User's Manual

RL78 Family

DALI-2 Input Device Library
User's Manual: Basic (103)

16-bit single chip microprocessor

Renesas Electronics
www.renesas.com

Rev.1.00 Nov. 2022

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

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/.

Trademarks

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

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 Intended Readers

This manual is intended for users who develop an Input Device of the DALI system by using the RL78 MCU. Readers of this manual are required to have a basic knowledge of electric circuits, logic circuits, and microcomputers.
This manual is mainly organized into the product overview, product specifications, and notes on usage.

Sufficiently confirm the precautions before using this library. Precautions are provided in the body of each chapter, at the end of each chapter, and the chapter of precautions.

Revision History is a summary of major corrections and additions made in the previous versions. Not all revisions are contained in Revision History. For details, refer to this manual.

The following documents are provided for the DALI library. Use the latest version of documents. The latest version is available on the website of Renesas Electronics.

Document type	Contents	Document title	Document number
User's Manual Hardware	Hardware specifications (pin assignment, memory map, peripheral function specifications, electrical characteristics, and timing) and operation description Note: For details about how to use peripheral functions, see the application note.	RL78/G23 User's Manual Hardware	R01UH0896JJ0100
User's Manual Software	Description of CPU instruction sets	RL78 family User's Manual Software	R01US0015JJ0220
Application notes	How to use peripheral functions, and their application examples Reference programs How to create programs in C language	These documents are available on the website of Renesas Electronics.	
Renesas Technical Update	Flash reports on product specifications and documentation		

2. Abbreviations

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

Contents

RL78 Family Input Device Library

User's Manual: Basic (103)

1. DALI103i Library Overview

1.1 Overview of Library Functions

As the library for Input Devices in DALI communication, this library implements processing of the hardware-independent parts of the Input Device specifications in the IEC62386-103 standard (DALI103 or later).

The DALI standard defines both the hardware-related standard such as the communication timing and variable saving and the software-related standard such as processing for receiving a Forward Frame. This library is mainly responsible for processing during transmission and reception and timing processing. Refer to the application note and ensure that the standard hardware-related parts (hardware dependent parts) and power supply control comply with the DALI standard.

To use this library, you need to understand the DALI standard.

Table 1-1 Range of processing

User-created processing	Library processing
• Hardware setting • DALI communication driver • Timer control • Memory bank entity and control • Non-volatile memory access • Error detection • Instance status update	• Processing of received 24-bit Forward Frames • Issuing transmission Backward Frames • Timing control • DALI variable operations • Memory bank operations • Processing of transmission Event Message Frames

This library performs processing based on the 24-bit Forward Frame received via DALI communication. Because the library does not have a hardware-dependent part, a 24-bit Forward Frame received by the user application is notified to and processed in the library by calling API functions.

If the settings need to be changed in the application due to, for example, a command that specifies or acquires DALI variable settings, a notification is sent to the application as required.

This library can implement multiple logical units with a single device. It also supports implementation of multiple memory banks. The following shows the maximum number of supported logical units and the numbers of memory banks that can be implemented.

Table 1-2 Logical unit and memory bank specifications

Item	Value
Maximum number of logical units	64Note
Memory bank numbers that can be implemented	0 to 199

Note: A maximum of 64 Input Devices can be connected to a single DALI network for the DALI system. Considering the DALI network configuration and hardware limitations, specify the setting so that the sum total does not exceed 64.

1.2 Software Structure

The following describes the software structure of an Input Device when this library is used.

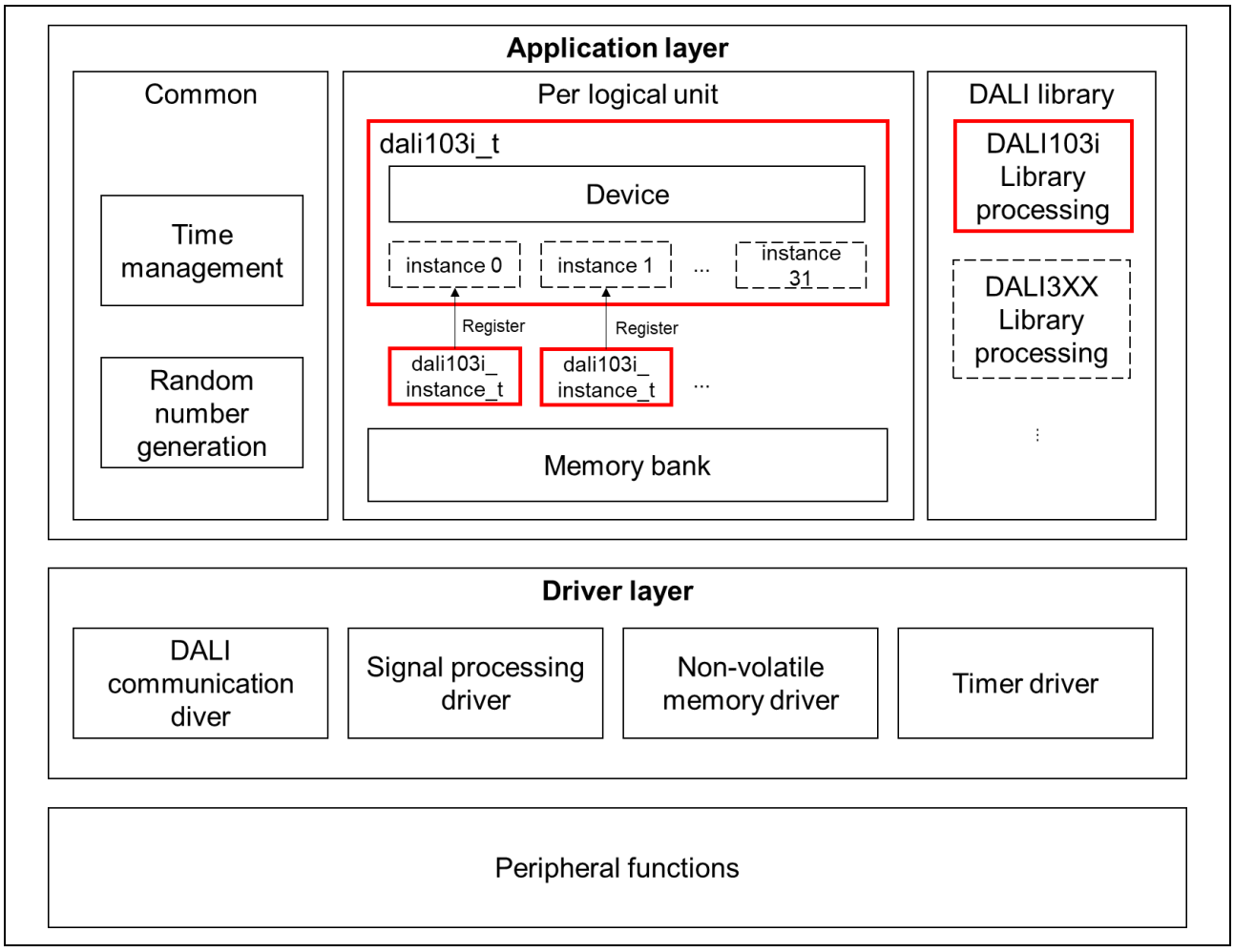
This library provides the parts in red frames. This library implements processing of the hardware-independent parts. This library works as a part of the application layer to perform processing such as DALI communication and access to non-volatile memory. Furthermore, this library can be extended by using the DALI3XX library configured based on the IEC62386-3XX standard specifications.

A dali103i_t type (details are described later) variable consists of logical unit parameters of the Input Device. This type of variable has one device entity in the Input Device and 32 sockets in which instance entities can be registered.

A dali103i_instance_t type (details are described later) variable has instance parameters that can be registered to the dali103i_t type.

The user can achieve a desired Input Device by implementing the required number of logical units and instances.

Figure 1-1 Input Device software structure diagram



1.3 Supported Standard

This library supports the following standard and compilers.

Table 1-3 Supported standard and library names

Supported Standard	Compiler	Library name
IEC62386-103 Edition 1.1	Renesas CC-RL V1.11.00	r_dali_103i_cc_gen2_v1_00.lib
	IAR C/C++ Compiler for Renesas RL78 V4.21.4	r_dali_103i_iar_gen2_v1_00.a

1.4 List of Files

The following shows a list of files supported by this library.

Table 1-4 List of files

File name	Description
r_dali_103i_cc_gen2_v1_00.lib r_dali_103i_iar_gen2_v1_00.a	CC-RL version library file IAR version library file
r_dali103i_api.h	Library header file
r_dali103i_dvar.h	Definition header file of the device variable module
r_dali103i_ivar.h	Definition header file of the instance variable module
r_dali103i_timer.h	Definition header file of the timer module
r_dali103i_list.h	Definition header file of the list module
r_dali103i_mb_if.h	Definition header file of the memory bank I/F module
r_dali103i_itx_if.h	Definition header file of the general-purpose instance type I/F module
r_dali103i_ftx_if.h	Definition header file of the general-purpose feature type I/F module
r_dali103i_device.h	Definition header file of the device module
r_dali103i_instance.h	Definition header file of the instance module
r_dali103i_common.h	Definition header file used by multiple modules

1.5 Resources

The following indicates the library resources required by this library.

Table 1-5 Library resources (fixed) shows the resources that do not depend on the implementation contents of the Input Device. Table 1-6 Library resources (variable) shows the resources that depend on the implementation contents of the Input Device.

Table 1-5 Library resources (fixed)

Compiler	Item		Size
CC-RL	Library resource	ROM size	13,432 [byte]
		RAM size	4 [bytes]
	Maximum stack size		74 [bytes] (R_DALI103I_ExecuteCommand)
IAR	Library resource	ROM size	16,088 [bytes]
		RAM size	4 [bytes]
	Maximum stack size		68 [bytes] (R_DALI103I_ExecuteCommand)

Table 1-6 Library resources (variable)

Compiler	Item	RAM size
CC-RL	dali103i_t	152 [bytes/logical unit]
	dali103i_instance_t	92 [bytes/instance]
IAR	dali103i_t	152 [bytes/logical unit]
	dali103i_instance_t	92 [bytes/instance]

1.6 Development Environment

The following describes the environments for developing this library.

Table 1-7 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	Size preferred
	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	Size preferred
	Language standard	GNU ISO C99

1.7 Precautions

1. It is prohibited to call an API function of this library from an interrupt handler in a user application.
2. Make sure that the maximum time required for loop processing for a program including this library is less than 1 ms. The DALI standard specifications are not satisfied in an environment where the loop processing time is 1 ms or more.
3. The dali103i_t type structure and dali103i_cmd_t type structure are provided only for reference.

2. Programming Environment

This chapter describes the hardware environment and software environment required for the user to perform Input Device operations by using this library.

2.1 Hardware Requirements

2.1.1 DALI communication circuit

Implementing DALI communication requires a communication circuit that performs operations defined in the IEC62386-101 standard.

2.1.2 Non-volatile area

The Input Device specifications state that data that is called NVM data is saved in a non-volatile area and the data value is retained after the power is turned on again. Therefore, you need to reserve the dedicated non-volatile area.

2.1.3 Error detection mechanism

The Input Device must detect abnormality in operation, retain the status in an internal variable, and then respond to the inquiry from the Application Controller. Therefore, a hardware mechanism that detects an error is required. Make sure that the detection part is implemented in the user application.

2.1.4 Signal processing unit

An instance is a component of the Input Device. Instances operate by handling input signals from switches and sensors.
Mount the signal processing unit according to your needs.

2.2 Software Requirements

2.2.1 DALI103i module definition

A logical bus unit (equivalent to Input Device in this manual) defined in a hardware device is called a logical unit. This library provides a structure type (dali103i_t) containing a collection of parameters required for configuring a logical unit of the Input Device. A dali103i_t type variable is called a DALI103i module.

Define as many DALI103i modules as the required number of logical units.

2.2.2 DALI103i Instance module definition

The DALI standard allows for implementation of 1 to 32 instances (signal processing units) per Input Device as required. Instances are classified into 32 instance types according to the input signals to be handled.

The following describes the instance types and defined standards.

Table 2-1 Instance types and defined standards

Instance Type	IEC 62386-	Used for
0	103	General-purpose instance
1	301	Push buttons
2	302	Absolute input devices
3	303	Occupancy sensors
4	304	Light sensors
. . .		

This library, which is created based on the DALI103 standard, provides a structure type (dali103i_instance_t) containing a collection of parameters required for configuring an instance of instance 0. A dali103i_instance_t type variable is called a DALI103i instance module. If you need an instance of instance type 0, define the required number of DALI103i instance modules.

In addition, if you want to install an instance of instance type 1 or greater, install the required parts according to your own specific needs, or separately use a DALI3XX library.

2.2.3 DALI communication driver

Install a driver that controls the DALI communication circuit described in the hardware requirements and satisfies the IEC62386-101 standard. For details about the DALI communication driver, refer to the following application note.

RL78/G23 Lighting Communication Master Board Initial Firmware (R01AN6460JJ0100)

2.2.4 Time management

This library contains an API function that performs time management. Implement the 1 ms interval timer in the user application and call the R_DALI103I_Tick1ms function.

A low precision timer might cause a violation to the standard. Make sure that the calling interval error is less than ±10%.

2.2.5 Random number generation

The Input Device is required to generate 24-bit random numbers without reproducibility. In the user application, implement a function that generates 4-bit random numbers without reproducibility in the range from 0x000000 to 0xFFFFFE. The following indicates the format of arguments and return values. This function is necessary for using this library.

The implemented function must be registered as a callback function in the Get Random Value member of the dali103i_general_callback_t type structure variable.

Function format	
Arguments	None
Return value	uint32_t A value in the range from 0x000000 to 0xFFFFFE

2.2.6 Non-volatile area access

Implement the processing that accesses the non-volatile area described in the hardware requirements.

In order to satisfy the IEC62396-103 standard, write processing must complete within 300 ms.

2.2.7 Error detection

If an error status is generated or cleared in the error detection mechanism described in the hardware requirements, call the following API functions.

- If an error related to the Input Device occurs:
  R_DALI103I_SetInputDeviceError function
- If an error related to the Input Device is cleared:
  R_DALI103I_ClearInputDeviceError function
- If an error related to the instance of instance type 0 occurs:
  R_DALI103I_SetInstanceError function
- If an error related to the instance of instance type 0 is cleared:
  R_DALI103I_ClearInstanceError function

2.2.8 Memory bank entity definition

This library does not contain information about the structure and contents of memory banks of the Input Device because such information is user dependent (with the exception of some parts). Implement the entity of the memory bank according to the IEC62386-103 standard.

Sections 2.2.8.1 to 2.2.8.3 describe the specifications defined in the standard.

2.2.8.1 Memory bank 0

Memory bank 0 is required for each logical unit. Information about the Input Device and logical unit is stored.

Table 2-2 Memory map of memory bank 0 (1/2)

Address	Description	Default value	Memory access
0x00	Address of the last accessible memory location	factory burn-in,	ROM
0x01	Reserved - not implemented	answer NO	n.a.
0x02	Number of the last accessible memory bank	factory burn-in, range [0,0xFF]	ROM
0x03	GTIN byte 0 (MSB)	factory burn-in	ROM
0x04	GTIN byte 1	factory burn-in	ROM
0x05	GTIN byte 2	factory burn-in	ROM
0x06	GTIN byte 3	factory burn-in	ROM
0x07	GTIN byte 4	factory burn-in	ROM
0x08	GTIN byte 5 (LSB)	factory burn-in	ROM
0x09	Firmware version (major)	factory burn-in	ROM
0x0A	Firmware version (minor)	factory burn-in	ROM
0x0B	Identification number byte 0 (MSB)	factory burn-in	ROM
0x0C	Identification number byte 1	factory burn-in	ROM
0x0D	Identification number byte 2	factory burn-in	ROM
0x0E	Identification number byte 3	factory burn-in	ROM
0x0F	Identification number byte 4	factory burn-in	ROM
0x10	Identification number byte 5	factory burn-in	ROM
0x11	Identification number byte 6	factory burn-in	ROM
0x12	Identification number byte 7 (MSB)	factory burn-in	ROM
0x13	Hardware version (major)	factory burn-in	ROM
0x14	Hardware version (minor)	factory burn-in	ROM
0x15	101 version number	factory burn-in, according to implemented version number	ROM
0x16	102 version number of all integrated control gear	factory burn-in, according to implemented version number	ROM
0x17	103 version number of all integrated control devices	factory burn-in, according to implemented version number	ROM

Table 2-3 Memory map of memory bank 0 (2/2)

Address	Description	Default value	Memory access
0x18	Number of logical control device units in the bus unit	factory burn-in, range [1, 64]	ROM
0x19	Number of logical control gear units in the bus unit	factory burn-in, range [0,64]	ROM
0x1A	Index number of this logical control gear unit	factory burn-in, range [0,location 0x19 - 1]	ROM
[0x1B, 0x7F]	Reserved - not implemented	answer NO	n.a.
[0x80, 0xFE]	Additional control gear information		ROM
0xFF	Reserved - not implemented	answer NO	n.a.

2.2.8.2 Memory bank 1

Memory bank 1 is an optional memory bank that can be added to each logical unit. This memory bank is reserved for additional information related to the OEM specifications.

Table 2-4 Memory map of memory bank 1 (1/2)

Address	Description	Default value	RESET value	Memory access
0x00	Address of the last accessible memory location	factory burn-in, range [0x10,0xFE]	no change	ROM
0x01	Indicator byte			any
0x02	Memory bank 1 lock byte. Lockable bytes in the memory bank shall be read-only while the lock byte has a value different from 0x55.	0xFF	0xFF	RAM
0x03	OEM GTIN byte 0 (MSB)	0xFF	no change	NVM (lockable)
0x04	OEM GTIN byte 1	0xFF	no change	NVM (lockable)
0x05	OEM GTIN byte 2	0xFF	no change	NVM (lockable)
0x06	OEM GTIN byte 3	0xFF	no change	NVM (lockable)
0x07	OEM GTIN byte 4	0xFF	no change	NVM (lockable)
0x08	OEM GTIN byte 5 (LSB)	0xFF	no change	NVM (lockable)

Table 2-5 Memory map of memory bank 1 (2/2)

Address	Description	Default value	RESET value	Memory access
0x09	OEM identification number byte 0 (MSB)	0xFF	no change	NVM (lockable)
0x0A	OEM identification number byte 1	0xFF	no change	NVM (lockable)
0x0B	OEM identification number byte 2	0xFF	no change	NVM (lockable)
0x0C	OEM identification number byte 3	0xFF	no change	NVM (lockable)
0x0D	OEM identification number byte 4	0xFF	no change	NVM (lockable)
0x0E	OEM identification number byte 5	0xFF	no change	NVM (lockable)
0x0F	OEM identification number byte 6	0xFF	no change	NVM (lockable)
0x10	OEM identification number byte 7 (MSB)	0xFF	no change	NVM (lockable)
≥ 0x11	Additional control device information			
0xFF	Reserved - not implemented	answer NO		n.a.

2.2.8.3 Memory banks 2 to 199

Memory banks 2 to 199 are optional memory banks that can be added to each logical unit. You can freely define the contents of each bank as far as the basic specifications are satisfied.

Table 2-6 Memory map of memory banks 2 to 199

Address	Description	Default value	RESET value	Memory access
0x00	Address of the last accessible memory location	factory burn-in, range [0x03,0xFE]	no change	ROM
0x01	Indicator byte			any
0x02	Memory bank lock byte. Lockable bytes in the memory bank shall be read-only while the lock byte has a value different from 0x55.	0xFF	0xFF	RAM
[0x03,0xFE]	Memory bank content			any
0xFF	Reserved - not implemented	answer NO	no change	n.a.

2.2.9 Implementing memory access functions

This library, by itself, cannot access a memory bank whose entity is defined by the user. Therefore, implement access functions according to the prototypes of the callback functions to be provided.

For details, refer to the function specifications in 3.5.1 R_DALI103I_InitLibrary.

To use this library, some access functions must be implemented while other functions are optional. The following table shows the required access functions and optional access functions.

Table 2-7 Required memory access functions

Access function	Required/optional
RESET function	Required
READ function	Required
WRITE function	Required
UNLATCH READ function	Optional
CANCEL WRITE function	Optional

2.2.9.1 RESET function

Implement the function that sets all locations of the function specified in the argument to the RESET value. The following indicates the format of arguments and return values. This function is required for using this library.

Function format	
Arguments	uint8_t unit Logical unit number uint8_t bank Bank number to be reset
Return value	void

2.2.9.2 READ function

Implement the function that reads the value of the location specified in the argument. The following indicates the format of arguments and return values. This function is required for using this library.

Function format	
Arguments	uint8_t unit Logical unit number uint8_t bank Read target bank number uint8_t location Specified location of the bank to be read
Return value	int16_t 0x00 - 0xFF: Read value DALI103I_MB_IS_NOT_IMPLEMENTED: The specified bank is not implemented. DALI103I_MB_LOCATION_IS_NOT_IMPLEMENTED: The specified location is not implemented. DALI103I_MB_EXECUTE_ERROR: Execution error

2.2.9.3 WRITE function

Implement the function that writes a value to the location specified in the argument. The following indicates the format of arguments and return values. This function is required for using this library.

Function format	
Arguments	uint8_t unit Logical unit number uint8_t bank Write target bank number uint8_t location Specified location of the bank to write data uint8_t data Data to be written
Return value	int16_t 0x00 to 0xFF: Written data value DALI103I_MB_IS_NOT_IMPLEMENTED: The specified bank is not implemented. DALI103I_MB_LOCATION_IS_NOT_IMPLEMENTED: The specified location is not implemented. DALI103I_MB_EXECUTE_ERROR: Execution error

2.2.9.4 UNLATCH READ function

The IEC62386-103 standard specifies memory banks that permit multibyte data (data from multiple consecutive locations of memory data combined to form singular meaningful data). If any data values are changed while multibyte data is being read, the validity of singular multibyte data is lost. To prevent values from being changed while allocating multibyte data in a memory bank, it is recommended to implement a data latch function. This function latches (retains) data from start to end of reading of target multibyte data. If this function is implemented, you also need to implement a function that cancels the data latch during reading of the logical unit specified in the argument. This function is optional for using this library.

Function format	
Arguments	uint8_t unit Logical unit number
Return value	void

2.2.9.5 CANCEL WRITE function

The IEC62386-103 standard specifies memory banks that permit multibyte data (data from multiple consecutive locations of memory data combined to form singular meaningful data). If writing is interrupted while multibyte data is being updated one byte at a time, the validity of singular multibyte data is lost. Therefore, when allocating multibyte data in a memory bank, the user requires a function that writes multibyte data in a batch only when data has been written from beginning to end, or resumes the data before update if the data is not written up to the end. This function is used to clearly notify that data has not been written up to the end.

Only when the above specifications are included in the memory bank entity definition, implement the function that cancels writing of multibyte data in the logical unit. This function is optional when using this library.

Function format	
Arguments	uint8_t unit Logical unit number
Return value	void

3. DALI103i Library Functions

This chapter describes the functions of this library.

3.1 Definitions of Data Types and Return Values

The following describes the data types provided by this library.

Table 3-1 List of data types

Type name	Description
dali103_t	DALI103i module type
dali103i_instance_t	DALI103i instance module type
dali103i_cmd_t	Command information type

The following describes the definition macros provided by this library.

Table 3-2 List of macros

Macro name	Macro value	Description
DALI103I_TEST_FRAME_MAX	(4)	Maximum number of test frames
DALI103I_INSTANCE_NUM_MAX	(32)	Maximum number of instances
DALI103I_NO_ANSWER	((int16_t)-1)	No backward frame
DALI103I_ANSWER_IS_CORRUPTED	((int16_t)-2)	Corrupted backward frame

The following describes the return values provided by this library.

Table 3-3 List of return values (dali103i_return_t)

Definition	Return value	Description
DALI103I_RETURN_OK	0	Normal termination
DALI103I_RETURN_ERR	-1	Abnormal termination

3.2 List of Structures

The following describes the structures provided by this library.

Definition of the general-purpose callback function type structure (dali103i_general_callback_t)

```
typdef struct
{
    uint32_t (*GetRandomValue)(void);
} dali103i_general_callback_t;
```

Definition of the memory bank operation callback function type structure (dali103i_mb_if_callback_t)

```
typedef struct
{
    void (*Reset)(uint8_t unit, uint8_t bank);
    int16_t (*Read)(uint8_t unit, uint8_t bank, uint8_t location);
    int16_t (*Write)(uint8_t unit, uint8_t bank, uint8_t location, uint8_t data);
    void (*UnlatchRead)(uint8_t unit);
    void (*CancelWrite)(uint8_t unit);
} dali103i_mb_if_callback_t;
```

Definition of the event message type structure (dali103i_event_t)

```
typedef struct
{
    bool is_exist;
    dali103i_forward_frame_t frame;
} dali103i_event_t;
```

Definition of the Test Frame type structure (dali103i_test_frame_t)

```
typedef struct
{
    uint8_t num;
    dali103i_forward_frame_t frame[DALI103I_TEST_FRAME_MAX];
} dali103i_test_frame_t;
```

Definition of the device NVM type structure (dali103i_device_nvm_t)

```
typedef struct
{
    uint8_t short_address;
    uint32_t device_groups;
    uint32_t random_address;
    uint8_t operating_mode;
    bool application_active;
    bool power_cycle_notification;
    uint8_t event_priority;
} dali103i_device_nvm_t;
```

Definition of the instance NVM type structure (dali103i_instance_nvm_t)

```
typedef struct
{
    uint8_t instance_group0;
    uint8_t instance_group1;
    uint8_t instance_group2;
    bool instance_active;
    uint32_t event_filter;
    uint8_t event_scheme;
    uint8_t event_priority;
} dali103i_instance_nvm_t;
```

3.3 List of API Functions

The following lists the API functions of this library.

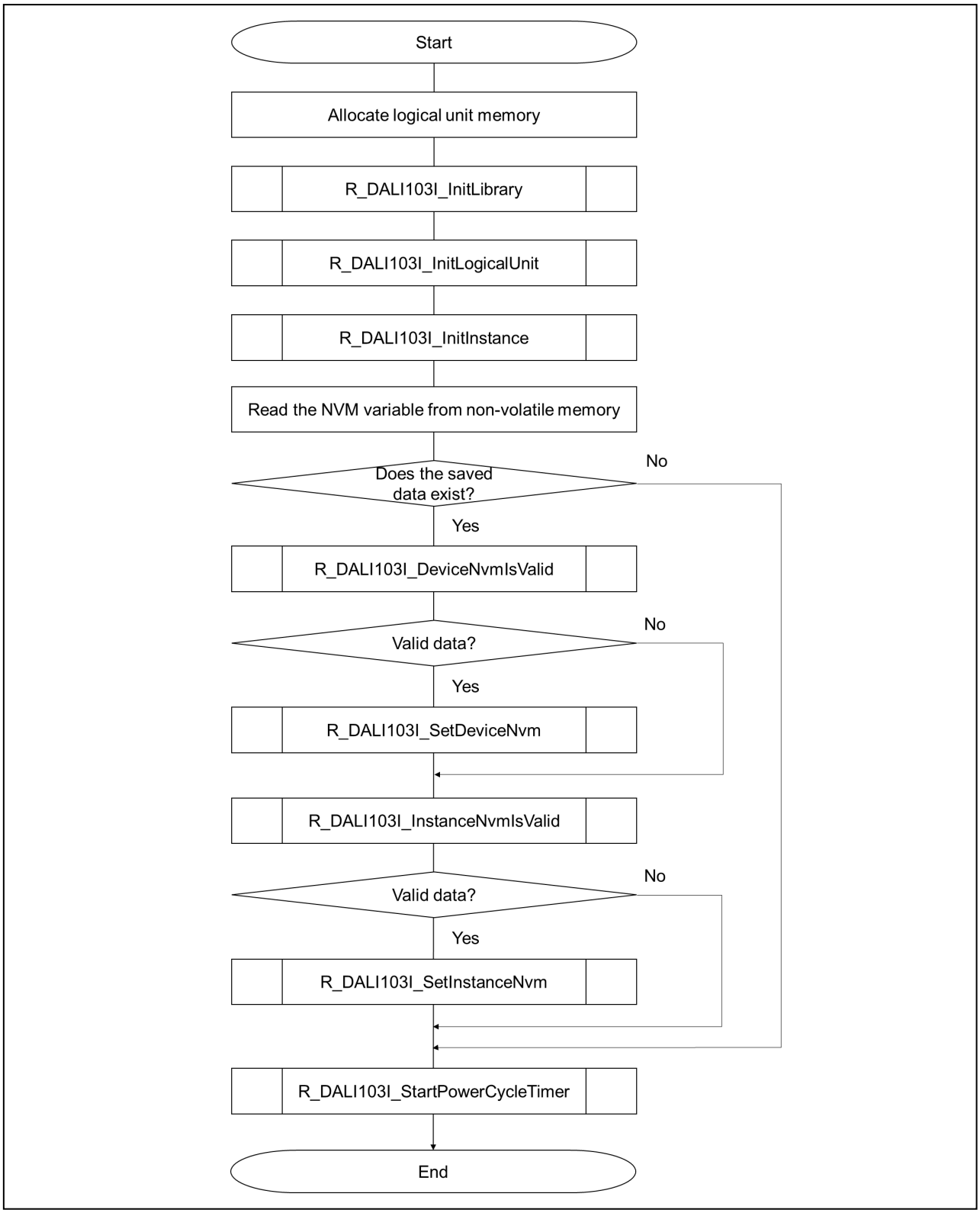
Table 3-4 List of API functions

Function name	Description
R_DALI103I_InitLibrary	Initializes the DALI103i library
R_DALI103I_InitLogicalUnit	Initializes a logical unit
R_DALI103I_InitInstance	Initializes the instance
R_DALI103I_DeviceNvmIsValid	Checks that device NVM variable values are in the valid range
R_DALI103I_InstanceNvmIsValid	Checks that instance NVM variable values are in the valid range
R_DALI103I_SetDeviceNvm	Sets device NVM variable values
R_DALI103I_SetInstanceNvm	Sets instance NVM variable values
R_DALI103I_GetDeviceNvm	Acquires device NVM variable values
R_DALI103I_GetInstanceNvm	Acquires instance NVM variable values
R_DALI103I_DeviceNvmIsChanged	Checks for changes in device NVM variable values
R_DALI103I_InstanceNvmIsChanged	Checks for changes in instance NVM variable values
R_DALI103I_NeedsToSaveNvm	Checks the necessity of saving NVM variables
R_DALI103I_NotifySaveNvm	Notification of NVM variable save completion
R_DALI103I_StartPowerCycleTimer	Starts the power cycle notification timer
R_DALI103I_GetOperatingMode	Acquires operating mode values
R_DALI103I_Tick1ms	Advances the internal operation by 1 ms
R_DALI103I_SetInputDeviceError	Sets detailed information for Input Device Error
R_DALI103I_ClearInputDeviceError	Clears InputDeviceError information
R_DALI103I_SetInstanceError	Sets an instance error
R_DALI103I_ClearInstanceError	Clears the instance error
R_DALI103I_InstanceIsActive	Checks the Instance Active status
R_DALI103I_GetPowerCycleNotification	Acquires the power cycle notification event
R_DALI103I_SetInputSignal	Sets input signals
R_DALI103I_GetInstanceEventFilter	Acquires an instance event filter
R_DALI103I_GenerateInputNotification	Generates an input notification event
R_DALI103I_GetTestFrame	Acquires test frames
R_DALI103I_IdentificationIsActive	Checks whether the identification is active
R_DALI103I_QuiescentModeIsActive	Checks whether the quiescent mode is active
R_DALI103I_CreateCommand	Creates command information
R_DALI103I_ExecuteCommand	Executes the received command
R_DALI103I_GetLibraryVersion	Acquires version information

3.4 Outline Flowcharts

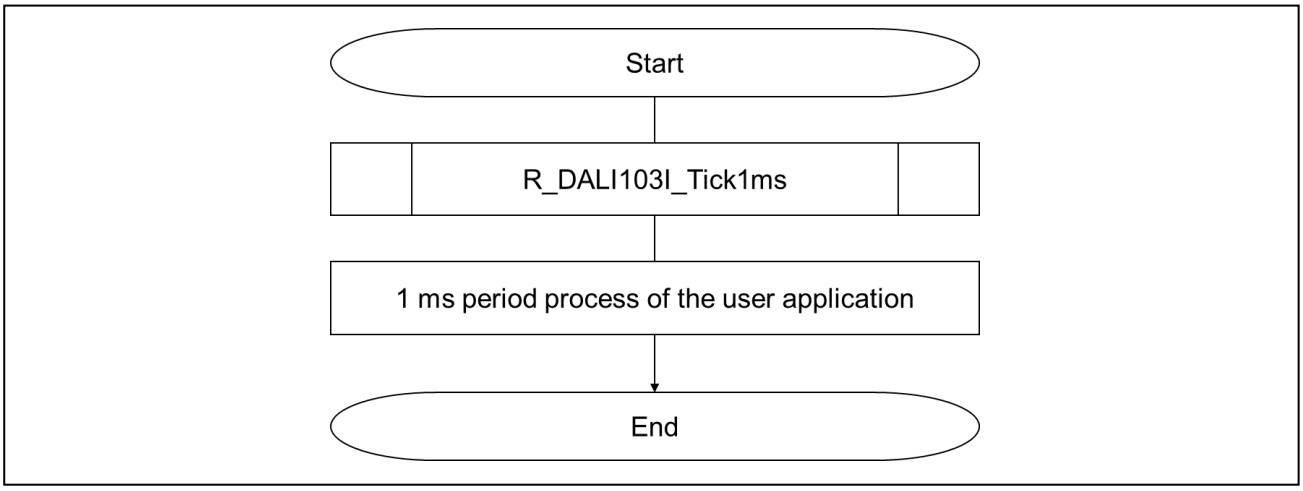
3.4.1 Initialization

The following shows the initialization flowchart.



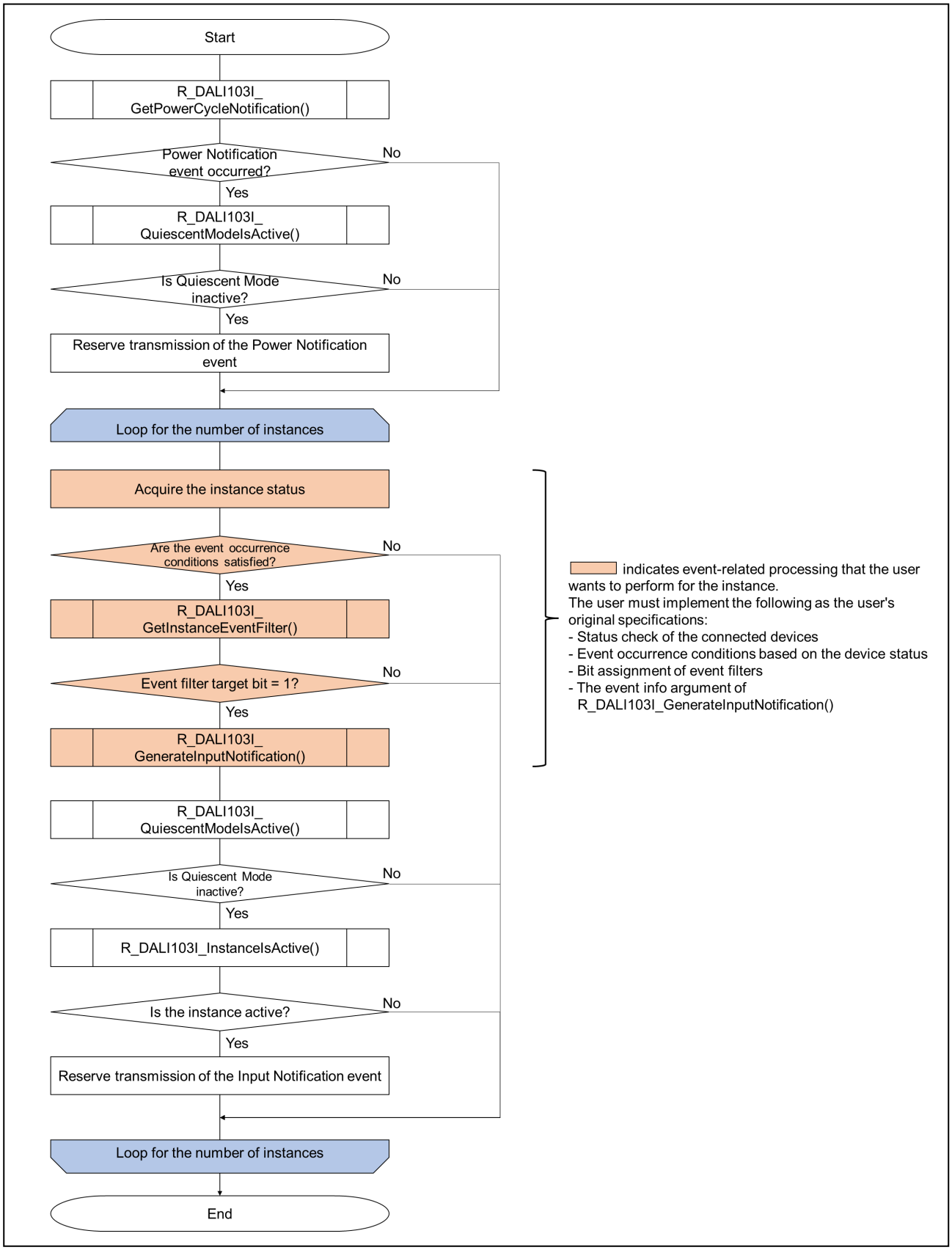
3.4.2 1 ms period process

The following shows the flowchart of 1 ms period process. Perform this process at 1 ms intervals.



3.4.3 Event Message processing

The following shows the flowchart of Event Message processing.

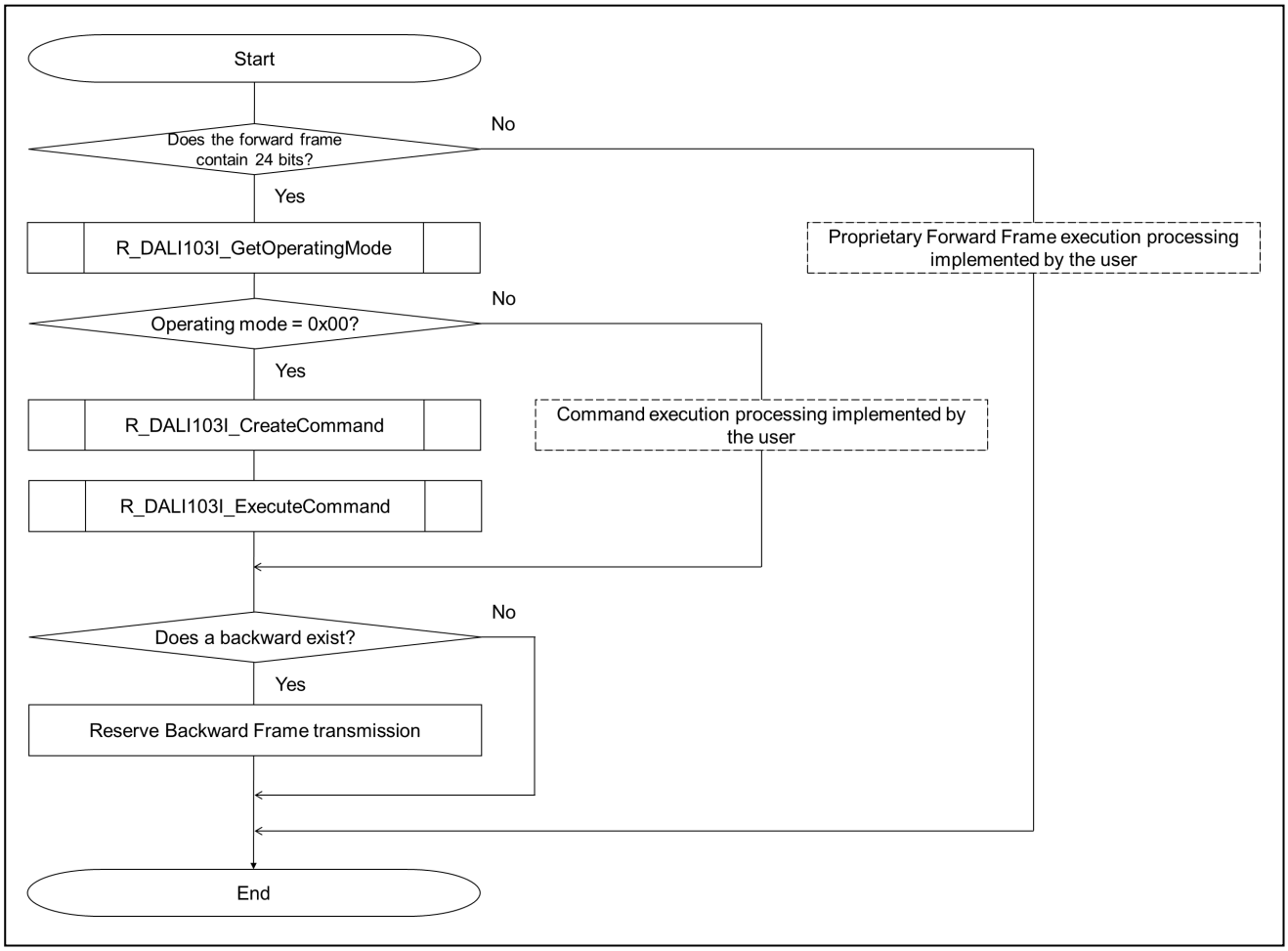


3.4.4 Processing when receiving a Forward Frame

The following shows the flowchart of processing when receiving a Forward Frame. Perform this processing then the DALI communication bus receives a Forward Frame.

Processing for the Proprietary Forward Frame (a forward frame whose length is more than 16 bits and other than 20 bits, 24 bits, or 32 bits) is an optional function. Implement the processing if it is supported by the DALI communication driver and application.

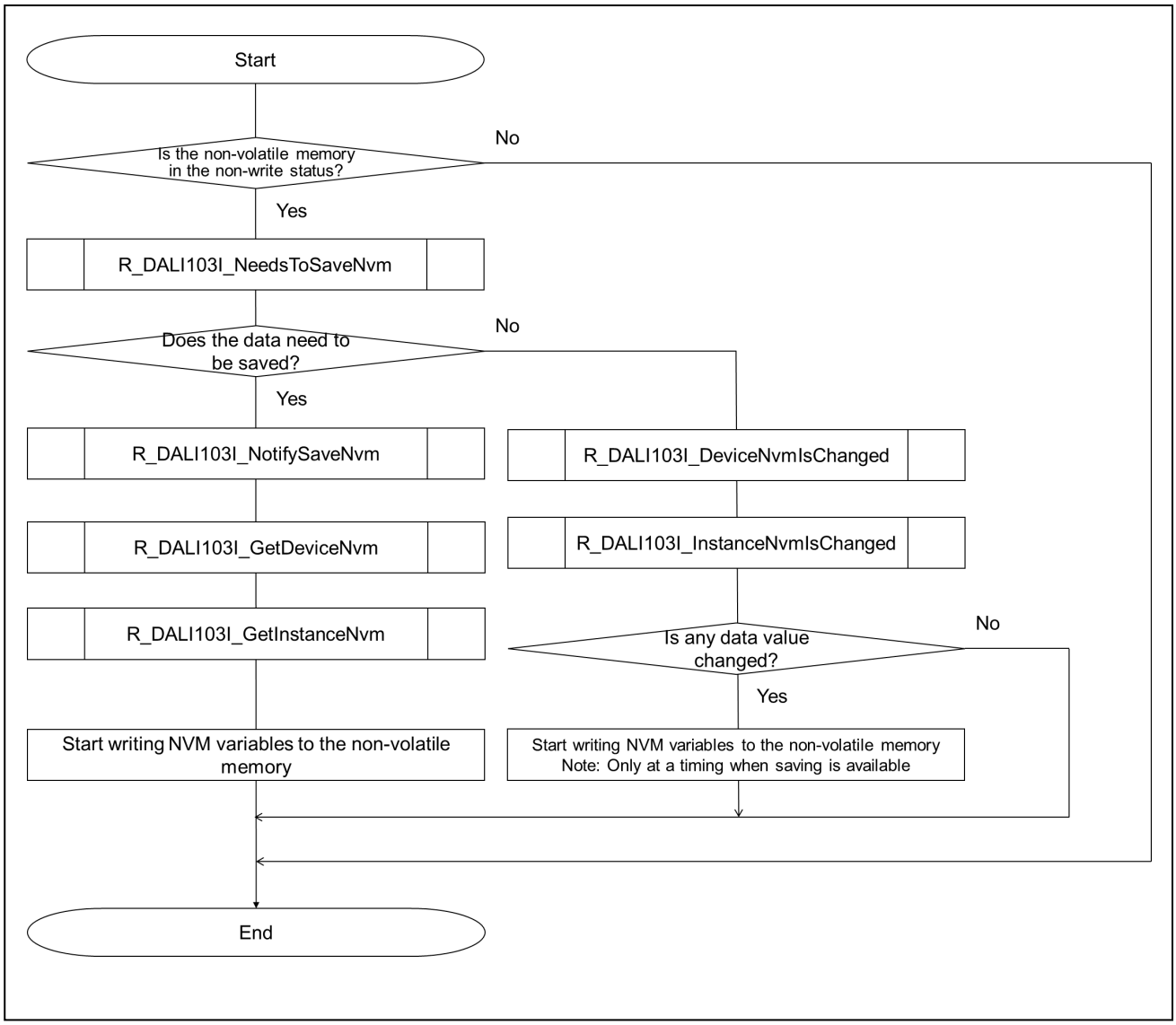
Note that any operating mode other than 0 is an optional function. If you need your original mode, implement that mode and then register the mode number by using the R_DALI103I_InitLogicalUnit function.



3.4.5 Non-volatile data processing

The following shows the flowchart of non-volatile data processing.

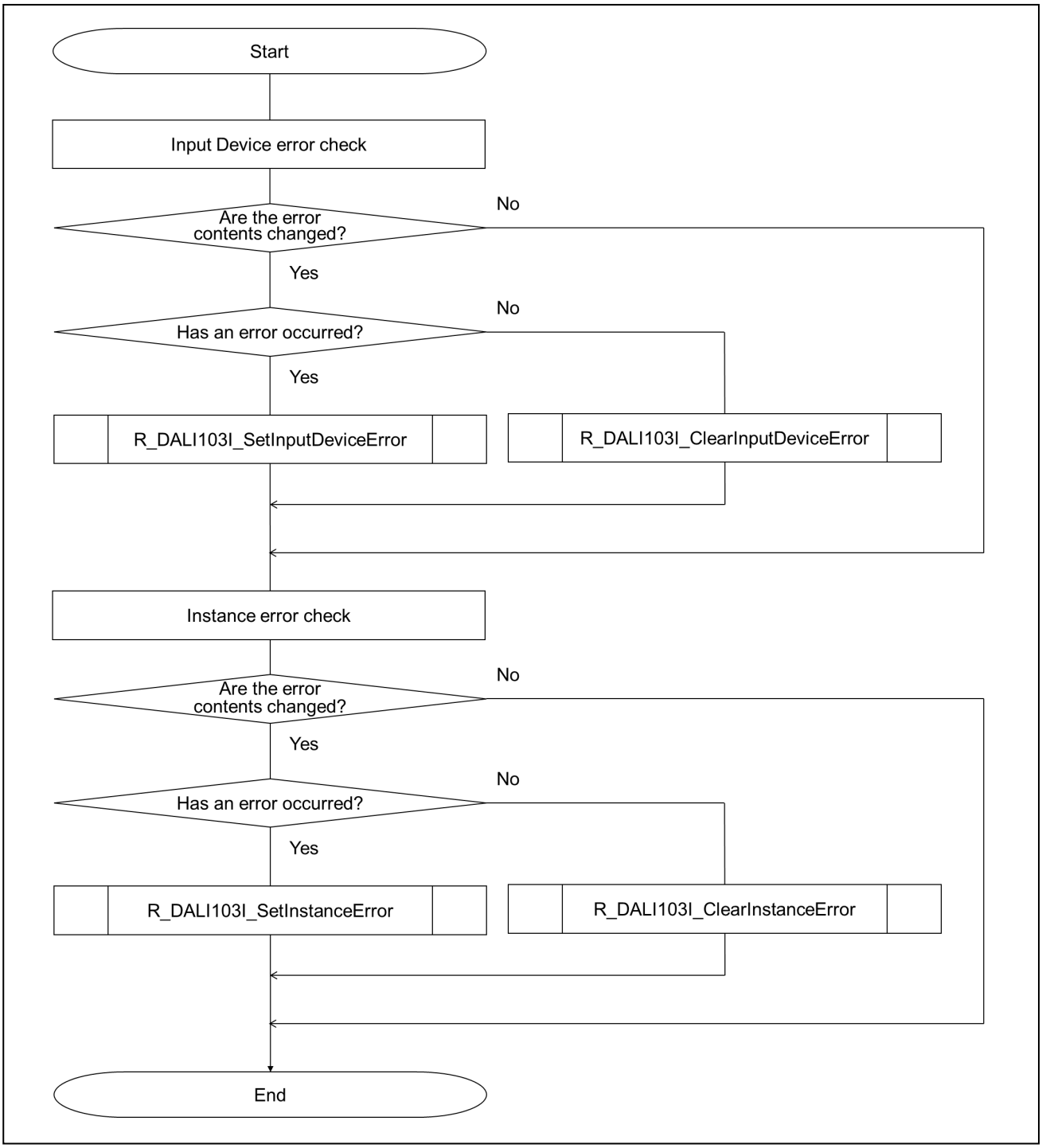
It is stipulated that saving of data in the non-volatile memory be complete within 300 ms after the SAVE PERSISTENT VARIABLES command is received. It is also stipulated that if any change is made in NVM variable values, the data must be saved within 30 seconds even if the SAVE PERSISTENT VARIABLES command is not received. Periodically check the data and perform processing to ensure that saving of the relevant data is complete within the specified time.



3.4.6 Error processing

The following shows the flowchart of error processing. Call this processing when an error state is updated.

The detailed specifications of Input Device error and Instance error depend on the hardware and software. Define the specifications and consider implementation according to the environment.



3.5 API Function Specifications

The following describes the API function specifications of this library.

3.5.1 R_DALI103I_InitLibrary

Overview:

This function initializes the DALI103i library.

Format:

```
dali103i_return_t R_DALI103I_InitLibrary(const dali103i_general_callback_t * p_gen_callback,
                                         const dali103i_mb_if_callback_t * p_mb_callback)
```

Prerequisites:

None

Arguments:

Arguments	Description
const dali103i_general_callback_t * p_gen_callback	Pointer to the general-purpose callback function [Members] .GetRandomValue Set the pointer of the function in accordance with the function format in 2.2.5. Setting of NULL is not permitted.
const dali103i_mb_if_callback_t * p_mb_callback	Pointer to the memory bank callback function [Members] .Reset Set the pointer of the function in accordance with the function format in 2.2.9.1. Setting of NULL is not permitted. .Read Set the pointer of the function in accordance with the function format in 2.2.9.2. Setting of NULL is not permitted. .Write Set the pointer of the function in accordance with the function format in 2.2.9.3. Setting of NULL is not permitted. .UnlatchRead Set the pointer of the function in accordance with the function format in 2.2.9.4. Because this is an optional function, setting of NULL is permitted. .CancelWrite Set the pointer of the function in accordance with the function format in 2.2.9.5. Because this is an optional function, setting of NULL is permitted.

Return values:

Value	Description
DALI103I_RETURN_OK	Normal termination
DALI103I_RETURN_ERR	Parameter error - Revise the argument settings.

3.5.2 R_DALI103I_InitLogicalUnit

Overview:

This function initializes the specified logical unit.

Call this function as many times as the required number of logical units.

Format:

```
dali103i_return_t R_DALI103I_InitLogicalUnit(dali103i_t * p_this,
                                             uint8_t unit_index,
                                             uint8_t default_operating_mode,
                                             const uint8_t * p_mode_list)
```

Prerequisites:

1. The R_DALI103I_InitLibrary function must have terminated normally.

Arguments:

Arguments	Description
dali103i_t * p_this	Pointer to the DALI103i module
uint8_t unit_index	Index number of the logical unit Valid range: 0x00 to 0x3F Note: Specify the same value as the index number specified for location 0x1A of memory bank 0 to be used.
uint8_t default_operating_mode	Default value of operating mode Valid range: - operating_mode: 0x00, 0x80 to 0xFF
const uint8_t * p_mode_list	Pointer to the operating mode array For details of how to specify the value, see the next page.

Return values:

Value	Description
DALI103I_RETURN_OK	Normal termination
DALI103I_RETURN_ERR	Parameter error - Revise the argument settings.

(1) Specifying the p_mode_list parameter

Specify the first pointer of the uint8_t array.
Assign the number of operating modes to be registered to the first element of the array, and assign the operating mode values to each of the second and subsequent elements. Satisfy the following requirements for use:

- Be sure to specify 0x00 (DALI specification mode).
- Register additional operating modes in the range of manufacturer-specific modes (0x80 to 0xFF).
- Set any of the registered operating modes to p_default_value.operating_mode.

The following provides examples of specifying arrays.

Example 1: To register the DALI specification mode only, specify as follows:

```
uint8_t operating_mode_list[2] = { 0x01, 0x00 };
```

Example 2: To register the DALI specification mode and manufacturer-specific modes 0x80 and 0x90, specify as follows:

```
uint8_t operating_mode_list[4] = { 0x03, 0x00, 0x80, 0x90 };
```

3.5.3 R_DALI103I_InitInstance

Overview:

This function initializes the DALI103i instance module and registers the instance to the DALI103i module (dali103i_t type).

The dali103i_instance_t type provides an instance of Instance Type 0. To use an instance of other than Instance Type 0, use the DALI3XX library that matches the instance type.

Format:

```
dali103i_return_t R_DALI103I_InitInstance(dali103i_t * p_this,
                                          dali103i_instance_t * p_instance,
                                          uint8_t resolution)
```

Prerequisites:

1. The R_DALI103I_InitLibrary function must have terminated normally.
2. The R_DALI103I_InitLogicalUnit function must have terminated normally.

Arguments:

Arguments	Description
dali103i_t * p_this	Pointer to the DALI103i module
dali103i_instance_t * p_instance	Pointer to the DALI103i instance module
uint8_t resolution	Resolution of input signals Valid range: 1 to 255

Return values:

Value	Description
DALI103I_RETURN_OK	Normal termination
DALI103I_RETURN_ERR	Parameter error - Revise the argument settings.

3.5.4 R_DALI103I_DeviceNvmIsValid

Overview:

This function returns whether all the values set for the members of the dali103i_device_nvm_t type variable are within the valid range.

You must call this function to check the values before setting a value in the R_DALI103I_SetDeviceNvm function described later.

Format:

```
bool R_DALI103I_DeviceNvmIsValid(const dali103i_t * p_this,
const dali103i_device_nvm_t * p_nvm)
```

Prerequisites:

1. The R_DALI103I_InitLibrary function must have terminated normally.
2. The R_DALI103I_InitLogicalUnit function must have terminated normally.

Arguments:

Arguments	Description
const dali103i_t * p_this	Pointer to the DALI103i module
const dali103i_device_nvm_t * p_nvm	Pointer to the DALI103i device NVM variable Valid range: - short_address: 0x00 to 0x3F, 0xFF - device_groups: 0x00000000 to 0xFFFFFFFF - random_address: 0x000000 to 0xFFFFFF - operating_mode: 0x00, 0x80 to 0xFF - application_active: true or false - power_cycle_notification: enable> or disable - event_priority: 0x2 to 0x5

Return values:

Value	Description
true	All the variables are within the valid range.
false	At least one variable is out of the valid range.

3.5.5 R_DALI103I_InstanceNvmIsValid

Overview:

This function returns whether all the values set for the members of the dali103i_instance_nvm_t type variable are within the valid range.

You must call this function to check the values before setting a value in the R_DALI103I_SetInstanceNvm function described later.

Format:

```
bool R_DALI103I_InstanceNvmIsValid(const dali103i_instance_t * p_this,
const dali103i_instance_nvm_t * p_nvm)
```

Prerequisites:

1. The R_DALI103I_InitLibrary function must have terminated normally.
2. The R_DALI103I_InitLogicalUnit function must have terminated normally.
3. The R_DALI103I_InitInstance function must have terminated normally.

Arguments:

Arguments	Description
const dali103i_instance_t * p_this	Pointer to the DALI103i instance module
const dali103i_instance_nvm_t * p_nvm	Pointer to the DALI103i instance NVM variable Valid range: - instance_group0: 0x00 to 0x1F, 0xFF - instance_group1: 0x00 to 0x1F, 0xFF - instance_group2: 0x00 to 0x1F, 0xFF - instance_active: true or false - event_filter: 0x000000 to 0xFFFFFF - event_scheme: 0x0 to 0x4 - event_priority: 0x2 to 0x5

Return values:

Value	Description
true	All the variables are within the valid range.
false	At least one variable is out of the valid range.

3.5.6 R_DALI103I_SetDeviceNvm

Overview:

This function sets the value of the device NVM variable in the DALI103i module.

Use this function to set the read-out data when data of the device NVM variable is saved in the non-volatile memory when the power is turned on.

Format:

```
void R_DALI103I_SetDeviceNvm(dali103i_t * p_this,
                             const dali103i_device_nvm_t * p_nvm)
```

Prerequisites:

1. The R_DALI103I_InitLibrary function must have terminated normally.
2. The R_DALI103I_InitLogicalUnit function must have terminated normally.
3. The device NVM variable must be confirmed to be within the valid range by using the R_DALI103I_DeviceNvmIsValid function.

Arguments:

Arguments	Description
dali103i_t * p_this	Pointer to the DALI103i module
const dali103i_device_nvm_t * p_nvm	Pointer to the DALI103i device NVM variable

Return values:

None

3.5.7 R_DALI103I_SetInstanceNvm

Overview:

This function sets the value of the instance NVM variable in the DALI103i instance module.

Use this function to set the read-out data when data of the device NVM instance variable is saved in the non-volatile memory when the power is turned on.

Format:

```
void R_DALI103I_SetInstanceNvm(dali103i_instance_t * p_this,
                               const dali103i_instance_nvm_t * p_nvm)
```

Prerequisites:

1. The R_DALI103I_InitLibrary function must have terminated normally.
2. The R_DALI103I_InitLogicalUnit function must have terminated normally.
3. The R_DALI103I_InitInstance function must have terminated normally.
4. The instance NVM variable must be confirmed to be within the valid range by using the R_DALI103I_InstanceNvmIsValid function.

Arguments:

Arguments	Description
dali103i_instance_t * p_this	Pointer to the DALI103i instance module
const dali103i_instance_nvm_t * p_nvm	Pointer to the DALI103i instance NVM variable

Return values:

None

3.5.8 R_DALI103I_GetDeviceNvm

Overview:

This function acquires the set value of the device NVM variable from the DALI103i module.

Use this function to save the latest value of the device NVM variable in the non-volatile memory.

Format:

```
void R_DALI103I_GetDeviceNvm(const dali103i_t * p_this,
                             dali103i_device_nvm_t * p_nvm)
```

Prerequisites:

1. The R_DALI103I_InitLibrary function must have terminated normally.
2. The R_DALI103I_InitLogicalUnit function must have terminated normally.
3. The power cycle notification timer must have started by using the R_DALI103I_StartPowerCycleTimer function.

Arguments:

Arguments	Description
const dali103i_t * p_this	Pointer to the DALI103i module
dali103i_device_nvm_t * p_nvm	Pointer to the DALI103i device NVM variable

Return values:

None

3.5.9 R_DALI103I_GetInstanceNvm

Overview:

This function acquires the set value of the device NVM variable from the DALI103i module.

Use this function to save the latest value of the instance NVM variable in the non-volatile memory.

Format:

```
void R_DALI103I_GetInstanceNvm(const dali103i_instance_t * p_this,
                               dali103i_instance_nvm_t * p_nvm)
```

Prerequisites:

1. The R_DALI103I_InitLibrary function must have terminated normally.
2. The R_DALI103I_InitLogicalUnit function must have terminated normally.
3. The R_DALI103I_InitInstance function must have terminated normally.
4. The power cycle notification timer must have started by using the R_DALI103I_StartPowerCycleTimer function.

Arguments:

Arguments	Description
const dali103i_instance_t * p_this	Pointer to the DALI103i instance module
dali103i_instance_nvm_t * p_nvm	Pointer to the DALI103i instance NVM variable

Return values:

None

3.5.10 R_DALI103I_DeviceNvmIsChanged

Overview:

This function acquires information about whether at least one device NVM variable value was changed.

If the return value of this function is true, save the device NVM variables in the non-volatile memory according to the status of hardware.

This function acquires the status from when the function was called the last time (or at startup when the function was called for the first time). Note that calling this function repeatedly might cause the return value to be false.

Format:

```
bool R_DALI103I_DeviceNvmIsChanged(dali103i_t * p_this)
```

Prerequisites:

1. The R_DALI103I_InitLibrary function must have terminated normally.
2. The R_DALI103I_InitLogicalUnit function must have terminated normally.
3. The power cycle notification timer must have started by using the R_DALI103I_StartPowerCycleTimer function.

Arguments:

Arguments	Description
dali103i_t * p_this	Pointer to the DALI103i module

Return values:

Value	Description
true	A value was changed.
false	No values were changed.

3.5.11 R_DALI103I_InstanceNvmIsChanged

Overview:

This function acquires information about whether at least one instance NVM variable value was changed.

If the return value of this function is true, save the instance NVM variables in the non-volatile memory according to the status of hardware.

This function acquires the status from when the function was called the last time (or at startup when the function was called for the first time). Note that calling this function repeatedly might cause the return value to be false.

Format:

```
bool R_DALI103I_InstanceNvmIsChanged(dali103i_instance_t * p_this)
```

Prerequisites:

1. The R_DALI103I_InitLibrary function must have terminated normally.
2. The R_DALI103I_InitLogicalUnit function must have terminated normally.
3. The R_DALI103I_InitInstance function must have terminated normally.
4. The power cycle notification timer must have started by using the R_DALI103I_StartPowerCycleTimer function.

Arguments:

Arguments	Description
dali103i_instance_t * p_this	Pointer to the DALI103i instance module

Return values:

Value	Description
true	A value was changed.
false	No values were changed.

3.5.12 R_DALI103I_NeedsToSaveNvm

Overview:

This function acquires information about whether a request for saving NVM variables was made (SAVE PERSISTENT VARIABLES command was received).

If the return value of this function is true, save the NVM variable values in the non-volatile memory according to the status of hardware. When save processing becomes possible in response to the save request with this function, issue a notification by calling the R_DALI103I_NotifySaveNvm function described later. The status that can be acquired by this function is not cleared until the notification is issued.

Format:

```
bool R_DALI103I_NeedsToSaveNvm(const dali103i_t * p_this)
```

Prerequisites:

1. The R_DALI103I_InitLibrary function must have terminated normally.
2. The R_DALI103I_InitLogicalUnit function must have terminated normally.
3. The R_DALI103I_InitInstance function must have terminated normally.
4. The power cycle notification timer must have started by using the R_DALI103I_StartPowerCycleTimer function.

Arguments:

Arguments	Description
const dali103i_t * p_this	Pointer to the DALI103i module

Return values:

Value	Description
true	Saving is required.
false	Saving is not required.

3.5.13 R_DALI103I_NotifySaveNvm

Overview:

Call this function when it becomes possible to perform processing that saves NVM variable values in the non-volatile memory in response to a save request from the R_DALI103I_NeedsToSaveNvm function.

The earlier you call this function, the more time you have to accept the next save request.

Format:

```
void R_DALI103I_NotifySaveNvm(dali103i_t * p_this)
```

Prerequisites:

1. The R_DALI103I_InitLibrary function must have terminated normally.
2. The R_DALI103I_InitLogicalUnit function must have terminated normally.
3. The R_DALI103I_InitInstance function must have terminated normally.
4. The power cycle notification timer must have started by using the R_DALI103I_StartPowerCycleTimer function.
5. The return value of the R_DALI103I_NeedsToSaveNvm function must be true.

Arguments:

Arguments	Description
dali103i_t * p_this	Pointer to the DALI103i module

Return values:

None

3.5.14 R_DALI103I_StartPowerCycleTimer

Overview:

This function sets each variable to the power on value and then starts the power cycle notification timer.

When this function is called, the variable group that contains the Power on value is set to the Power on value.

Considering the time required after the hardware is turned on until this function is called, specify a value so that the sum total is in the range from 1.3 to 5.0 s.

Format:

```
void R_DALI103I_StartPowerCycleTimer(dali103i_t * p_this,
                                     uint16_t msec)
```

Prerequisites:

1. The R_DALI103I_InitLibrary function must have terminated normally.
2. The R_DALI103I_InitLogicalUnit function must have terminated normally.
3. The R_DALI103I_InitInstance function must have terminated normally.

Arguments:

Arguments	Description
dali103i_t * p_this	Pointer to the DALI103i module
uint16_t msec	Time that can elapse before a power cycle notification is generated

Return values:

None

3.5.15 R_DALI103I_GetOperatingMode

Overview:

This function acquires the operating mode value during operation.

Change the processing for the received Forward Frame in accordance with the acquired operating mode value. The R_DALI103I_ExecuteCommand function provided by this library is a processing function when operating mode is 0.

Format:

```
uint8_t R_DALI103I_GetOperatingMode(const dali103i_t * p_this)
```

Prerequisites:

1. The R_DALI103I_InitLibrary function must have terminated normally.
2. The R_DALI103I_InitLogicalUnit function must have terminated normally.
3. The R_DALI103I_InitInstance function must have terminated normally.
4. The power cycle notification timer must have started by using the R_DALI103I_StartPowerCycleTimer function.

Arguments:

Arguments	Description
const dali103i_t * p_this	Pointer to the DALI103i module

Return values:

Value	Description
0x00	Mode in which processing is performed for received Forward Frames defined in the IEC62386-103 standard. Perform processing by executing the R_DALI103I_ExecuteCommand function described later.
0x80 to 0xFF	Mode implemented by the user. Handle the received Forward Frame by using user-implemented processing corresponding to the acquired operating mode value.

3.5.16 R_DALI103I_Tick1ms

Overview:

This function advances the internal operation of the DALI103i module by 1 ms.

Call this function at 1 ms intervals.

Format:

```
void R_DALI103I_Tick1ms(dali103i_t * p_this)
```

Prerequisites:

1. The R_DALI103I_InitLibrary function must have terminated normally.
2. The R_DALI103I_InitLogicalUnit function must have terminated normally.
3. The R_DALI103I_InitInstance function must have terminated normally.
4. The power cycle notification timer must have started by using the R_DALI103I_StartPowerCycleTimer function.

Arguments:

Arguments	Description
dali103i_t * p_this	Pointer to the DALI103i module

Return values:

None

3.5.17 R_DALI103I_SetInputDeviceError

Overview:

This function sets detailed information for Input Device Error.

Format:

```
void R_DALI103I_SetInputDeviceError(dali103i_t * p_this,
                                    uint8_t error)
```

Prerequisites:

1. The R_DALI103I_InitLibrary function must have terminated normally.
2. The R_DALI103I_InitLogicalUnit function must have terminated normally.
3. The R_DALI103I_InitInstance function must have terminated normally.
4. The power cycle notification timer must have started by using the R_DALI103I_StartPowerCycleTimer function.

Arguments:

Arguments	Description
dali103i_t * p_this	Pointer to the DALI103i module
uint8_t error	Detailed error information 0x00 to - 0xFE: Detailed error number Note: Error information is defined by the user. 0xFF: Error with unknown details

Return values:

None

3.5.18 R_DALI103I_ClearInputDeviceError

Overview:

This function clears the Input Device Error.

Format:

```
void R_DALI103I_ClearInputDeviceError(dali103i_t * p_this)
```

Prerequisites:

1. The R_DALI103I_InitLibrary function must have terminated normally.
2. The R_DALI103I_InitLogicalUnit function must have terminated normally.
3. The R_DALI103I_InitInstance function must have terminated normally.
4. The power cycle notification timer must have started by using the R_DALI103I_StartPowerCycleTimer function.

Arguments:

Arguments	Description
dali103i_t * p_this	Pointer to the DALI103i module

Return values:

None

3.5.19 R_DALI103I_SetInstanceError

Overview:

This function sets an instance error for the DALI103i instance module.

Format:

```
void R_DALI103I_SetInstanceError(dali103i_instance_t * p_this)
```

Prerequisites:

1. The R_DALI103I_InitLibrary function must have terminated normally.
2. The R_DALI103I_InitLogicalUnit function must have terminated normally.
3. The R_DALI103I_InitInstance function must have terminated normally.
4. The power cycle notification timer must have started by using the R_DALI103I_StartPowerCycleTimer function.

Arguments:

Arguments	Description
dali103i_instance_t * p_this	Pointer to the DALI103i instance module

Return values:

None

3.5.20 R_DALI103I_ClearInstanceError

Overview:

This function clears the instance error of the specified DALI103i instance module.

Format:

```
void R_DALI103I_ClearInstanceError(dali103i_instance_t * p_this)
```

Prerequisites:

1. The R_DALI103I_InitLibrary function must have terminated normally.
2. The R_DALI103I_InitLogicalUnit function must have terminated normally.
3. The R_DALI103I_InitInstance function must have terminated normally.
4. The power cycle notification timer must have started by using the R_DALI103I_StartPowerCycleTimer function.

Arguments:

Arguments	Description
dali103i_instance_t * p_this	Pointer to the DALI103i instance module

Return values:

None

3.5.21 R_DALI103I_InstanceIsActive

Overview:

This function acquires information about whether the specified DALI103i instance module is active.

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

Format:

```
bool R_DALI103I_InstanceIsActive(const dali103i_instance_t * p_this)
```

Prerequisites:

1. The R_DALI103I_InitLibrary function must have terminated normally.
2. The R_DALI103I_InitLogicalUnit function must have terminated normally.
3. The R_DALI103I_InitInstance function must have terminated normally.
4. The power cycle notification timer must have started by using the R_DALI103I_StartPowerCycleTimer function.

Arguments:

Arguments	Description
const dali103i_instance_t	Pointer to the DALI103i instance module

Return values:

Value	Description
true	The instance is active.
false	The instance is inactive.

3.5.22 R_DALI103I_GetPowerCycleNotification

Overview:

This function acquires the Power Cycle Notification event.

Format:

```
dali103i_event_t R_DALI103I_GetPowerCycleNotification(dali103i_t * p_this)
```

Prerequisites:

1. The R_DALI103I_InitLibrary function must have terminated normally.
2. The R_DALI103I_InitLogicalUnit function must have terminated normally.
3. The R_DALI103I_InitInstance function must have terminated normally.
4. The power cycle notification timer must have started by using the R_DALI103I_StartPowerCycleTimer function.

Arguments:

Arguments	Description
dali103i_t * p_this	Pointer to the DALI103i module

Return values:

Value	Description
bool is_exist	Information about whether the event exists
dali103i_forward_frame_t frame	If is_exist is set to true, the power cycle notification event is stored.

3.5.23 R_DALI103I_SetInputSignal

Overview:

This function sets an input signal for the specified DALI103i instance module.

Format:

```
void R_DALI103I_SetInputSignal(dali103i_instance_t * p_this,
                               const uint8_t * p_signal)
```

Prerequisites:

1. The R_DALI103I_InitLibrary function must have terminated normally.
2. The R_DALI103I_InitLogicalUnit function must have terminated normally.
3. The R_DALI103I_InitInstance function must have terminated normally.
4. The power cycle notification timer must have started by using the R_DALI103I_StartPowerCycleTimer function.

Arguments:

Arguments	Description
dali103i_instance_t * p_this	Pointer to the DALI103i instance module
const uint8_t * p_signal	Pointer to the input signal array

Return values:

None

(1) Setting of p_signal parameters

Please set start pointer as uint8_t type dimension.

Please keep following condition.

- Set the number of elements in the array to the number obtained by rounding up parameter, “resolution” of the instance that was divided by 8 to the nearest integer.

- The input signal value storage method for the array should be remodeling little endian and LSB aligned.

Example of setting for dimension is below:

Ex.1 Case of resolution = 3, input signal = 0x05

```
uint8_t input_signal[1] = { 0x05 };
```

Ex.2 Case of resolution = 10, input signal = 0x45

```
uint8_t input_signal[2] = { 0x45, 0x01 };
```

Ex.3 Case of resolution = 100, input signal = 0x123456789ABCDEF

```
uint8_t input_signal[13] = { 0xEF, 0xCD, 0xAB, 0x89, 0x67, 0x45, 0x23, 0x01,
                             0x00, 0x00, 0x00, 0x00, 0x00 };
```

3.5.24 R_DALI103I_GetInstanceEventFilter

Overview:

This function notifies an event filter of the specified DALI103i instance module.

Format:

uint32_t R_DALI103I_GetInstanceEventFilter(const dali103i_instance_t * p_this)

Prerequisites:

1. The R_DALI103I_InitLibrary function must have terminated normally.
2. The R_DALI103I_InitLogicalUnit function must have terminated normally.
3. The R_DALI103I_InitInstance function must have terminated normally.
4. The power cycle notification timer must have started by using the R_DALI103I_StartPowerCycleTimer function.

Arguments:

Arguments	Description
const dali103i_instance_t * p_this	Pointer to the DALI103i instance module

Return values:

Value	Description
uint32_t	Value specified for the instance event filter

3.5.25 R_DALI103I_GenerateInputNotification

Overview:

This function generates an input notification event for the specified DALI103i instance module.

For the dali103i_instance_t type instance with Instance Type 0, an input notification event is not defined in the standard. Therefore, you can use this function to generate a user-defined event. Whether to send the generated event affects the event filter settings. Therefore, use this function together with the R_DALI103I_GetInstanceEventFilter function.

Format:

```
dali103i_event_t R_DALI103I_GenerateInputNotification(const dali103i_t * p_this,
                                                      const dali103i_instance_t * p_instance,
                                                      uint16_t event_info)
```

Prerequisites:

1. The R_DALI103I_InitLibrary function must have terminated normally.
2. The R_DALI103I_InitLogicalUnit function must have terminated normally.
3. The R_DALI103I_InitInstance function must have terminated normally.
4. The power cycle notification timer must have started by using the R_DALI103I_StartPowerCycleTimer function.

Arguments:

Arguments	Description
const dali103i_t * p_this	Pointer to the DALI103i module
const dali103i_instance_t * p_instance	Pointer to the DALI103i instance module
uint16_t event_info	Event information (lower 10 bits)

Return values:

Value	Description
bool is_exist	Information about whether the event exists
dali103i_forward_frame_t frame	If is_exist is set to true, the input notification event is stored.

3.5.26 R_DALI103I_GetTestFrame

Overview:

This function acquires test frames.

Format:

dali103i_test_frame_t R_DALI103I_GetTestFrame(dali103i_t * p_this)

Prerequisites:

1. The R_DALI103I_InitLibrary function must have terminated normally.
2. The R_DALI103I_InitLogicalUnit function must have terminated normally.
3. The R_DALI103I_InitInstance function must have terminated normally.
4. The power cycle notification timer must have started by using the R_DALI103I_StartPowerCycleTimer function.

Arguments:

Arguments	Description
dali103i_t * p_this	Pointer to the DALI103i module

Return values:

Value	Description
uint8_t num	Number of test frames
dali103i_forward_frame_t frame	Array of test frames

3.5.27 R_DALI103I_IdentificationIsActive

Overview:

This function acquires information about whether the identification is active. Identification is a temporary status used in a test run so that the Input Device provider can identify a specific Input Device.

As long as the return value of this function is true, the user application must control the lighting device by using discriminable operations. To do this, visual or auditory means, such as blinking LEDs or generated sounds, and wireless transmission to a smart device and tools can be used.

Note that processing for the identification is an optional function.

Format:

```
bool R_DALI103I_IdentificationIsActive(const dali103i_t * p_this)
```

Prerequisites:

1. The R_DALI103I_InitLibrary function must have terminated normally.
2. The R_DALI103I_InitLogicalUnit function must have terminated normally.
3. The R_DALI103I_InitInstance function must have terminated normally.
4. The power cycle notification timer must have started by using the R_DALI103I_StartPowerCycleTimer function.

Arguments:

Arguments	Description
const dali103i_t * p_this	Pointer to the DALI103i module

Return values:

Value	Description
true	Identification is active.
false	Identification is inactive.

3.5.28 R_DALI103I_QuiescentModeIsActive

Overview:

This function acquires information about whether the Quiescent Mode is active.

The user application is prohibited from sending a Forward Frame as long as the return value of this function is true.

Format:

```
bool R_DALI103I_QuiescentModeIsActive(const dali103i_t * p_this)
```

Prerequisites:

1. The R_DALI103I_InitLibrary function must have terminated normally.
2. The R_DALI103I_InitLogicalUnit function must have terminated normally.
3. The R_DALI103I_InitInstance function must have terminated normally.
4. The power cycle notification timer must have started by using the R_DALI103I_StartPowerCycleTimer function.

Arguments:

Arguments	Description
const dali103i_t * p_this	Pointer to the DALI103i module

Return values:

Value	Description
true	The Quiescent Mode is active.
false	The Quiescent Mode is inactive.

3.5.29 R_DALI103I_CreateCommand

Overview:

This function creates command information that allows this library to process the 24-bit Forward Frame received by the R_DALI103I_CreateCommandDALI communication driver.

Format:

```
dali103i_cmd_t R_DALI103I_CreateCommand(dali103i_t * p_this,
                                        uint32_t forward,
                                        bool twice)
```

Prerequisites:

1. The R_DALI103I_InitLibrary function must have terminated normally.
2. The R_DALI103I_InitLogicalUnit function must have terminated normally.
3. The R_DALI103I_InitInstance function must have terminated normally.
4. The power cycle notification timer must have started by using the R_DALI103I_StartPowerCycleTimer function.

Arguments:

Arguments	Description
const dali103i_t * p_this	Pointer to the DALI103i module
uint32_t forward	Received 24-bit Forward Frame
bool twice	Twice state true: The same frame as the one received within the last 100 ms false: Other than the above

Return values:

Member variables	Description
uint8_t address_byte	Address byte data
uint8_t instance_byte	Instance byte data
uint8_t opcode_byte	Opcode byte data
bool is_received_twice	Twice state
dali103i_daddr_t device_addressing	Device addressing
dali103i_iaddr_t instance_addressing	Instance addressing
dali103i_cmd_type_t type	Command type
dali103i_target_t target	Command execution target

3.5.30 R_DALI103I_ExecuteCommand

Overview:

This function executes the received DALI command.

Send the Backward Frame to the DALI communication bus according to the return value of this function.

Format:

```
int16_t R_DALI103I_ExecuteCommand(dali103i_t * p_this,
                                  const dali103i_cmd_t * p_cmd)
```

Prerequisites:

1. The R_DALI103I_InitLibrary function must have terminated normally.
2. The R_DALI103I_InitLogicalUnit function must have terminated normally.
3. The R_DALI103I_InitInstance function must have terminated normally.
4. The power cycle notification timer must have started by using the R_DALI103I_StartPowerCycleTimer function.
5. Command information must have been acquired by using the R_DALI103I_CreateCommand function.

Arguments:

Arguments	Description
dali103i_t * p_this	Pointer to the DALI103i module
const dali103i_cmd_t * p_cmd	Pointer to the command information

Return values:

Value	Description
0x00 to 0xFF	Backward frame
DALI103I_NO_ANSWER	No backward frames
DALI103I_BACKWARD_IS_CORRUPTED	Corrupted backward frame

3.5.31 R_DALI103I_GetLibraryVersion

Overview:

This function acquires the version number of this library.

Format:

```
uint16_t R_DALI103I_GetLibraryVersion(void)
```

Prerequisites:

None

Arguments:

None

Return values:

Value	Description
uint16_t	Version number (format: 0x*XXYY*) *XX*: Major version *YY*: Minor version

Revision History	RL78 Family Input Device Library User's Manual: Basic (103)

Rev.	Date	Description	
		Page	Summary
1.00	8th, Nov., 22	—	First Edition issued

RL78 Family Input Device Library
User's Manual: Basic (103)

Publication Date: Rev.1.00 8th, Nov., 22

Published by: Renesas Electronics Corporation

RL78 Family

Renesas Electronics Corporation

R01US0593EJ0100