User's Manual

RL78 Family

DALI-2 Control Gear Library
User's Manual: Basic (102)

16-bit single chip microprocessor

All information contained in these materials, including products and product specifications, represents information on the product at the time of publication and is subject to change by Renesas Electronics Corp. without notice. Please review the latest information published by Renesas Electronics Corp. through various means, including the Renesas Electronics Corp. website (http://www.renesas.com).

Renesas Electronics
www.renesas.com

Rev.1.01 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.

© 2022 Renesas Electronics Corporation. All rights reserved.

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 Control Gear 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	RL78/I1A User's Manual Hardware	R01UH0169EJ0320
User's Manual Software	Description of CPU instruction set	RL78/I1A 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 Lighing Interface	International Standard for Lighting Control
NVM	Non-Volatile Memory	Non-volatile memory

Table of Contents

1. DALI102 Library Overview

1.1 Overview of library features

As a slave (Control Gear) library for DALI communication, this library realizes the processing of the hardware-independent part of the DALI102 standard.
The DALI standard defines both hardware standards, such as communication timing and variable storage, and software standards, such as processing when receiving a forward frame. This library mainly omits the hardware-related part (hardware-dependent part) and power supply control to provide general purpose.
To use this library, it is necessary to understand the DALI standard.

Table 1.1 Processing range

User creation processing	Library processing
・H/W setting ・DALI communication driver ・Timer control ・Memory bank entity/control ・Dimmer control ・Non-volatile data access ・Error detection	・Received 16-bit forward frame processing ・Transmitted backward frame issuance ・Timing control ・DALI variable manipulation ・Memory bank operation

This library performs processing based on the 16-bit forward frame received through DALI communication. Since the library does not have any hardware-dependent part, the processing is performed by specifying the 16-bit forward frame received by the user application to the library using the specified API function.
There are various commands specified in the 16-bit forward frame, such as DALI variable setting commands and DALI variable setting value acquisition commands. If the application needs to change the settings, the application will be notified as necessary.
This library can realize multiple logical units in one device. It can also support multiple memory bank implementations. The maximum number of logical units that can be supported and the bank numbers of the memory banks that can be implemented are as follows

Table 1.2 logical unit and memory bank specifications

Item	Value
Maximum number of logical units	64Note
Implementable memory bank number	0 to 199

Note : As a DALI system, the maximum number of Control Gear connections for a single DALI network is 64, and should not exceed 64 in total, taking into account the configuration of the DALI network and hardware limitations.

1.2 Software configuration

The Control Gear software configuration when using this library is shown below.
The part surrounded by the red line is this library. This library realizes the processing of the hardware-independent part. This library is part of the application layer and performs DALI communication, non-volatile data access, etc. In addition, this library can be extended by using the DALI2XX library, which is a library of IEC62386 Part 2XX specifications.

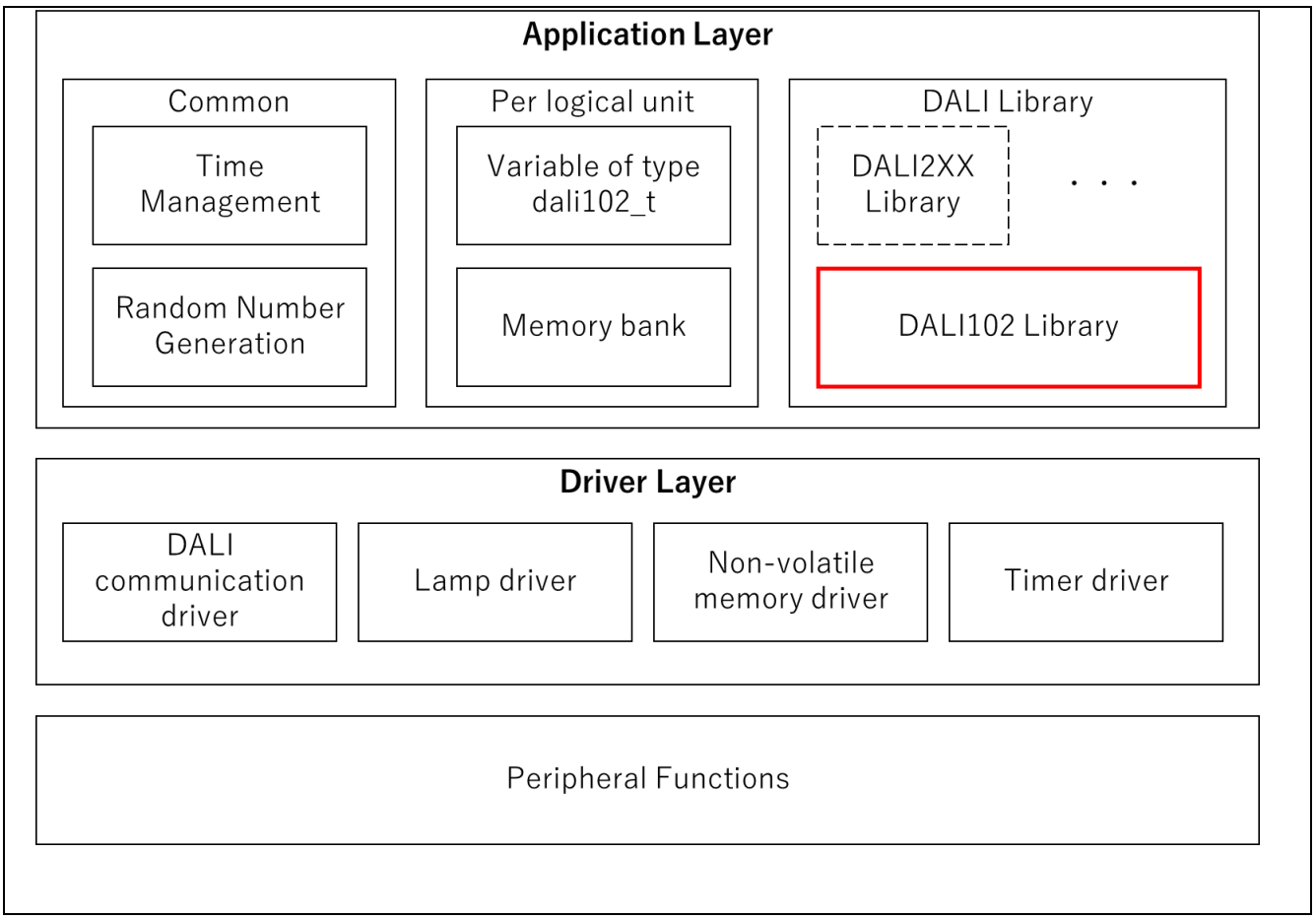


Figure 1.1 Control Gear software configuration diagram

1.3 Supported standard

The standards supported by this library are as follows.

Table 1.3 Supported standard and library name

Supported standard	Compiler	Library name
IEC62386-102 Edition 2.0	Renesas CC-RL V1.10.00	r_dali_102_cc_gen2_v1_00.lib
	IAR C/C++ Compiler for Renesas RL78 V4.21.2.2420	r_dali_102_iar_gen2_v1_00.a

1.4 File list

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

Table 1.4 File list

File name	Description
r_dali_102_cc_gen2_v1_00.lib r_dali_102_iar_gen2_v1_00.a	CC-RL version library file IAR version library file
r_dali102_api.h	Library header file
r_dali102_common.h	Definition header files used in multiple modules
r_dali102_var.h	Definition header files for variable modules
r_dali102_timer.h	Definition header files for timer modules
r_dali102_fade.h	Definition header files for fade modules
r_dali102_mb_if.h	Definition header files of memory bank I/F modules
r_dali102_list.h	Definition header files of list modules
r_dali102_dtx_if.h	Definition header files of generic Device Type I/F modules
r_dali102_dt6_if.h	Definition header files for Device Type 6 I/F modules
r_dali102_dt8_if.h	Definition header files for Device Type 8 I/F modules

1.5 Resource

The library resources (ROM/RAM size and maximum stack size) required by this library are shown below. Table 1.5 Library resource(fixed) lists resources that do not depend on Control Gear implementation content, and Table 1.6 Library resource(variable) lists resources that do not depend on Control Gear implementation content.

Table 1.5 Library resource(fixed)

Compiler	Item		Size
CC-RL	Library resource	ROM size	11,561 [bytes]
		RAM size	4 [bytes]
	Maximum stack size		98 [bytes] (R_DALI102_Tick1ms function)
IAR	Library resource	ROM size	13,245 [bytes]
		RAM size	6 [bytes]
	Maximum stack size		116 [bytes] (R_DALI102_Tick1ms function)

Table 1.6 Library resource(variable)

Compiler	Item	Size
CC-RL	dali102_t	108 [bytes / logical unit]
IAR	dali102_t	108 [bytes / logical unit]

1.6 Development environment

The environment when developing this library is described below.

Table 1.7 Library development environment

Compiler	Item	Description
CC-RL	Integrated development environment	e2studio V2021-04
	C compiler	Renesas CC-RL V1.10.00
	CPU core	RL78-S2 core
	Optimization level	Code size precedence
	Language standard	GNU ISO C99
IAR	Integrated development environment	IAR Embedded Workbench for Renesas RL78 V8.5.2.7561
	C compiler	IAR C/C++ Compiler For Renesas RL78 V4.21.3.2447
	CPU core	RL78-S2 core
	Optimization level	Code size precedence
	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. The loop processing of programs containing this library should be able to run for less than 1 ms at maximum. An environment in which loop processing runs for more than 1 ms will not meet the DALI standard specifications.
3. The dali102_t type structure and dali102_cmd_t type structure are reference-only structures.

2. Programming environment

The hardware and software environments required for users to perform Control Gear operations using this library are described in this chapter.

2.1 Hardware requirement

2.1.1 DALI communication circuit

To realize DALI communication, a communication circuit that performs the operations specified in the IEC62386-101 standard is required.

2.1.2 Non-volatile area

Since Control Gear has specifications to save data in a non-volatile area called NVM data and retain data values even after power is reconnected, a dedicated non-volatile area should be secured.

2.1.3 Dimmer Control Circuit

Control Gear is specified the brightness of the lighting equipment by the command from the Application Controller. A circuit is required to control the lighting according to the brightness obtained via this library.

2.1.4 Failure detection mechanism

Control Gear needs to detect operational failures, retain the status in an internal variable, and then respond to the Application Controller's inquiry. For this reason, a hardware failure detection mechanism (e.g., lighting fixture failure) is required.

2.2 Software requirement

2.2.1 DALI102 Module Definition

A unit of a logical Bus unit (corresponding to Control Gear in this document) defined in one piece of hardware is called a logical unit.This library provides a structure type (dali102_t) that summarizes the parameters necessary to configure the logical unit of Control Gear. A dali102_t type variable is called a DALI102 module.

Please define DALI102 modules for as many logical units as you need.

2.2.2 DALI Communication Driver

Implement a driver that controls the DALI communication circuit described in the hardware requirements and satisfies the IEC62386-101 standard. For details of the DALI communication driver, see RL78/I1A DALI Control Gear Basic (102) Dimming (207) Colour Control (209Tc) Sample Application (R01AN6177).

2.2.3 Time Management

This library has API functions that are called periodically for 1ms to perform time management. Implement the 1ms interval timer in the user application and call the R_DALI102_Tick1ms function.
Note that if the timer is inaccurate, it may violate the standard, so make sure that the error in the call interval is less than ±10%.

2.2.4 Random Number Generation

Control Gear is required to generate a non-repeatable 24-bit random number. Implement a function to generate a non-repeatable random number in the range of 0x000000 to 0xFFFFFE in the user application.
The argument and return value formats are as follows. This function is required to use this library.
The implemented function should be registered as a callback function in the GetRandomValue member of the dali102_general_callback_t type structure variable.

Function format	
Argument	None
Return value	uint32_t Value in the range of 0x000000 to 0xFFFFFE

2.2.5 Non-volatile Area Access

Implement a process that performs access to the nonvolatile area described in the hardware requirements. To meet the IEC62396-102 standard, ensure that the write process is completed within 300 ms.

2.2.6 Failure Notification

Call the following API function when an error condition occurs or is resolved by the failure detection mechanism described in the hardware requirements.

- Lamp failure occurs
  R_DALI102_SetLampFailure function
- Lamp failure is cleared
  R_DALI102_ClearLampFailure function
- Failure related to the entire Control Gear occurs
  R_DALI102_SetControlGearFailure function
- Failure related to the entire Control Gear is cleared
  R_DALI102_ClearControlGearFailure function

2.2.7 Dimming Control

Control the DALI communication circuit as described in the hardware requirements, and periodically call the R_DALI102_GetActualLevel function or R_DALI102_GetActualLevelHighRes function to dim the light according to the obtained actual level.
Basically, the dimmer should be dimmed immediately to the dimming rate indicated by the obtained actual level. However, it is necessary to set an appropriate startup time only when the dimming rate is changed from 0% to other than 0%.
The startup time is the time from the time a light is instructed to be turned on to the time it is turned off. Set the time according to the characteristics of the light fixture to be used.
Call the R_DALI102_NotifyBeginStartup function at the start of startup and the R_DALI102_NotifyEndStartup function at the end of startup.

Also, call the R_DALI102_SetLampOn function when the light is turned on, and call the R_DALI102_ClearLampOn function when the light is turned off.

2.2.8 Memory Bank Entity Definition

The structure and contents of the memory bank of the Control Gear are user-dependent, except for some parts, and are not included in this library. Implement the entity of the memory bank by referring to the IEC62386-102 standard.
The specifications specified in the standard are shown in 2.2.8.1 to 2.2.8.3.

2.2.8.1 Memory Bank 0

Memory bank 0 is required for each logical unit and contains information about Control Gear and logical units.

Table 2.1 Memory map of memory bank 0 (1/2)

Address	Description	Default value	Memory access
0x00	Address of last accessible memory location	factory burn-in,	ROM
0x01	Reserved - not implemented	answer NO	n.a.
0x02	Number of 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 2.2 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 and is reserved for setting additional information about the OEM specifications.

Table 2.3 Memory map of memory bank 1 (1/2)

Address	Description	Default value	RESET value	Memory access
0x00	Address of 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 2.4 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 Bank 2 to 199

Memory banks 2 to 199 are memory banks that can be added arbitrarily to each logical unit. The contents of each bank can be freely defined if they meet the basic specifications.

Table 2.5 Memory map of memory bank 2 to 199

Address	Description	Default value	RESET value	Memory access
0x00	Address of 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 Memory Bank Access Function Implementation

This library by itself cannot access the memory bank defined by the user. Therefore, implement each access function according to the prototype of the callback function provided.
For details, refer to the function specification of the R_DALI102_InitLibrary function.
Some access functions are required to use this library, and some are optional. The following table shows whether the implementation of each access function is required or not.

Table 2.6 List of memory bank access function requirements

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 a function that sets the RESET value to all locations of the bank specified by the argument. The format of the argument and return value is as follows. This function is required to use this library.

Function format	
Argument	uint8_t unit Number of logical unit uint8_t bank Bank number to reset
Return value	void

2.2.9.2 READ function

Implement a function to read out the value of the location specified in the argument. The format of the argument and return value is as follows. This function is required to use this library.

Function format	
Argument	uint8_t unit Number of logical unit uint8_t bank Bank number to be read uint8_t location Location of the bank to be read
Return value	int16_t 0x00 - 0xFF: Read value DALI102_MB_IS_NOT_IMPLEMENTED: The specified bank is not implemented DALI102_MB_LOCATION_IS_NOT_IMPLEMENTED: The specified location is not implemented DALI102_MB_EXECUTE_ERROR: Execution error

2.2.9.3 WRITE function

Implement a function to write the value to the location specified in the argument. The format of the argument and return value is as follows. This function is required to use this library.

Function format	
Argument	uint8_t unit Number of logical unit uint8_t bank Bank number to be written to uint8_t location Location of the bank to be written uint8_t data Data to be written
Return value	int16_t 0x00 - 0xFF: Data value written DALI102_MB_IS_NOT_IMPLEMENTED: The specified bank is not implemented DALI102_MB_LOCATION_IS_NOT_IMPLEMENTED: The specified location is not implemented DALI102_MB_EXECUTE_ERROR: Execution error

2.2.9.4 UNLATCH READ function

The IEC62386-102 standard for memory banks allows multibyte data (i.e., data that makes sense as a single piece of data by combining multiple consecutive locations of memory data). If a part of the data value is changed while reading out multibyte data one byte at a time, it will not be valid as a single multibyte data. Therefore, when placing multibyte data in a memory bank, it is recommended that the user implement a function to latch (hold) the data so that the value is not changed between the start and end of reading the target multibyte data. Only when the data latch function for multibyte data is implemented, implement the function to release the data latch during reading of the logical unit specified by the argument. This function is optional to use this library.

Function format	
Argument	uint8_t unit Number of logical unit
Return value	void

2.2.9.5 CANCEL WRITE function

The IEC62386-102 standard for memory banks allows multibyte data (i.e., data that makes sense as a single piece of data by combining multiple consecutive locations of memory data). If a multibyte data is stopped in the middle of updating one byte at a time, it will no longer be valid as a single multibyte data. Therefore, when writing multibyte data, the user needs to have a function to "write data in a batch only when the data from the start to the end has been written" or "write back to the data before updating when the last data has not been written". This function is used to clearly notify the user that the last data has not been written.
Only when the above specification is implemented in the entity definition of the memory bank, implement the function that cancels the writing of the multibyte data of the logical unit specified by the argument. This function is optional to use this library.

Function format	
Argument	uint8_t unit Number of logical unit
Return value	void

3. DALI102 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
dali102_t	DALI102 module type
dali102_cmd_t	Command information type

The definition macros provided by this library are described below.

Table 3.2 List of macros

Macro name	Macro value	Description
DALI102_NO_ANSWER	(int16_t)-1	No Backward data
DALI102_MB_IS_NOT_IMPLEMENTED	(-1)	Memory bank not implemented
DALI102_MB_LOCATION_IS_NOT_IMPLEMENTED	(-2)	Location not implemented
DALI102_MB_EXECUTE_ERROR	(-3)	Execution error

Table 3.3 List of light source type(dali102_light_source_type_t)

Macro name	Macro value	Description
LIGHT_SOURCE_TYPE_LOW_PRESSURE_FLUORESCENT	0	Low Pressure Fluorescent
LIGHT_SOURCE_TYPE_HID	2	HID
LIGHT_SOURCE_TYPE_LOW_VOLTAGE_HALOGEN	3	Low Voltage Halogen
LIGHT_SOURCE_TYPE_INCANDESCENT	4	Incandescent
LIGHT_SOURCE_TYPE_LED	6	LED
LIGHT_SOURCE_TYPE_OLED	7	OLED
LIGHT_SOURCE_TYPE_OTHER	252	Other than listed above
LIGHT_SOURCE_TYPE_UNKNOWN	253	Unknown light source type
LIGHT_SOURCE_TYPE_NO_LIGHT	254	No light source

The return values provided by this library are described below.

Table 3.4 List of return values (dali102_return_t)

Definition	Return value	Description
DALI102_RETURN_OK	0	Normal end
DALI102_RETURN_ERR	-1	Error end

3.2 List of structures

The structures provided by this library are described below.

Definition of the general callback function type structure (dali102_general_callback_t)

```
typdef struct
{
    uint32_t (*GetRandomValue)(void);
} dali102_general_callback_t;
```

Definition of memory bank operation callback function type structure (dali102_mb_if_callback_t)

```
typdef 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);
} dali102_mb_if_callback_t;
```

Definition of the default value type structure (dali102_default_t)

```
typdef struct
{
    uint8_t operating_mode;
    uint8_t phm;
} dali102_default_t;
```

Definition of the NVM variable type structure (dali102_nvm_t)

```
typdef struct
{
	uint8_t last_light_level;
	uint8_t power_on_level;
	uint8_t system_failure_level;
	uint8_t min_level;
	uint8_t max_level;
	uint8_t fade_rate;
	uint8_t fade_time;
	struct
	{
		uint8_t base : 4;
		uint8_t multiplier : 3;
	} extended_fade_time;
	uint8_t short_address;
	uint32_t random_address;
	uint8_t operating_mode;
	uint16_t gear_groups;
	uint8_t scene[SCENE_SIZE];
} dali102_nvm_t;
```

3.3 List of API Functions

The API functions of this library are described below.

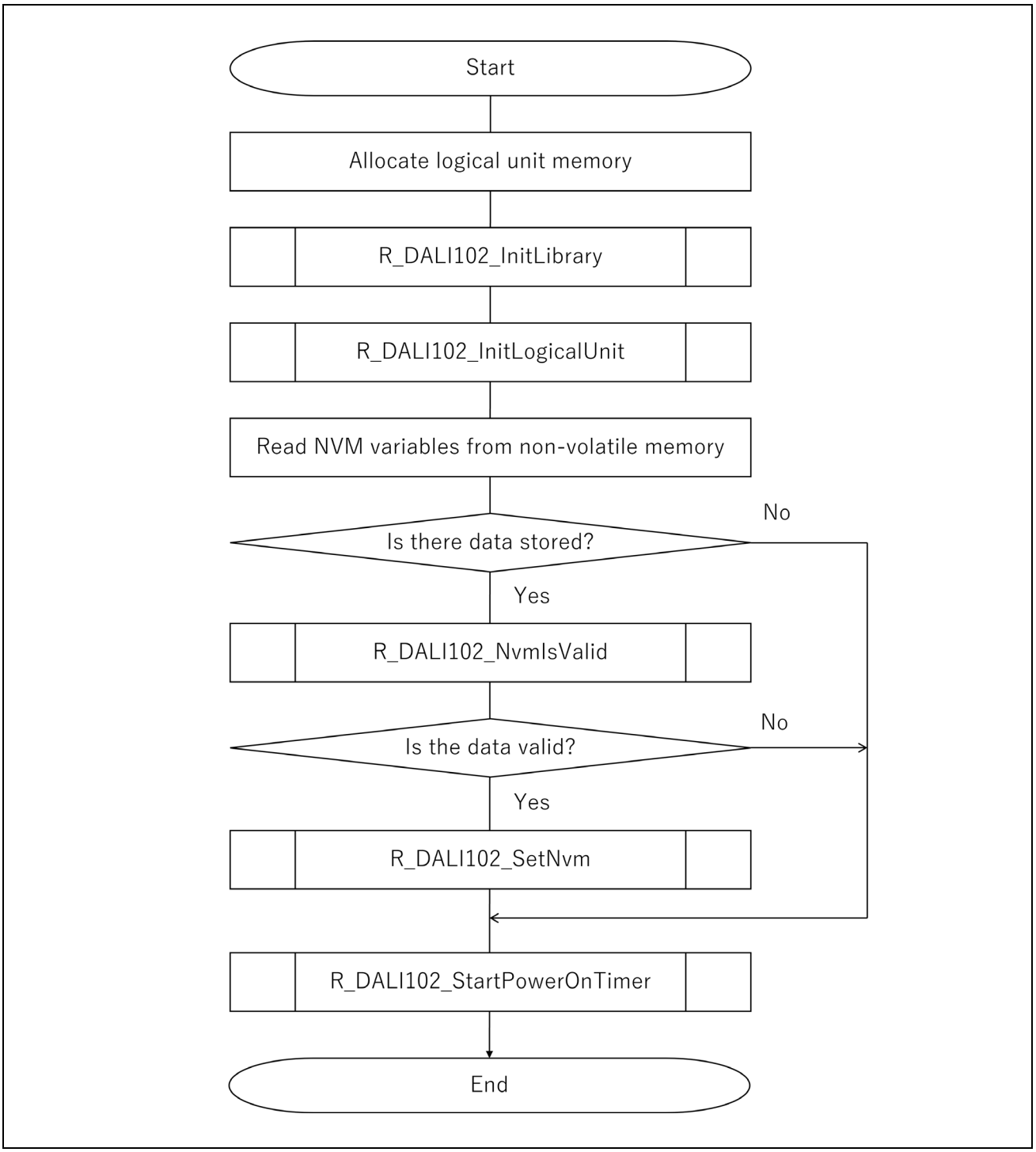
Table 3.5 List of API functions

Function name	Description
R_DALI102_InitLibrary	Initialize the DALI102 library
R_DALI102_InitLogicalUnit	Initialize the logical unit
R_DALI102_NvmIsValid	Check NVM variable values within valid range
R_DALI102_SetNvm	Set the NVM variable value
R_DALI102_GetNvm	Get the NVM variable value
R_DALI102_NvmIsChanged	Check for NVM variable value change
R_DALI102_NeedsToSaveNvm	Check for the need to save an NVM variable
R_DALI102_NotifySaveNvm	Notify saving NVM variables
R_DALI102_StartPowerOnTimer	Start power on timer
R_DALI102_GetOperatingMode	Get operating mode value
R_DALI102_Tick1ms	Progress internal operation for 1 ms
R_DALI102_NotifyBeginStartup	Notify the start of startup
R_DALI102_NotifyEndStartup	Notify end of startup
R_DALI102_SetLampOn	Set lamp on state
R_DALI102_ClearLampOn	Clear lamp on state
R_DALI102_SetLampFailure	Set lamp failure state
R_DALI102_ClearLampFailure	Clear lamp failure state
R_DALI102_SetControlGearFailure	Set control gear failure state
R_DALI102_ClearControlGearFailure	Clear control gear failure state
R_DALI102_NotifySystemFailure	Notify system failure
R_DALI102_GetActualLevel	Get actual level
R_DALI102_GetActualLevelHighRes	Get high-resolution actual level
R_DALI102_IdentificationIsActive	Check for active identification
R_DALI102_CreateCommand	Create command information
R_DALI102_ExecuteCommand	Execute received command
R_DALI102_GetLibraryVersion	Get library version

3.4 Schematic flowchart

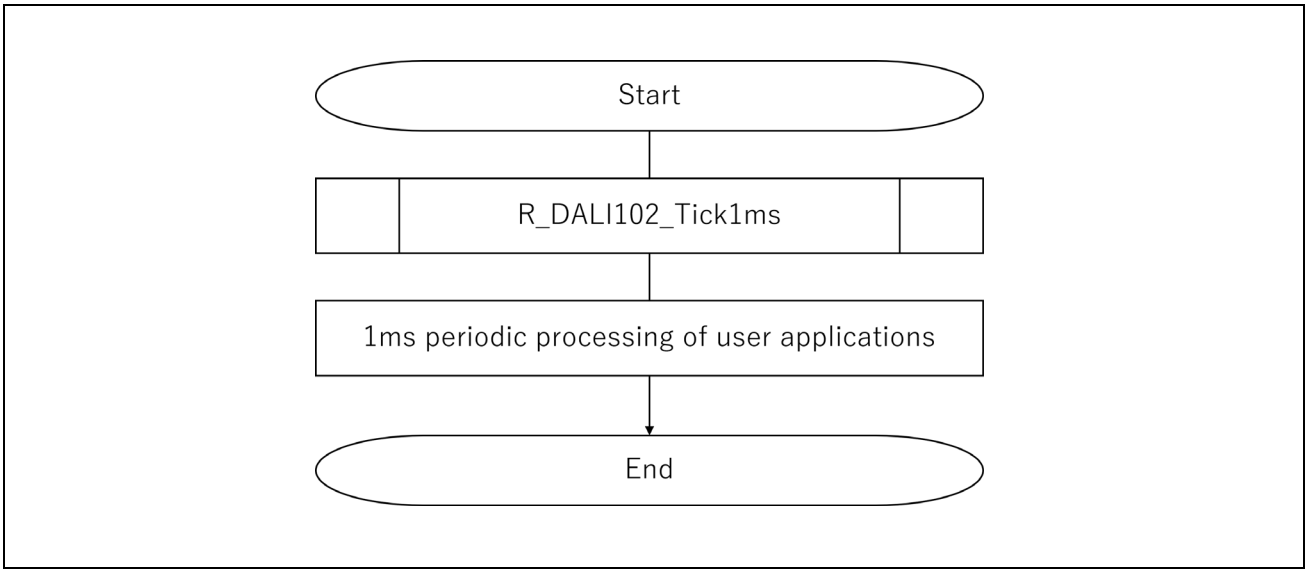
3.4.1 Initialization

The initialization flow is described below.



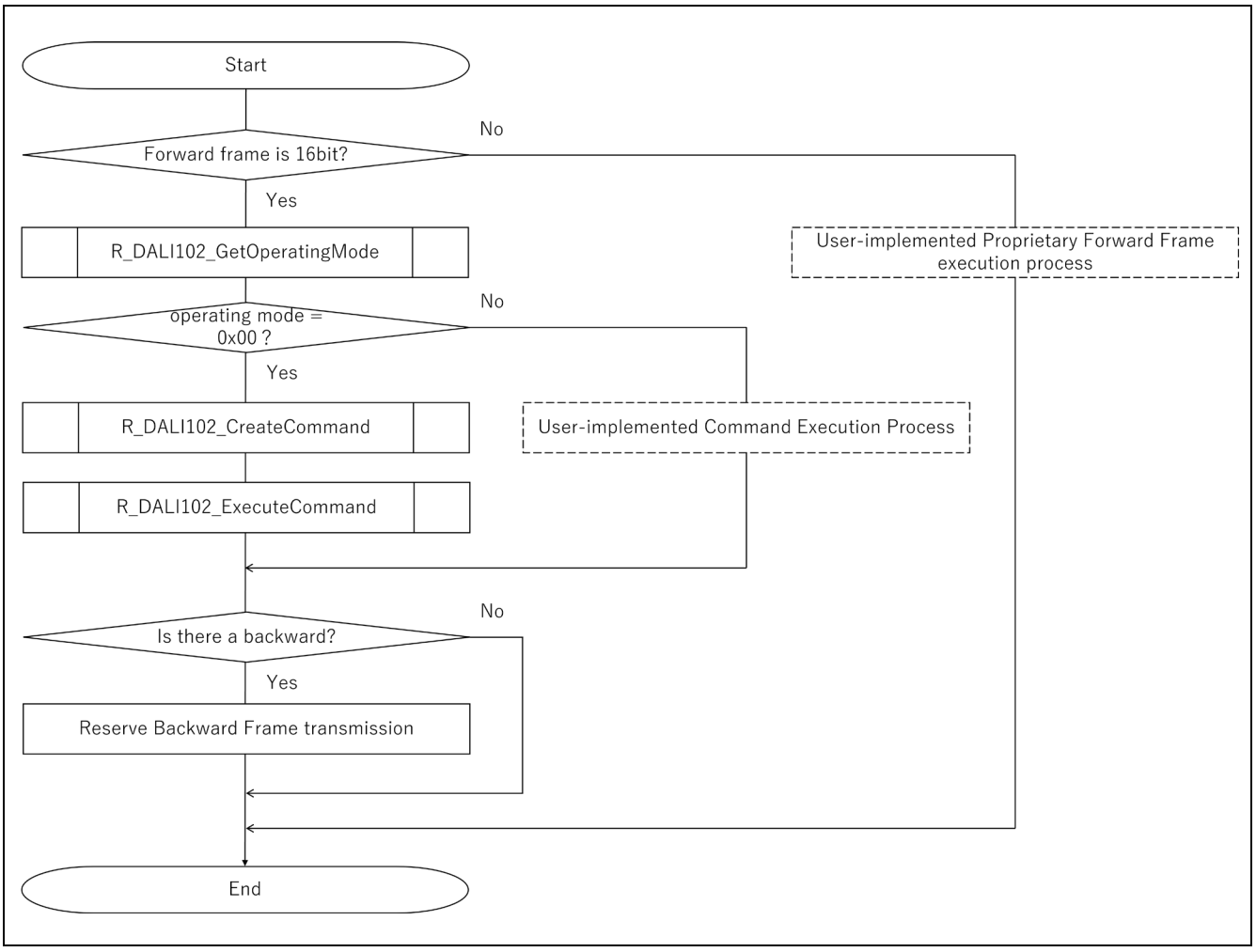
3.4.2 1 ms periodic processing

The flow of the 1ms definition process is described below. This process should be performed at 1ms intervals.



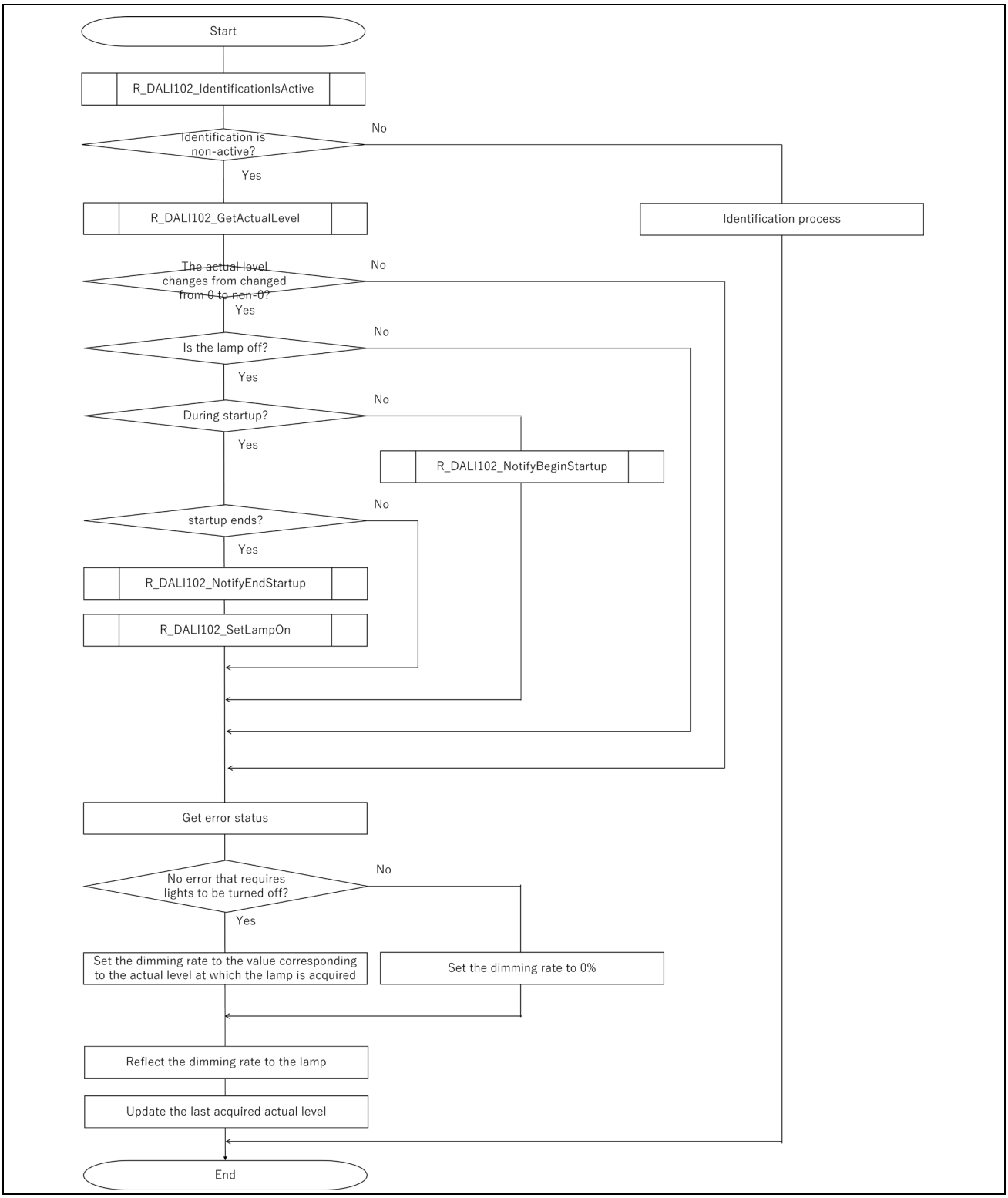
3.4.3 Receiving Forward Frame

The flow of forward frame reception processing is described below and should be performed when a forward frame is received by the DALI communication bus.
Processing for Proprietary Forward Frames (forward frames of more than 16 bits and other than 20 bits and 24 bits) 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 an optional function. If an original mode is required, implement it and register the mode number with the R_DALI102_InitLogicalUnit function.



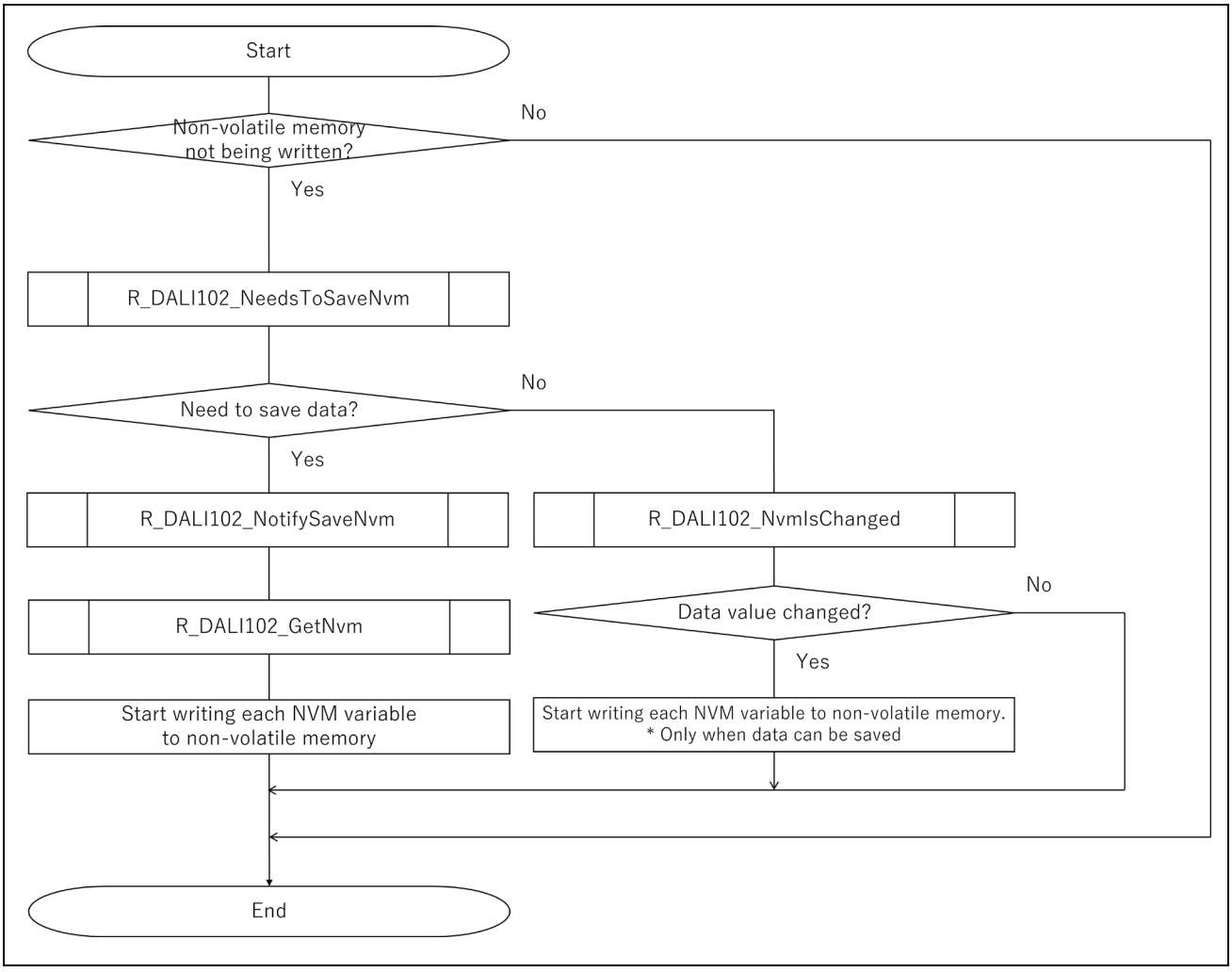
3.4.4 Dimming process

The flow of the dimming process is described below. Perform the process periodically.
The identification process is a user-dependent process. The user must define and implement its use.



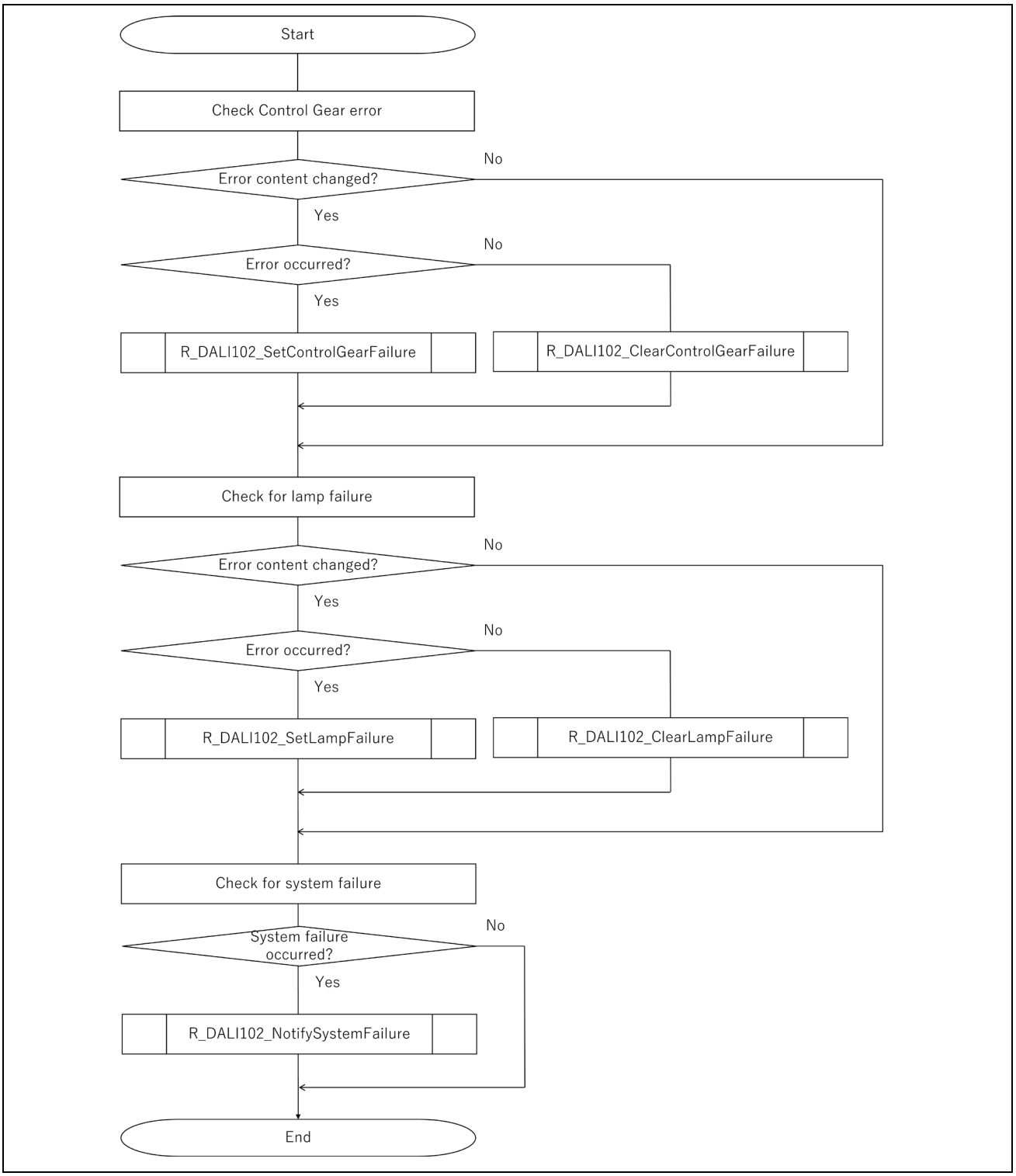
3.4.5 Non-volatile Data Processing

The flow of non-volatile data processing is described below.
It is specified that saving to nonvolatile memory must be completed within 300 ms after receiving the SAVE PERSISTENT VARIABLES command. In addition, if there is a change in the NVM variable value without receiving a SAVE PERSISTANT VARIABLES VARIABLES command, the data must be saved within 30 seconds. Check periodically to ensure that saving is completed within the specified time, and then perform the processing.



3.4.6 Error Handling

The flow of error handling is described below. Call this function when the error status is updated.
The detailed specifications of Control Gear error and Lamp error depend on the hardware and software.
Define the specifications according to the environment and consider implementation.



3.5 API Function Specifications

The API function specifications for this library are listed below.

3.5.1.1 R_DALI102_InitLibrary

[Overview]

Initializes the DALI102 library.

[Format]

```
dali102_return_t R_DALI102_InitLibrary ( const dali102_general_callback_t * p_gen_callback,
                                         const dali102_mb_if_callback_t * p_mb_callback)
```

[Prerequisite]

None in particular.

[Arguments]

Argument	Description
const dali102_general_callback_t * p_gen_callback	Pointer to a generic callback function [Member] .GetRandomValue Set the pointer to the function according to the function format described in Chapter 2.2.4. Setting it to NULL is not acceptable.
const dali102_mb_if_callback_t * p_mb_callback	Pointer to memory bank callback function [Member] .Reset Set the pointer to the function according to the function format described in Chapter 2.2.9.1. Setting it to NULL is not acceptable. .Read Set the pointer to the function according to the function format described in Chapter 2.2.9.2. Setting it to NULL is not acceptable. .Write Set the pointer to the function according to the function format described in Chapter 2.2.9.3. Setting it to NULL is not acceptable. .UnlatchRead Set the pointer to the function according to the function format described in Chapter 2.2.9.4. Since this is an optional feature, setting it to NULL is acceptable. .CancelWrite Set the pointer to the function according to the function format described in Chapter 2.2.9.5. Since this is an optional feature, setting it to NULL is acceptable.

【Return values】

Value	Description
DALI102_RETURN_OK	Normal end
DALI102_RETURN_ERR	Parameter error - Review the argument settings.

3.5.1.2 R_DALI102_InitLogicalUnit

[Overview]

Initializes the specified logical unit.

Call only the number of logical units needed.

[Format]

```
dali102_return_t R_DALI102_InitLogicalUnit ( dali102_t * p_this,
                                             uint8_t unit_index,
                                             uint16_t phm_fade_time_ms,
                                             const dali102_default_t * p_default_value,
                                             const uint8_t * p_light_source_list,
                                             const uint8_t * p_mode_list )
```

[Prerequisite]

1. R_DALI102_InitLibrary function must have ended normally.

[Arguments]

Argument	Description
dali102_t * p_this	Pointer to DALI102 module
uint8_t unit_index	Index number of logical unit Valid range : 0x00 to 0x3F * Set the same value as the index number specified in location 0x1A of memory bank 0 to be used.
uint16_t phm_fade_time_ms	Minimum time that can be physically faded [unit: ms] Valid range : 1 to 700
const dali102_default_t * p_default_value	Default value of factory burn-in Valid range : - phm : 1 to 254 - operating_mode : 0x00, 0x80 to 0xFF
const uint8_t * p_light_source_list	Pointer to the light source type array Refer to the next page for the setting method.
const uint8_t * p_mode_list	Pointer to operating mode array Refer to the next page for the setting method.

[Return values]

Value	Description
DALI102_RETURN_OK	Normal end
DALI102_RETURN_ERR	Parameter error - Review the argument settings.

(1) Set p_light_source_list parameter

Set the first pointer of the uint8_t type array.
Assign the number of light sources to be registered to the first element of the array and the light source type defined in the dali102_light_source_type_t enumeration type to the second and subsequent elements. An example of the array configuration is shown below.

e.g. 1) When only LED is registered

```
uint8_t light_source_list[2] = { 0x01 , LIGHT_SOURCE_TYPE_LED };
```

e.g. 2) When HID and OLED are registered

```
uint8_t light_source_list[3] = { 0x02, LIGHT_SOURCE_TYPE_HID, LIGHT_SOURCE_TYPE_OLED };
```

(2) Set p_mode_list parameter

Set the first pointer of the uint8_t type array.
Assign the number of operating modes to be registered to the first element of the array, and the operating mode value to the second and subsequent elements. The following requirements must be met for use.

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

The following is an example of an array setting.

e.g. 1) When only the DALI specified mode is to be registered

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

e.g. 2) When 0x80 and 0x90 are registered as DALI specified mode and manufacturer specific mode

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

3.5.1.3 R_DALI102_NvmIsValid

[Overview]

Returns whether or not all the values set in the members of the dali102_nvm_t type variable are within the valid range.

Be sure to call and check this function before setting values to the R_DALI102_SetNvm function described below.

[Format]

```
bool R_DALI102_NvmIsValid ( const dali102_t * p_this,
                            const dali102_nvm_t * p_nvm )
```

[Prerequisite]

1. R_DALI102_InitLibrary function must have ended normally.
2. R_DALI102_InitLogicalUnit function has ended normally.

[Arguments]

Argument	Description
const dali102_t * p_this	Pointer to DALI102 module
const dali102_nvm_t * p_nvm	Pointer to NVM variable for DALI102 module Valid range : - last_light_level : 0x00, min_level to max_level - power_on_level : 0x00 to 0xFF - system_failure_level : 0x00～0xFF - min_level : phm to max_level - max_level : min_level to 0xFE - fade_rate : 0x01～0x0F - fade_time : 0x00 to 0x0F - extended_fade_time.base : 0x00 to 0x0F - extended_fade_time.multiplier : 0x00 to 0x04 - short_address : 0x00 to 0x3F, 0xFF - random_address : 0x000000 to 0xFFFFFF - operating_mode : 0x00, 0x80 to 0xFF - gear_groups : 0x0000 to 0xFFFF - scene : 0x00 to 0xFF

[Return values]

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

3.5.1.4 R_DALI102_SetNvm

[Overview]

Sets the NVM variable value to the DALI102 module.
Use this function to set the read data when the NVM variable data is stored in the non-volatile memory at power-on.

[Format]

```
void R_DALI102_SetNvm ( dali102_t * p_this,
                        const dali102_nvm_t * p_nvm )
```

[Prerequisite]

1. R_DALI102_InitLibrary function must have ended normally.
2. R_DALI102_InitLogicalUnit function must have ended normally.
3. R_DALI102_NvmIsValid function verifies that the NVM variable is within the valid range.

[Arguments]

Argument	Description
dali102_t * p_this	Pointer to DALI102 module
const dali102_nvm_t * p_nvm	Pointer to NVM variable for DALI102 module

[Return values]

None

3.5.1.5 R_DALI102_GetNvm

[Overview]

Gets the value of the NVM variable setting from the DALI102 module.
Use this function to store the latest NVM variable values in non-volatile memory.

[Format]

```
void R_DALI102_GetNvm (const dali102_t * p_this,
                       dali102_nvm_t * p_nvm )
```

[Prerequisite]

1. R_DALI102_InitLibrary function must have ended normally
2. R_DALI102_InitLogicalUnit function must have ended normally.

[Arguments]

Argument	Description
const dali102_t * p_this	Pointer to DALI102 module
dali102_nvm_t * p_nvm	Pointer to NVM variable for DALI102 module

[Return values]

None

3.5.1.6 R_DALI102_NvmIsChanged

[Overview]

Gets whether there has been a change in at least one NVM variable value.
If the return value of this function is true, save the NVM variable to the non-volatile memory according to the hardware status.
The status that can be obtained by this function is from the last time this function is called (or at startup when called for the first time). Note that successive calls to this function will result in a false return value.

[Format]

```
bool R_DALI102_NvmIsChanged ( dali102_t * p_this )
```

[Prerequisite]

1. R_DALI102_InitLibrary function must have ended normally.
2. R_DALI102_InitLogicalUnit function must have ended normally.

[Arguments]

Argument	Description
dali102_t * p_this	Pointer to DALI102 module

[Return values]

Value	Description
true	Value change
false	No value change

3.5.1.7 R_DALI102_NvmIsChanged

[Overview]

Get whether or not there was a request to save NVM variables (SAVE PERSISTENT VARIABLES command received).

If the return value of this function is true, save the NVM variable value to the non-volatile memory according to the hardware status. When the saving process becomes possible in response to the save request by this function, notify it by calling the R_DALI102_NotifySaveNvm function described later. The status that can be obtained by this function will not be cleared until notification is made.

[Format]

```
bool R_DALI102_NeedsToSaveNvm ( const dali102_t * p_this )
```

[Prerequisite]

1. R_DALI102_InitLibrary function must have ended normally.
2. R_DALI102_InitLogicalUnit function must have ended normally.

[Arguments]

Argument	Description
const dali102_t * p_this	Pointer to DALI102 module

[Return values]

Value	Description
true	Need to save
false	No need to save

3.5.1.8 R_DALI102_NotifySaveNvm

[Overview]

Call this function when it becomes possible to save NVM variable values to non-volatile memory in response to a save request by the R_DALI102_NeedsToSaveNvm function.
By calling this function as early as possible, it will take longer to accept the next save request.

[Format]

```
void R_DALI102_NotifySaveNvm ( dali102_t * p_this )
```

[Prerequisite]

1. R_DALI102_InitLibrary function must have ended normally.
2. R_DALI102_InitLogicalUnit function must have ended normally.
3. Must be in the true state in the R_DALI102_NeedsToSaveNvm function.

[Arguments]

Argument	Description
dali102_t * p_this	Pointer to DALI102 module

[Return values]

None

3.5.1.9 R_DALI102_StartPowerOnTimer

[Overview]

Starts a timer from power-on until the Power On Level is reflected.
According to the IEC62386-102 standard, the time from power-on until the Power on Level is reflected must be within 600ms±10%.
Set a time that satisfies the standard, considering the time from hardware power-on to the time this function is called.

[Format]

```
void R_DALI102_StartPowerOnTimer ( dali102_t * p_this,
                                   uint16_t msec )
```

[Prerequisite]

1. R_DALI102_InitLibrary function must have ended normally.
2. R_DALI102_InitLogicalUnit function must have ended normally.

[Arguments]

Argument	Description
dali102_t * p_this	Pointer to DALI102 module
uint16_t msec	Time until Power on level is applied

[Return values]

None

3.5.1.10 R_DALI102_GetOperatingMode

[Overview]

Gets the operating mode value during operation.
Switch the processing for the received Forward Frame according to the obtained operating mode setting value. This library provides R_DALI102_ExecuteCommand function for the case where the operating mode is 0.

[Format]

uint8_t R_DALI102_GetOperatingMode (const dali102_t * p_this)

[Prerequisite]

1. R_DALI102_InitLibrary function must have ended normally.
2. R_DALI102_InitLogicalUnit function must have ended normally.

[Arguments]

Argument	Description
const dali102_t * p_this	Pointer to DALI102 module

[Return values]

Value	Description
0x00	This mode is used to process the receive Forward Frame specified in the IEC62386-102 standard. Execute R_DALI102_ExecuteCommand function described below to perform the processing.
0x80 to 0xFF	This is a user-implemented mode. The received Forward Frame should be processed by the user-implemented process corresponding to the operating mode value obtained.

3.5.1.11 R_DALI102_Tick1ms

[Overview]

Advances the internal operation of the DALI102 module by 1 ms.
Call it periodically every 1ms.

[Format]

void R_DALI102_Tick1ms (dali102_t * p_this)

[Prerequisite]

1. R_DALI102_InitLibrary function must have ended normally.
2. R_DALI102_InitLogicalUnit function must have ended normally.

[Arguments]

Argument	Description
dali102_t * p_this	Pointer to DALI102 module

[Return values]

None

3.5.1.12 R_DALI102_NotifyBeginStartup

[Overview]

Notifies the start-up of the lamp.

Call this at the beginning of the preparatory phase that it takes to turn on the light, when the lamp goes from being off to being on.

[Format]

void R_DALI102_NotifyBeginStartup (dali102_t * p_ths)

[Prerequisite]

1. R_DALI102_InitLibrary function must have ended normally.
2. R_DALI102_InitLogicalUnit function must have ended normally.

[Arguments]

Argument	Description
dali102_t * p_this	Pointer to DALI102 module

[Return values]

None

3.5.1.13 R_DALI102_NotifyEndStartup

[Overview]

Notifies the end of lamp startup.
Call this function immediately after the lamp is actually turned on after calling R_DALI102_NotifyBeginStartup function.

[Format]

```
void R_DALI102_NotifyEndStartup ( dali102_t * p_this )
```

[Prerequisite]

1. R_DALI102_InitLibrary function must have ended normally.
2. R_DALI102_InitLogicalUnit function must have ended normally.
3. R_DALI102_NotifyBeginStartup function must have been called in advance.

[Arguments]

Argument	Description
dali102_t * p_this	Pointer to DALI102 module

[Return values]

None

3.5.1.14 R_DALI102_SetLampOn

[Overview]

Notifies the lamp of its lighting status.

Call it immediately after the lamp is turned on.

[Format]

void R_DALI102_SetLampOn (dali102_t * p_this)

[Prerequisite]

1. R_DALI102_InitLibrary function must have ended normally.
2. R_DALI102_InitLogicalUnit function must have ended normally.

[Arguments]

Argument	Description
dali102_t * p_this	Pointer to DALI102 module

[Return values]

None

3.5.1.15 R_DALI102_ClearLampOn

[Overview]

Notifies lamp off.

Call it immediately after the lamp is turned off.

[Format]

void R_DALI102_ClearLampOn (dali102_t * p_this)

[Prerequisite]

1. R_DALI102_InitLibrary function must have ended normally.
2. R_DALI102_InitLogicalUnit function must have ended normally.

[Arguments]

Argument	Description
dali102_t * p_this	Pointer to DALI102 module

[Return values]

None

3.5.1.16 R_DALI102_SetLampFailure

[Overview]

Notifies the occurrence of an abnormality related to the lamp.
Call this function when an abnormality related to the lamp occurs.
The specific abnormality of the lamp depends on the hardware and software. Use this function in relation to the abnormality defined by the user. (e.g., short circuit/open circuit with lamp)

[Format]

```
void R_DALI102_SetLampFailure ( dali102_t * p_this )
```

[Prerequisite]

1. R_DALI102_InitLibrary function must have ended normally.
2. R_DALI102_InitLogicalUnit function must have ended normally.

[Arguments]

Argument	Description
dali102_t * p_this	Pointer to DALI102 module

[Return values]

None

3.5.1.17 R_DALI102_ClearLampFailure

[Overview]

Notifies the clearing of failure related to the lamp.

Call this function when all failures related to the lamp have been cleared.

[Format]

void R_DALI102_ClearLampFailure (dali102_t * p_this)

[Prerequisite]

1. R_DALI102_InitLibrary function must have ended normally.
2. R_DALI102_InitLogicalUnit function must have ended normally.
3. R_DALI102_SetLampFailure function must have been called in advance.

[Arguments]

Argument	Description
dali102_t * p_this	Pointer to DALI102 module

[Return values]

None

3.5.1.18 R_DALI102_SetControlGearFailure

[Overview]

Notifies the failure of the entire Control Gear.
Call this function when a failure related to the entire Control Gear has occurred.
The specific failure details for the entire Control Gear depend on the hardware and software. Use this function in conjunction with the user-defined failure details. (e.g. temperature rise above the guaranteed operating temperature)

[Format]

```
void R_DALI102_SetControlGearFailure (dali102_t * p_this )
```

[Prerequisite]

1. R_DALI102_InitLibrary function must have ended normally.
2. R_DALI102_InitLogicalUnit function must have ended normally.

[Arguments]

Argument	Description
dali102_t * p_this	Pointer to DALI102 module

[Return values]

None

3.5.1.19 R_DALI102_ClearControlGearFailure

[Overview]

Notify the clearing of failure related to the entire Control Gear.

Call this function when all failures related to the entire Control Gear have been cleared.

[Format]

```
void R_DALI102_ClearControlGearFailure ( dali102_t * p_this )
```

[Prerequisite]

1. R_DALI102_InitLibrary function must have ended normally.
2. R_DALI102_InitLogicalUnit function must have ended normally.
3. R_DALI102_SetControlGearFailure function must have been called in advance.

[Arguments]

Argument	Description
dali102_t * p_this	Pointer to DALI102 module

[Return values]

None

3.5.1.20 R_DALI102_NotifySystemFailure

[Overview]

Notifies the occurrence of a system failure.
Call when the DALI communication bus is in the ACTIVE state (LOW level) for 500ms±10%.
Even if the system failure state continues after calling this function, there is no need to call it periodically, and even if the DALI communication bus changes to the IDLE state (HIGH level), the library does not need to perform any processing.

[Format]

```
void R_DALI102_NotifySystemFailure (dali102_t * p_this )
```

[Prerequisite]

1. R_DALI102_InitLibrary function must have ended normally.
2. R_DALI102_InitLogicalUnit function must have ended normally.

[Arguments]

Argument	Description
dali102_t * p_this	Pointer to DALI102 module

[Return values]

None

3.5.1.21 R_DALI102_GetActualLevel

[Overview]

Get the actual level.

Call this function periodically to control the lamp with the dimming rate corresponding to the obtained actual level.

For the correspondence between the actual level and the dimming rate, refer to the IEC62386-102 standard.

[Format]

void R_DALI102_GetActualLevel (const dali102_t * p_this)

[Prerequisite]

1. R_DALI102_InitLibrary function must have ended normally.
2. R_DALI102_InitLogicalUnit function must have ended normally.

[Arguments]

Argument	Description
const dali102_t * p_this	Pointer to DALI102 module

[Return values]

Value	Description
0x00 to 0xFE	actual level value

3.5.1.22 R_DALI102_GetActualLevelHighRes

[Overview]

Get the high-resolution actual level. This function can get the value in the range of 0x0000 to 0xFE00, which is 256 times larger than the actual level range of 0x00 to 0xFE. It is used when more detailed dimming control is required.

Call this function periodically to dim the lamp with the dimming rate corresponding to the acquired actual level.

For the correspondence between the actual level and the dimming rate, refer to the IEC62386-102 standard.

[Format]

void R_DALI102_GetActualLevelHighRes (const dali102_t * p_this)

[Prerequisite]

1. R_DALI102_InitLibrary function must have ended normally.
2. R_DALI102_InitLogicalUnit function must have ended normally.

[Arguments]

Argument	Description
const dali102_t * p_this	Pointer to DALI102 module

[Return values]

Value	Description
0x0000 to 0xFE00	actual level value (unit: actual level / 256)

3.5.1.23 R_DALI102_IdentificationIsActive

[Overview]

Gets whether Identification is in the active state or not. Identification refers to a temporary state used during commissioning that allows the Control Gear installer to identify a specific Control Gear.
While the return value of this function is true, execute the process that can identify the specified logical unit. The content of the identification process during identification is to temporarily ignore the actual level and to dim the lamp at an arbitrary dimming rate from 0 to 100%. Since the specific processing content is user-dependent, the user should implement the content taking into account the set specification. (e.g., blinking at 1-second intervals)

[Format]

```
bool R_DALI102_IdentificationIsActive ( const dali102_t * p_this )
```

[Prerequisite]

1. R_DALI102_InitLibrary function must have ended normally.
2. R_DALI102_InitLogicalUnit function must have ended normally.

[Arguments]

Argument	Description
const dali102_t * p_this	Pointer to DALI102 module

[Return values]

Value	Description
true	Identification is active
false	Identification is not active

3.5.1.24 R_DALI102_CreateCommand

[Overview]

Creates command information that can be processed by this library for 16-bit Forward Frames received through the DALI communication driver.

[Format]

```
dali102_cmd_t R_DALI102_CreateCommand ( uint16_t forward,
                                        bool twice )
```

[Prerequisite]

1. R_DALI102_InitLibrary function must have ended normally.
2. R_DALI102_InitLogicalUnit function must have ended normally.

[Arguments]

Argument	Description
uint16_t forward	Received 16-bit Forward Frame
bool twice	twice status Same frame as the frame received within the last 100ms: true Otherwise: false

[Return values]

Member Variable	Description
dali102_cmd_num_t num	Command number
uint8_t address_byte	address byte data
uint8_t opcode_byte	opcode byte data
bool is_received_twice	twice status
uint16_t execute_condition	Command execution condition

3.5.1.25 R_DALI102_ExecuteCommand

[Overview]

Executes the received DALI command.

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

[Format]

```
int16_t R_DALI102_ExecuteCommand ( dali102_t * p_this,
                                   const dali102_cmd_t *_p_cmd )
```

[Prerequisite]

1. R_DALI102_InitLibrary function must have ended normally.
2. R_DALI102_InitLogicalUnit function must have ended normally.
3. The command information must have been obtained with the R_DALI102_CreateCommand function.

[Arguments]

Argument	Description
dali102_t * p_this	Pointer to DALI102 module
const dali102_cmd_t * p_cmd	Pointer to command information

[Return values]

Value	Description
0x00 to 0xFF	backward frame
DALI102_NO_ANSWER	No backward frame

3.5.1.26 R_DALI102_GetLibraryVersion

[Overview]

Gets the version number of this library.

[Format]

```
uint8_t R_DALI102_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 Control Gear Library User's Manual: Basic (102)

Rev.	Date	Description	
		Page	Summary
1.00	15th, Jun., 22	—	First Edition issued
1.01	1st, Nov., 22	—	Changed several expressions

RL78 Family Control Gear Library
User's Manual: Basic (102)

Publication Date: Rev.1.01 1st, Nov., 22

Published by: Renesas Electronics Corporation

RL78 Family

Renesas Electronics Corporation

R01US0535EJ0101