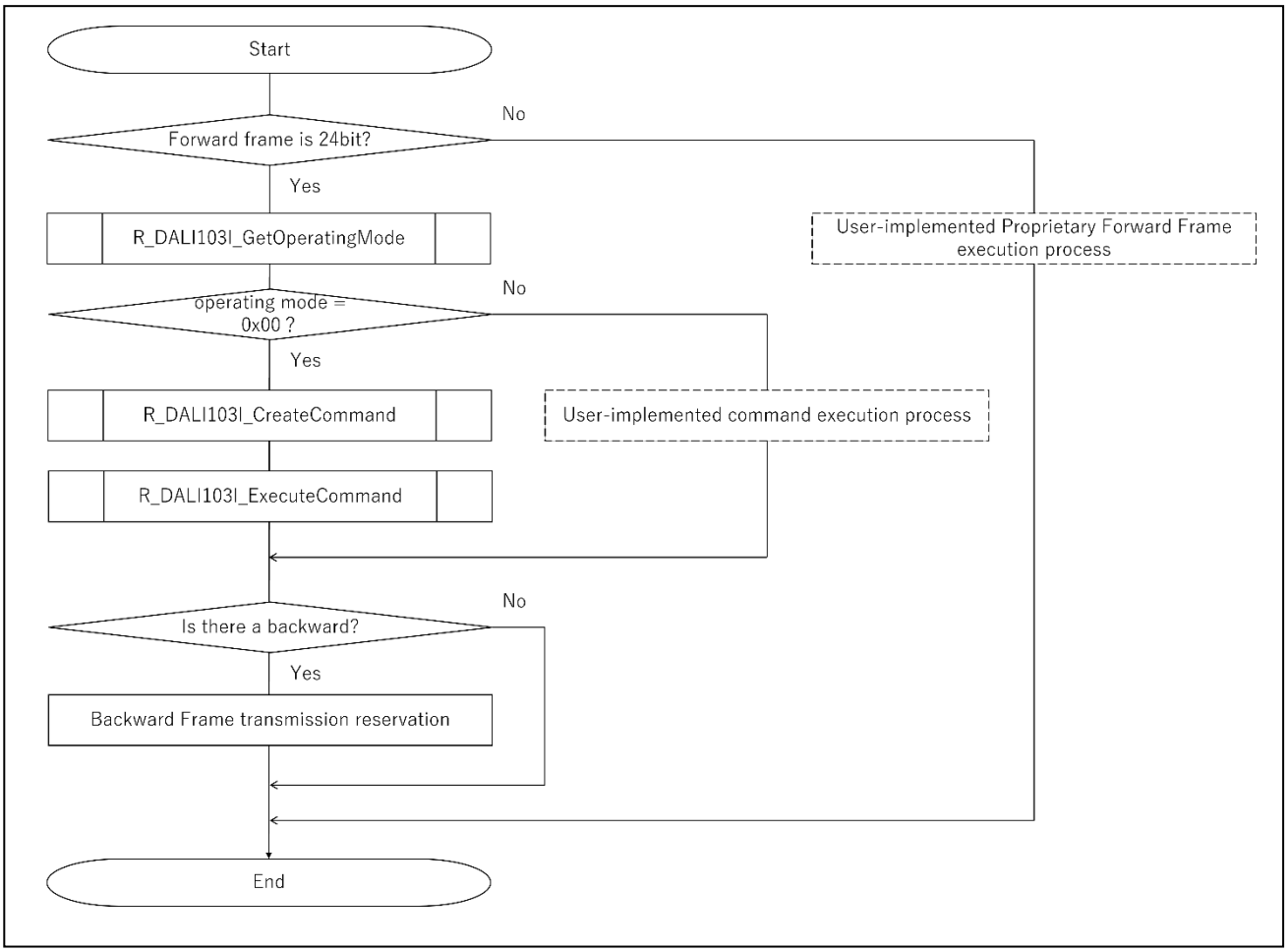
3.4.4 Event Message processing

The flow of Event Message processing is described below. The functions circled in red are those provided by this library.



3.4.5 Receiving Forward Frame

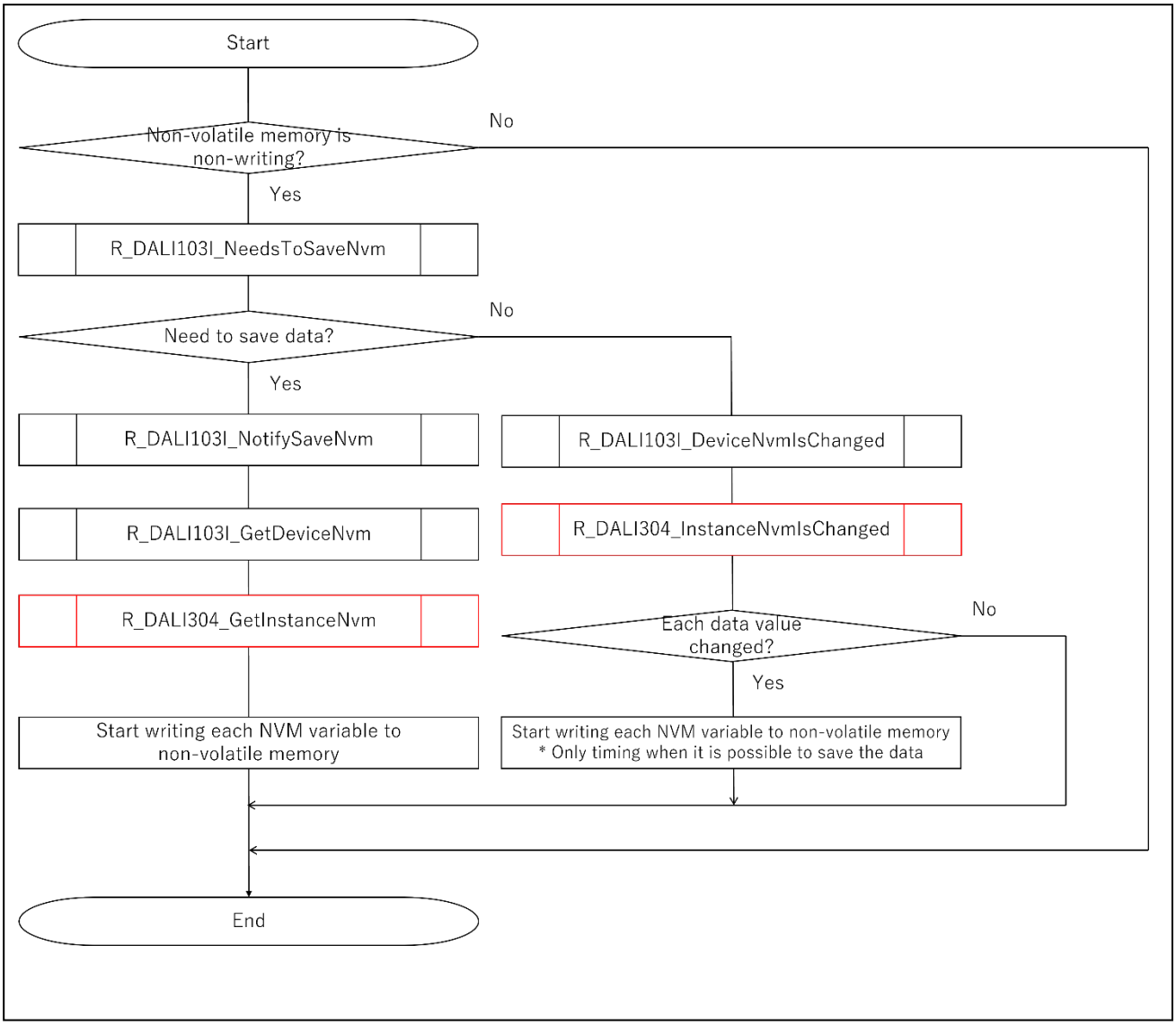
The following describes the flow of processing when a Forward Frame is received. The processing should be performed when a forward frame is received by the DALI communication bus.
The processing for Proprietary Forward Frames (more than 16 bits and other than 20-bit, 24-bit, and 32-bit Forward Frames) is an optional feature and should be implemented if the DALI communication driver and the application support it.
The operating mode other than 0 is also an optional feature. Register the mode number with the R_DALI103I_InitLogicalUnit function after implementation if an original mode is required.



3.4.6 Non-volatile data processing

This section describes the flow of non-volatile data processing.

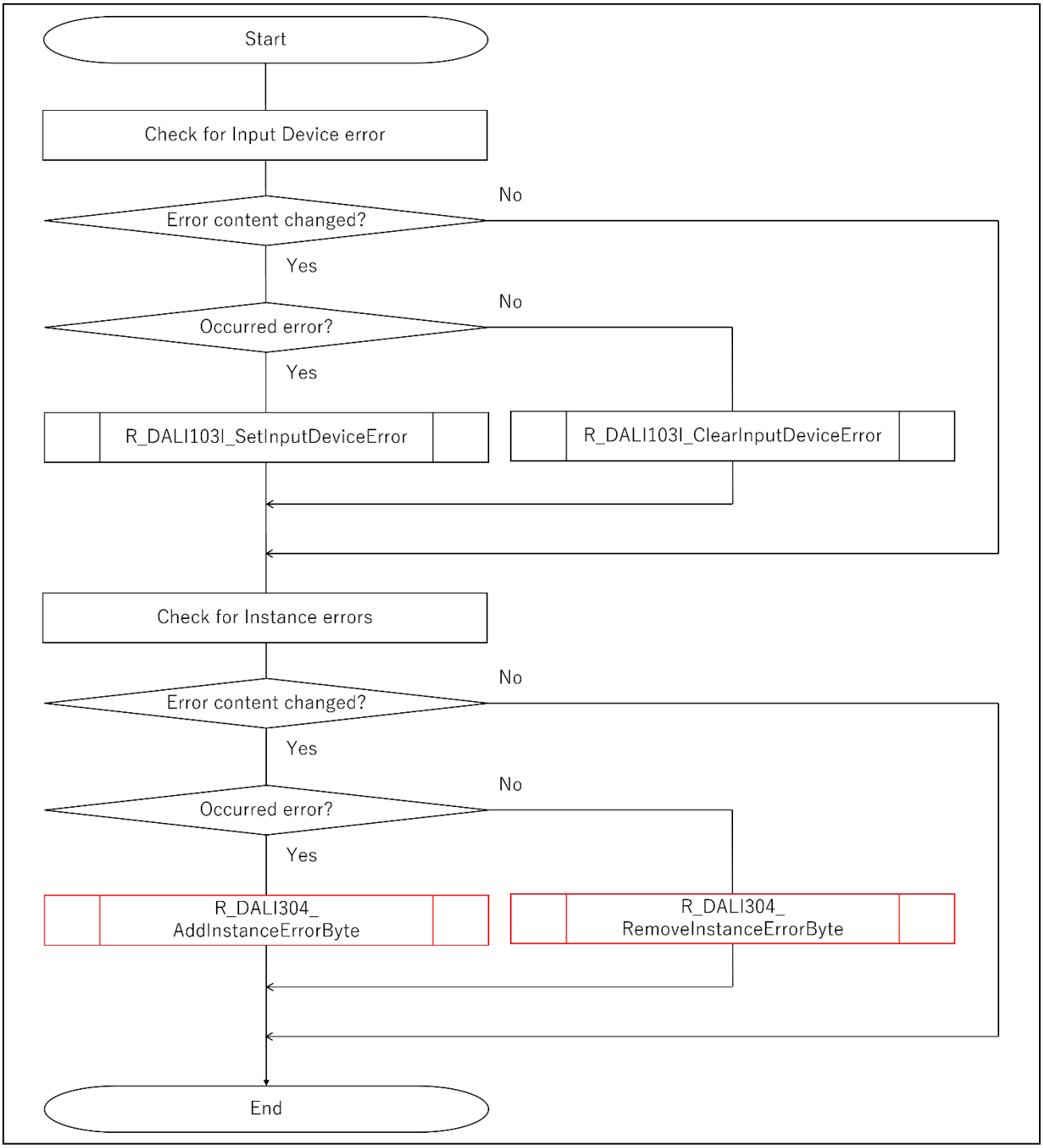
It is specified that saving to non-volatile memory is to be completed within 300 ms after the SAVE PERSISTENT VARIABLES command is received. It is also specified that even if the SAVE PERSISTANT VARIABLES command is not received, if there is a change in the NVM variable value, it must be saved within 30 seconds. Please check periodically and perform processing to ensure that the saving is completed within the specified time. The functions circled in red are those provided by this library.



3.4.7 Error handling

This section describes the flow of the error handling process. Call this function when the error status is updated.

Detailed specifications for Input Device error and Instance error depend on the hardware and software. Please define the specifications and implement them according to the environment. The functions circled in red are the functions provided by this library.



3.5 API Function Specifications

The API function specifications for this library are listed below.

3.5.1 R_DALI304_InitLibrary

[Overview]

It Initializes the DALI304 library.

[Format]

dali304_return_t R_DALI304_InitLibrary(void)

[Prerequisite]

1. R_DALI103I_InitLibrary must have ended normally.

[Arguments]

None

[Return values]

Value	Description
DALI304_RETURN_OK	Normal end
DALI304_RETURN_ERR	Parameter error

3.5.2 R_DALI304_InitInstance

[Overview]

It Initializes the DALI304 instance module and register the instance in the DALI103i module (type dali103i_t). Type dali304_instance_t provides an instance of InstanceType 4.

[Format]

```
dali304_return_t R_DALI304_InitInstance(dali103i_t * p_this,
                                        dali304_instance_t * p_instance,
                                        const dali304_instance_default_t * p_default_value)
```

[Prerequisite]

1. R_DALI103I_InitLibrary function must have ended normally.
2. R_DALI304_InitLibrary function must have ended normally.
3. R_DALI103I_InitLogicalUnit function must have ended normally.

[Arguments]

Argument	Description
dali103i_t * p_this	Pointer to DALI103i module
dali304_instance_t * p_instance	Pointer to DALI304 instance module
const dali304_instance_default_t * p_default_value	User-defined default value Valid range - resolution : 1 - 255

[Return values]

Value	Description
DALI304_RETURN_OK	Normal end
DALI304_RETURN_ERR	Parameter error - Review the argument settings.

3.5.3 R_DALI304_InstanceNvmIsValid

[Overview]

It returns whether all the values set to the members of the dali304_instance_nvm_t type variable are within the valid range.

Be sure to call and check the R_DALI304_SetInstanceNvm function described below before setting values.

[Format]

```
bool R_DALI304_InstanceNvmIsValid(const dali304_instance_t * p_this,
                                  const dali304_instance_nvm_t * p_nvm)
```

[Prerequisite]

1. R_DALI103I_InitLibrary function must have ended normally.
2. R_DALI304_InitLibrary function must have ended normally.
3. R_DALI103I_InitLogicalUnit function must have ended normally.
4. R_DALI304_InitInstance function must have ended normally.

[Arguments]

Argument	Description
const dali304_instance_t * p_this	Pointer to DALI304 instance module
const dali304_instance_nvm_t * p_nvm	Pointer to DALI304 instance NVM variable Valid range: - base.instance_group0：0x00 - 0x1F, 0xFF - base.instance_group1：0x00 - 0x1F, 0xFF - base.instance_group2：0x00 - 0x1F, 0xFF - base.instance_active：true, false - base.event_filter：0x00000000 - 0x000000FF - base.event_scheme：0x00 - 0x04 - base.event_priority：0x02 - 0x05 - add. t_deadtime：0x00 - 0xFF - add. t_report：0x00 - 0xFF - add. hysteresis_min：0x00 - 0xFF - add. hysteresis：0 - 25

[Return values]

Value	Description
true	All variables are in the valid range
false	At least one variable is outside the valid range

3.5.4 R_DALI304_SetInstanceNvm

[Overview]

It sets the instance NVM variable value in the DALI304 instance module.
Use to set the read data when the data of the instance NVM variable is saved in non-volatile memory at power-on.

[Format]

```
void R_DALI304_SetInstanceNvm(dali304_instance_t * p_this,
                              const dali304_instance_nvm_t * p_nvm)
```

[Prerequisite]

1. R_DALI103I_InitLibrary function must have ended normally.
2. R_DALI304_InitLibrary function must have ended normally.
3. R_DALI103I_InitLogicalUnit function must have ended normally.
4. R_DALI304_InitInstance function must have ended normally.
5. Be sure that the instance NVM variable is within the valid range with the R_DALI304_InstanceNvmIsValid function.

[Arguments]

Argument	Description
dali304_instance_t * p_this	Pointer to DALI304 instance module
const dali304_instance_nvm_t * p_nvm	Pointer to DALI304 instance NVM variable

[Return values]

None

3.5.5 R_DALI304_GetInstanceNvm

[Overview]

It gets the instance NVM variable setting values from the DALI304 instance module.
Use to save the latest instance NVM variable values to non-volatile memory.

[Format]

```
void R_DALI304_GetInstanceNvm(const dali304_instance_t * p_this,
                              dali304_instance_nvm_t * p_nvm)
```

[Prerequisite]

1. R_DALI103I_InitLibrary function must have ended normally.
2. R_DALI304_InitLibrary function must have ended normally.
3. R_DALI103I_InitLogicalUnit function must have ended normally.
4. R_DALI304_InitInstance function must have ended normally.
5. The power cycle notification timer must have been started at R_DALI103I_StartPowerCycleTimer function.

[Arguments]

Argument	Description
const dali304_instance_t * p_this	Pointer to DALI304 instance module
dali304_instance_nvm_t * p_nvm	Pointer to DALI304 instance NVM variable

[Return values]

None

3.5.6 R_DALI304_InstanceNvmIsChanged

[Overview]

Get whether at least one instance NVM variable value has changed.
If the return value of this function is true, the instance NVM variable should be saved in non-volatile memory according to the hardware status.
The status that can be obtained by this function is the status from the last time this function was called (at startup for the first call). Note that consecutive calls will return false.

[Format]

```
bool R_DALI304_InstanceNvmIsChanged(dali304_instance_t * p_this)
```

[Prerequisite]

1. R_DALI103I_InitLibrary function must have ended normally.
2. R_DALI304_InitLibrary function must have ended normally.
3. R_DALI103I_InitLogicalUnit function must have ended normally.
4. R_DALI304_InitInstance function must have ended normally.
5. The power cycle notification timer must have been started at R_DALI103I_StartPowerCycleTimer function.

[Arguments]

Argument	Description
dali304_instance_t * p_this	Pointer to DALI304 instance module

[Return values]

Value	Description
true	Value changed
false	Value not changed

3.5.7 R_DALI304_InstanceIsActive

[Overview]

It gets whether the specified DALI304 instance module is "Active" or not.

If the return value of this function is false, the input notification event cannot be sent.

[Format]

bool R_DALI304_InstanceIsActive(const dali304_instance_t * p_this)

[Prerequisite]

1. R_DALI103I_InitLibrary function must have ended normally.
2. R_DALI304_InitLibrary function must have ended normally.
3. R_DALI103I_InitLogicalUnit function must have ended normally.
4. R_DALI304_InitInstance function must have ended normally.
5. The power cycle notification timer must have been started at R_DALI103I_StartPowerCycleTimer function.

[Arguments]

Argument	Description
const dali304_instance_t	Pointer to DALI304 instance module

[Return values]

Value	Description
true	instance is "Active"
false	instance is not "Active"

3.5.8 R_DALI304_EncodeToInputSignal

[Overview]
It encodes the sensor input value (0-$2^{resolution}$ -1) to the Input Signal value (0-$2^{resolution}$ -2) considering the MASK value.

[Format]

```
void R_DALI304_EncodeToInputSignal(dali304_instance_t * p_this,
                                   const uint8_t * p_input_level, uint8_t * p_signal);
```

[Prerequisite]

1. R_DALI103I_InitLibrary function must have ended normally.
2. R_DALI304_InitLibrary function must have ended normally.
3. R_DALI103I_InitLogicalUnit function must have ended normally.
4. R_DALI304_InitInstance function must have ended normally.
5. The power cycle notification timer must have been started at R_DALI103I_StartPowerCycleTimer function.

[Arguments]

Argument	Description
const dali304_instance_t	Pointer to DALI304 instance module
const uint8_t * p_input_level	Input information to perform the conversion (Pointer to array variable where sensor input values is stored)
uint8_t * p_signal	Destination for storing the conversion result (Pointer to array variable where the Input Signal value is stored)

[Return values]
None

(1) Encoding of InputSignal

The maximum value of InputSignal ($2^{resolution}$ - 1) is treated as a MASK value for instance type 4 instances, which means that the sensor value is in an indeterminate state (between startup and initial sensor value acquisition or in case of sensor failure).

Therefore, if a valid sensor value is acquired, the InputSinal value must be set within the range of 0 to $2^{resolution}$ - 2.

By using the R_DALI304_EncodeToInputSignal function, the acquired sensor value (0 to $2^{resolution}$ - 1) can be encoded into a valid InputSinal value (0 to $2^{resolution}$ - 2).

(2) Setting p_input_level and p_signal parameters

The p_input_level parameter should be set to the first pointer of a uint8_t type array.

The following requirements must be met

- The number of array elements should be the number of the instance's resolution/8 rounded up to the nearest integer.
- The array variable should be set to the acquired sensor value in the range of 0 to $2^{resolution}$ - 1.
- Sensor values for arrays should be stored little-endian and LSB-packed.

The p_signal parameter should be set to the first pointer of a uint8_t type array.

The requirements are as follows

- The number of elements in the array should be the resolution/8 of the instance rounded up to the nearest integer.

An example of array settings is shown below.

e.g.1) When resolution = 3 and the acquired sensor value is 0x07

```
uint8_t input_level[1] = { 0x07 };
uint8_t input_signal[1];

R_DALI304_EncodeToInputSignal(&dali304_instance, input_level, input_signal);
```

By setting the first pointer of the above array as a parameter of the R_DALI304_EncodeToInputSignal function, the encoding result { 0x06 } is stored in the input_signal array.

e.g.2) When resolution = 10 and the acquired sensor value is 0x2F5

```
uint8_t input_level[2] = { 0xF5, 0x02 };
uint8_t input_signal[2];

R_DALI304_EncodeToInputSignal(&dali304_instance, input_level, input_signal);
```

By setting the first pointer of the above array as a parameter of the R_DALI304_EncodeToInputSignal function, the encoding result { 0xF4, 0x02 } is stored in the input_signal array.

3.5.9 R_DALI304_SetInputSignal

[Overview]

It sets the input signal to the specified DALI304 instance module.

Set the illuminance level of the Light Sensor corresponding to the instance as needed.

[Format]

```
void R_DALI304_SetInputSignal(dali304_instance_t * p_this,
                              uint8_t * p_signal)
```

[Prerequisite]

6. R_DALI103I_InitLibrary function must have ended normally.
7. R_DALI304_InitLibrary function must have ended normally.
8. R_DALI103I_InitLogicalUnit function must have ended normally.
9. R_DALI304_InitInstance function must have ended normally.
10. The power cycle notification timer must have been started at R_DALI103I_StartPowerCycleTimer function.

[Arguments]

Argument	Description
dali304_instance_t * p_this	Pointer to DALI304 module
uint8_t * p_signal	Pointer to input signal array

[Return values]

None

3.5.10 R_DALI304_GetInputNotification

[Overview]

It gets the input notification event for the specified DALI304 instance module.

Send the input notification event acquired by this function in the time according to the priority setting.

[Format]

```
dali103i_event_t R_DALI304_GetInputNotification(dali103i_t * p_this,
                                                dali304_instance_t * p_instance)
```

[Prerequisite]

1. R_DALI103I_InitLibrary function must have ended normally.
2. R_DALI304_InitLibrary function must have ended normally.
3. R_DALI103I_InitLogicalUnit function must have ended normally.
4. R_DALI304_InitInstance function must have ended normally.
5. The power cycle notification timer must have been started at R_DALI103I_StartPowerCycleTimer function.

[Arguments]

Argument	Description
dali103i_t * p_this	Pointer to DALI103i module
dali304_instance_t * p_instance	Pointer to DALI304 instance module

[Return values]

Member	Description
bool is_exist	Whether or not an event exists
dali103i_forward_frame_t frame	Stores input notification events when is_exist=true

3.5.11 R_DALI304_AddInstanceErrorByte

[Overview]

It sets the specified error for the specified DALI304 instance module additionally to the instanceErrorByte. When a specific error occurs, call it using the corresponding macro.

[Format]

```
void R_DALI304_AddInstanceErrorByte(dali304_instance_t * p_this,
                                    uint8_t error)
```

[Prerequisite]

1. R_DALI103I_InitLibrary function must have ended normally.
2. R_DALI304_InitLibrary function must have ended normally.
3. R_DALI103I_InitLogicalUnit function must have ended normally.
4. R_DALI304_InitInstance function must have ended normally.
5. The power cycle notification timer must have been started by the R_DALI103I_StartPowerCycleTimer function.

[Arguments]

Argument	Description
dali304_instance_t * p_this	Pointer to DALI304 instance module
uint8_t error	Additional setting values for instance error Valid range - DALI304_ERRBYTE_MANUFACTURER_SPECIFIC_ERROR_1 - DALI304_ERRBYTE_MANUFACTURER_SPECIFIC_ERROR_2 - DALI304_ERRBYTE_MANUFACTURER_SPECIFIC_ERROR_3 - DALI304_ERRBYTE_MANUFACTURER_SPECIFIC_ERROR_4 * Multiple specifications are possible by specifying OR for the above macros

[Return values]

None

3.5.12 R_DALI304_RemoveInstanceErrorByte

[Overview]

It sets the specified error for the specified DALI304 instance module to be removed from the instanceErrorByte.

When the specified error is resolved, call this function using the corresponding macro.

[Format]

```
void R_DALI304_RemoveInstanceErrorByte(dali304_instance_t * p_this,
                                       uint8_t error)
```

[Prerequisite]

1. R_DALI103I_InitLibrary function must have ended normally.
2. R_DALI304_InitLibrary function must have ended normally.
3. R_DALI103I_InitLogicalUnit function must have ended normally.
4. R_DALI304_InitInstance function must have ended normally.
5. The power cycle notification timer must have been started by the R_DALI103I_StartPowerCycleTimer function.

[Arguments]

Argument	Description
dali304_instance_t * p_this	Pointer to DALI304 instance module
uint8_t error	Setting value for removal of instance error Valid range - DALI304_ERRBYTE_MANUFACTURER_SPECIFIC_ERROR_1 - DALI304_ERRBYTE_MANUFACTURER_SPECIFIC_ERROR_2 - DALI304_ERRBYTE_MANUFACTURER_SPECIFIC_ERROR_3 - DALI304_ERRBYTE_MANUFACTURER_SPECIFIC_ERROR_4 * Multiple specifications are possible by specifying OR for the above macros

[Return values]

None

3.5.13 R_DALI304_GetInstanceErrorByte

[Overview]

It gets the instanceErrorByte setting value.

[Format]

uint8_t R_DALI304_GetInstanceErrorByte(const dali304_instance_t * p_this)

[Prerequisite]

1. R_DALI103I_InitLibrary function must have ended normally.
2. R_DALI304_InitLibrary function must have ended normally.
3. R_DALI103I_InitLogicalUnit function must have ended normally.
4. R_DALI304_InitInstance function must have ended normally.
5. The power cycle notification timer must have been started by the R_DALI103I_StartPowerCycleTimer function.

[Arguments]

Argument	Description
const dali304_instance_t * p_this	Pointer to DALI304 module

[Return values]

Value	Description
uint8_t	instanceErrorByte setting value

3.5.14 R_DALI304_GetLibraryVersion

[Overview]

It gets the version number of this library.

[Format]

uint16_t R_DALI304_GetLibraryVersion(void)

[Prerequisite]

None

[Arguments]

None

[Return values]

Value	Description
uint16_t	Version number (format: 0xXXYY) XX: Major version YY: Minor version

Revision History	RL78 Family DALI-2 Input Device Library User's Manual: Light Sensor (304)

Rev.	Date	Description	
		Page	Summary
1.00	Oct.04.23	—	First Edition issued

RL78 Family DALI-2 Input Device Library
User's Manual: Light Sensor (304)

Publication Date:Rev.1.00 Oct.04.23

Published by: Renesas Electronics Corporation

RL78 Family

Renesas Electronics Corporation

R01US0652EJ0100