RX65N Group APPLICATION NOTE

R01AN5396EJ0101

DALI-2 lighting communication using RX65N Cloud kit (ControlDevice/ApplicationController)

Rev.1.01

Nov.21.22

Introduction

This application note describes the operation of a demonstration project equipped with the Application Controller function of the lighting communication standard DALI using the RX65N Cloud Kit.

This application note can be referred to for each purpose.

- Those who want to run the demo environment
  See Chapters 2, 3, and 0.
- Those who want to develop using the demo project
  See in order from Chapter 1.

Target Device

RX65N

Contents

1. Overview of DALI

1.1 About DALI

DALI (Digital Addressable Lighting Interface) is an international open communication protocol for lighting control and is mainly used for controlling multiple fluorescent lights and LED lamps. Lighting equipment made in accordance with the DALI standard can communicate between products of different manufacturers.

Refer to the following application notes for an overview of the standard and for Control Gear and Control Device.

About Control Gear：
Lighting Communications Using RL78/I1A(Reception) Application Note(r01an1115)
About Control Device：
Lighting Communications Using RL78/I1A (Transmission) Application Note(r01an3193)

This application note describes the contents related to operation with a Control Device that is not described in the above application notes.

1.2 Receiver function and Transmitter function

When transmitting and receiving a Frame, there are a Receiver function to receive a Frame and a Transmitter function to send a Frame. The Receiver function and the Transmitter function differ in the types of frames and bit timing definitions that must be supported depending on the BUS Unit used.

The following table shows the summary of Receiver and Transmitter specifications in BUS Unit.

Table 1 Transmitters and Receivers in BUS units

<table>
<tr><th>BUS Unit</th><th>Receiver</th><th colspan="2">Transmitter</th></tr>
<tr><td>Control Gear</td><td>16bit Forward Frame</td><td colspan="2">Backward frames, following the single master timing requirements [a]</td></tr>
<tr><td rowspan="2">Input Device</td><td rowspan="2">24bit Forward Frame</td><td>24bit Forward Frame</td><td rowspan="4">Following the multi-master timing requirements</td></tr>
<tr><td>Backward [a]</td></tr>
<tr><td rowspan="3">Multi-master Application Controller</td><td>24bit Forward Frame</td><td>24bit Forward Frame</td></tr>
<tr><td>16bit Forward Frame [b]</td><td>16bit Forward Frame</td></tr>
<tr><td>Backward Frame</td><td>Backward Frame [a]</td></tr>
<tr><td>Single-master Application Controller</td><td>Backward Frame [c]</td><td colspan="2">16bit forward frames, following the single master timing requirements [d]</td></tr>
<tr><td colspan="4">a No collision detection or collision avoidance methods shall be applied to backward frame transmissions.
b Only applicable when the multi-master application controller is able to process 16bit forward frames transmitted by other application controllers.
c Only required if the single master application controller uses addressing or queries.
d A single master application controller can also send 24bit frames if polling input devices.</td></tr>
</table>

Multi-master Application Controller needs to be equipped with Multi-master transmission, and Single-master Application Controller needs to be equipped with Single-master transmission.

1.3 Overview of DALI Frame

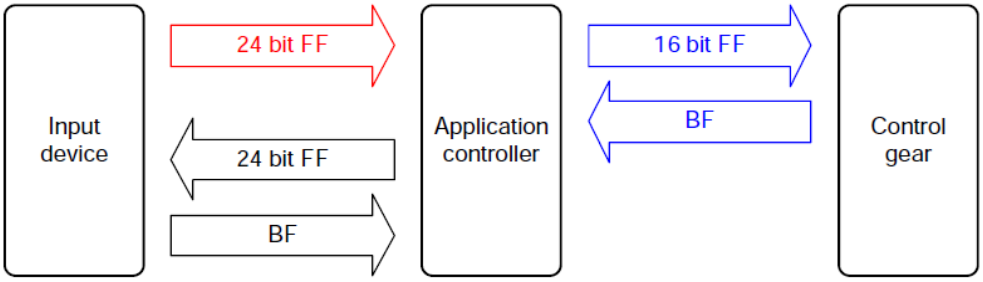

Figure 1 Communication between bus units

By definition, there are Forward Frame (16bit, 24bit) and Backward Frame (8bit). In addition, there are Reserved Forward Frame (20bit, 32bit) and Proprietary Forward Frame as user's own implementation.
Proprietary Forward Frames are classified into two types: a case where the number of data bits is other than the prescribed value, and a case where the start bit, the number of data bits, and the encoding of the stop condition are different. For each BUS Unit that is not designed to interpret the Proprietary Forward Frame, it is necessary to determine the frame size violation and bit timing violation.

2. System overview

In this demonstration project, DALI communication and dimming control of lamps via Control Gear are performed using multiple connection services or devices from the RX65N Cloud Kit + DALI-2 Option board. Three types of operations can be performed depending on the connection service or device.

DALI GUI operation mode: Connects to DALI master controller GUI to control dimming
AWS Web operation mode: Connect to AWS Web application to control dimming
Switch operation mode: Dimming control with matrix button

The system outline diagram of this demonstration project is shown below.

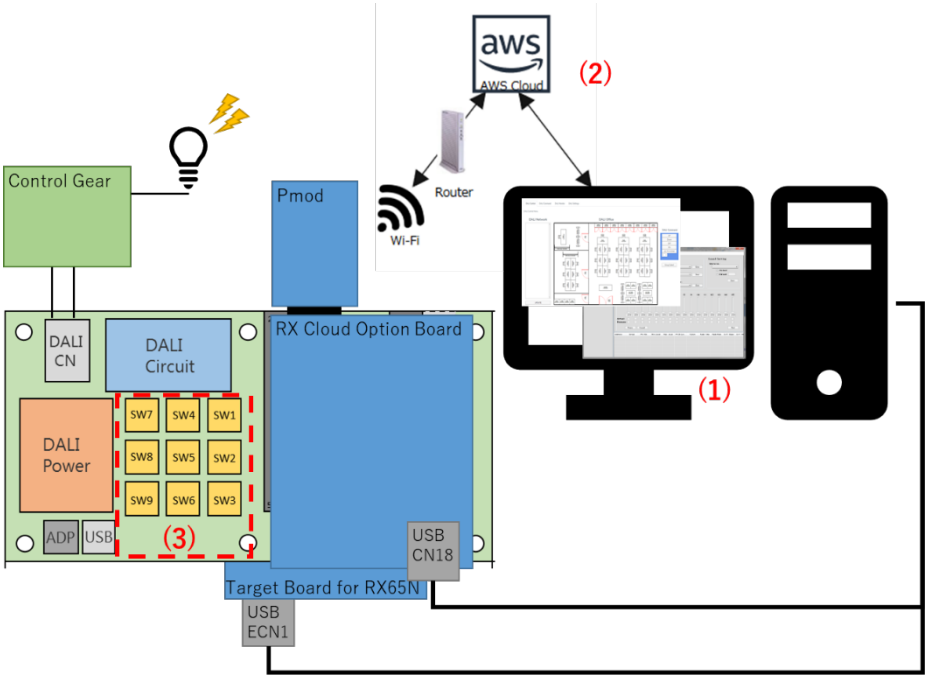

Figure 2 System outline diagram

2.1 DALI GUI operation mode

By connecting and operating the DALI Master Controller GUI to the RX65N Cloud kit + DALI-2 Option board, the dimming of the lamp connected to the Control Gear can be controlled. The DALI Master Controller GUI is a GUI (Graphical User Interface) that can communicate on the PC in accordance with the DALI standard.

2.2 AWS Web operation mode

AWS web applications run on the Google Chrome browser using AWS IoT Core from the AWS Cloud. For communication between the RX65N Cloud kit + DALI-2 Option board and WEB application, use MQTT to communicate via Wi-Fi connection. Simple dimming of the lamp connected to Control Gear can be controlled by operation from the browser.

2.3 Switch operation mode

It works with RX65N Cloud kit + RX65N DALI-2 Option board alone without using connection service. When the matrix button is pressed, the corresponding 16-bit Forward Frame is transmitted, and simple dimming of the lamp connected to the Control Gear can be controlled.

2.4 Download file structure

The file structure for downloading is as follows.
File name: r01an5396xx0ZZZ-dali-2-rx-application-controller.zip
ZZZZ: varies depending on version.

Folder structure
Workspace/
r_aws_setting AWS setting file
r_aws_web AWS-WEB source file
rx65b_dali Rx65N project file
*The folder of rx65b_dali needs to have a smaller hierarchy.

3. Hardware configuration

3.1 Hardware environment

The following table shows the hardware environment used in this demonstration project.

Table 2 Hardware environment list

Item	Content	Provider	Description
Board used	Target Board for RX65N	Renesas Electronics Corporation	Evaluation board with RX65N MCU [a]
	RX Cloud Option Board		AWS Connectable Cloud Communication evaluation board [a]
	Silex Pmod Module		Communication board with wireless LAN module [a] Used only in AWS Web operation mode.
	EZ-0012		Lighting control evaluation board with DALI communication circuit [b]
	RX65N DALI-2 Option board	TESSERA TECHNOLOGY	DALI evaluation board with DALI communication circuit for connecting RX65N Cloud kit [c]
Wi-Fi	Wireless router	-	Used as a Wi-Fi connection destination in AWS Web operation mode. Wireless LAN standard: IEEE 802.11b/g/n (2.4GHz) Encryption method: None, AES
PC	Windows10	-	Recommended OS
	Google Chrome	-	Browser to use Used only in AWS Web operation mode.

a Target Board for RX65N, RX65N Cloud Option Board, and SILEX UART Pmod are included in the RX65N Cloud Option kit.
Please contact Renesas for purchasing.https:/www.renesas.com/rx65n-cloud/

b EZ-0012 is an evaluation board for lighting control equipped with RL78 / I1A. In this demonstration project, it is used as Control Gear. You do not need to prepare a separate light fixture because the on-board LED turns on and off.
* Separate AC power adapter for EZ-0012 is required.
For EZ-0012, it is necessary to make settings using a separate tool. For the setting of EZ-0012, refer to Chapter 5.3.

c RX65N DALI-2 Option board is used as Control Device (Application Controller) in combination with RX65N Cloud Option kit.
The board provider is TESSERA TECHNOLOGY. Please contact TESSERA TECHNOLOGY for purchasing.
https://www.tessera.co.jp/eng/product.html

3.2 Board Configuration Diagram

3.2.1 RX Cloud Option Board

Refer to the following user's manual for the configuration diagram of the RX Cloud Option Board.

RX Family Cloud Option Board User's Manual (r12um0039)

3.2.2 Target Board for RX65N

Refer to the following user's manual for the configuration diagram of Target Board for RX65N.

RX65N Group Target Board for RX65N User's Manual(r12um0038)

3.2.3 RX65N DALI-2 Option board

The following figures show the diagram and dimensions of RX65N DALI-2 Option board.

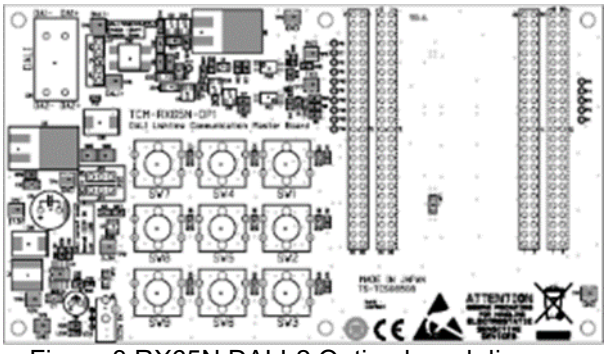

Figure 3 RX65N DALI-2 Option board diagram

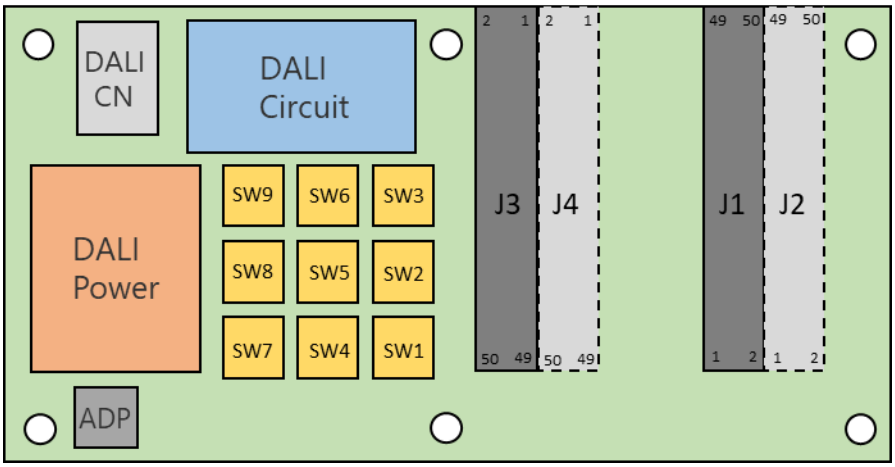

Figure 4 RX65N DALI-2 Option board dimensions

Before connecting each board to be used, RX65N DALI-2 Option board needs to confirm the board settings. Refer to section 0 for board settings.

4. Software configuration

4.1 Overall configuration

The following figure shows the overall software configuration of this demo project.
By using Amazon Free-RTOS function, RX65N peripheral function, and FIT driver, operate the functions in the red frame.

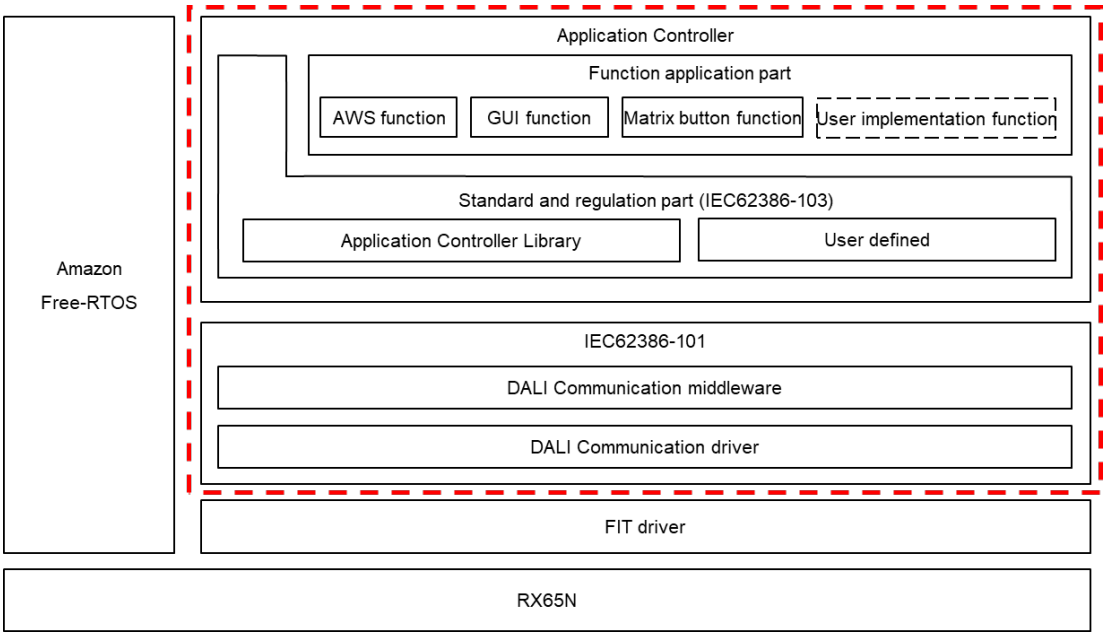

Figure 5 Overall software configuration diagram

Application Controller

Function application part

When transmitting and receiving Frame by using each connection service or device (AWS web application, DALI master controller GUI, matrix button), exchange with each connection service or device.

In the user-implemented function, this is the place where you can implement your own processing operation using this demo project, such as when you want to process the Frame issued from the Input Device.

Standard and regulation part (IEC62386-103)

it operates Application Controller specified by IEC62386-103 standard.

It interprets received Forward Frames and issues Backward Frames for them, and uses the Application Controller Library made by Renesas Electronics for part of its operation.

The user-defined area manages identification operations, memory bank definitions, and data stored in nonvolatile memory.

IEC62386-101

DALI communication middleware

It controls the DALI communication driver, and notifies and manages the Frame with Application Controller or DALI communication driver.

DALI communication driver

By using the RX65N peripheral, it receives Frames issued on the DALI BUS and transmits Frames received from DALI communication middleware.

4.2 Software environment

The following table shows the software environment of this demonstration project.

Table 3 Software environment list

Item	Content	version
language	C	C99
Integrated development environment	e2studio Made by the Renesas electronics corporation	7.6.0
Compiler	GNU RX GCC	8.3.0.2019.4
FIT driver	RX Driver Package	122
	BSP	5.20
	BYTEQ	1.80
	FLASH	4.20
	SCI	3.20
	ADC (S12AD)	4.20
	GPIO	3.20
	CMT	4.20
	Wi-Fi (SX-ULPGN WIFI FIT Module)	1.00
Cord auto-generation	CRC	1.7.0
	MTU	1.7.0
	PORT	1.9.0

4.3 RX65N peripheral functions and FIT module settings

The following figure shows an overview of the FIT module used and the RX65N peripheral functions.

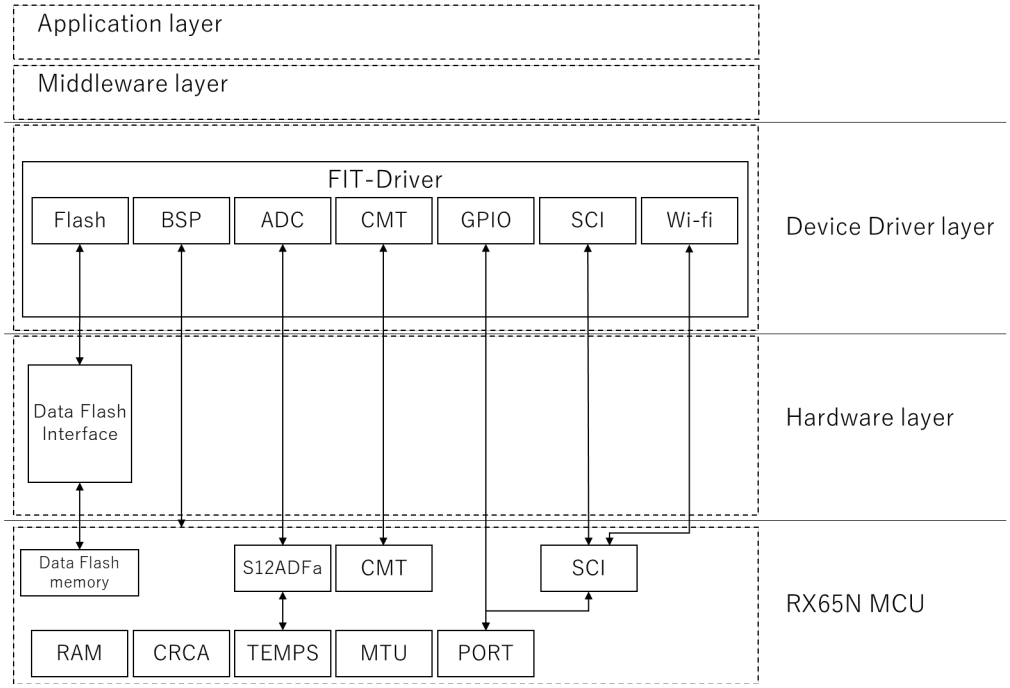

Figure 6 Overview of peripheral functions

4.3.1 Setting of peripheral functions

Table 4 List of peripheral function settings 1/2

item	Settings
CRC	Generator polynomial: CRC_32 Bit order: LSB Defaults: 0x00000000 Do not invert the operation result
MTU0	TCNT0 counter setting TGRA0 compare match / input capture (Use TGRA0 as cycle register) Counter clock: PLCK/4 Falling edge General register settings TGRA0: Input capture register TGRB0~F0: Output compare register / 100us I / O terminal settings Input capture by rising edge of MTIOC0A pin input (Use noise filter) Interrupt settings TGRA Input capture / Enable compare match interrupt Priority: Level 3
MTU1	TCNT1 counter setting TGRA1 compare match / input capture (Use TGRA1 as cycle register) Counter clock: PLCK General register settings TGRA1: Output compare register / 419700ns TGRB1: Output compare register / 396700ns I / O terminal settings Invalid Interrupt settings TGRA Input capture / Enable compare match interrupt Priority: Level 6 TGRB Input capture / Enable compare match interrupt Priority: Level 6
MTU2	TCNT2 counter setting TGRA2 compare match / input capture (Use TGRA2 as cycle register) Counter clock: PLCK/32 Rising edge General register settings TGRA2: Output compare register / 5700us TGRB2: Output compare register / 10190us I / O terminal settings Invalid Interrupt settings TGRA Input capture / Enable compare match interrupt Priority: Level 3 TGRB Input capture / Enable compare match interrupt Priority: Level 3

Table 5 List of peripheral function settings 2/2

item	Settings
MTU3	TCNT3 counter setting Cleared by clearing the counter of another channel operating synchronously Counter clock：PLCK/256 Falling edge General register settings TGRA3: Output compare register / 4150us TGRB3: Output compare register / 14000us TGRC3: Output compare register / 15400us TGRD3: Output compare register / 16800us I / O terminal settings Invalid Interrupt settings TGRA~TGRD Input capture / Enable compare match interrupt Priority: Level 3
MTU4	TCNT4 counter setting TGRA4 compare match / input capture (Use TGRA4 as cycle register) Counter clock: PLCK/256 Rising edge General register settings TGRA4: Output compare register / 18500us TGRB4: Output compare register / 20100us TGRC4: Output compare register / 97000us TGRD4: Output compare register / 12800us I / O terminal settings Invalid Interrupt settings TGRA~TGRD Input capture / Enable compare match interrupt Priority: Level 3
PORT	Output terminal PB1/PB0 P15/P17/P25/P24 Input terminal PC4/PC5/PC6/PE0/PE1/PE2 I / O terminal MTIOC0A

4.3.2 FIT module settings

Table 6 List of FIT module settings

item	Settings
BSP	BSP_CFG_STARTUP_DISABLE="0" BSP_CFG_USER_STACK_ENABLE="1" BSP_CFG_USTACK_BYTES="0x1000" BSP_CFG_ISTACK_BYTES="0x400" BSP_CFG_HEAP_BYTES="0x2000" Other default settings
BYTEQ	BYTEQ_CFG_PARAM_CHECKING_ENABLE="0" BYTEQ_CFG_USE_HEAP_FOR_CTRL_BLKS="0" BYTEQ_CFG_MAX_CTRL_BLKS="32"
FLASH	FLASH_CFG_PARAM_CHECKING_ENABLE="1" FLASH_CFG_CODE_FLASH_ENABLE="1" FLASH_CFG_DATA_FLASH_BGO="1" FLASH_CFG_CODE_FLASH_BGO="1" FLASH_CFG_CODE_FLASH_RUN_FROM_ROM="1"
SCI	Asynchronous communication No parameter check Channels used: CH0, CH1, CH5 SCI_CFG_CH0_TX_BUFSIZ="2180" SCI_CFG_CH1_TX_BUFSIZ="2180" SCI_CFG_CH0_RX_BUFSIZ="4096" SCI_CFG_CH1_RX_BUFSIZ="4096" SCI_CFG_RXERR_PRIORITY="3" SCI_CFG_ERI_TEI_PRIORITY="3" Other default settings
ADC	Default settings
GPIO	Default settings
CMT	CMT interrupt priority level: 3
Wi-Fi	WIFI_CFG_SCI_CHANNEL="0" WIFI_CFG_SCI_SECOND_CHANNEL="1" WIFI_CFG_SCI_INTERRUPT_LEVEL="14" WIFI_CFG_SCI_BAUDRATE="460800" WIFI_CFG_SCI_USE_FLOW_CONTROL="1" WIFI_CFG_RESET_PORT="13" WIFI_CFG_RESET_PIN="0" WIFI_CFG_RTS_PORT="2" WIFI_CFG_RTS_PIN="2" WIFI_CFG_CREATABLE_SOCKETS="4" WIFI_CFG_SOCKETS_RECEIVE_BUFFER_SIZE="8192" WIFI_CFG_USE_CALLBACK_FUNCTION="0" WIFI_CFG_CALLBACK_FUNCTION_NAME="NULL"

4.3.3 Configuration of software components

Table 7 List of software component settings

item	Settings
Device selection	Board: CloudKitRX65N (V.1.00) Device: R5F565NEDxFP

4.3.4 FreeRTOS Resources

FreeRTOS Resources are described below.

Table 8 List of FreeRTOS Resources(Task)

item	Name	priority	Stack size	content
Task	r_app_event_control_task	1	150	Manages notifications for each application task.
Task	r_app_1msec_interval_task	3	150	Perform 1ms periodic processing.
Task	r_app_recv_dali_frame_task	1	150	Each application task is notified from the contents of the receive queue.
Task	r_gui_recv_com_task	2	150	Performs command processing of GUI application.
Task	r_gui_report_com_task	2	150	Assign a command to the specified matrix button from AWS Matrix button input sets the command to the send queue of softDALI middleware.
Task	r_btn_sw_input_task	1	150	Assign a command to the specified matrix button from AWS.
Task	r_btn_recv_com_task	2	150	Matrix button input sets the command to the send queue of softDALI middleware.
Task	r_sdmdl_transfer_control_rx_task	3	256	Forward frames are sorted by priority by DALI transfer control and set in the transmission queue.
Task	r_sdmdl_transfer_control_task	1	256	Performs software DALI middleware transmission processing
Task	r_sdmdl_transfer_control_tx_task	1	256	Software DALI middleware reception processing
Task	prvAWStoDALITask	0	1024	Controls between AWS and DALI tasks.
Task	prvDALILoggerTask	0	1024	Control logs between AWS-DALI tasks.

Table 9 List of FreeRTOS Resources(Message Queue)

item	Name	num	size	content
Message Queue	gs_app_qhandle_rx_frame_to_gui	4	50	Queue that passes received frames to GUI
Message Queue	gs_app_qhandle_rx_frame_to_aws_for_log	5	50	Queue that passes the received frame to AWS (for log)
Message Queue	gs_app_qhandle_rx_frame_to_aws_for_cmd	2	50	Queue for passing received frames to AWS (for AWS command)
Message Queue	gs_app_qhandle_aws_to_btn	2	50	Queue to pass frame data from AWS to BTN
Message Queue	gs_app_qhandle_btn_to_aws	2	50	Queue to pass frame data from BTN to AWS
Message Queue	g_sdmdl_qhandle_tx_forward_frame	10	50	Queue to pass frame data from application to MDL
Message Queue	g_sdmdl_qhandle_tx_backward_frame	5	50	Queue to pass backward data from application to MDL
Message Queue	g_sdmdl_qhandle_rx_frame_sdmdl_to_app	10	50	Queue to pass frame data from MDL to application
Message Queue	g_sdmdl_qhandle_rx_frame_sddrv_to_sdmdl	5	36	Queue to pass frame data from software DALI driver to MDL
Message Queue	g_sdmdl_qhandle_rx_resp_frame_type	5	4	Queue used inside MDL
Message Queue	g_sdmdl_qhandle_sddrv_rx_event	20	4	Queue to pass incoming events from software DALI driver to MDL

Table 10 List of FreeRTOS Resources(Event Flag)

item	Name	content
Event Flag	g_app_ehandle_task_information	Management of each APP task event
Event Flag	g_sdmdl_ehandle_sdmdl_to_sddrv	Event flag for sending notification to MDL

Table 11 List of FreeRTOS Resources (Timer)

item	Name	Time(ms)	content
Timer	g_sdmdl_thandle_transfer_rx	1000	MDL receive data timeout
Timer	gs_app_thandle_1msec_timer	1	Software timer 1ms
Timer	gs_gac_sci_timer_handle	20	For UART timeout
Timer	g_sys_fail_handle	534	For system failure detection

4.4 Folder Structure

The following table shows the file structure of this demo project.

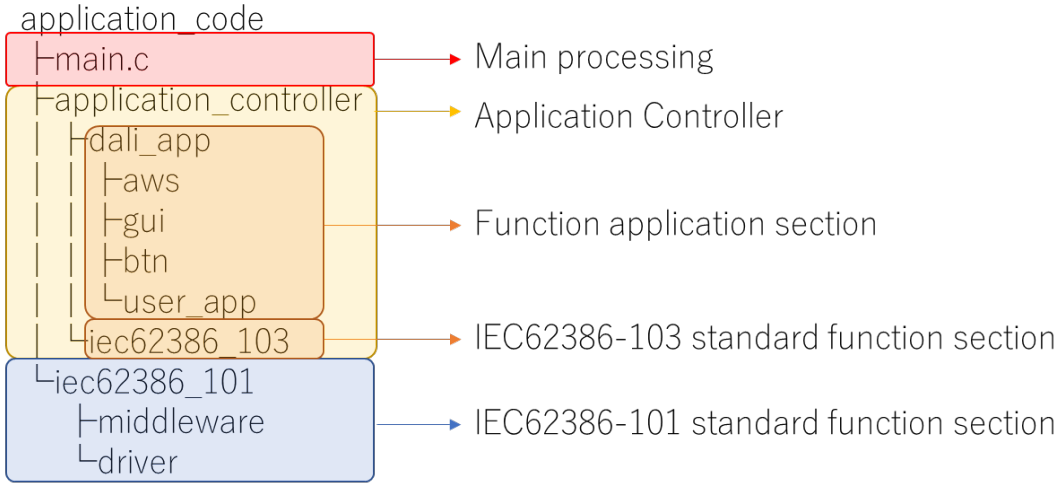

Figure 7 Folder structure

4.5 File Structure

The file list of each function described in this demo project is shown below.

Table 12 File organization 1/3

Function name	File name	Description
Main process	main.c	Main file of this demo project
Application Controller	r_app_api_main.c	Main file for Application Controller
	r_app_api_main.h	Main header file for Application Controller
	r_app_task.c	Source file for Application Controller task
	r_app_task.h	Header file for Application Controller task
	r_app_queue.c	Source file for Queue used by Application Controller
	r_app_queue.h	Header file for Queue used by Application Controller
	r_app_df_access.c	Source file to access data flash
	r_app_df_access.h	Header file to access data flash
	r_app_df_access_user.h	Definition header file used to access data flash
Function application part	r_aws_control_task.c	Source file that controls connection with AWS via MQTT
	r_aws_control_task.h	Header file that controls connection with AWS via MQTT
	r_aws_dali_task.c	Source file for tasks used in AWS web applications
	r_aws_raa_process.c	Source file for random address allocation via AWS.
	r_aws_raa_process.h	Header file for random address allocation via AWS.

Table 13 File organization 2/3

Function name	File name	Description
Function application part	r_gui_api.c	Source file for GUI application
	r_gui_api.h	Header file for GUI application
	r_gui_task.c	Source file for GUI application task
	r_gui_task.h	Header file for GUI application task
	r_btn_api.c	Source file for matrix button application
	r_btn_api.h	Header file for matrix button application
	r_btn_task.c	Source file for matrix button application task
	r_btn_task.h	Header file for matrix button application task
	r_mng_user.c	Source file for user implementation
	r_mng_user.h	Header file for user implementation
IEC62386-103 standard functions	r_dali103_appctrl_api.a	Library file for Application Controller library
	r_dali103_appctrl_api.h	Header file for Application Controller library
	r_dali103_api.c	Source file for Application Controller application
	r_dali103_api.h	Header file for Application Controller application
	r_dali103_mbanks_access.c	Source file for memory bank (external) used by Application Controller
	r_dali103_mbanks_access.h	Header file for memory bank (external) used by Application Controller
	r_dali103_mbanks_entify.c	Source file for memory bank (definition) used by Application Controller
	r_dali103_mbanks_entify.h	Header file for memory bank (definition) used by Application Controller
	r_dali103_mbank_access.c	Source file for memory bank (internal) used by Application Controller
	r_dali103_mbank_access.h	Header file for memory bank (internal) used by Application Controller
	r_dali103_random_seed.c	Source file for generating Seed values used for random numbers
	r_dali103_random_seed.h	Header file for generating Seed values used for random numbers

Table 14 File organization 3/3

Function name	File name	Description
IEC62386-101 standard functions	r_sdmdl_api.c	Source file for DALI communication middleware
	r_sdmdl_api.h	Header file for DALI communication middleware
	r_sdmdl_transfer.c	Source file for error handling of DALI communication middleware
	r_sdmdl_transfer.h	Definition header file used by DALI communication middleware
	r_sdmdl_transfer_list.c	Source file for list control used in DALI communication middleware
	r_sdmdl_transfer_list.h	Header file for list control used in DALI communication middleware
	r_sdmdl_task_transfer_control.c	Source file for DALI communication middleware task
	r_sdmdl_task_transfer_control_rx.c	Source file for receiving DALI communication middleware
	r_sdmdl_task_transfer_control_tx.c	Source file for transmitting DALI communication middleware
	r_sddrv_def.h	Internal definition header file for DALI communication driver
	r_sddrv_user.h	Header file related to user definition of DALI communication driver
	r_sddrv_api.c	Source file for DALI communication driver.
	r_sddrv_api.h	Header file for DALI communication driver
	r_sddrv_com.c	Source file for DALI communication driver application side communication related
	r_sddrv_com.h	Header file for DALI communication driver application side communication related
	r_sddrv_frame.c	Source file for Frame decoding / encoding of DALI communication driver
	r_sddrv_frame.h	Header file for Frame decoding / encoding of DALI communication driver
	r_sddrv_gpio.c	Source files related to DALI communication driver ports
	r_sddrv_gpio.h	Header files related to DALI communication driver ports
	r_sddrv_rx.c	Source file for receiving DALI communication driver
	r_sddrv_rx.h	Header file for receiving DALI communication driver
	r_sddrv_timer.c	Source file related to timer of DALI communication driver
	r_sddrv_timer.h	Header file related to timer of DALI communication driver
	r_sddrv_tx.c	Source file for transmitting DALI communication driver
	r_sddrv_tx.h	Header file for transmitting DALI communication driver

4.6 Function Overview

In this demonstration project, Application Controller is operated by using the event notification and Queue notification function of the task of Free-RTOS. Tasks are divided for each function of Application Controller, and each function uses multiple tasks.
However, the DALI communication driver is controlled using timer interrupts, not tasks.
The basic operation of the demo project is shown below.

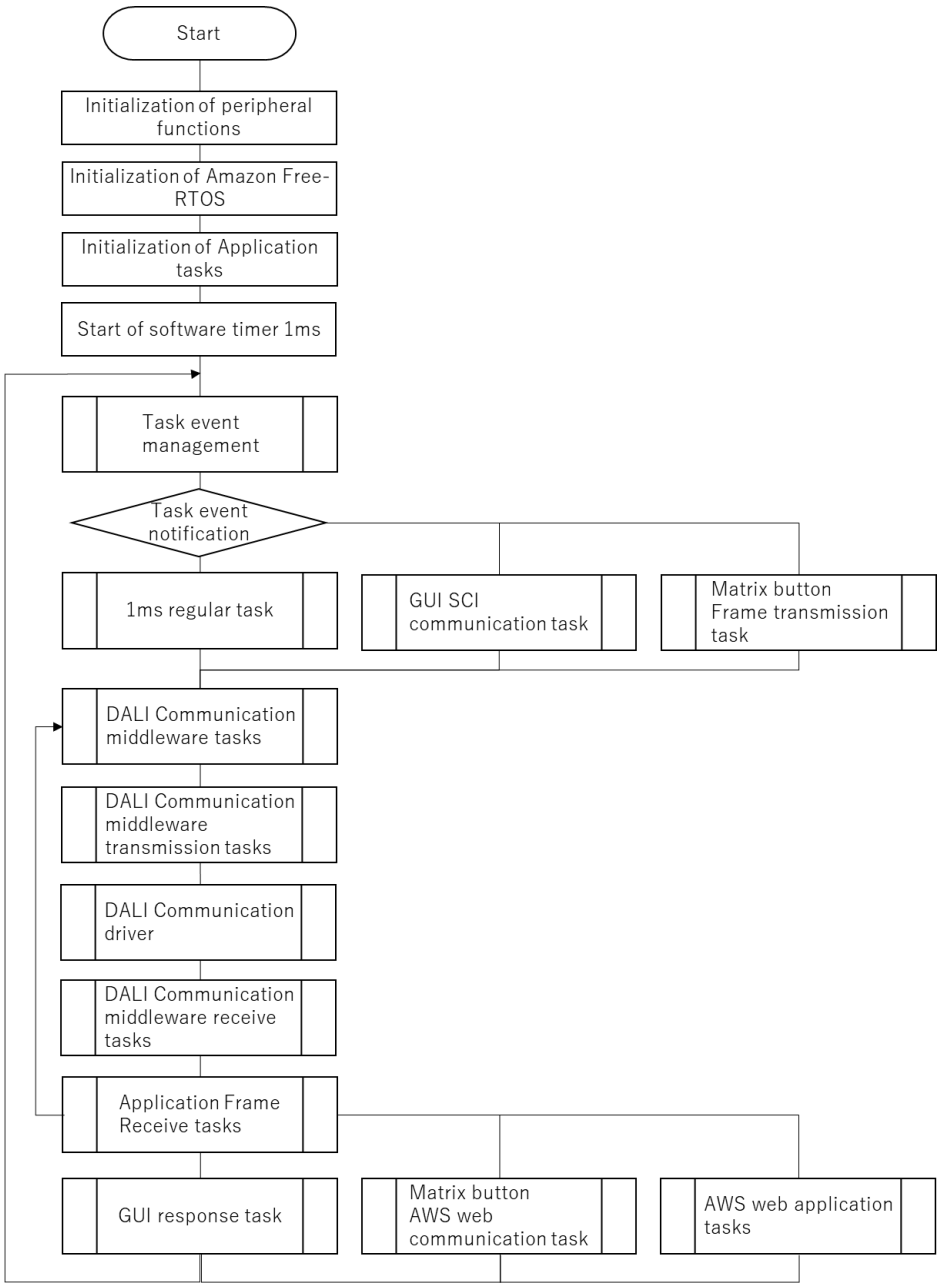

Figure 8 Operation flow diagram of each application

- Application Controller task: operate Application Controller
  －1ms regular task, Frame receive task

- GUI application task: communicate with DALI master controller GUI
  －GUI SCI communication task, GUI response task

- AWS web application tasks: communicate with AWS web applications.
  －AWS DALI communication task, Logger task

- Matrix button application task: work on matrix button
  －Frame transmission task, AWS Web communication task

- DALI communication middleware task: control with DALI communication driver, manage and notify Frame
  －DALI communication driver control task, Frame transmission task, Frame reception task

- DALI communication driver: transmit and receive Frame using peripheral functions
  －DALI communication control

4.6.1 IEC62386-101 standard and regulation part

The DALI communication function included in this demonstration project is a Single-master application controller based on the DALI standard IEC62386-101 ed2.1 and Receiver / Transmitter function that enables the use of Multi-master application controller in Chapter 4.6.5.

1. DALI communication driver

This driver is a DALI communication protocol confirmation driver (Hereafter referred to as softDALI driver) to realize the DALI-2 communication protocol (each frame transmission / reception) by software. The use of this driver is intended for Application Controller of Control Device.
For reception, it checks that the Forward Frame received by DALI communication conforms to the DALI-2 Receiver timing standard, and determines whether to accept or discard the received Frame.
For transmission, it transmits Forward Frame and Backward Frame transmitted by DALI communication according to Transmitter timing standard. As for Forward Frame transmission during Multi-master operation, it transmits according to priority. It also implements collision processing for avoiding, detecting, and recovering from collision with frames transmitted from other Control Devices.
It has a proprietary function and the effective bit upper limit is 256 bits. (With valid / invalid specification)

Table 15 Soft DALI driver processing overview list

Soft DALI driver processing overview
Receiver ・8bit　Receive Backward Frame ・16bit,24bit,32bit　Receive Forward Frame ・64bit,128bit,256bit　Receive Proprietary Forward Frame(User changeable: within 3 types) ・Detect Bit timing Violation (Reception bit timing error) ・Detect Receiver Frame Size Violation(Reception size error) ・Check Stop condition ・Check Twice Frame Transmitter ・8bit　Transmit Backward Frame ・16bit,24bit,32bit　Transmit Forward Frame ・64bit,128bit,256bit　Transmit Proprietary Forward Frame ・Transmit at transmission bit timing ・Avoid collision when transmitting (Multi-master only) ・Detect collision when transmitting (Multi-master only) ・Recover when collision occurs (Multi-master only) ・Retransmit when collision occurs (Multi-master only) ・Adjust the transmission timing by priority (Multi-master only)

This application note mainly describes the following parts of the DALI communication driver.

- Collision processing
- Twice Frame
- System failure
- Functional restriction

(1) How to realize SoftDALI driver

(a) Driver name definition

The names used in this driver are described below.

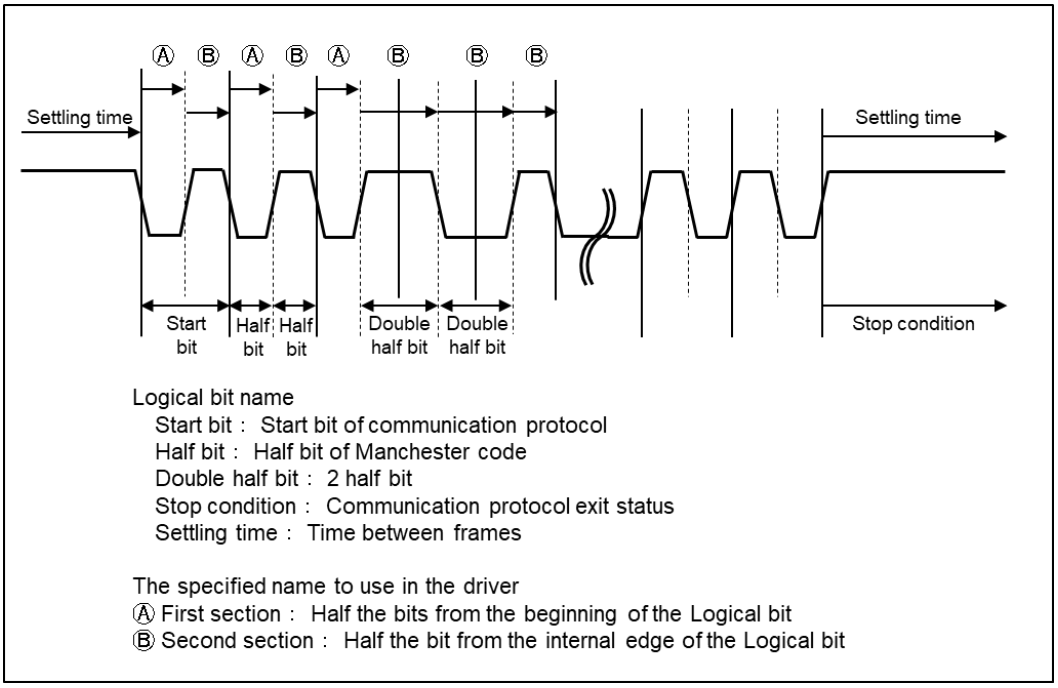

Figure 9 The specified name to use in the driver

(b) Judgment of received Frame data

The following shows how to obtain the data set during Frame decoding with this driver.

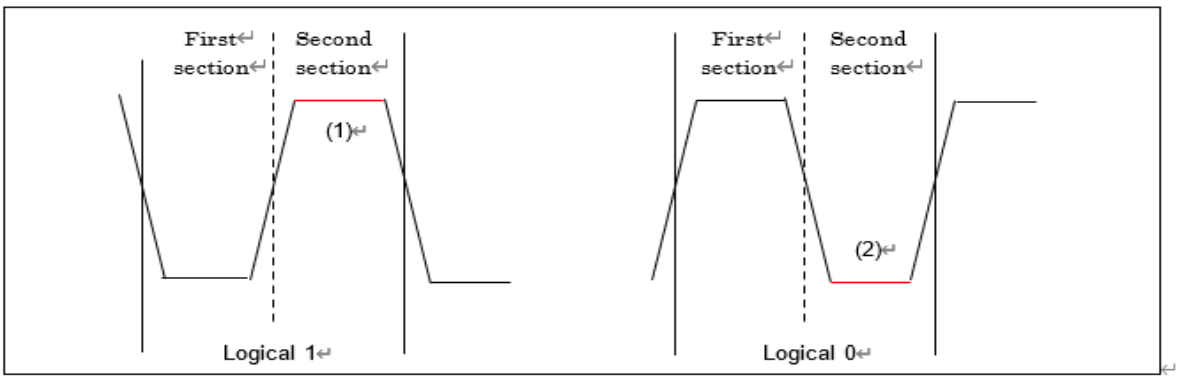

Figure 10 How to determine Half bit frame data

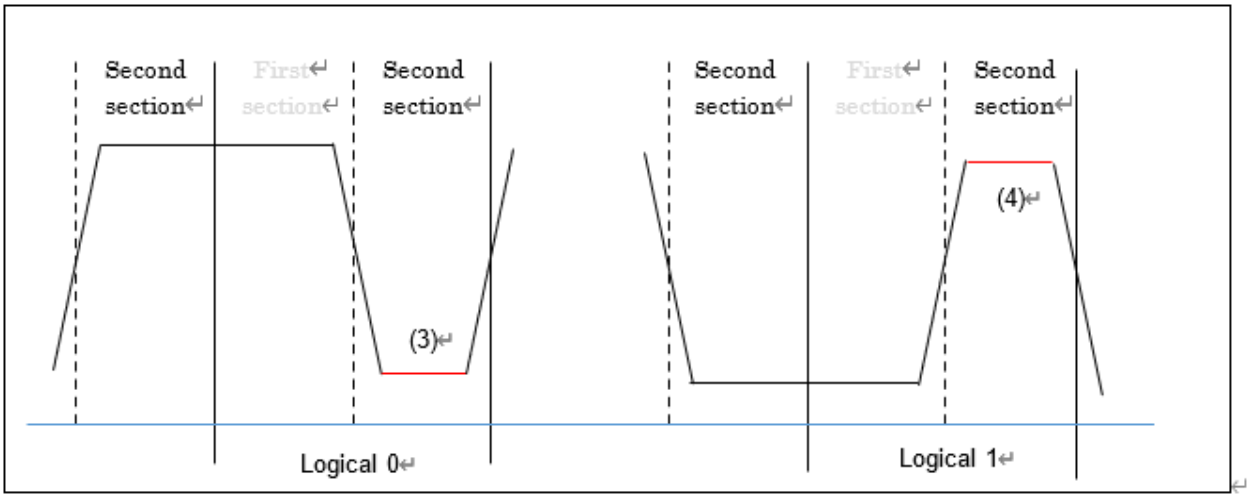

Figure 11 How to determine Double half bit frame data

Check the status of pin when interrupt of First section occurs.
Create frame data as “1” in High state (1) and “0” in Low state (2).
In the case of the Second section interrupt, check the pins only at the time of Double half bit.
Create frame data as "0" in Low state (3) and "1" in High state (4).

Table 16 Pin status and data to set

Frame decoding status	Pin status	Setting data
Half bit	LOW	“1”
	HIGH	“0”
Double half bit	LOW	“0”
	HIGH	“1”

Other frame decoding conditions

- First bit is not saved as data.
- The frame is 8 bits,16 bits, 24 bits and others (20 bits and 32 bits of Reserve) in Proprietary designated size. If any other size is received, a received size error will occur.
- If a stop condition is received with a size other than the above, a receive size error will occur.

(c) Transmission bit timing

Definition of Single-master Transmitter bit timing

The transmission bit timing and name for Single-master used in this driver are described below.

Table 17 Single-master Transmitter bit timing

Minimum	Typical	Maximum	Description
366.7us	416.7us	466.7us	Half bit
733.3us	833.3us	933.3us	Double half bit
2450us			Stop condition time

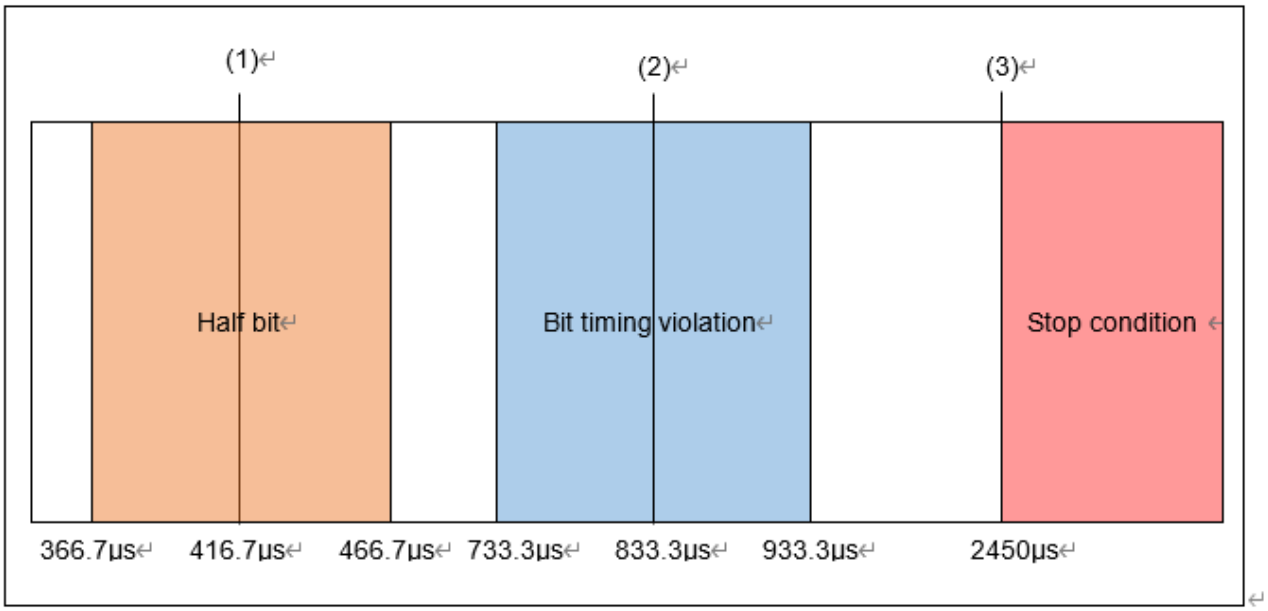

Figure 12 Single-master Transmitter bit timing

Definition of Single-master Transmitter Settling time

The following table shows the settling time for Single-master used in this driver.

Table 18 Single-master Transmitter settling time timing

Minimum	Typical	Maximum	Description
5.5ms	-	10.5ms	Backward frame
13.5ms	-	75.0ms	Forward frame

The backward frame is the duration from the start of stop condition measurement. It keeps measuring even if BUS state changes to active state on the way. It must always be transmitted ignoring the occurrence of collision. It can be transmitted only within 5.5ms ~ 10.5ms.

The forward frame can be transmitted if it is after 13.5ms.

Definition of Multi-master Transmitter bit timing
The transmission bit timing and names for Multi-master used in this driver are described below.

Table 19 Multi-master Transmitter bit timing

Minimum	Typical	Maximum	Description
400.0us	416.7us	433.3us	Half bit
800.0us	833.3us	866.7us	Double half bit
2450us			Stop condition time

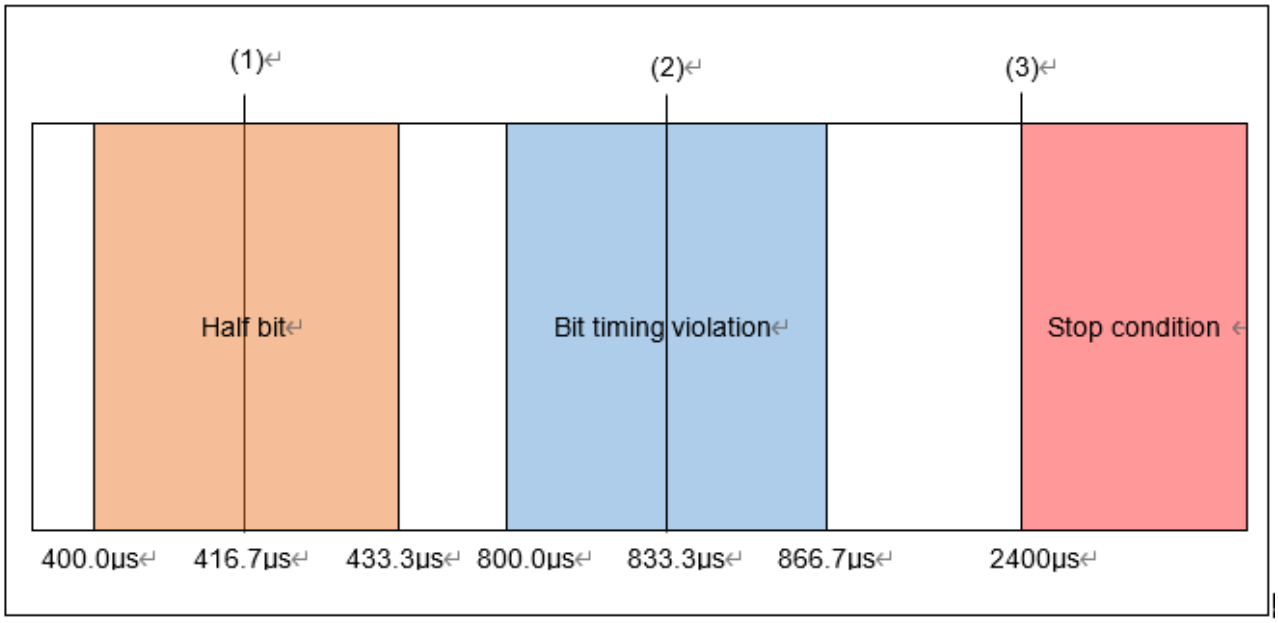

Figure 13 Multi-master Transmitter bit timing

Definition of Multi-master Transmitter Settling time
The following table shows the settling time for Multi-master used in this driver.

Table 20 Multi-master Transmitter settling time timing

Minimum	Typical	Maximum	Description
5.5ms		10.5ms	Backward frame
13.5ms		14.7ms	Forward frame (Priority 1)
14.9ms		16.1ms	Forward frame (Priority 2)
16.3ms		17.7ms	Forward frame (Priority 3)
17.9ms		19.3ms	Forward frame (Priority 4)
19.5ms		21.1ms	Forward frame (Priority 5)

The backward frame is the duration from the start of stop condition measurement. It keeps measuring even if BUS state changes to active state on the way (only when the Stop condition has been measured).
The backward frame must always be transmitted ignoring the occurrence of collision. Transmission of backward frame is possible only within 5.5ms ~ 10.5ms. It can be transmitted only within 5.5ms ~ 10.5ms.
The forward frame uses Settling time according to priority.
It transmits at any time within each range. If fixed, the timing of devices using the same software will be aligned, which may cause collisions.
The forward frame can be transmitted after minimum time.

(d) Measurement of Transmitter bit timing

The method of measuring the transmitter bit timing of DALI-2 communication with this driver is described below.

Measurement method of Transmitter bit timing

In this driver, the transmission bit timing is measured using the following method.

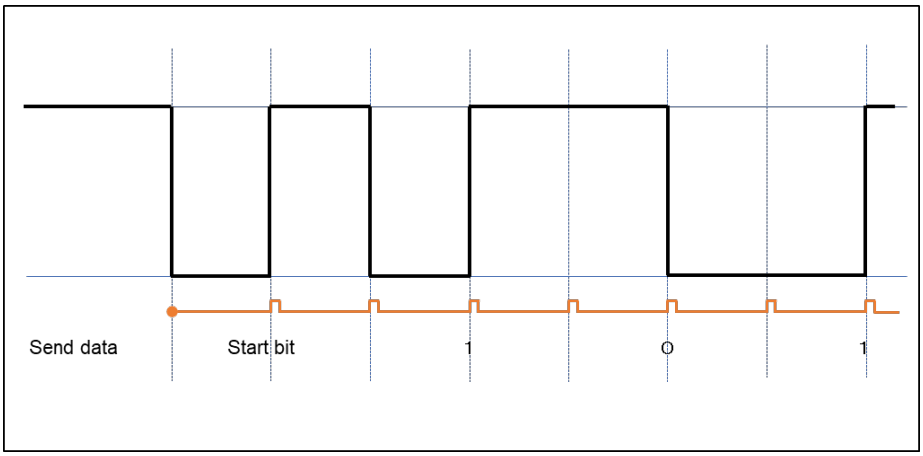

Figure 14 Transmitter bit timing acquisition timing

This driver uses the following timer to measure the transmitter bit timing.

- Use the cycle count operation timer and set the half bit value as the counter value.

(The timer changes for each device. This specification is for the RX65n.)

Since the cycle count timer does not generate an interrupt at the start, set the transmission pin low at the timer start timing. After that, set High / Low to the transmission pin using the interrupt of the cycle timer.
Since the transmission cycle differs between Single-master and Multi-master, set the transmission cycle according to the type of master.

(e) Collision

Collision processing

This driver is intended for use with Control devices.
If the Control device is a Multi master Transmitter, it is necessary to perform collision processing when transmitting Forward Frame.

There are three types of collision processing as shown below.

- Collision avoidance
- Collision detection
- Collision recovery

Collision avoidance
There is a standard for the transmission timing to avoid Collision before the Multi-master transmitter transmits the Forward Frame.

- Be a Multi-master transmitter
- The time more than the Settling time specified by the priority of the Forward Frame to be transmitted has elapsed
  (Settling time is to create random timing for each device within the specified range for each priority.)
- BUS is in Idle state (Settling time becomes 0 by becoming Active state)

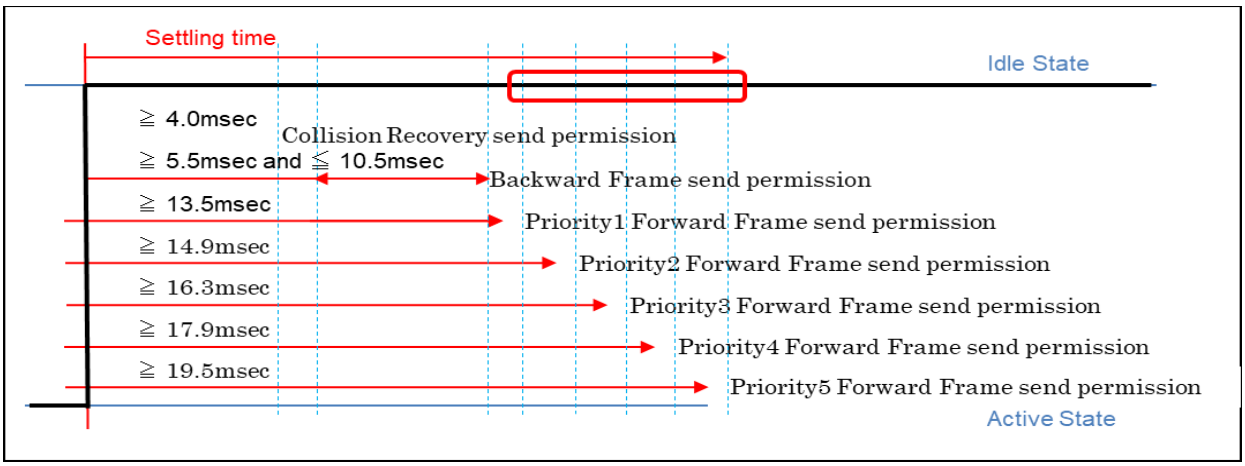

Figure 15 Collision avoidance transmission timing

When the Settling time set in the specified Priority of the transmission Forward Frame has elapsed, that Forward Frame can be transmitted. In addition, you must ensure that the State of BUS at the start of transmission is Idle State. In this driver, whether or not it is in the Idle State is determined by the State obtained by the receiver reception interrupt.

When Collision Recovery has occurred, for the Device connected to the BUS including the Control Gear, the Forward Frame will cause a Bit violation due to the occurrence of the Collision, and it is considered that the transmission of the Backward Frame will not occur. As the Backward Frame is transmitted with a higher priority than the Forward Frame, progress of the transmission possible Settling time of Forward Frame does not occur.

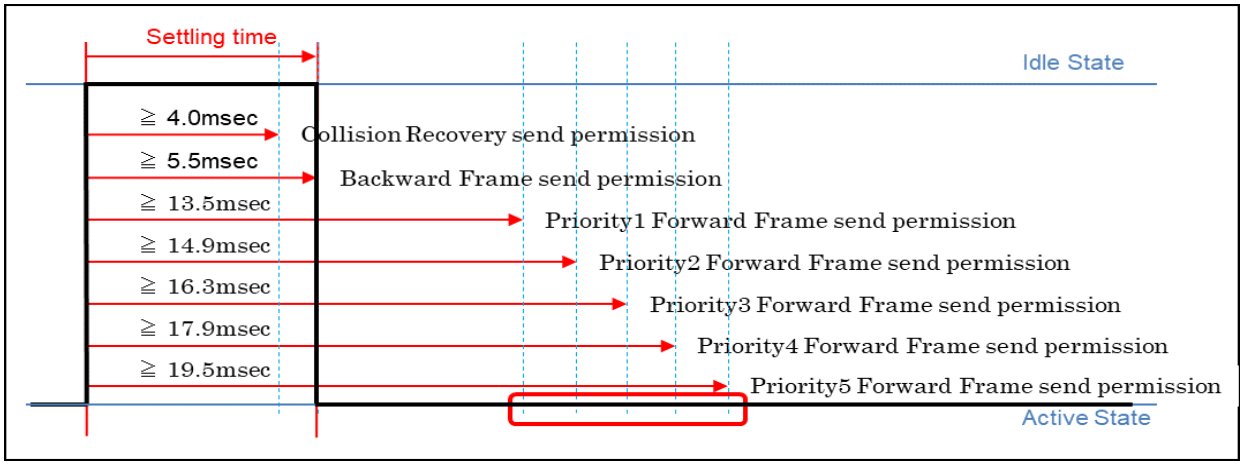

Figure 16 Collision avoidance transmission timing

The timer of Settling time used for the collision avoidance process starts on the Receiver side.
On the transmitting side, check that the timer for Settling timer is running, and check the elapsed time from the timer counter value before processing.
However, the Backward Frame should be transmitted in the deemed time (5.5ms) even if the settling time ends within 5.5ms. Backward Frame does not detect Collision with other Application Controllers.

Collision detection
Collision detection is performed at the time of Forward Frame transmission, but detection is performed at the Receiver because it is necessary to check the data on the BUS.
Since the Receiver only performs the Forward Frame transmission, it is necessary to recognize that the transmission is being performed.
The conditions for collision detection are shown below.

- Be a multi-master transmitter
- Forward Frame is being transmitted

Collision detection is performed under the above conditions.
Collision has a different generation pattern depending on the transmission timing. The recovery method differs depending on the occurrence pattern.
The following patterns exist for collision detection.

- Collision detection when the bit width of BUS State is not Destroy area
- Collision detection when the bit width of BUS State is Destroy area

Collision detection is performed with the above pattern.

The bit timing used by this driver for collision detection is shown below.

Table 21 First section bit timing

Minimum	Typical	Maximum	Description
		<100us	Grey area
100us		356.7us	Destroy area
>356.7us	(1)365us	400.0us	Grey area
400.0us		433.3us	Valid half bit
>433.3us	②467us	<476.7us	Grey area
476.7us			Destroy area

In the first section, the range from (1) to ③ is judged as Half bit, and the others are judged as Destroy area.
Although the specification states that only the Active state is applied for 476.7 us or more, this driver will determine the Destroy area even in the Idle state if a signal is received before the Stop condition elapses.

Table 22 Second section bit timing

Minimum	Typical	Maximum	Description
		<100us	Grey area
100us		356.7us	Destroy area
>356.7us	(1)365us	400.0us	Grey area
400.0us		433.3us	Valid half bit
>433.3us	②467us	<476.7us	Grey area
476.7us		723.3us	Destroy area
>723.3us	③762us	<800.0us	Grey area
800.0us		866.7us	2 valid half bits
>866.7us	④905us	943.3us	Grey area
943.3us			Destroy area

In the second section, the range from (1) to ③ is judged as Half bit, the range from ③ to ④ is judged as Double half bit, and the others are judged as Destroy area.
Although the specification states that only the active state is applied for 943.3 us or more, this driver will determine that the Destroy area even in the idle state if a signal is received before the Stop condition elapses.

Collision detection when the BUS State bit width is not in the Destroy area

If there is no Destroy area in the bit width of the BUS State, but there is a difference between the transmitting State and the Bus State, it is determined that a collision has occurred.
In this pattern, Collision cannot be detected only by bit width measurement by reception interrupt. State is compared to detect the occurrence of Collision.

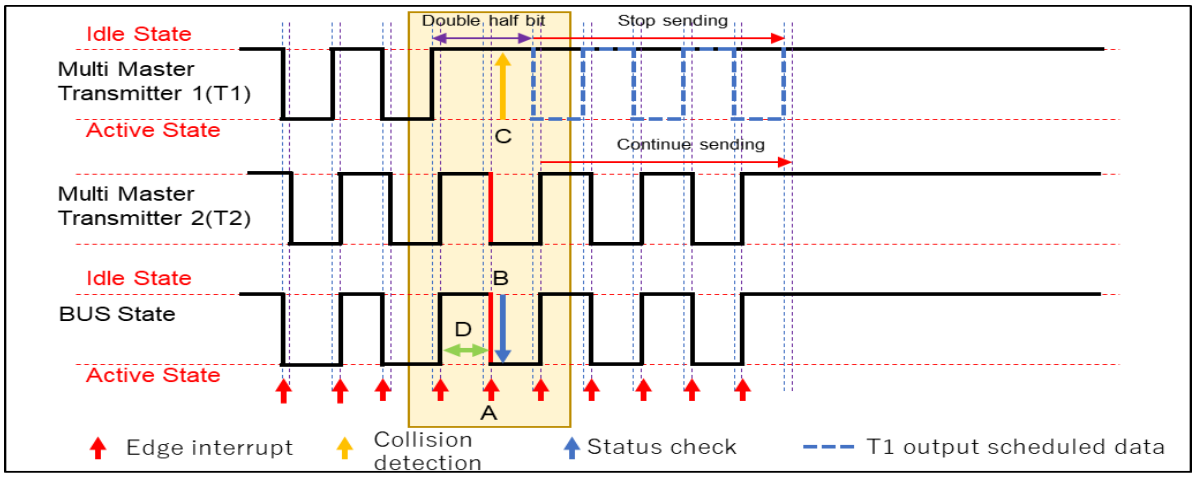

Figure 17 Collision detection when not in Destroy area

"Figure 17 Collision detection when not in Destroy area" above shows Collision detection when the area is not Destroy area.
Transmitter1 (hereafter T1) is a self-machine, and it's assumed that data transmission has been begun at the same time in T1 and Transmitter2 (hereafter T2).

At some point, T1 transmits at Double half bit and T2 transmits at Half bit.
If Idle State and Active State are transmitted at the same time, Active State has priority, so the waveform transmitted by T2 flows on BUS. Therefore, a reception edge interrupt (A) occurs at both T1 and T2.
At T1, the transmission state at the time of occurrence of the interrupt (A) is the Idle State, while the BUS State (B) is the Active State, and the T1 detects the difference between the states and detects the occurrence of Collision (C).

In such a pattern, the bit width (D) detected by the reception interrupt (A) is within the normal range of the Half bit, and collision detection cannot be performed from the bit width.

T1 stops transmission when it detects a collision, but in T2, the transmission state and the BUS state are the same, so it does not recognize that a collision has occurred and continues transmission.

Collision detection when BUS State bit width is Destroy area

If a Destroy area exists in the bit width of the BUS State, it is determined that a collision has occurred. In this pattern, the collision is detected by the bit width measurement by the reception interrupt.

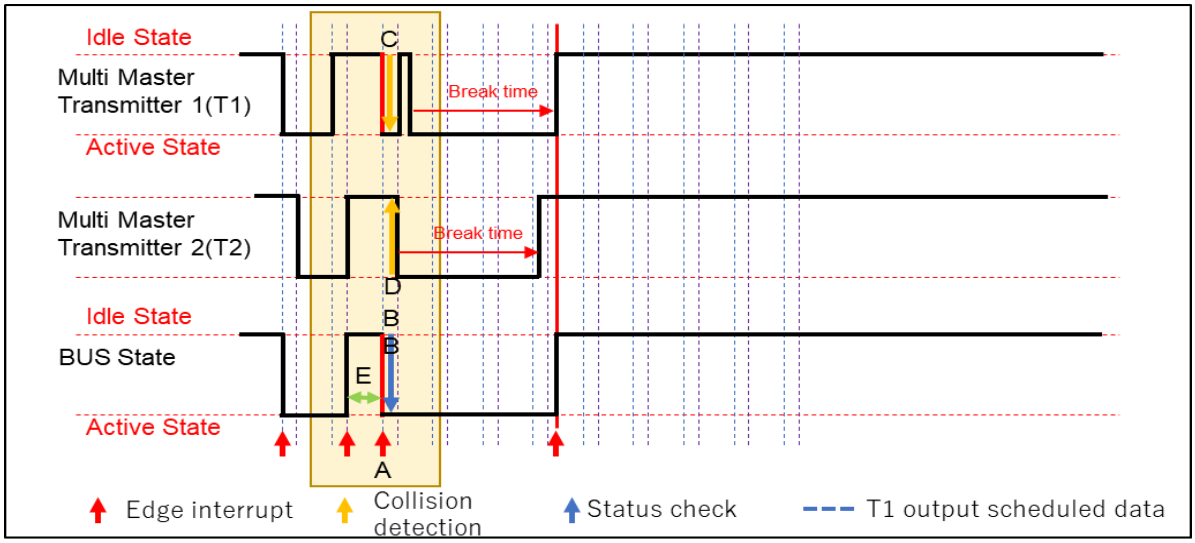

Figure 18 Collision detection in case of Destroy area

"Figure 18 Collision detection in case of Destroy area" above shows Collision detection when a Destroy area occurs.
Assume that data transmission starts with a slight time difference between T1 and T2.

At some point, T1 switches to Active State and transmits. Since Active State has priority, the waveform transmitted by T1 flows on BUS. Therefore, a reception edge interrupt (A) is generated for both T1 and T2.
At T1, the transmission state when the interrupt (A) occurs is the Active State, and the BUS State (B) is the Active State. There is no difference between the states. Since the bit width (E) detected by the receive interrupt (A) is the Destroy area, T1 detects (C) the occurrence of Collision.
T2 can detect the occurrence of Collision in both the state difference of the receive interrupt (A) and the Destroy area of the bit width (E).
Transmission stops when both T1 and T2 detect Collision.

Unlike the detection of "Figure 18 Collision detection in case of Destroy area" above, there are cases where no interrupt is generated.
If no interrupt occurs in the Idle State, it is regarded as a Stop condition, but if no interrupt occurs in the Active State, the Destroy area must be detected.

The pattern that does not generate an interrupt is shown below.

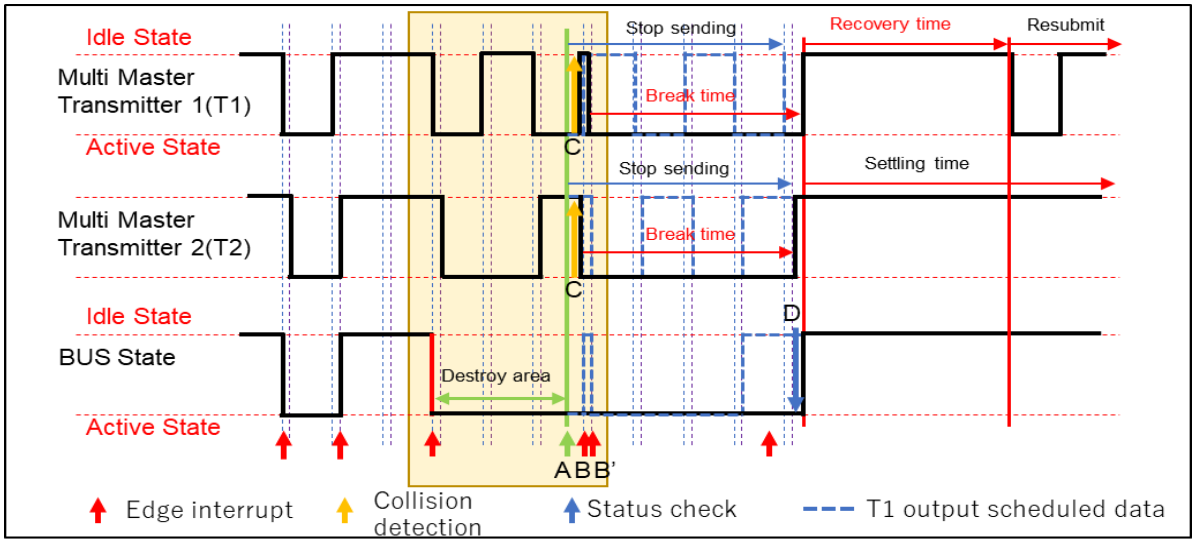

Figure 19 Collision detection in case of Destroy area

The above “Figure 19 Collision detection in case of Destroy area” shows detection of Collision occurrence in Destroy area under the condition that no reception interrupt occurs. This method is valid only when BUS State is Active State.

If the Active State continues alternately in T1 and T2 as described above, the Active State may be held in the BUS State beyond the Destroy area. In this case, Collision cannot be detected at T1 and T2 because no reception interrupt occurs.
The Transmitter activates the Destroy area detection timer at the time of the reception interrupt due to the fall (at the time of transition to Active State), the timer interrupt (A) occurs, and (C) detects the occurrence of Collision.
This detection method is valid only when the BUS State is the Active State. In the Idle State, another timer is used to detect the Stop condition.

The maximum Destroy area start value (943.3 us) for the Second section is used as the timer value for detecting the Destroy area used by the timer. This is the maximum waiting time for reception interrupt in Active State. This detects the continuation of the Active State of BUS State by multiple transmitters.
Also, when the Transmitter detects the occurrence of Collision due to the Destroy area, it must generate the Active State for 1.2ms. Even if it cannot be detected first, Collision can be detected with this Break time.

Collision recovery
In Collision recovery, the recovery procedure differs depending on whether or not the detection of Collision includes the occurrence of Destroy area.
The following patterns exist for collision detection.

- Collision detection of the case where the bit width of the BUS State is not in Destroy area
- Collision detection of the case where the bit width of the BUS State is in Destroy area

In the occurrence of Destroy area, all transmitters on the BUS have detected the occurrence of collision. If the Destroy area has not occurred, there is a transmitter that has not detected the occurrence of collision.

Collision recovery when not in Destroy area

The recovery method when the Destroy area does not exist in the BUS State bit width is shown below.

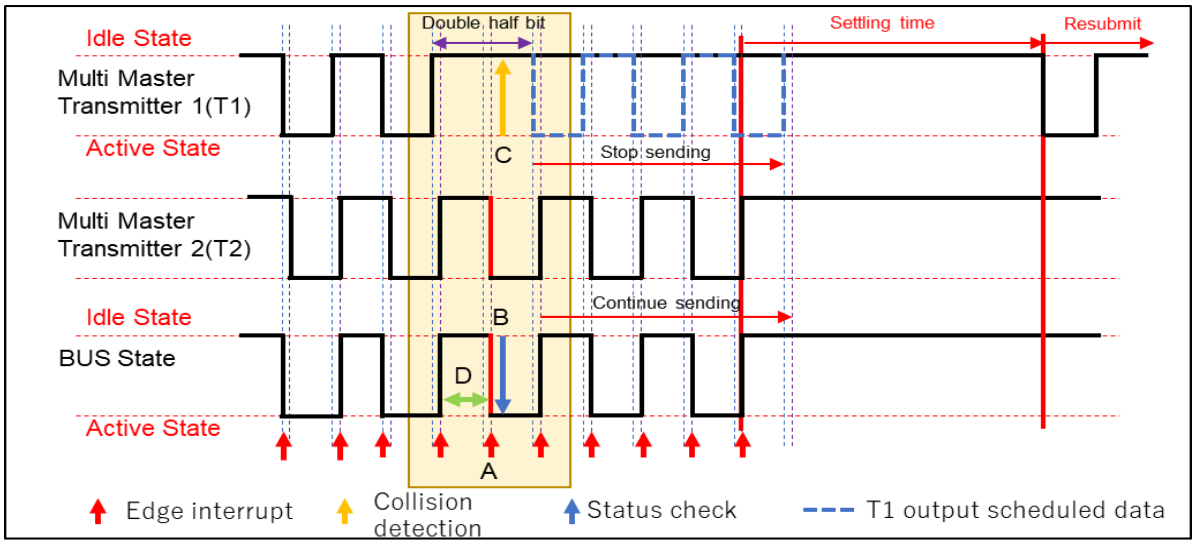

Figure 20 Collision recovery when not in Destroy area

The above figure, "Figure 20 Collision recovery when not in Destroy area", shows the Collision recovery processing when no Destroy area occurs.
In the case of the above figure, T1 confirms the abnormality of State in (C), detects Collision, and stops transmission. T2 continues transmission without detecting the occurrence of Collision because no state abnormality has occurred.

The processing when a collision occurs when the Destroy area has not occurred is shown below.

- Stop transmitting.
- Continue receiving. (Same reception processing as normal data)
- Re-transmit in the same way as the transmission procedure after receiving normal data

T1 detects Collision in (C) and stops transmission. Since T2 has not detected the occurrence of Collision, transmission is continuing. Therefore, T2 transmission processing is normally performed on the BUS. When transmission stops, T1 passes BUS to T2 and shifts to normal reception operation.
Since T1 has stopped transmitting, it does not detect Collision, performs normal receiving operation, and waits for the elapse of Settling time after detecting the Stop condition.
After confirming that the Settling time has elapsed, it starts re-transmitting the data that has stopped transmitting. The retransmission starts from the beginning, not from the middle.

Collision recovery in Destroy area

The following shows the Collision recovery method when a Destroy area is detected.

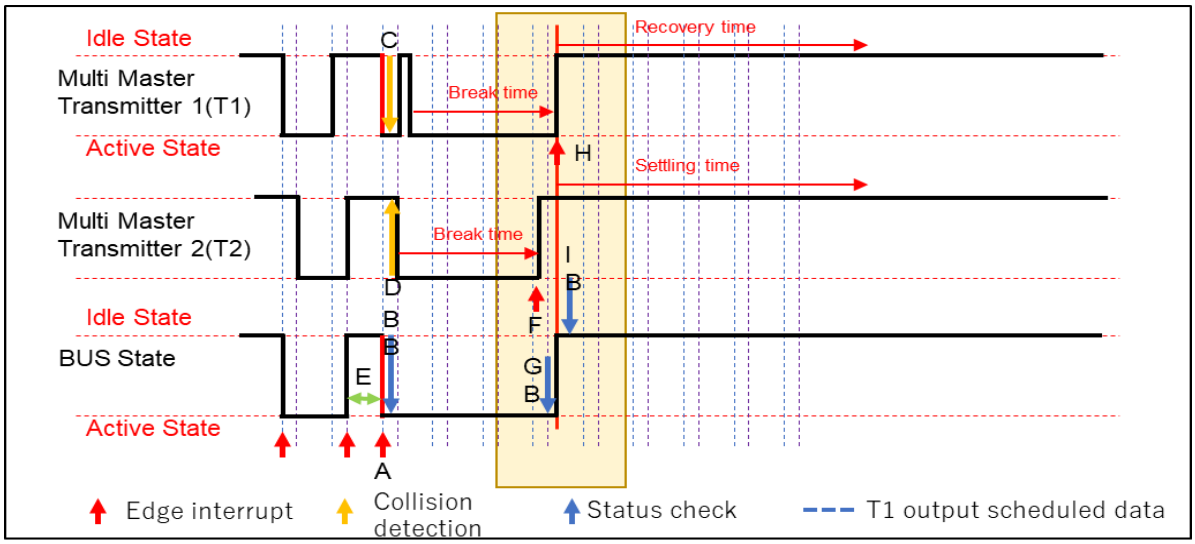

Figure 21 Collision recovery in Destroy area

"Figure 21 Collision recovery in Destroy area" above shows Collision detection when a Destroy area occurs. Collision is detected at T1 and T2 when Destroy area occurs.

When Collision including Destroy area is detected, Transmitter must force BUS to Active State as Break time within 450µs. This is done to make the Transmitter that has not detected the collision detect the Collision.

In the above figure, you can see that T2 generated the Break time first, and the end of the Break time is slightly earlier than T2. Transmitter checks the BUS State when the Break time that it has created ends. In the case of T2, it confirms the BUS State in (G) and confirms that it is in Active State. In the case of T1, it confirms it in (I) and confirms that it is in Idle State.
T1 whose BUS state confirmed at the end of Break time is Idle State is given priority for retransmission by Collision Recovery.

- The T1 retransmission settling time uses the Recovery time.
- The T2 retransmission settling time uses the time specified in the priority of the transmitted data (see Figure 13)

Transmitter will recover Collision according to this rule.

Table 23 Collision Recovery Timing

Minimum	Typical	Maximum	Description
1.2ms		1.4ms	Break time
4.0ms		4.6ms	Recovery time
Transmitter should start transmitting at any time between the minimum and maximum Recovery time to avoid Collision.			

(f) Twice Frame

Twice Frame is established by transmitting the same data continuously within the specified time.
In the driver, the transmission is performed within a specified time by a normal transmission method, so a special thing isn't done. Regarding reception, it is judged as a Twice Frame by comparing with the previous reception Frame and receiving within the specified Settling time below.
However, since the frame contents are not checked in this driver, those that are not Twice Frames are also judged as Twice, and the final judgment is to be made by the application.
The specified reception time of Twice Frame used by this driver is shown below.

Table 24 Twice Frame Timing

Minimum	Typical	Maximum	Description
	(1)Stop condition value	2.4ms	Grey area
2.4ms		94ms	Twice Forward Frame
>94ms	②100ms	<105ms	Grey area
In this driver, the Settling time from the (1)Stop condition value to ②100ms is the Twice Frame valid period.			

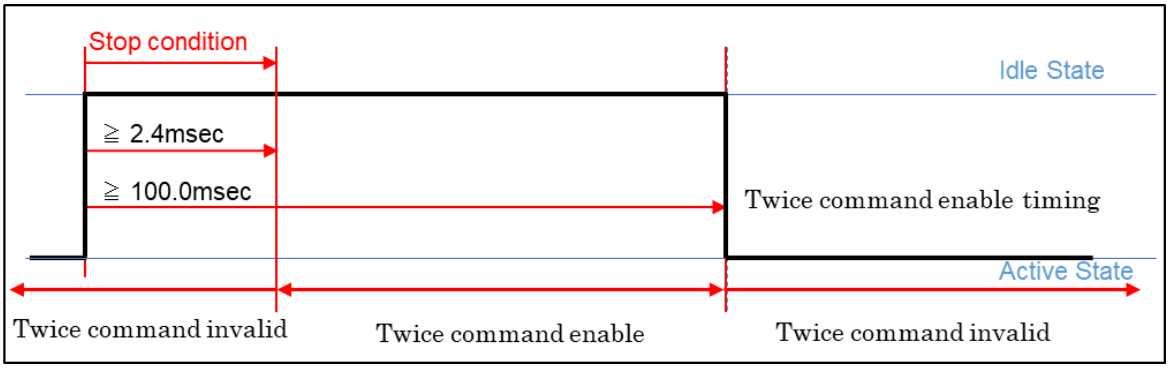

Figure 22 Twice frame effective timing

(g) System failure

On the applicable board, System failure is detected by checking the DALI circuit.
System failure occurs when the Active State continues for 500ms or more.
For measurement, the timer for System failure starts when a falling edge is received, and measures the Active state period.
It detects the occurrence by System failure timer interrupt and notifies the application side.

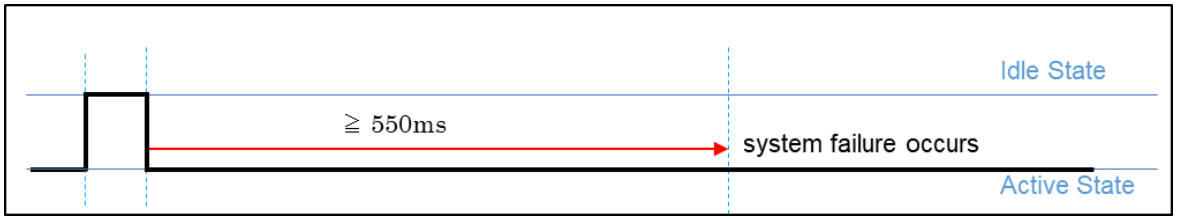

Figure 23 System failure timing

Table 25 System failure Timing

Minimum	Typical	Maximum	Description
>550ms			System failure

(h) Function restrictions

This driver has limited functions. The restrictions are shown below.

Number of defined Proprietary Frames
In the DALI standard, there is no limit on the number of Proprietary Frames specified, but this driver limits it to a maximum of three frame sizes.

Default value is as follows

- 64bit
- 128bit
- 256bit

About the handling size of Proprietary Frame, you can change the number of types such as only one type, two types, or three types if it is less than three types.
It is also possible to change the Proprietary Frame size from the above Default value.

You can change it by modifying the Define value in the user disclosure header file.

(2) SoftDALI driver

(a) Status transition

The transition diagram of the internal state transition of this driver is shown below.

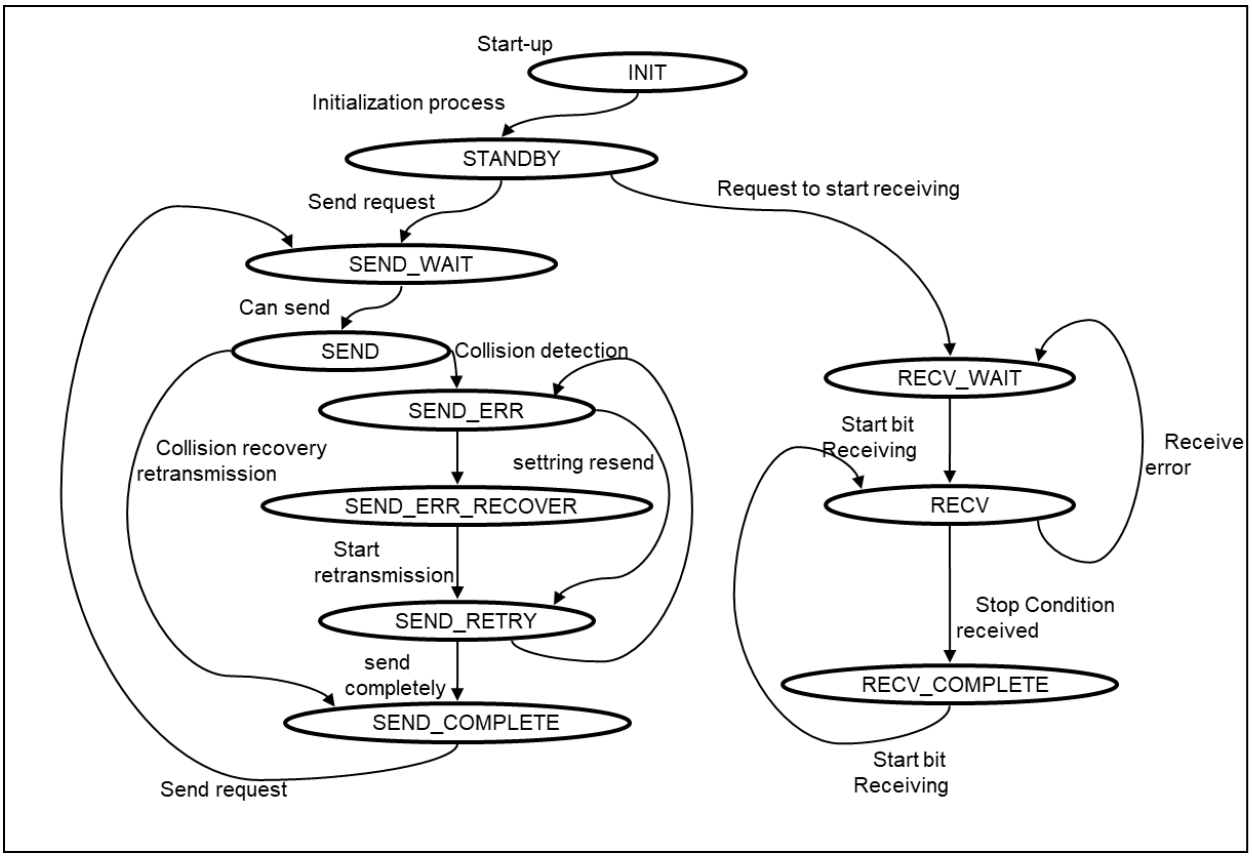

Figure 24 SoftDALI state transition diagram

SDDRV_STATE_INIT
A state when this driver is started.

SDDRV_STATE_STANDBY
A state in which initialization processing of this driver has been completed normally and peripherals for transmission and reception have been prepared.
Transmit requests and receive requests can be accepted. In this state, reception processing is started by making a reception start request. The state transition on the receiving side is started by the reception start processing request, and the receiver enters the reception waiting state (SDDRV_STATE_RECV_WAIT).

SDDRV_STATE_SEND_WAIT
A state in which a transmission request has been accepted normally and the transmission start condition has been met.
Check Settling time and BUS State, and wait for transmission.
When the transmission start condition is satisfied, the state transits to the transmitting state (SDDRV_STATE_SEND).

SDDRV_STATE_SEND
A state in which transmission is enabled, transmission is started, and transmission is being performed.
This state continues until the reception detects Collision or a Stop condition is received normally.

SDDRV_SEND_ERR
A state where an error such as Collision is detected during transmission.
Immediately after transitioning to this state, transmission is stopped and recovery operation is selected. If Destroy area has detected, the state transits to Collision recovery state (SDDRV_STATE_SEND_ERR_RECOVERY). If the Destroy area has not been detected, set the settling time according to the normal priority and transit to the retransmission state (SDDRV_STATE_SEND_RETRY).

SDDRV_STATE_SEND_ERR_RECOVERY
A state where recovery operation is performed when a Destroy area is detected.
Break time is generated immediately after the transition. Check the BUS State at the end of Break time. In the case of Active State, set Settling time according to normal priority. In the case of Idle State, set the Collision Recovery time to Settling time, then transit to the retransmission state (SDDERV_STATE_SEND_RETRY).

SDDRV_STATE_SEND_RETRY
A state in which retransmission processing is being performed
Check the elapse of the settling time and the BUS state, and start transmission. This state continues until the reception detects Collision or a Stop condition is received normally.

SDDRV_STATE_SEND_COMPLETE
A state the transmission is completed.
This state is entered when the receiving side has successfully received the Stop condition. In this state, transmission completion is returned to the application.

SDDRV_STATE_RECV_WAIT
A state waiting for reception.
This state is entered when a reception start request or reception data error occurs. In this state, there is no received data.

SDDRV_STATE_RECV
A state in which reception of the Start bit has been started.

SDDRV_STATE_RECV_COMPLETE
A state Stop condition is received and that reception has normally ended.
In this state, reception completion is returned to the application.

(b) Timer

The following describes the timer used in this driver.

Table 26 List of timers used by SoftDALI driver

Timer name	Timer type	Timer value	Description
Transmission timing timer	Cycle count (TIMER:MTU1)	416us	Timer for bit transmission timing. Generates interrupts at regular intervals and creates bit transmission timing. The timer starts when the Start bit is set, and stops when the transmission is completed and Collision is detected.
Reception interrupt timer	Input capture (TIMER:MTU0)	—	Starts when a reception start request is made. There is no stop operation.
Timer for timing measurement	Compare match (TIMER:CMT)	2400us 944us 1200us	Stop condition measurement timer Active state measurement timer Break time measurement timer Used for three types of timers. It starts for Stop condition measurement at the rising edge of the reception interrupt, and starts for Active state measurement at the falling edge. When Collision occurs, it starts to measure Break time when falling. * The timer for Active state and Break time measurement is used only when there is transmission.
Settling time measurement timer (for Backward)	Compare match (TIMER:MTU2)	5.5ms 10.5ms	Backward settling time measurement Use as transmission start timing confirmation processing.
Settling time measurement timer	Compare match (TIMER:MTU3) (TIMER:MTU4)	4.0ms 13.5ms 14.9ms 16.3ms 17.9ms 19.5ms 100.0ms	Use as Settling time measurement (including Recovery time, Twice) Transmission start timing confirmation processing Twice Frame invalidation confirmation processing

Transmission timing timer
Timer for bit transmission timing of this driver.
The following shows how to use the transmission timing timer.

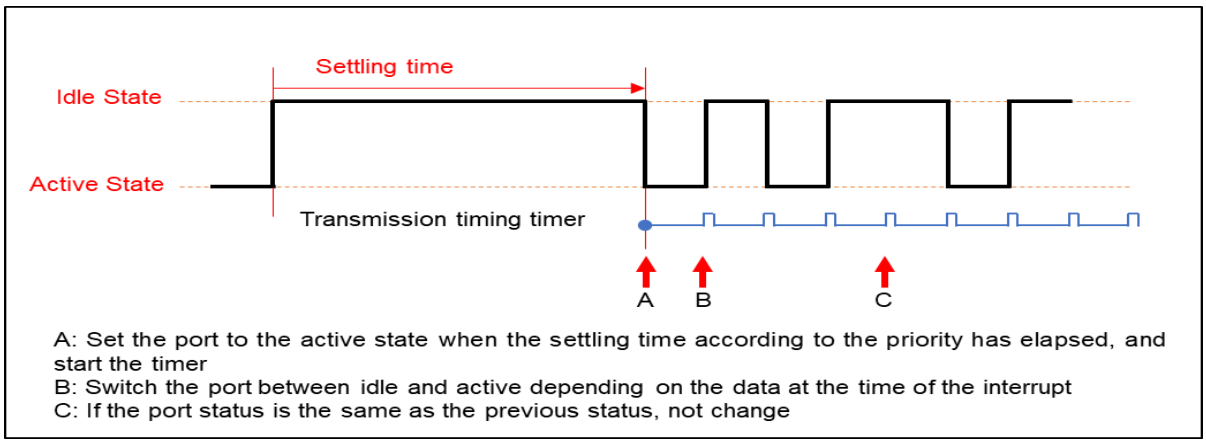

Figure 25 Start of transmission timing timer

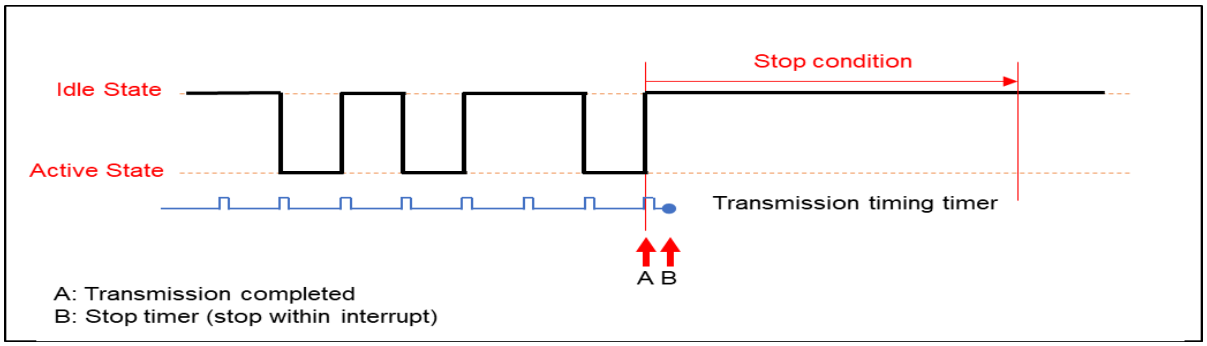

Figure 26 Stop of transmission timing timer

Use this timer to create DALI communication bit transmission timing.
Creates bit transmission timing by performing periodic operation at the Half bit timing specified by the standard.
DALI communication is performed by operating the transmission pin during the interrupt processing.
When the transmission data ends, the timer stops.

Reception interrupt timer
Timer for measuring the receive bit width of this driver.
Use the input capture timer that generates an interrupt at both edges during reception.

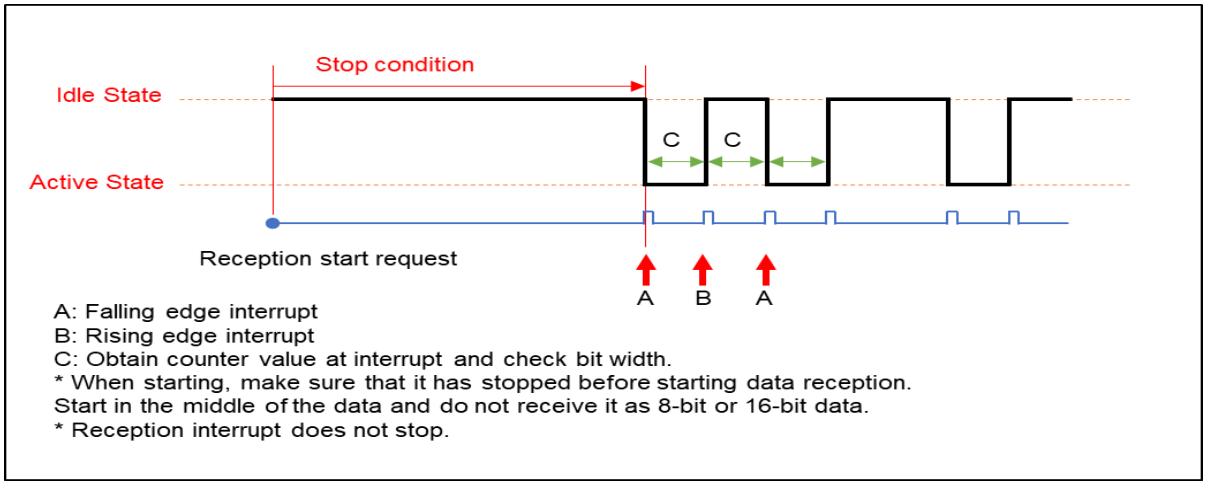

Figure 27 Start reception interrupt timer

Timer for timing measurement
Timer for bit timing measurement of this driver.
The counter match timer is used, and the timer is used for measuring two types of bit width.

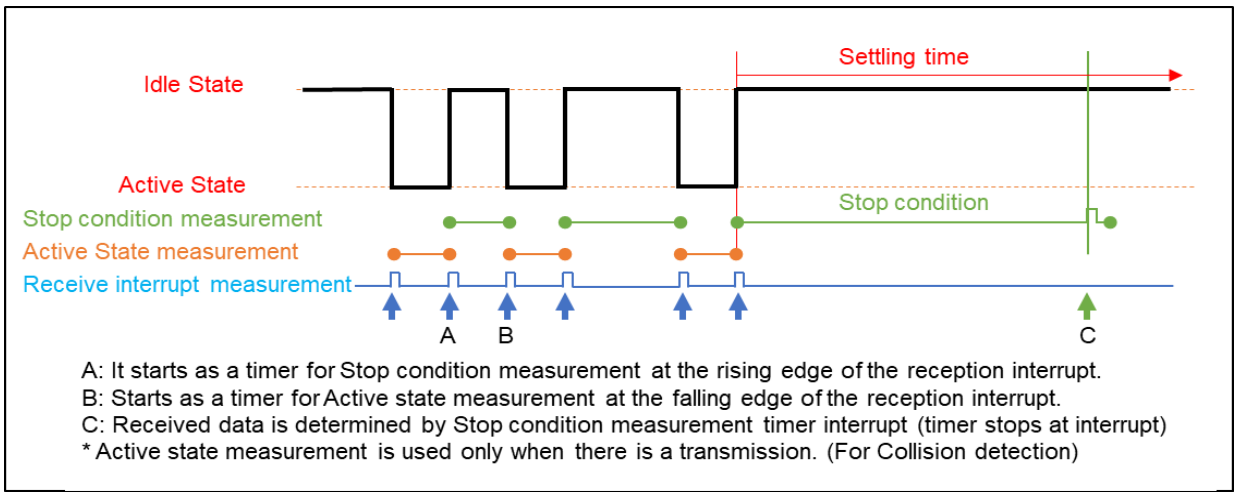

Figure 28 Start / stop of timer for timing measurement (Stop condition)

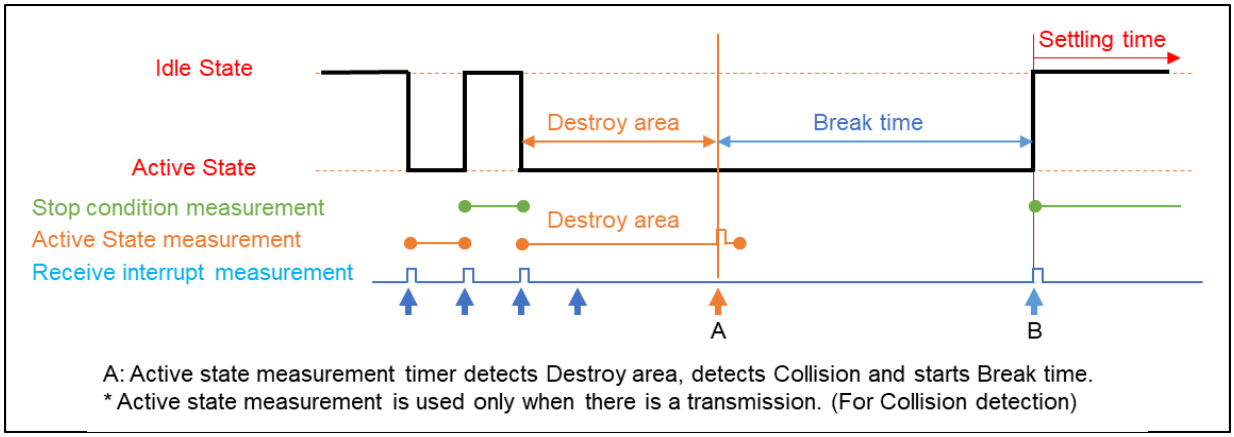

Figure 29 Start / stop of timer for timing measurement (Destroy area)

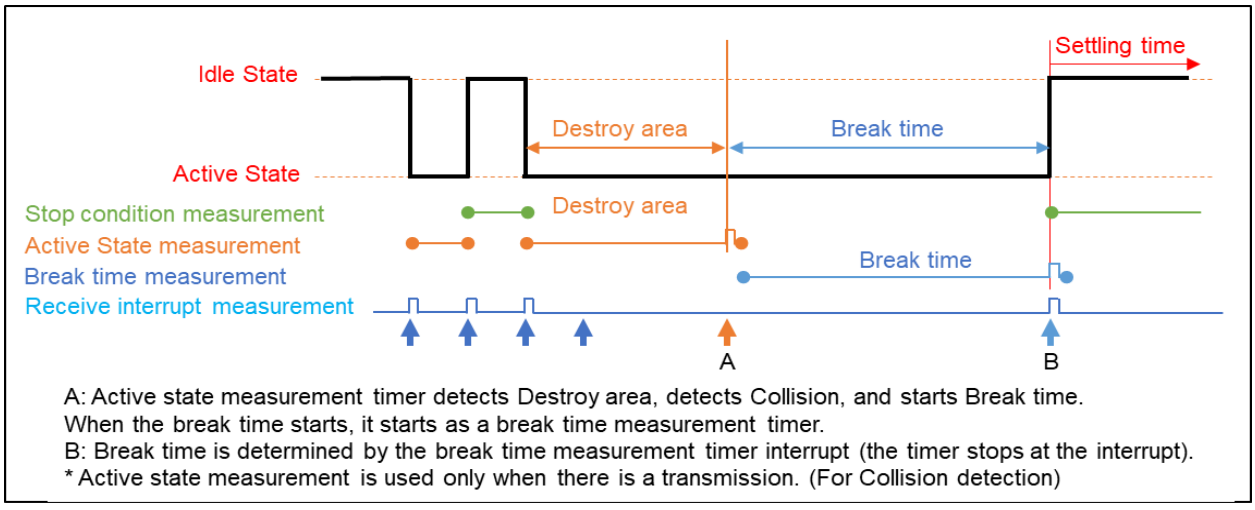

Figure 30 Start / stop of timer for timing measurement (Break time)

Settling time measurement timer
Timer for measuring Settling time (including for Backward) used in this driver.
Used for various timing measurements starting from the last rising interrupt of received data.

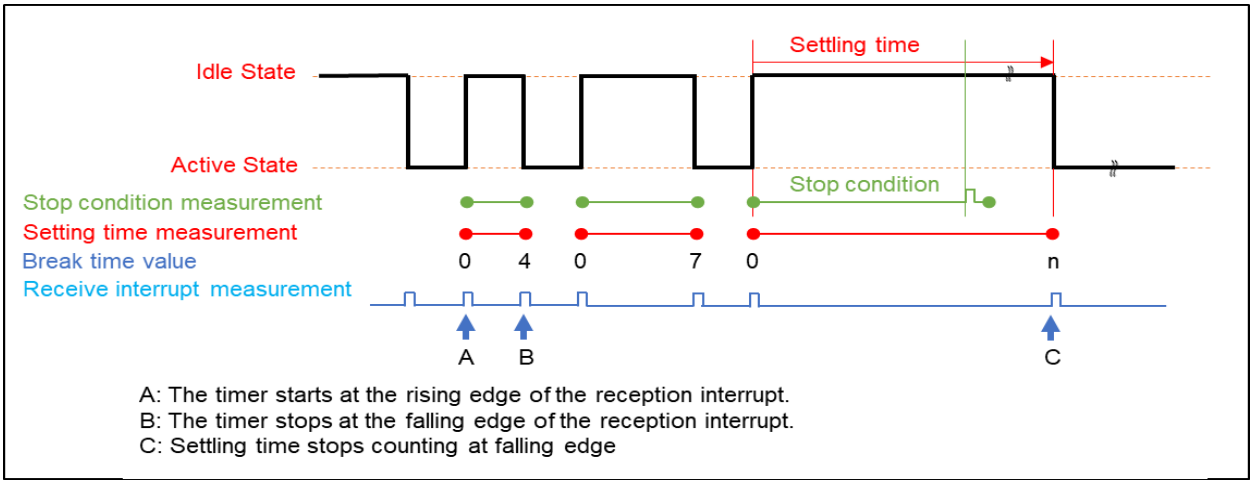

Figure 31 How to use the Settling time timer

Settling time measurement is to measure the timing of transmission start.
This is the timing required for all devices.
Please refer to "Definition of Single-master Transmitter Settling time" and "Definition of Multi-master Transmitter Settling time" for the definition of Settling time.
If the measurement time has passed the Settling time, the transmittable priority will be updated.

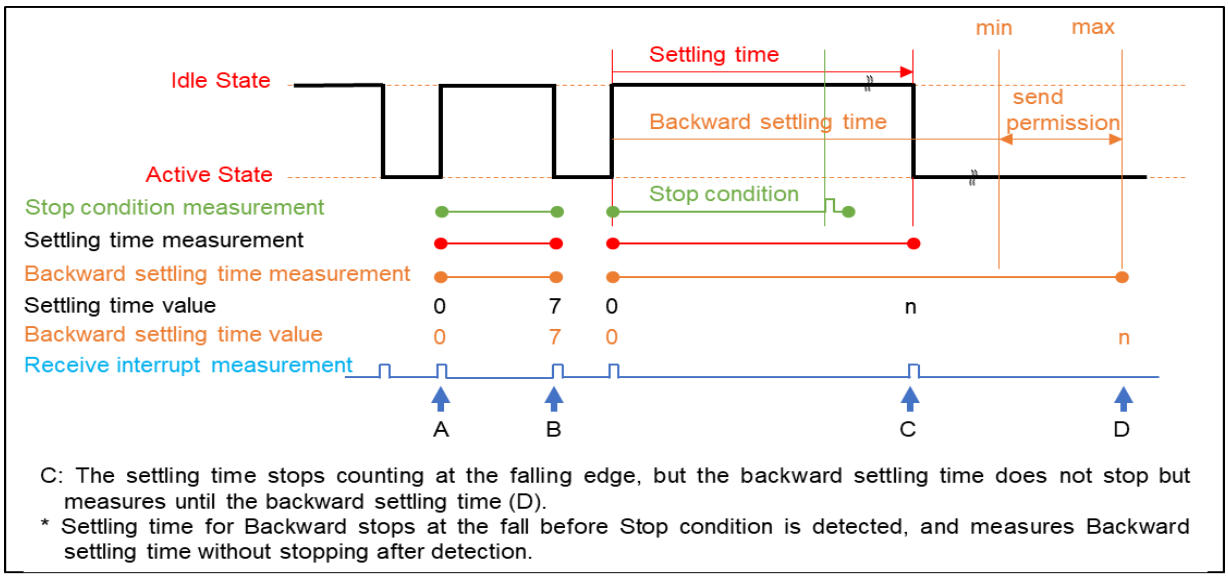

Figure 32 How to use Settling time timer for Backward (Backward settling time)

Settling time used for Backward Frame is the elapsed time from the last rising edge of the previous received data.
Normal Settling time measurement stops when the BUS State Active state is detected, but Backward Settling time does not stop because it measures the elapsed time. However, it is assumed that the Stop condition has been received and the previous data has been successfully received. If the BUS state changes to the Active state before detecting the Stop condition, the measurement stops. The measurement of Backward Frame stops at the maximum time of the specified transmission time, after which transmission is prohibited.
Note that the Backward Frame must be transmitted within the specified time (5.5 to 10.5 ms) even if it is known that collision will occur.

(c) Interrupt processing

The following shows the processing when an interrupt occurs that is used in this driver.

Table 27 SoftDALI driver interrupt list

Interrupt name	Timer generation condition	Remarks
Transmission timing timer interrupt	When the period counter has elapsed	Every 416.7 us
Receive interrupt	When both edges of DALI BUS occur	
Timing measurement timer interrupt	When the timer elapses When the Stop condition timer has elapsed Active state timer has elapsed When Break time timer expires	Use the same timer depending on conditions 2400us 944us 1200us
Backward settling time measurement timer interrupt	When the timer elapses	5500us 10500us
Settling time measurement timer interrupt	When the timer elapses	4000us 13500us 14900us 16300us 17900us 19500us 100000us

Transmission timing timer interrupt
The processing when a transmission timing timer interrupt occurs is shown below.

Table 28 List of interrupt processing of transmission timing timer

Condition	Processing
Not the last data transmitted	・Set transmission pin according to transmission data 0: Low (Active state) 1: High (Idle state) ・Set transmission state Set the transmitted state in the internal variable
The last data transmitted	・Set transmission pin according to transmission data 0: Low 1: High ・Set transmission state Set the transmitted state in the internal variable ・Stop transmission timing timer

Receive interrupt timer interrupt

The following shows the processing when a reception timer interrupt occurs.

Table 29 List of interrupt processing of reception timing timer

Condition	Processing
Falling edge interrupt	・Bus state acquisition ・Collision detection ・Confirmation of bit width ・Stop condition timer stop ・Activate Active state timer (only when there is transmission) ・Stop the transmission timing timer (only when a collision occurs) ・Transmission pin Low (Active state setting) (only when Collision occurs) ・Break time timer start (only when Collision occurs) ・Settling time measurement stop ・Save received data (when receiving normally) ・Status setting according to the status (both transmission and reception)
Rising edge interrupt	・Bus state acquisition ・Collision detection ・Confirmation of bit width ・Active state timer stopped (only when there is transmission) ・Stop condition timer start (except when Collision occurs) ・Stop the transmission timing timer (only when a collision occurs) ・Transmission pin Low (Active state setting) (only when Collision occurs) ・Break time timer start (only when Collision occurs) ・Start Settling time measurement (except when Collision occurs) ・Backward settling time measurement start (other than when collision occurs) ・Save received data (when receiving normally) ・Status setting according to the status (both transmission and reception)

Timing measurement timer interrupt

The processing when a timer interrupt for timing measurement occurs is shown below.

Table 30 List of timer interrupt processing for timing measurement

Condition	Processing
Stop condition interrupt	・Transmission complete setting (only when there is transmission) ・Confirmation of received data (data conversion) ・Reception complete setting ・Status setting according to the status (both transmission and reception) ・Comparison with previous received data ・Twice confirmation setting
Active state interrupt	・Collision detection setting (internal variable) ・Stop the transmission timing timer (only when a collision occurs) ・Transmission pin Low (Active state setting) (only when Collision occurs) ・Break time timer start (only when Collision occurs) ・Status setting according to the status (both transmission and reception)
Break time interrupt	・Bus state acquisition ・Settling time setting (Recovery time, usually Settling time) ・Status setting according to the status (both transmission and reception)

Free-run timer interrupt

The following shows the processing when a free-run timer interrupt occurs.

Table 31 List of free-run timer interrupt processing

Condition	Processing
Backward settling time measurement timer interrupt	・In case of Minimum time Backward transmission possible setting ・In case of Maximum time Backward transmission prohibited setting
Settling time measurement timer interrupt	・In case of Collision Recovery Retransmission processing of Frame when collision occurs ・In case of Priority1 Priority1 transmission processing ・In case of Priority2 Priority 2 or higher transmission processing ・In case of Priority3 Priority 3 or higher transmission processing ・In case of Priority4 Priority 4 or higher transmission processing ・In case of Priority5 Priority 5 or higher transmission processing ・In case of Twice Disable Twice Frame

(d) Transmit / receive data conversion

The handling of the transmitting and receiving data to be used in the present driver are shown below.

Transmission data

Transmission data is set from the application.

The following shows how to arrange transmission data.

Figure 33 Example of transmission data arrangement

Assume that the above 24-bit data is created on the application.

In DALI communication, data is normally transmitted in the order of bit23 to bit0.

This driver allows data up to 256 bits (32 bytes) to target Proprietary data. Normal DALI data is up to 32 bits, so uint32_t can be used, but for Proprietary, it must be a uint8_t array.

In this driver, transmission data is handled by the pointer of uint8_t, so data conversion is required.

The data array when setting 24-bit data to this driver is as follows.

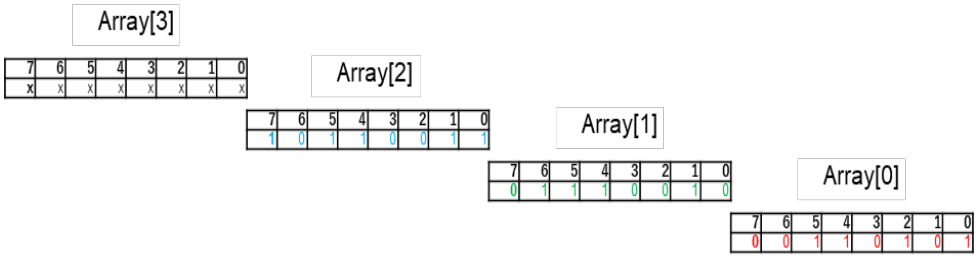

Figure 34 How to store in 8-bit data array

In this driver, data is transferred using the head pointer of the above data array.

Encoding of transmitted data

In this driver, the data set as above is converted to Manchester code used in DALI communication and used.

The conversion method to Manchester code used for DALI transmission is shown below.

Table 32 Manchester code

	Logical 0		Logical 1	
Data bits	0		1	
Manchester code	1	0	0	1

In addition, the order of data is swapped to make it easier to understand the use of the processing counter in transmission.

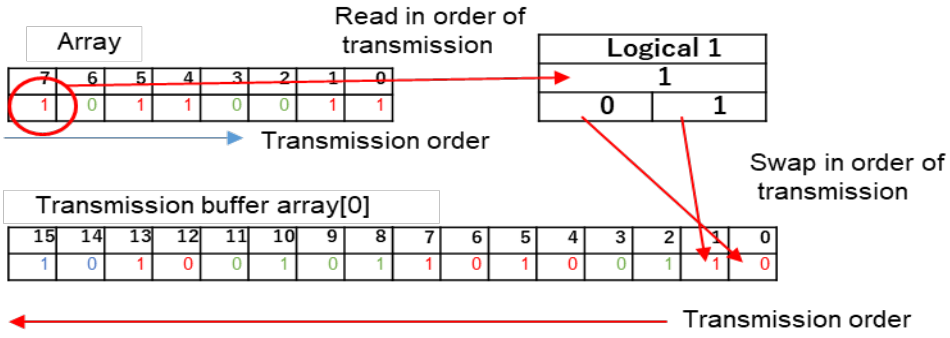

Figure 35 Conversion to Manchester code

Decode received data
The receive data is set by decoding the Manchester code obtained from DALI BUS.
The following shows how to arrange received data.

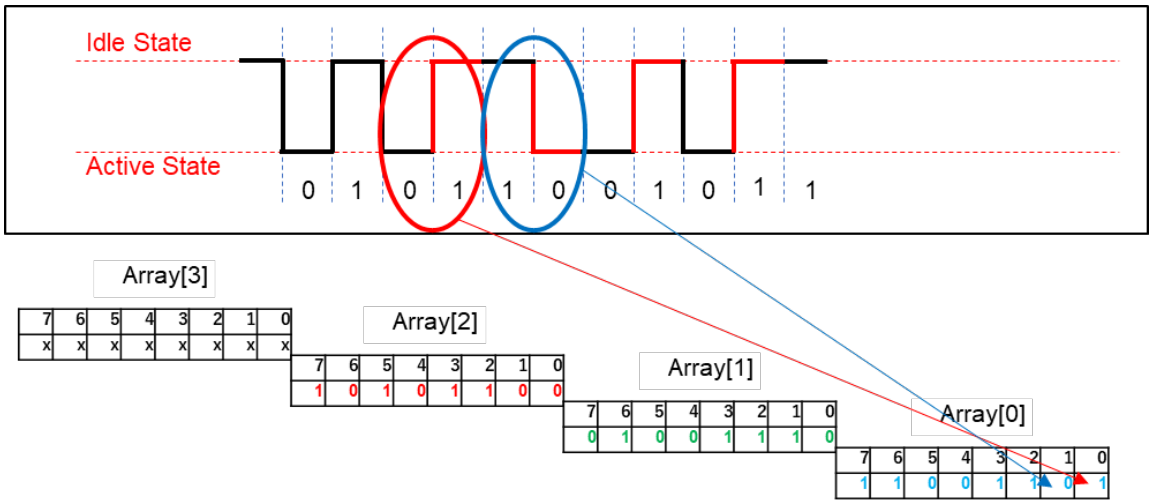

Figure 36 Decoding received data

Since it is not known how many bits the received data is transmitted, set the received data from the top of the array. You can see that 24bit data was transmitted as above when the Stop condition was received.
If the number of bits is not specified at this point, a reception error will occur.
By the number of bits is known, you can create the data to pass to the application.

Inversion of received data
When reception is completed, the stored data is the reverse of the normal DALI data bit order.
When passing to an application, the bit order must be rearranged to that of normal DALI data.

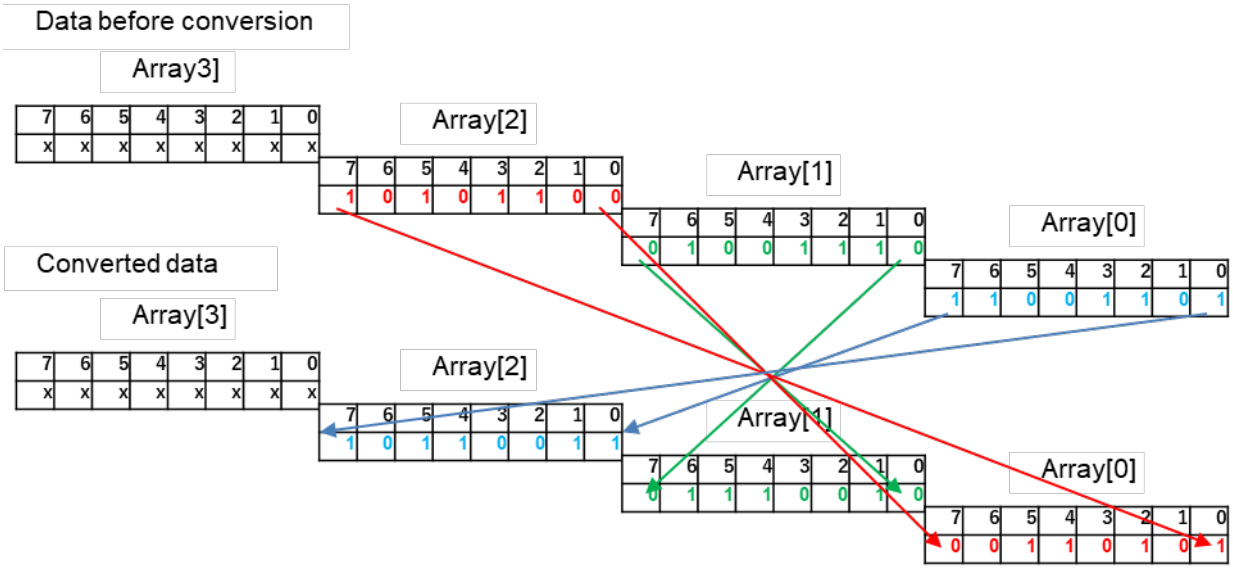

Figure 37 Inversion of received data

Converts the received data as described above when a Stop condition reception interrupt occurs.

(3) SoftDALI driver configuration
The driver configuration is described below.

(a) File structure
The file configuration of this driver is shown below.

Table 33 SoftDALI driver configuration file list

File name	Description
r_sddrv_api.c	SoftDALI API section source file
r_sddrv_api.h	SoftDALI API section header file
r_sddrv_tx.c	SoftDALI transmission source file
r_sddrv_tx.h	SoftDALI transmission header file
r_sddrv_rx.c	SoftDALI reception source file
r_sddrv_rx.h	SoftDALI reception header file
r_sddrv_timer.c	SoftDALI timer related source file
r_sddrv_timer.h	SoftDALI timer related header file
r_sddrv_frame.c	SoftDALI Frame decode encoding processing source file
r_sddrv_frame.h	SoftDALI Frame decode encoding processing header file
r_sddrv_com.c	SoftDALI application communication related source file
r_sddrv_com.h	SoftDALI application communication related header file
r_sddrv_gpio.c	SoftDALI pin related source file
r_sddrv_gpio.h	SoftDALI pin related header file
r_sddrv_def.h	SoftDALI driver internal definition header file
r_sddrv_user.h	SoftDALI user definition related header file

(b) Public macro definition
The macros provided by this driver are shown below.

Table 34 SoftDALI driver function return value list

macro	value	Description
SDDRV_RET_OK	(0)	Successful completion
SDDRV_RET_WAIT	(1)	Waiting for transmission completion, waiting for reception completion
SDDRV_RET_NG	(-1)	Abnormal termination
SDDRV_RET_ERR_PARAM	(-2)	Parameter error
SDDRV_RET_BUFF_FULL	(-3)	Transmission buffer in use
SDDRV_RET_SEND_NONE	(-4)	No transmission data
SDDRV_RET_ERR_SEND_TIMING	(-5)	Transmission timing error (Backward only)

※int16_t

Table 35 SoftDALI driver buffer usage types

macro	value	Description
SDDRV_FRAME_BUFF_USED_NONE	(0)	All buffers not used
SDDRV_FRAME_BUFF_USED_BF	(1)	Backward Frame buffer in use
SDDRV_FRAME_BUFF_USED_FF	(2)	Forward Frame buffer in use

* When both Backward and Forward buffers are used, it is assumed that an OR operation is returned.
`SDDRV_FRAME_BUFF_USED_BF | SDDRV_FRAME_BUFF_USED_FF`

(c) Structure / Union / Enum

The structures, unions, and enumerations provided by this driver are shown below.

Table 36 App communication structure (st_sddrv_com_param_t)

Member variable type	Member variable name	Description
EventGroupHandle_t	sendCompleteEventHandl	Receive and send completion event handle
EventBits_t	sendCompleteBFEventBit	Backward Frame send completion event bit
EventBits_t	sendCompleteFFEventBit	Forward Frame send completion event bit
EventBits_t	sendErrorCompleteEventBit	Send completion event bit at sending error
QueueHandle_t	recvCompleteQueueHandl	Receive completion data Queue handle
QueueHandle_t	recvCompleteEventQueueHandl	Receive completion event Queue handle
EventBits_t	recvCompleteEventBit	Receive complete event bit
EventBits_t	recvISendCompleteEventBit	Reception complete event bit (self-sent data)
EventBits_t	systemFailureEventBit	System failure detection event bit
EventBits_t	backwardTimeoutEventBit	Backward Frame timeout event bit

* Set Queue and event at each timing.

Table 37 Transmission Frame structure (st_sddrv_send_param_t)

Member variable type	Member variable name	Description
uint8_t	priority	Transmission Frame Priority (1-5)
uint8_t	frameType	Type of transmission Frame SDDRV_FRAME_BACKWARD SDDRV_FRAME_FORWARD SDDRV_FRAME_COLLAPSED
uint8_t*	pframe_data	Top pointer of transmission Frame data
uint16_t	frame_bit_num	Size of transmission Frame (bit size specified)

* The transmission frame priority when SDDRV_FRAME_BACKWARD is specified is ignored.

Table 38 Receive Frame structure (st_sddrv_recv_param_t)

Member variable type	Member variable name	Description
uint16_t	frame_bit_num	Size of teception Frame (bit size)
bool	is_twice	Twice timing (true: twice false: no twice)
bool	isError	Data status (true: error false: normal)
uint8_t	frame_data[SDDRVE_DATA_SIZE_MAX]	Top pointer of reception Frame data

* IsTwice: Set true if the same Frame as before was received within the Twice Frame timing.
* IsError: Set false if the received data is normal data, and true if the received data is error data.

Table 39 Device type enumeration (e_device_type_t)

macro	value	Description
SDDRV_DEV_SINGLE_APP	(0)	Single master
SDDRV_DEV_MULTI_APP	(1)	Multi master

* uint8_t
* Single master：application controler Standalone / Multi master：Multiple Application controller

Table 40 Frame type enumeration (e_frame_type_t)

macro	value	Description
SDDRV_FRAME_BACKWARD	(0)	Backward Frame
SDDRV_FRAME_FORWARD	(1)	Forward Frame
SDDRV_FRAME_COLLAPSE	(2)	Collapsed Frame

Table 41 Operation status enumeration (e_sddrv_state_t)

macro	value	Description
SDDRV_STATE_INIT	0	Initial state
SDDRV_STATE_STANDBY	1	Waiting for completion of initialization
SDDRV_STATE_SEND_WAIT	2	Waiting for sending
SDDRV_STATE_SEND	3	Sending
SDDRV_STATE_SEND_ERR	4	Send error (collision occurred)
SDDRV_STATE_SEND_ERR_RECOVER	5	During sending error recovery (collision recovery)
SDDRV_STATE_SEND_RETRY	6	During resending (settling time start: recovery time included)
SDDRV_STATE_SEND_COMPLETE	7	Send completely
SDDRV_STATE_RECV_WAIT	8	Wait for reception (not received, abnormal data received)
SDDRV_STATE_RECV	9	Receiving
SDDRV_STATE_RECV_COMPLETE	10	Receive completely (locked data reception completed)

(d) Macro for user-defined operation definition build

Set the definition related to the operation when this driver is used by the user.
Be sure to set at build time.

Table 42 Symbol for operation setting build

macro	Description
SDDRV_MODE_PROPRIETARY	Set Extended Frame 0: Disabled / 1: Enabled Default 0: Disabled

* This symbol is used to set the operation of this driver when building. Be sure to set it when building.

Table 43 Macro for operation setting build

macro	default	Description
SDDRV_PROPRIETARY_NUM	3	Type of Proprietary Frame used (1-3)
SDDRV_PROPRIETARY_BIT_1	64	Number of user-specified bits 1
SDDRV_PROPRIETARY_BIT_2	128	Number of user-specified bits 2
SDDRV_PROPRIETARY_BIT_3	256	Number of user-specified bits 3

* This macro is used to define the Proprietary permission bits of this driver when building. If you want to change it, be sure to set it when building.
* It is valid only when SDDRV_MODE_PROPRIETARY is valid (1).

Table 44 Build macro for timer clock

macro	default	Description
INPUT_CAPTURE_CLOCK_MHZ	30	Clock (MHz) used for receive input capture timer.

Table 45 Build macro for timer adjustment

macro	default	Description
SDDRV_BREAK_TIME_INTERVAL_OFFSET	(40)	One shot timer adjustment macro for Stop condition, Active state, and Break time.
SDDRV_SYSTEM_FAILURE_INTERVAL_OFFSET	(-16)	Macro for timer adjustment for system failure

(e) Macro for internal definition

The macros that define the bit timing used in this driver are shown below.

Table 46 Macro for internal operation setting build

macro	default Description
SDDRV_STOP_CONDITION_TIMER_ BASE_INTERVAL	2150 Timer base value for stop condition measurement (ns)
SDDRV_ACTIVE_STATTE_TIMER_ BASE_INTERVAL	944 Timer base value for Active state measurement (ns)
SDDRV_BREAK_TIME_TIMER_ BASE_INTERVAL	1200 Timer base value for Break time measurement (ns)
SDDRV_SYSTEM_FAILURE_ BASE_INTERVAL	550 Timer base value for System failure measurement (ns)
SDDRV_STOP_CONDITION_ TIMER_INTERVAL	SDDRV_STOP_CONDITION_TIMER_BASE_INTERVAL + SDDRV_STOP_CONDITION_INTERVAL_OFFSET
	Stop condition measurement timer base value plus offset value (ns) Use this value.
SDDRV_ACTIVE_STATTE_ TIMER_INTERVAL	SDDRV_ACTIVE_STATE_TIMER_BASE_INTERVAL + SDDRV_ACTIVE_STATE_INTERVAL_OFFSET
	Active state measurement timer base value plus offset value (ns) Use this value.
SDDRV_BREAK_TIME_ TIMER_INTERVAL	SDDRV_BREAK_TIME_TIMER_BASE_INTERVAL + SDDRV_BREAK_TIME_INTERVAL_OFFSET
	Break time measurement timer base value plus offset value (ns) Use this value.
SDDRV_SYSTEM_ FAILURE_INTERVAL	SDDRV_SYSTEM_FAILURE_BASE_INTERVAL + SDDRV_SYSTEM_FAILURE_INTERVAL_OFFSET
	System failure measurement timer base value plus offset value (ns) Use this value.
SDDRV_RISING_TIME_ TIMER_INTERVAL	SDDRV_RISING_TIME_VALUE + SDDRV_INTERRUPT_SYSTEM_DELAY_VALUE
	Value obtained by adding start-up delay time to pin rise time (us)

Table 47 Bit width threshold macro (without Collision measurement)

macro	default	Description
SDDRV_BIT_WIDE_1st_HALF_MIN	300000	Minimum value of First section half bit (ns)
SDDRV_BIT_WIDE_1st_HALF_MAX	625000	Maximum value of First section half bit (ns)
SDDRV_BIT_WIDE_2nd_HALF_MIN	300000	Minimum value of half bit of Second section (ns)
SDDRV_BIT_WIDE_2nd_HALF_MAX	580000	Maximum value of half bit of Second section (ns)
SDDRV_BIT_WIDE_2nd_D_HALF_MIN	580000	Minimum value of Double half bit of Second section (ns)
SDDRV_BIT_WIDE_2nd_D_HALF_MAX	1100000	Maximum value of Double half bit of Second section (ns)

Table 48 Bit width threshold macro (when there is Collision measurement)

macro	default	Description
SDDRV_BIT_WIDE_1st_HALF_COL_MIN	365000	Minimum value of First section half bit (ns)
SDDRV_BIT_WIDE_1st_HALF_COL_MAX	476000	Maximum value of First section half bit (ns)
SDDRV_BIT_WIDE_2nd_HALF_COL_MIN	365000	Minimum value of half bit of Second section (ns)
SDDRV_BIT_WIDE_2nd_HALF_COL_MAX	467000	Maximum value of half bit of Second section (ns)
SDDRV_BIT_WIDE_2nd_D_HALF_COL_MIN	762000	Minimum value of Double half bit of Second section (ns)
SDDRV_BIT_WIDE_2nd_D_HALF_COL_MAX	905000	Maximum value of Double half bit of Second section (ns)

(f) API function list

The list of API functions provided by this driver is shown below.

Table 49 API function list

Function name	Description
R_SDDRV_Init	Initialize the SoftDALI driver.
R_SDDRV_RecvStart	Start the receiving process.
R_SDDRV_CheckSendBuff	Check the usage status of the transmission data buffer.
R_SDDRV_Send	Request start of transmission Return immediately without waiting for transmission to be completed
R_SDDRV_SendCancel	Stop sending and cancel if not sent.
R_SDDRV_GetVersion	Get the version number of SoftDALI driver.

(g) API function specification

R_SDDRV_Init

Function name	void R_SDDRV_Init (st_sddrv_com_param_t* p_param, e_device_type_t dev_type)
Argument	st_sddrv_com_param_t* p_param The head pointer of the application communication structure e_device_type_t dev_type App communication structure SDDRV_DEV_SINGLE_APP SDDRV_DEV_MULTI_APP SDDRV_DEV_INPUT_DEVICE
Return value	void
Description	Initialize SoftDALI driver. Set device type Initialize each component

R_SDDRV_RecvStart

Function name	int16_t R_SDDRV_RecvStart (void)
Argument	None
Return value	Int16_t SDDRV_RET_OK: Normal termination SDDRV_RET_NG: Abnormal termination
Description	Start reception with SoftDALI driver. Enables both edges interrupt input capture interrupt for reception.

R_SDDRV_CheckSendBuff

Function name	uint8_t R_SDDRV_CheckSendBuff (void)
Argument	None
Return value	uint16_t bit0: Backward frame buffer use flag 0: not used 1: used bit1: Forward frame buffer use flag 0: not used 1: used
Description	Check the usage status of the transmission data buffer.

R_SDDRV_Send

Function name	int16_t R_SDDRV_Send (const st_sddrv_send_param_t* p_send)
Argument	const st_sddrv_send_param_t* p_send The head pointer of the transmission Frame structure
Return value	int16_t SDDRV_RET_OK: Normal termination SDDRV_RET_ERR_PARAM: Parameter error SDDRV_RET_BUFF_FULL: Sending previous data SDDRV_RET_ERR_SEND_TIMING: Transmission timing error occurred
Description	Set the data to be sent and start sending. It returns immediately without waiting for transmission completion. If the previously set data has not been transmitted, “SDDRV_RET_BUFF_FULL” is returned. "SDDRV_RET_ERR_SEND_TIMING" is returned if the specified time has already been exceeded at the time of Backward transmission request. When the Forward Frame is in the transmission waiting state, the Backward Frame can be additionally set. In this case, the backward frame is transmitted first. Forward Frame cannot be additionally set when Backward Frame is in the transmission waiting state. In this case, "SDDRV_RET_BUFF_FULL" is returned.

R_SDDRV_SendCancel

Function name	uint8_t R_SDDRV_SendCancel (void)
Argument	None
Return value	Int16_t SDDRV_STATE_OK: Normal termination SDDRV_RET_SEND_NONE: No transmission
Description	Stop transmitting Transmitting is stopped while waiting for transmission or during transmission.

R_SDDRV_GetVersion

Function name	uint16_t R_SDDRV_GetVersion (void)
Argument	None
Return value	uint16_t Upper 8 bits: major version Lower 8 bits: minor version
Description	Returns the version of the SoftDALI driver.

(h) List of internal global variables

The internal global variables used in this driver is shown below.

Table 50 Internal Global List 1/7

Member variable type	Member variable name	Description
e_sddrv_state_t	g_transmitter_status	Transmitter status SDDRV_STATE_INIT SDDRV_STATE_STANDBY SDDRV_STATE_SEND_WAIT SDDRV_STATE_SEND SDDRV_STATE_SEND_ERR SDDRV_STATE_SEND_ERR_RECOVER SDDRV_STATE_SEND_RETRY SDDRV_STATE_SEND_COMPLETE Setting timing When processing status changes Initial value SDDRV_STATE_INIT
e_sddrv_state_t	g_receiver_status	Receiver status SDDRV_STATE_INIT SDDRV_STATE_STANDBY SDDRV_STATE_RECV_WAIT SDDRV_STATE_RECV SDDRV_STATE_RECV_COMPLETE Setting timing When processing status changes Initial value SDDRV_STATE_INIT
e_device_type_t	g_device_type	Specified device type SDDRV_DEV_SINGLE_APP SDDRV_DEV_MULTI_APP SDDRV_DEV_INPUT_DEVICE Setting timing During initialization processing Initial value SDDRV_DEV_SINGLE_APP
st_sddrv_com_param_t	g_com_info	Application communication structure storage area Setting timing During initialization processing Initial value ALL 0

Table 51 Internal Global List 2/7

Member variable type	Member variable name	Description
uint8_t	g_recv_data_buff[]	Receive buffer (received data) Buffer size SDDRVE_DATA_SIZE_MAX Setting timing At reception interrupt Decode timing Delete timing When decoding data creation completion How to use Used as decoding source data when reception is completed
uint8_t	g_prev_data_buff[]	Previous receive buffer (received data) Buffer size SDDRVE_DATA_SIZE_MAX Setting timing At Stop condition interrupt After the previous data comparison Delete timing None How to use Used to compare the last received data with the current received data and to judge Twice.
uint16_t	g_cnt_recv_bit	Number of received bits Setting timing At reception interrupt Decode timing Delete timing At the start of reception How to use Confirmation and notification of the number of received bits
bool	g_stop_condition_flg	Stop condition reception flag false: Not received true: Received true setting timing At Stop condition interrupt false setting timing At the start of reception How to use Confirmation of Stop condition reception

Table 52 Internal Global List 3/7

Member variable type	Member variable name	Description
bool	g_recv_section_flg	Stop condition reception flag false：first section true：second section true setting timing At reception interrupt false setting timing None How to use Decode timing judgment
uint8_t	g_cnt_start_bit	Nomber of transmittion Start bit 0: Initial value 1: Start bit first section transmission 2: Start bit second section transmission Setting timing At reception interrupt Delete timing Set 1 at reception start How to use Judge data reception start timing during reception
bool	g_collision_flg	Collision detection flag false: Collision not detected true: Collision detected true setting timing When the receiver interrupts reception When the difference between the transmission state and the BUS state occurs && When Destroy area occurs (bit violation) When the Destroy area measurement timer interrupts false setting timing When a break time measurement interrupt is received How to use Stop condition measurement start possible / impossible judgment Initial value False If a difference occurs between the transmission state and the BUS state, but no Destroy area occurs, reception continues as normal transmission error processing.

Table 53 Internal Global List 4/7

Member variable type	Member variable name	Description
bool	g_recv_err_flg	Receive data error flag false: No receive data error true: Receive data error true setting timing At reception interrupt At Stop condition interrupt Active state interrupt false setting timing At the start of reception How to use Judgment of Bit violation, number of bits, and Destroy area detection of received data
bool	g_collision_retry_flg	Collision recovery judgment flag false: Normal recovery true: Collision recovery true setting timing Break time interrupt false setting timing At the start of reception How to use Confirm Settling time used for retransmission
bool	g_recv_start_flg	Receive start flag false: Not received true: Receiving true setting timing At the start of reception false setting timing At Stop condition interrupt When Active state timer interrupts How to use Judge decoding of received data, judge data decoding at Stop condition interrupt.
bool	g_twice_flg	Twice judgment flag false: Twice not received true: Twice received true setting timing At Stop condition interrupt false setting timing At twice timeout interrupt twice How to use Twice Frame reception notification to the application

Table 54 Internal Global List 5/7

Member variable type	Member variable name	Description
uint16_t	g_ff_data_buff[]	Forward Frame transmission buffer (encode) Buffer size SDDRVE_DATA_SIZE_MAX Setting timing When requesting transmission Delete timing When confirmation of transmission completion (At Stop condition interrupt) How to use Used as transmission data at transmission timing timer interrupt.
uint16_t	g_bf_data_buff	Backward Frame transmission buffer (encode) Setting timing When requesting transmission Delete timing When confirmation of transmission completion (At Stop condition interrupt) How to use Used as transmission data at transmission timing timer interrupt.
bool	g_is_ffset	Forward Frame transmission buffer setting flag true setting timing When requesting transmission false setting timing When confirmation of transmission completion (At Stop condition interrupt) How to use Used to confirm Forward Frame settings
bool	g_is_bfset	Backward Frame transmission buffer setting flag true setting timing When requesting transmission false setting timing When confirmation of transmission completion (At Stop condition interrupt) How to use Used to confirm Backward Frame settings
uint8_t	g_ff_priority	Forward Frame transmission priority Setting timing When requesting transmission Delete timing When confirmation of transmission completion (At Stop condition interrupt) How to use Used to determine Settling time used

Table 55 Internal Global List 6/7

Member variable type	Member variable name	Description
uint16_t	g_ff_size	Forward Frame size Setting timing When requesting transmission Delete timing When confirmation of transmission completion (At Stop condition interrupt) How to use Used to determine the size of transmitted data
uint16_t	g_encode_size	Transmit Frame Manchester Size Setting timing When requesting transmission Delete timing When confirmation of transmission completion (At Stop condition interrupt) How to use Used to determine the transmission completion of transmission data
uint16_t	g_transmitted_num	Number of transmitted bits Setting timing At transmission interrupt Delete timing At the start of transmission How to use Used as a transmitted data counter to determine transmission completion
uint8_t	g_send_state	Frame setting during transmission SDDRV_FRAME_SENDING_NONE: Not sent SDDRV_FRAME_SENDING_BACKWARD SDDRV_FRAME_SENDING_FORWARD Setting timing At the start of transmission Delete timing At the end of transmission How to use Used for frame determination during transmission

Table 56 Internal Global List 7/7

Member variable type	Member variable name	Description
uint8_t	g_bf_permit_state	Judge Backward Frame transmission permission SDDRV_BACKWARD_SEND_WAIT SDDRV_BACKWARD_SEND_POSSIBLE SDDRV_BACKWARD_SEND_IMPOSSIBLE Setting timing At timing measurement counter interrupt 5.5ms elapsed in Backward settling time measurement 10.5ms elapsed in Backward settling time measurement How to use Used for transmission setting and transmission start judgment
uint8_t	g_ff_permit_priority_num	Specify Forward Frame transmission permission priority 0：Transmission prohibited 1：Only priority 1 can be transmitted 2：Priority 2 or higher transmission allowed 3：Priority 3 or higher transmission allowed 4：Priority 4 or higher transmission allowed 5：Priority 5 or higher transmission allowed 6：Collision recovery Setting timing 0: When transmission is prohibited At receiving Start bit 1-5: Priority transmission permission At each Settling time measurement interrupt Collision recovery transmission permission Settling time measurement interrupt How to use Judge Forward frame transmission start permission
bool	g_tx_send_short_int_flg	High level transmission flag False: Not sent True: Sent Setting timing At interrupt for transmission rise At the start of transmission How to use Used to determine whether to transmit at normal transmission timing

2. DALI communication middleware

DALI communication middleware is a DALI communication driver middleware function. It notifies the DALI communication driver of the Frame requested by the Application Controller, notifies the Application Controller of the Frame received from the DALI communication driver, and others.

(1) Function

DALI communication middleware implements the following functions.

Table 57 Function list of DALI communication middleware

Function	Description
Transmission processing	Control transmission Queue, control transmission Frame priority, control transaction.
Reception processing	Control reception Queue
Soft DALI driver communication	Transmit and receive frames with Soft DALI driver.

(2) DALI communication middleware task: Transmission processing

This section describes the contents of transmission processing in the DALI communication middleware task.

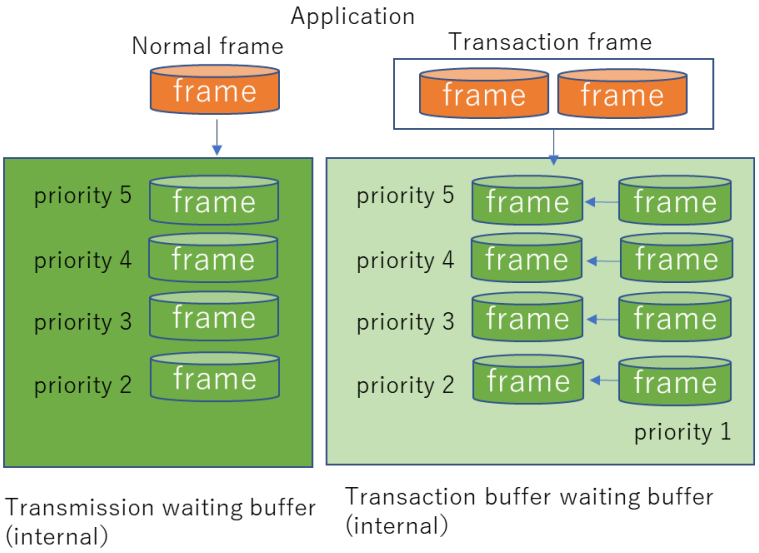

Figure 38 Overview of transmission processing

[Control contents]

1. Extract Frame from Queue set by application.
2. Determine whether the extracted Frame is a transaction and register it in the middleware buffer
   Register in the transaction buffer for transactions, and in the normal buffer for normal frames.
3. Sort by priority as internal processing after registration.
4. Return to 1 if Frame is in Queue.
5. Monitor the middleware's transmittion queue status to see if there is a frame to transmit next.
   If there is a transmission frame in the transmission queue and the buffer of the DALI communication driver is empty, pass the transmission frame to the DALI communication driver.
6. If the DALI communication driver can transmit, it notifies the start of transmission. If transmission is not possible, do nothing.
7. Wait for transmission completion of DALI communication driver (return to 1 in case of timeout)
8. Return to step 6 for transaction frame.
9. When transmitting Backward Frame, pass data to DALI communication driver without using internal buffer (reordering by priority).

(3) DALI communication middleware task: Reception processing
This section describes the contents of reception processing in the DALI communication middleware task.

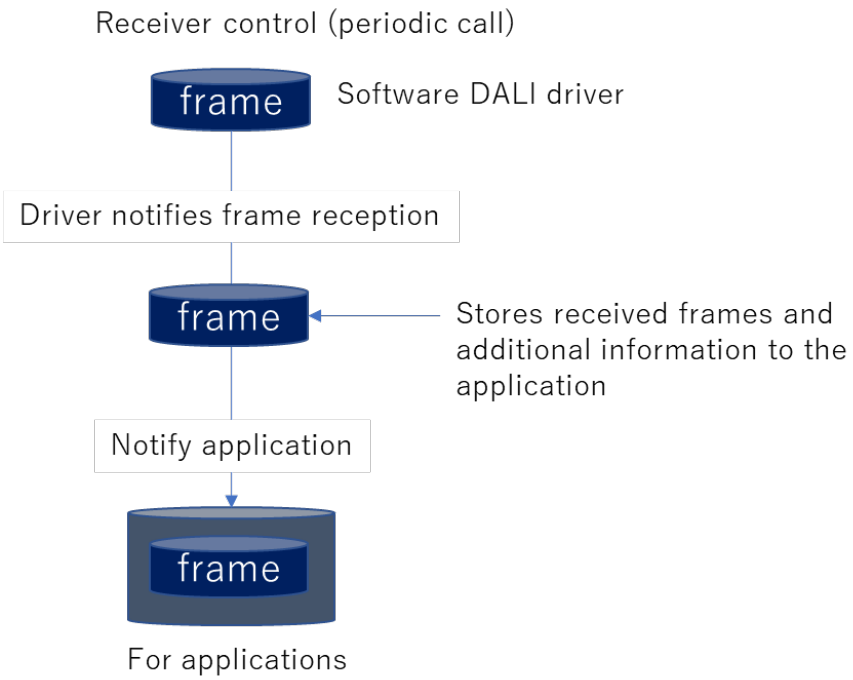

Figure 39 Overview of reception processing

[Control contents]
1. Receive receipt notification from DALI communication driver.
2. When receiving Frame
   Notify application controller of received frame using Queue.
3. For notifications such as Backward timeout
   Store timeout information and notify Application Controller using Queue.

4.6.2 IEC62386-103 standard and regulation part

The IEC62386-103 standard and regulation part is composed of an Application Controller library, an Application Controller application that runs the Application Controller library, and peripheral applications. For the Application Controller library, refer to the user's manual.

1. Application Controller application task

Application Controller Application task is a task that performs Application Controller operation. There are tasks that operate periodically for 1 ms and tasks that operate on receiving frames. Use the API functions provided by the Application Controller library to perform Application Controller operations and Frame processing.

The API function of Application Controller library corresponds to the black frame in the following figure.

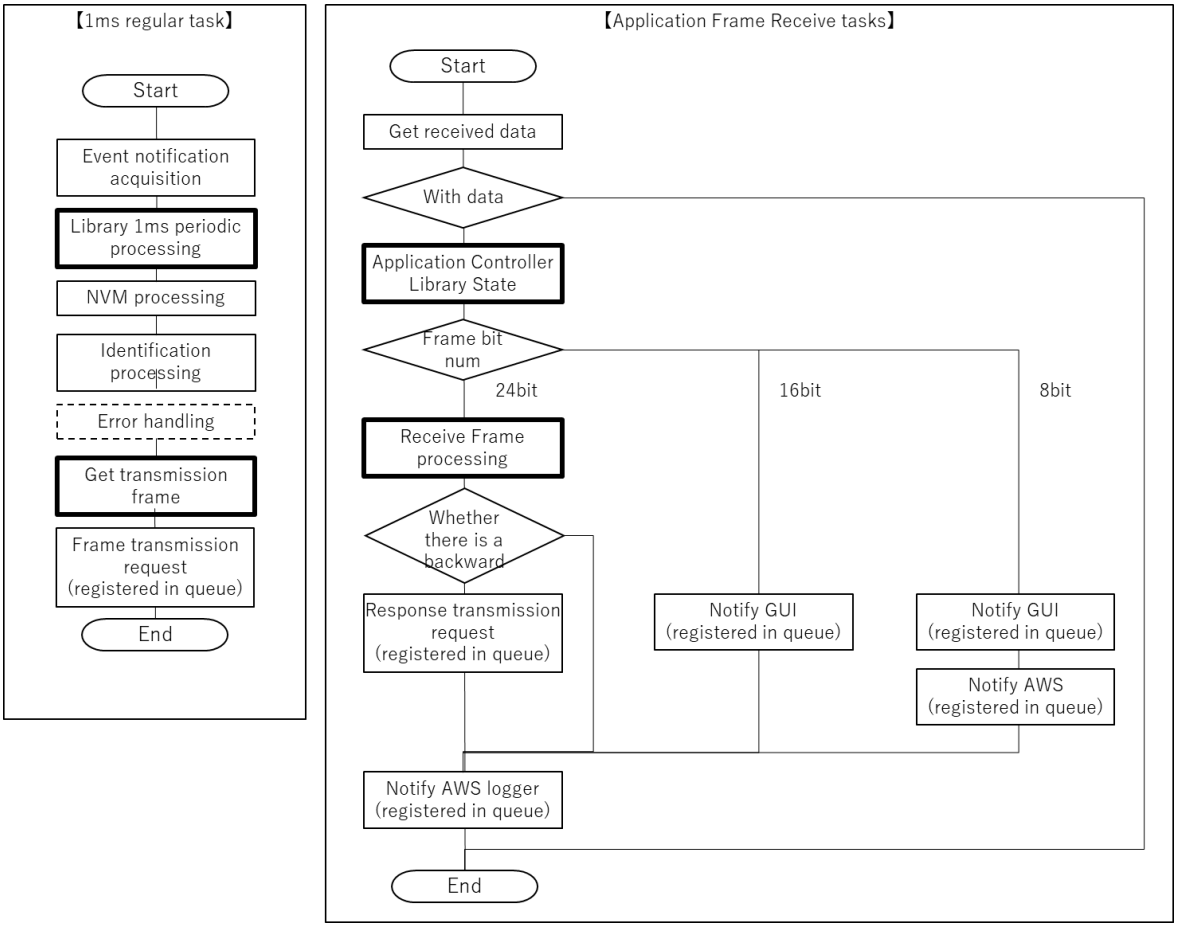

Figure 40 Application controller task flow diagram

1ms regular task

Tasks are processed periodically with 1ms regular events by software timer with Free-RTOS function. Performs 1ms library periodic processing specified in Application Controller library.
In this demo project, acquisition processing of the transmission frame is also performed in the 1ms regular task.
* Error processing of the broken line is implemented by the user. There is a sample code that notifies the library of the error status. When performing error processing, implement it referring to the sample code in Section(4).

Frame receiving task

Depending on the size of the received frame, notify the target Application Controller task of the Frame or process the Frame.

(1) Function specifications

r_app_1msec_interval_task

Function name	void r_app_1msec_interval_task (void * pvParameters)
Argument	void * pvParameters Free-RTOS task handler
Return value	void
Description	When a task notification is received, 1ms periodic processing is performed for each Logical Unit used. ● Perform the following processing for each Logical Unit to be used. ● Perform 1ms periodic processing of Application Controller library. ● Perform NVM periodic processing. ● Performs identification processing. ● If Forward Frame has occurred, get Forward Frame. ● Register Forward Frame to the transmission queue of DALI communication middleware task.

r_app_recv_dali_frame_task

Function name	void r_app_recv_dali_frame_task (void * pvParameters)
Argument	void * pvParameters Free-RTOS task handler
Return value	void
Description	When the target data is stored in the reception queue of the DALI communication middleware task, the transfer and notification of the frame to the Application Controller task by queue are performed according to the bit size or reception state of the frame and the state of the logical unit. Also, if the Logger function of the AWS web application is enabled, the notification to the Logger function is also performed. Process or notify by frame size 24bit frame From the received data, determine whether the Application Controller library is to execute the Frame process. Frame processing is performed for each Logical Unit to be used. If there is a response result (Backward) of the Frame processing, register Backward in the Queue for Backward of the DALI communication middleware task. 16bit frame In the case of self-issued Frame, notify the GUI application task. 8bit frame Notify GUI application and AWS Web application. Process or notify depending on reception status Backward reception TIMEOUT Notify GUI application and AWS Web application.

2. User Defined

Some of the behavior of the Application Controller needs to be defined by the user.
In this demo project, parts related to user dependencies are defined by symbols, and some sample codes are provided. Change and use according to the purpose of use.

(1) Explain each symbol

The following table shows the contents of each symbol.

Table 58 List of contents of each symbol

Symbol name	Default value	Description
IS_SINGLE_MODE	False (Multi-master mode)	Set Multi / Single master operation of Logical Unit used
LOGI_UNIT_NUM	1	Number of Logical Units used
IDENTIFY_LED_PORT	GPIO_PORT_B_PIN_0	LED lighting pin number used for identification
DEFAULT_OPERATING_MODE	0x00	Default value of Operating Mode of Logical Unit used
USE_PROPRIETARY_FRAME_SIZE_NUM	3	Number of Proprietary Frames used
PROPRIETARY_FRAME_SIZE_1	64	Number of bytes of Proprietary Frame transmitted
PROPRIETARY_FRAME_SIZE_2	128	Number of bytes of Proprietary Frame transmitted
PROPRIETARY_FRAME_SIZE_3	256	Number of bytes of Proprietary Frame transmitted

(2) How to change symbol settings

Change the value of the specified symbol from [Properties] -> [General] -> [Path and Symbol] -> [Symbol] tab.

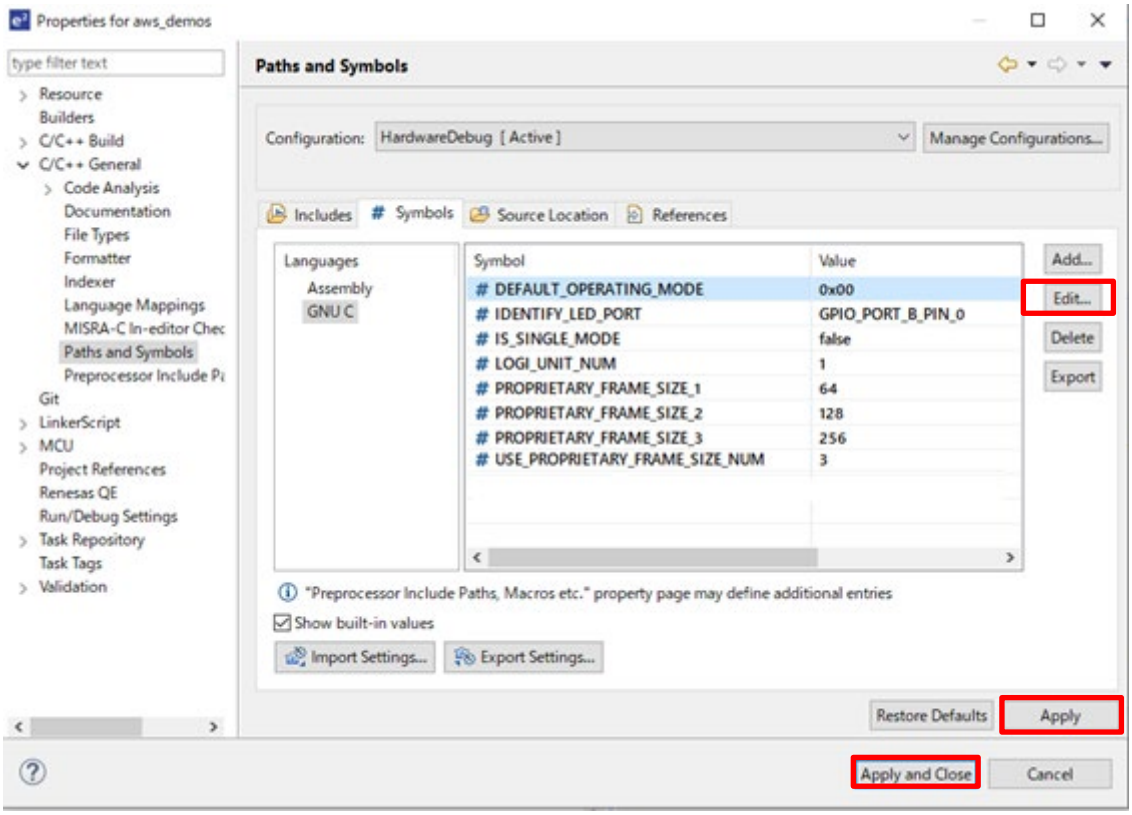

Figure 41 How to change the setting of each symbol

(3) When changing to multiple Logical Units

In this demonstration project, Logical Unit is defined as a single unit.
When using in multiple Logical Units, it is necessary to add the symbol value of the number of used Logical Units and the initialization definition of the entity.
The following is the sample code of Logical Unit definition of this demo project.

```
R_dali103_api.c

static st_appctrl_unit_t gs_logi_unit[LOGI_UNIT_NUM] =
{
 [0] = {.p_logi_unit = (st_appctrl_logiunit_t*)gs_logi_unit_buff[0],
       .is_always_active = true,
       .is_single_master = IS_SINGLE_MODE,
       .default_ope_mode = DEFAULT_OPERATING_MODE,
       .p_mbank_list = g_mbank_list,
       .p_mbank_info_list = g_mbank_info_list,
       .auto_save_timer = TIMER_STOP,
       },
};
```

The following shows an example of changing to multiple Logical Units. Change and use them according to your purpose.

Change procedure when three Logical Units are defined and used

1. Change LOGI_UNIT_NUM to 3 on the "Path and Symbol" screen
2. Add initialization definition of static st_appctrl_unit_t gs_logi_unit [LOGI_UNIT_NUM] as follows

```
R_dali103_api.c

static st_appctrl_unit_t gs_logi_unit[LOGI_UNIT_NUM] =
{
 [0] = {.p_logi_unit = (st_appctrl_logiunit_t*)gs_logi_unit_buff[0],
       .is_always_active = true,
       .is_single_master = IS_SINGLE_MODE,
       .default_ope_mode = DEFAULT_OPERATING_MODE,
       .p_mbank_list = g_mbank_list,
       .p_mbank_info_list = g_mbank_info_list,
       .auto_save_timer = TIMER_STOP,
       },
 [1] = {.p_logi_unit = (st_appctrl_logiunit_t*)gs_logi_unit_buff[1],
       .is_always_active = false,
       .is_single_master = IS_SINGLE_MODE,
       .default_ope_mode = DEFAULT_OPERATING_MODE,
       .p_mbank_list = g_mbank_list,
       .p_mbank_info_list = g_mbank_info_list,
       .auto_save_timer = TIMER_STOP,
       },
 [2] = {.p_logi_unit = (st_appctrl_logiunit_t*)gs_logi_unit_buff[2],
       .is_always_active = false,
       .is_single_master = IS_SINGLE_MODE,
       .default_ope_mode = DEFAULT_OPERATING_MODE,
       .p_mbank_list = g_mbank_list,
       .p_mbank_info_list = g_mbank_info_list,
       .auto_save_timer = TIMER_STOP,
       },
};
```

Note: For multiple Logical Units, pay attention to the Active setting of the Logical Unit. For the Active setting, refer to the IEC62386-103 standard document.

(4) Sample code for error handling

The DALI standard stipulates that the state of the Application Controller be made public, which includes error conditions. The error status includes whether an error has occurred and the details of the error, and the definition of the details of the error depends on the user.
Since this demo project uses the Application Controller library, it is necessary to notify and register error information.

The following sample code is provided for error notification to the Application Controller library. Change and use it according to the purpose.

```
void R_DALI103_UpdateErrorStatus (uint8_t unit, bool error)
{
  if(error == true)
  {
   R_DALI103_APPCTRL_SetAppCtrlError(gs_logi_unit[unit].p_logi_unit, 0xFF);
  }
  else
  {
   R_DALI103_APPCTRL_ClearAppCtrlError(gs_logi_unit[unit].p_logi_unit);
  }
}
```

(5) Sample code for identification processing

The DALI standard has an "IDENTIFY" operation for identifying connected devices. This operation differs depending on the hardware environment of the Control Device or Control Gear used, so the user must implement it.

This demo project has the following sample code. Change and use it according to the user's purpose.

```
void R_DALI103_IdentifyProcess (uint8_t unit)
{
  st_appctrl_status_t status;

  status =
R_DALI103_APPCTRL_GetLogiUnitStatus(gs_logi_unit[unit].p_logi_unit);
  if( status.is_identify_active == true )
  {

   R_GPIO_PinWrite(IDENTIFY_LED_PORT, GPIO_LEVEL_LOW);// LOW is Light up.
  }
  else if( status.is_identify_active == false )
  {
   R_GPIO_PinWrite(IDENTIFY_LED_PORT, GPIO_LEVEL_HIGH);// HIGH is off.
  }
}
```

(6) Sample code for operation in each Operating Mode

The DALI standard has an “Operating Mode”. In addition to the basic operation of the standard, if you want to perform your own operation, you can operate by setting the Operating Mode separately.

The following sample code is provided for interpreting Frame by each Operating Mode when receiving Frame. Change and use it according to the purpose.

```
int16_t R_DALI103_ExecuteRxForward (uint8_t unit, st_appctrl_rx_forward_t forward)
{
  uint8_t mode;
  st_appctrl_backward_t backward;
  int16_t retval = TAC_NO_ANSWER;
#if 0
  mode = R_DALI103_APPCTRL_GetOperatingMode(gs_logi_unit[unit].p_logi_unit);
   if(mode == 0)
   {
#endif
      backward =
   R_DALI103_APPCTRL_ExecuteRxForwardFrame(gs_logi_unit[unit].p_logi_unit,forward);
      switch( backward.result )
     {
      /***** Omitting *****/
      }
#if 0
   }
   else
   {
   /* user-defined processing */
   }
#endif
  return retval;
}
```

3. Implement a memory bank

(1) Memory bank setting

One of the specifications of the Application Controller is called a memory bank, which is specified in the IEC62386-103 standard “9.10 Memory banks”. Each data value that constitutes a memory bank is user-dependent and must be defined for each Logical Unit.

In this demonstration project, memory bank 0 and memory bank 1 that need to be implemented are set as shown in the table below.

Table 59 Macro definition

Macro	Value	Description
NO_CHANGE	(-1)	resetValue for locations that will not be reset
UNUSED_VALUE	(0x00)	defaultValue value of unimplemented location
MBANK_NUM	(2)	Number of memory banks
MBANK0_SIZE	(0x1B)	Size used by memory bank 0
MBANK1_SIZE	(0x11)	Size used by memory bank 1

Table 60 Value set in memory bank 0

Address	Description	Default value	Reset value	Memory type
0x00	Address of last accessible memory location	(MBANK0_SIZE - 1)	NO_CHANGE	ROM
0x01	Reserved	UNUSED_VALUE	NO_CHANGE	n.a.
0x02	Number of last accessible memory bank	(MBANK_NUM -1)	NO_CHANGE	ROM
0x03	GTIN byte 0 (MSB)	0x00	NO_CHANGE	ROM
0x04	GTIN byte 1	0x11	NO_CHANGE	ROM
0x05	GTIN byte 2	0x22	NO_CHANGE	ROM
0x06	GTIN byte 3	0x33	NO_CHANGE	ROM
0x07	GTIN byte 4	0x44	NO_CHANGE	ROM
0x08	GTIN byte 5 (LSB)	0x55	NO_CHANGE	ROM
0x09	Firmware version (major)	0x01	NO_CHANGE	ROM
0x0A	Firmware version (minor)	0x00	NO_CHANGE	ROM
0x0B	Identification number byte 0 (MSB)	0x00	NO_CHANGE	ROM
0x0C	Identification number byte 1	0x11	NO_CHANGE	ROM
0x0D	Identification number byte 2	0x22	NO_CHANGE	ROM
0x0E	Identification number byte 3	0x33	NO_CHANGE	ROM
0x0F	Identification number byte 4	0x44	NO_CHANGE	ROM
0x10	Identification number byte 5	0x55	NO_CHANGE	ROM
0x11	Identification number byte 6	0x66	NO_CHANGE	ROM
0x12	Identification number byte 7 (LSB)	0x77	NO_CHANGE	ROM
0x13	Hardware version (major)	0x01	NO_CHANGE	ROM
0x14	Hardware version (minor)	0x00	NO_CHANGE	ROM
0x15	101 version number	0x08	NO_CHANGE	ROM
0x16	102 version number of all integrated control gear	0xFF	NO_CHANGE	ROM
0x17	103 version number of all integrated control devices	0x08	NO_CHANGE	ROM
0x18	Number of logical control device units in the bus unit	LOGI_UNIT_NUM	NO_CHANGE	ROM
0x19	Number of logical control gear units in the bus unit	0	NO_CHANGE	ROM
0x1A	Index number of this logical control device unit	LOGI_UNIT_NUM-1	NO_CHANGE	ROM

Table 61 Value set in memory bank 1

Address	Description	Default value	Reset value	Memory type
0x00	Address of last accessible memory location	(MBANK1_SIZE - 1)	NO_CHANGE	ROM
0x01	Indicator byte	UNUSED_VALUE	NO_CHANGE	any [a]
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)
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 (LSB)	0xFF	NO_CHANGE	NVM (lockable)
a Purpose, default/ power on/reset value and memory access of these bytes shall be defined by the manufacturer.				

(2) Memory bank operation functions

Since the Application Controller implemented in this demonstration project uses the Application Controller library, it is necessary to implement the operation function for the memory bank and register it in the Application Controller library.

In this demonstration project, the functions to operate the memory bank are implemented as follows and registered in the library.

(a) Callback function type structure

The structure used when registering in the Application Controller library is shown below.

Table 62 Definition of memory bank access callback function type structure (st_appctrl_mbank_callback_t)

```
typedef struct
{
void (*p_Reset)(uint8_t unit, uint8_t bank);
int16_t (*p_Read)(uint8_t unit, uint8_t bank, uint8_t location);
int16_t (*p_Write)(uint8_t unit, uint8_t bank, uint8_t location, uint8_t
data);
void (*p_UnlatchRead)(uint8_t unit);
void (*p_CancelWrite)(uint8_t unit);
} st_appctrl_mbank_callback_t;
```

Table 63 List of functions to be registered in the library

Registration destination	Function name	Description
*p_Reset	r_callback_mbank_reset	Set the RESET target data of the memory bank to the RESET value.
*p_Read	r_callback_mbank_read	Load a memory bank.
*p_Write	r_callback_mbank_write	Write to memory bank.
*p_UnlatchRead	NULL ※	Release the data latch (hold).
*p_CancelWrite	NULL ※	Cancel writing multi-byte data.
※Since this demo project does not use multi-byte data, it is a function not implemented.		

(b) Function specifications

r_callback_mbank_reset

Argument	uint8_t unit Logical unit number uint8_t bank Memory bank number to be reset
Return value	void
Description	Set the RESET target data of the memory bank number specified by the argument to the RESET value. The RESET process differs depending on the MEMORY TYPE of the memory bank. ROM: do nothing RAM: Set the RESET value according to the status of the Lock byte. NVM: After setting the RESET value according to the status of the Lock byte, turn on the NVM save request.

r_callback_mbank_read

Argument	uint8_t unit Logical unit number uint8_t bank Memory bank number to be read uint8_t location The specified address of the memory bank to read
Return value	int16_t 0x00 - 0xFF: Read value DALI103_APPCTRL_MBANK_BANK_IS_NOT_IMPLEMENT: The specified memory bank is not implemented DALI103_APPCTRL_MBANK_LOCATION_IS_NOT_IMPLEMENT: The specified location is not implemented
Description	Get the data at the specified address of the memory bank number specified by the argument. READ processing differs depending on the MEMORY TYPE of the memory bank. ROM: Return the default value. RAM and NVM: Return the read value. Return an error in the following cases: If the specified memory bank is not implemented: DALI103_APPCTRL_MBANK_BANK_IS_NOT_IMPLEMENT is returned. If the address of the specified memory bank is not implemented: DALI103_APPCTRL_MBANK_LOCATION_IS_NOT_IMPLEMENT is returned.

r_callback_mbank_write

Argument	uint8_t unit Logical unit number uint8_t bank Memory bank number to be written uint8_t location The specified address of the memory bank to write uint8_t data Written data
Return value	int16_t 0x00 - 0xFF: Data value written DALI103_APPCTRL_MBANK_BANK_IS_NOT_IMPLEMENT: The specified memory bank is not implemented DALI103_APPCTRL_MBANK_LOCATION_IS_NOT_IMPLEMENT: The specified location is not implemented DALI103_APPCTRL_MBANK_EXECUTE_ERROR: Execution error
Description	Write data to the specified address of the memory bank number specified by the argument. Write only when MEMORY TYPE of memory bank is RAM or NVM. In the case of NVM, turn on the NVM save request after writing. Returns an error in the following cases: If the specified memory bank is not implemented: DALI103_APPCTRL_MBANK_BANK_IS_NOT_IMPLEMENT is returned. If the address of the specified memory bank is not implemented: DALI103_APPCTRL_MBANK_LOCATION_IS_NOT_IMPLEMENT is returned. When the MEMORY TYPE of the specified memory bank is ROM: When the specified memory bank is in the read-only state: DALI103_APPCTRL_MBANK_EXECUTE_ERROR is returned.

4.6.3 Function application part

1. GUI application tasks

The GUI application task is a task for sending and receiving frames with the DALI master controller GUI. There are two tasks, an SCI communication task and a response task.

For SCI communication, use the corresponding FIT module "RX Family SCI Module FIT".

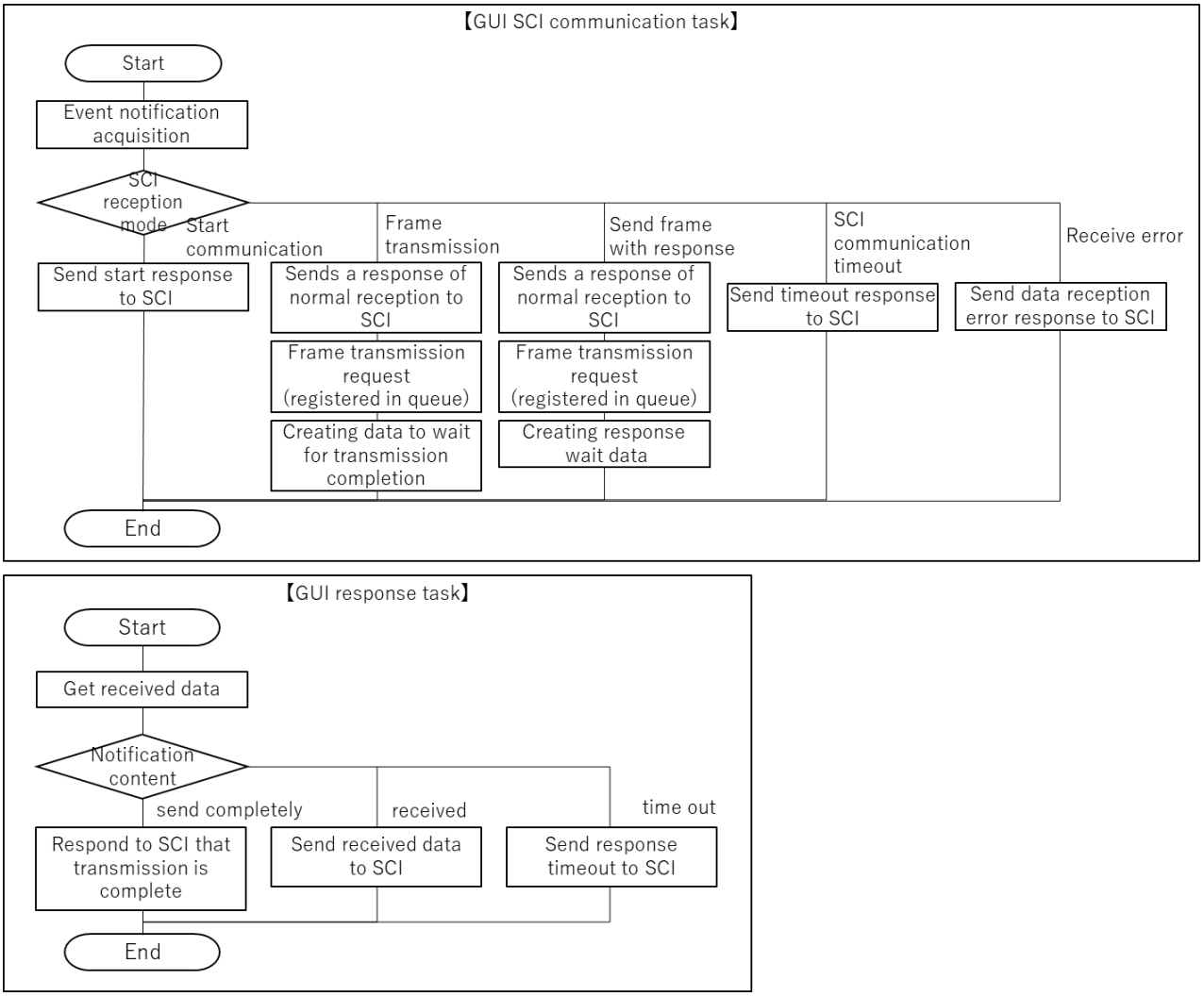

Figure 42 GUI application task flow diagram

- SCI communication task
  Performs SCI communication with the DALI Master Controller GUI and requests for Frame reception and transmission to the DALI communication middleware task. When making a transmission request, create a frame transmission completion or Backward response waiting state.

- Response task
  A response is returned to the DALI Master Controller GUI via SCI communication from "Waiting for transmission completion" and "Waiting for Backward response" created by the SCI communication task and the Frame transmission result. The transmission result of Frame is acquired by the notification by Queue from the Frame reception process of Application Controller application task.

(1) Function specifications

r_gui_recv_com_task

Function name	void r_gui_recv_com_task (void * pvParameters)
Argument	void * pvParameters Free-RTOS task handler
Return value	void
Description	Register the frame acquired by SCI communication in the response to the DALI master controller GUI and the transmission queue of the DALI communication middleware task, when receiving the task notification. GUI_MESSAGE_START It returns the version to the DALI Master Controller GUI as a response. GUI_MESSAGE_REQ_SEND_ONCE GUI_MESSAGE_REQ_SEND_TWICE After returning NORMAL to the DALI Master Controller GUI as a response, it registers the SCI reception data in the transmission Queue. * If registration to the transmission Queue fails, NORMAL is returned again. GUI_MESSAGE_REQ_NEED_ANSWER After returning NORMAL to the DALI Master Controller GUI as a response, it registers the SCI reception data in the transmission Queue. * If registration to the transmission Queue fails, NORMAL is returned again. GUI_MESSAGE_REQ_DATA_ANSWER After returning NORMAL to the DALI Master Controller GUI as a response, it registers the SCI reception data in the transmission Queue. * If registration to the transmission Queue fails, nothing is returned and it causes a timeout. GUI_MESSAGE_REPLY_ERROR_TIMEOUT If a reception timeout occurs when the request reception by SCI is 3 bytes or less, it returns a timeout error (0xE2) to the DALI master controller GUI. Other: Since the receive data is abnormal, it returns a receive error (0xE1) to the DALI master controller GUI as a response.

r_gui_report_com_task

Function name	void r_gui_report_com_task (void * pvParameters)
Argument	void * pvParameters Free-RTOS task handler
Return value	void
Description	When the target data is stored in the reception queue, it returns the reception data to the DALI master controller GUI. If a notification arrives while waiting for transmission completion: If the transmitted Frame is GUI_MESSAGE_REQ_SEND_ONCE or GUI_MESSAGE_REQ_SEND_TWICE, It returns NORMAL to the DALI Master Controller GUI when transmission is completed. If a timeout notification arrives while waiting for a Frame response: If the transmitted Frame is GUI_MESSAGE_REQ_NEED_ANSWER, it returns NORMAL to DALI Master Controller GUI. If waiting for the Frame result is GUI_MESSAGE_REQ_NEED_ANSWER, GUI_MESSAGE_REQ_DATA_ANSWER: It returns the data obtained from the reception queue to the DALI master controller GUI, and clears the waiting state for the frame result.

2. AWS Web application task

The AWS web application task is a task that transmits and receives Frames to and from the AWS web application. There are two tasks, the AWS-DALI communication task and the Logger task.

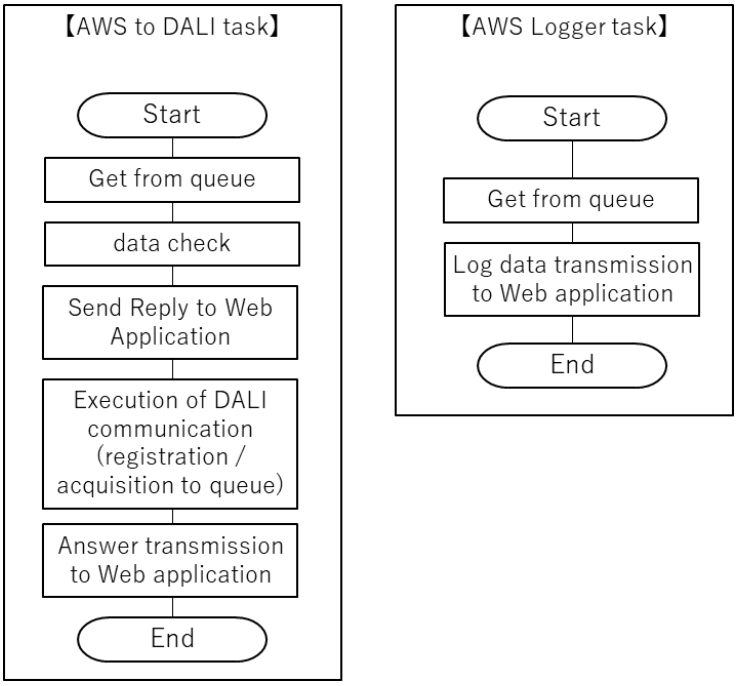

Figure 43 AWS Web application task flow diagram

AWStoDALI task

Start operation by notification from Queue from MQTT. Generate a REPLY message based on the frame check result and publishes it to MQTT. After that, DALI communication is executed, an ANSWER message is generated based on the notification contents of the reception queue by the Application Controller application, and the message is published to MQTT. After this, wait for the Queue notification from MQTT again.

Logger task

Get Frame output on DALI BUS and publish to MQTT.

(1) Topic

For MQTT communication, topics (data transmission destination and Subscribe registration destination) are required.
The list of topics is shown in the table below.

Table 64 Topic List

Publisher	Subscriber	Topic name
WEB App	RX65N	rx65n-cloud-kit-dali/<thingName>/toDevice
RX65N	WEB App	rx65n-cloud-kit-dali/<thingName>/fromDevice
RX65N	WEB App	rx65n-cloud-kit-dali/<thingName>/logger

* The thingName is a name (thing name) to identify rx65n-cloud-kit on AWS.

(2) Function specifications

prvAWStoDALITask

Function name	static void prvAWStoDALITask(void * pvParameters)
Argument	void * pvParameters Free-RTOS task handler
Return value	void
Description	When receiving the notification by Queue from MQTT, it registers the response to the AWS web application via MQTT whether it was received and register the acquired Frame in the transmission queue of the DALI communication middleware task. When receiving a notification by Queue from Frame reception processing of Application Controller application, it creates response data and returns it to AWS web application via MQTT. If the content of the notified frame is an abnormal value, DALI communication is not performed and an error is returned to the AWS web application via MQTT.

prvDALILoggerTask

Function name	static void prvDALILoggerTask(void * pvParameters)
Argument	void * pvParameters Free-RTOS task handler
Return value	void
Description	It transmits the Frame received by the Queue notification from the Frame reception process of the Application Controller application to the AWS web application via MQTT. If the Logger function is not enabled, notification by Queue will not be performed.

3. Matrix button application task

The matrix button application task has two tasks: the task of transmitting a frame by matrix button input (only 16-bit Forward Frame), and the task of changing or reading the Frame allocated to the matrix button from the AWS web application.
The allocated Frame is saved in the data flash, so there is no need to change the settings at startup.
Use the corresponding FIT modules "RX Family GPIO Module FIT" and "RX Family Flash Module FIT".

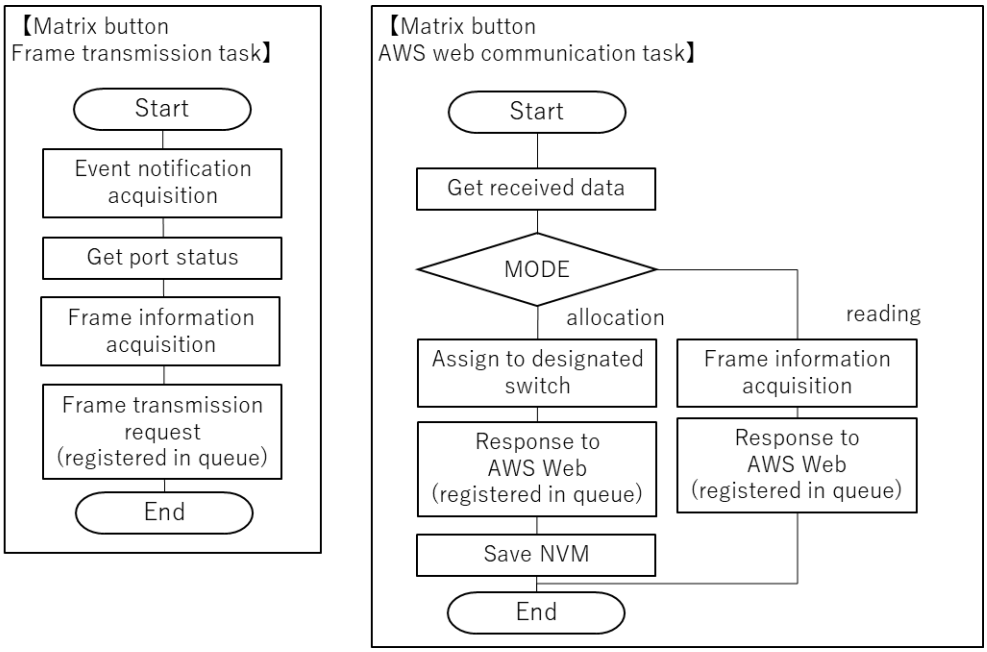

Figure 44 Matrix button application task flow diagram

- Frame transmission task
  If a matrix button has been pressed, it requests the transmission of the Frame assigned to that matrix button.

- AWS web communication task
  It processes according to the MODE received from the AWS web application.
  Allocate: Allocates the specified Frame to the matrix button.
  Read: Returns the Frame allocated to the specified matrix button to the AWS web application.

(1) Function specification

r_btn_recv_com_task

Function name	void r_btn_recv_com_task (void * pvParameters)
Argument	void * pvParameters Free-RTOS task handler
Return value	void
Description	Returns the Frame settings or Frame information for the matrix button specified from the AWS web application. Set Frame Register the result (OK / NG) of setting Frame on the matrix button in the AWS Web Application Notification Queue. Turn on the NVM write flag and write NVM. Read Frame Get the Frame assigned to the matrix button and register it in the AWS Web Application Notification Queue.

r_btn_sw_input_task

Function name	void r_btn_sw_input_task (void * pvParameters)
Argument	void * pvParameters Free-RTOS task handler
Return value	void
Description	Judge the input state of the matrix button from each pin, when an event is received notification of 10ms regularly, When a matrix button is input, obtain a Frame from the corresponding number and register the Frame in the transmission queue of DALI communication middleware. The number of times the frame is transmitted varies depending on the input state of the matrix button. ● Short press: Transmit Frame once ● Long press: Transmit Frame repeatedly ● Press multiple times: Invalid

4. Operation by User Implementation

In this demo project, the following empty source files and header files that the user can implement independently are prepared.

Table 65 File name list

File	Description
application_code¥application_controller¥dali_app¥user_app¥	
r_user_app.c	Source file for user implementation.
r_user_app.h	Header file for user implementation.

Use this when implementing applications not specified by the DALI standard, such as "By analyzing the Event received from the Input Device and the Frame received from another Application Controller, it grasps the DALI network environment and determines the Forward Frame to be transmitted to manage the network".

4.6.4 Main / Initialization

The following figure shows the initialization process of Application Controller.

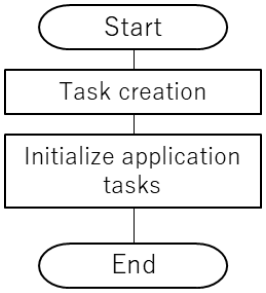

Figure 45 Initialization flow chart

1. Function specification

r_rxdali_app_task_create

Function name	bool r_rxdali_app_task_create (void)
Argument	void
Return value	bool [out]: true - task created successfully false - task created unsuccessfully
Description	Generate a task to be used in Application Controller. Generate Application Controller task and task used in DALI communication middleware. In the following cases, false is returned as an error. ● Return value of task creation is other than psPASS

r_rxdali_app_init

Function name	bool r_rxdali_app_init (void)
Argument	void
Return value	bool [out]: true – initialize successfully false – initialize unsuccessfully
Description	Initialize Application Controller. • Specify the ID data size of the data flash. • Generate Queue to be used in application layer. • Generate a task management event handler. • Initialize the Application Controller application and Application Controller library. • Initialize GUI application tasks. • Initialize the matrix button application task. • Generates and starts 1ms software timer for Free-RTOS function. In the following cases, false is returned as an error. • When an event handler cannot be created • When the initialization process of each APP fails • When a 1ms software timer handler cannot be created

r_rxdali_app_main

Function name	void r_rxdali_app_main (void)
Argument	void
Return value	void
Description	This is the main function of Application Controller. Call the task generation process and the initialization process of Application Controller. Starts the 1ms periodic software timer generated in the initialization processing. In the following cases, an error condition occurs and the operation transitions to an infinite loop. ● Initialization error ● Task generation error

4.6.5 Tasks and task management

The task that runs Application Controller uses the event notification function of Free-RTOS.
The following shows the occurrence of periodic events created by the callback of the software timer and the notification task that notifies the target task of the event that occurred.

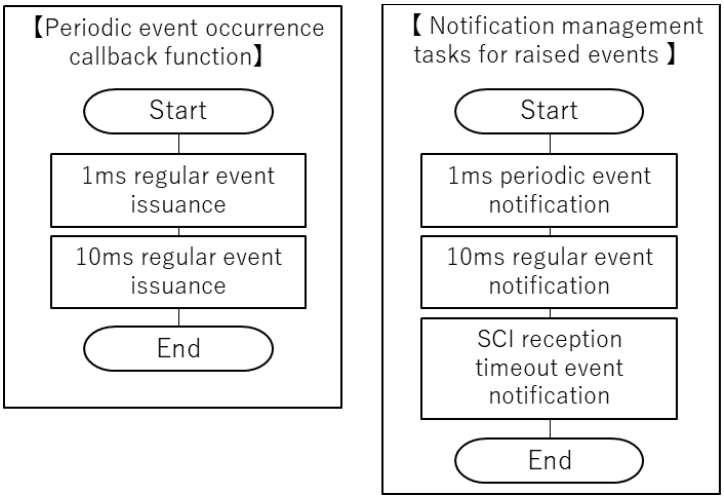

Figure 46 Task management flow diagram

1. Function specification

r_app_1msec_timer_interrupt

Function name	void r_app_1msec_timer_interrupt (TimerHandle_t xTimer)
Argument	TimerHandle_t xTimer Free-RTOS Timer handler with software timer function
Return value	void
Description	When 1ms elapses, 1ms processing is performed by the callback function of the software timer. 1ms interrupt occurs periodically. ● Issues a task notification event when 1 ms has elapsed. ● Issues a task notification event when 10 ms has elapsed.

r_app_event_control_task

Function name	void r_app_event_control_task (void * pvParameters)
Argument	void * pvParameters Free-RTOS Task Handler
Return value	void
Description	When a task notification event is issued, clear the event after notifying the target task. ・APP_EVENT_TASK_10MS_SW: Notify r_btn_sw_input_task. ・APP_EVENT_TASK_GUI_SCI: Notify r_gui_recive_com_task. * SCI events are issued during SCI communication interrupt processing.

4.6.6 Access to nonvolatile memory

The Application Controller application and the matrix button application access non-volatile memory.
Use the corresponding FIT module "RX Family II Flash Module FIT" to access non-volatile memory.

1. Function specification

r_df_init

Function name	bool r_df_init (uint8_t id_num, callback_write_finish_t p_callback)
Argument	uint8_t id_num [in] ：ID to use callback_write_finish_t p_callback [in]：Callback function pointer to call when writing is completed
Return value	bool [out] Result of initialization true: Initialization completed normally false: Initialization abnormal termination
Description	Initializes the area used for data flash.

r_df_finish

Function name	void r_df_finish (void)
Argument	void
Return value	void
Description	Terminates access to the data flash.

r_df_read

Function name	bool r_df_read (uint8_t id, void* p_data)
Argument	e_df_id_t id [in] data id: ID to be read void * p_data: Pointer to read and store
Return value	bool [out] Result of reading true: Read succeeded false: Read failed
Description	Reads the specified ID data from the data flash.

r_df_write_with_bgo

Function name	bool r_df_write_with_bgo (uint8_t id, void* p_data)
Argument	e_df_id_t id [in] data id: ID to be write void * p_data: Pointer to write and store
Return value	bool [out] Result of writing true: Write succeeded false: Write failed
Description	Writes the specified ID data to the data flash using bgo (background operation).

r_df_handler

Function name	r_df_handler(void)
Argument	void
Return value	e_df_status_t [out] Operation status DF_STATUS_OK: Operable DF_STATUS_BUSY: Busy status DF_STATUS_NG: Error occurred
Description	Check the access status of the data flash and start the operation.

r_df_is_write_process_idle

Function name	bool r_df_is_write_process_idle (void)
Argument	void
Return value	bool true - FLASH write control is idle false - FLASH write control is running
Description	This function checks whether the FLASH write control is operating.

5. Building the environment

This section describes how to install the software used in this demo project and how to connect the board.
Chapters 5.1 to 5.5 are commonly used regardless of the operation mode.

5.1 How to install e2studio

This demo project works on e2studio, so how to install e2studio is described.

1. Double-click on e2studio installer to invoke the e2studio installation wizard page.
   Click the [Next] to continue.
2. Install Folder
   Specify installation destination.
3. Device Families
   Select RX device support, click [Next].
4. Extra Components
   Select the language pack and RTOS support, click [Next].
5. Components
   Check that GCC for Renesas RX Build Support is checked in the optional components, and click [Next].
6. Additional Software
   Select GCC for Renesas RX 8.3.0 201904 compiler and click [Next].
   * If it is not displayed, register the user with the following URL and install the compiler.

   GNU toolchain download site:
   https://gcc-renesas.com/

7. License Agreement
   Read and agree to the license agreement, click [Next].
8. Shortcuts
   Select shortcut name for start menu and click [Next].
9. Summary
   A list of components to be installed is displayed, so check the contents and click [Install].
10. Installing…
    During installation, a dialog box for installing the software opens according to the item selected in additional software, so follow the on-screen instructions
11. Results
    The result of the installation is displayed. Check that there are no errors, then click the [Finish].

For details on installation and other basic operations, refer to the following user's manual.
e2studio User's Manual: Getting Started Guide for V7.0(r20ut4374)

5.2 How to import e2studio project

This section describes how to import this demo project.

1. Unzip the provided project file and place it in any location (folder).
   * If the path hierarchy is deep, the build may not pass. Pay attention to the location. Please move ¥workspace¥rx65n_dali folder to a place that has shorten directry name like C drive root.
2. Start e2studio and right-click or select [File] tab → [Import] in Project Explorer.
3. Select [Existing Projects into Workspace] from [General] and click [Next] button.
4. Select [Select root directory] and select the project file located in 1. from [Browse].
5. Select only "/projects/renesas/rx65n-cloud-kit-uart-sx-ulpgn/e2studio-gcc "
6. Check be checked in both two projects, “aws_demos” and “boot” on Project window.
7. Uncheck "Copy project to workspace"
8. Click [Finish].

5.3 How to set up EZ-0012

EZ-0012 is an evaluation board for lighting control equipped with DMX512 communication and infrared communication function in addition to DALI communication.
Using the software automatic generation tool Applilet EZ for HCD Controller, after generating / writing the code that can perform DALI communication, it can be operated as Control Gear.
When using the DALI-2 EXPANSION BOARD FOR EZ-0012, use Applilet EZ for DALI Control Gear.

This chapter explains the setting method using the following user's manual.
Appliret EZ for HCD Conttoller Ver.9.00 User's Manual(r20ut0435)
Appliret EZ for DALI Control Gear User's Manual(r11ut0078)

Note: When using EZ-0012 and DALI-2 EXPANSION BOARD or other boards, please refer to their respective user's manuals.

5.3.1 Install Applilet EZ for HCD Controller

For details on installing Applilet EZ for HCD Controller, refer to Chapter 2 of the User's Manual.

5.3.2 Generate / write code with Applilet EZ for HCD Controller

Based on the contents of the Applilet EZ for HCD Controller User's Manual, the operations and settings for DALI communication are described below.

1. Start Applilet EZ for HCD Controller and make the initial startup settings. (User's Manual Chapter 3)
2. Set up the evaluation board. (User's Manual Chapter 4.2)
   The figure below shows an example of evaluation board settings.

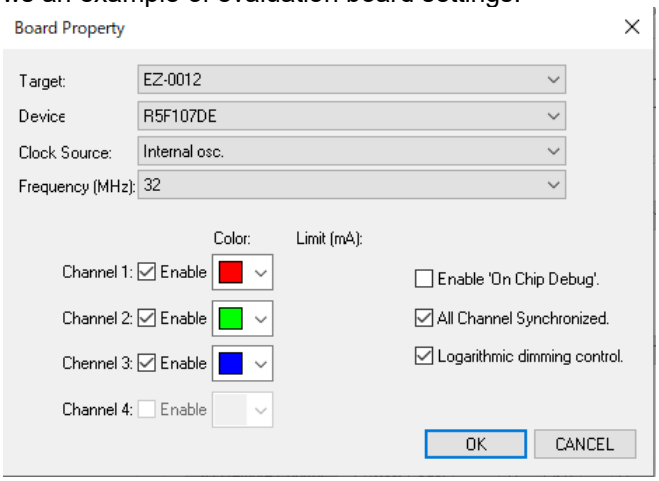

Figure 47 Board Property

3. Set the mode. (User's Manual Chapter 4.3.6)
   The figure below shows an example of the mode settings.

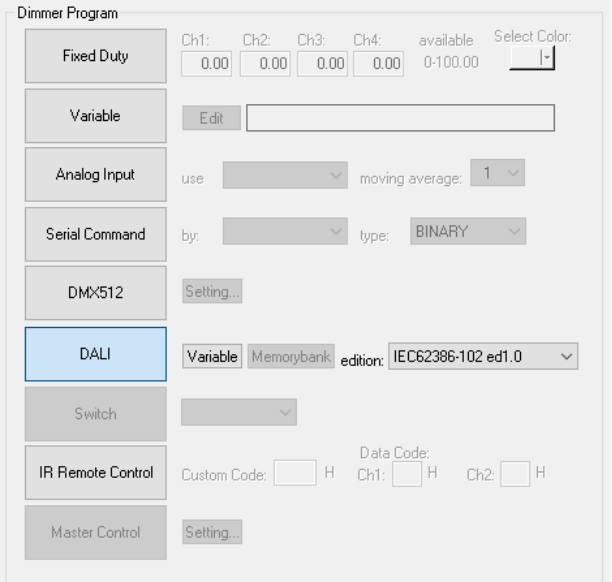

Figure 48 Dimmer Program

4. Generate and write. (User's Manual Chapter 4.4)

5.3.3 Set peration mode of EZ-0012

To perform DALI communication, it is necessary to change the state of the setting switches (SW1, SW2) from the time of initial setting.

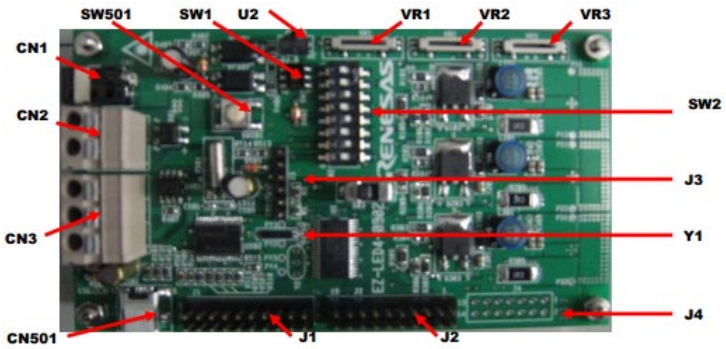

Figure 49 RL78 / I1A DC / DC LED Control Evaluation Board Components

1. Setting of setting switch
   Change SW1 to the “DALI” side in the figure below, and change SW2 to the settings in the table below.

Figure 50 Setting switch SW1

Table 66 Setting switch SW2

Number	Setting
1	ON
2	ON
3	OFF
4	OFF
5	OFF
6	OFF
7	ON
8	ON

For other details of EZ-0012, refer to the following User's Manual.
EZ-0012 RL78/I1A DC/DC LED Control Evaluation Board User's Manual(r01uh0363)
Note: When using EZ-0012 and DALI-2 EXPANSION BOARD or other boards, please refer to their respective user's manuals.

5.4 How to connect the board

The following figure shows how to connect the boards used in this demonstration project.

The power of X65N Cloud Kit + DALI-2 Option board is supplied by USB connection (ECN1), and the power of EZ-0012 is supplied by AC adapter power connection. The method of wiring connection is common regardless of the operation mode.

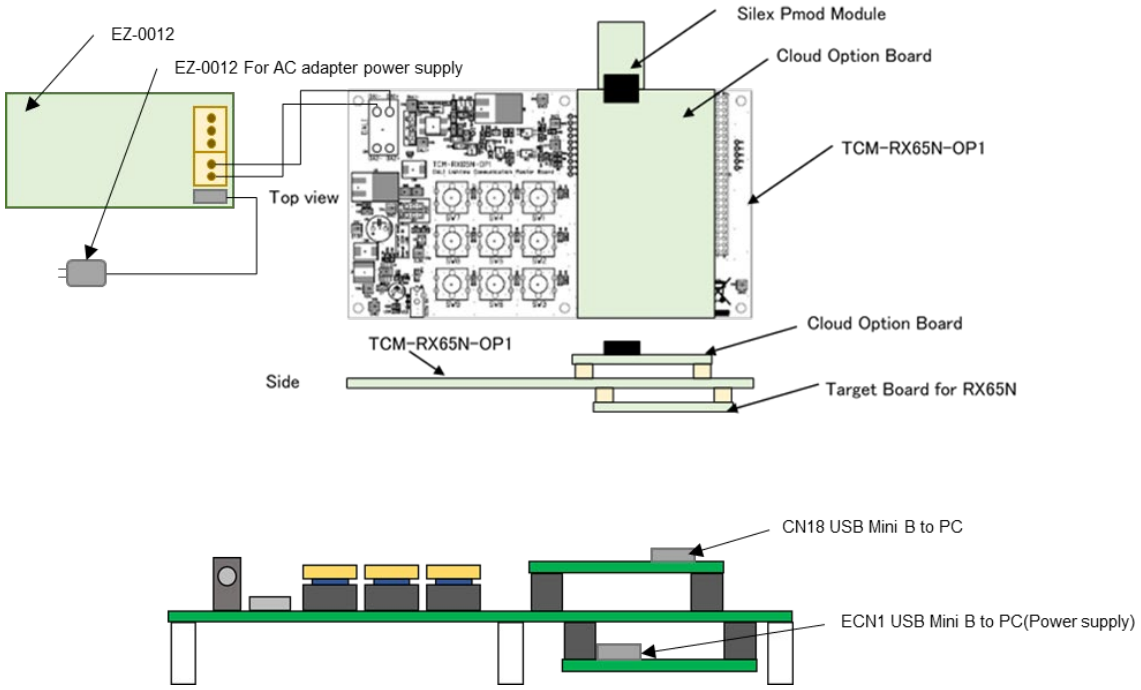

Figure 51 Overall hardware configuration diagram

Change the jumper connection of the DALI-2 Option boardto the following status in order to set the power supply via USB connection.

JP1 and JP2 are each jumper connected
JP3 and JP4 are each jumper connected to USB side

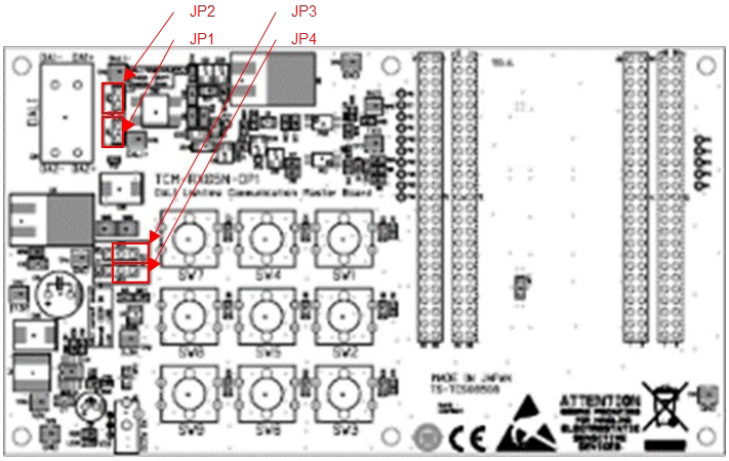

Figure 52 RX65N DALI-2 Option board jumper connection

5.5 Run the demo project

5.5.1 Set build options

1. Right-click on the project name and select menu -> [Property].
2. Click [C/C ++ Build] -> [Settings] -> [Toolchain] tab, and check the toolchain and version.

- Toolchain : GCC for Renesas RX
- Version : 8.3.0.201904

5.5.2 Build the project

1. Right-click the project in the Project Explorer and select [Build project].
2. The build starts and the console displays the status of the build. When the message “Build completed” is displayed, the build is complete.

5.5.3 Debug

1. Click the  button to download the program to the microcontroller.
2. Select [Run] -> [Debug Configuration⋯] to open the [Debug configuration] window.
3. In the [Debug configuration] window, expand the display of the [Renesas GDB Hardware Debugging] debug configuration and click on “ aws_demos HardwareDebug” configuration.
4. Switch to [Debugger] -> [Connection Settings] tab and check that the settings are as shown below.

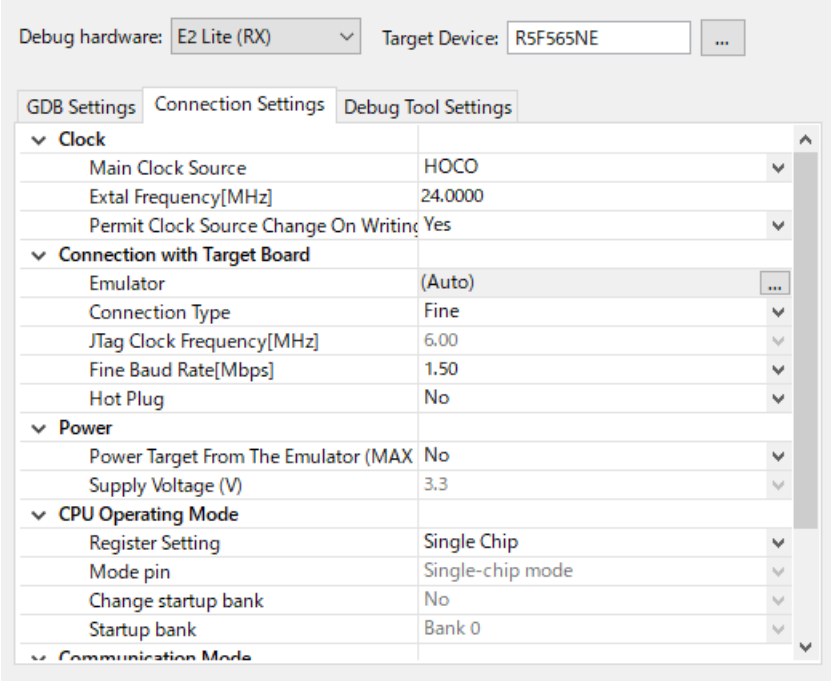

Figure 53 Debug screen settings pickup

5. Select [Start debugging] and the [Debug] view screen is displayed. Preparation for debugging is complete.

5.5.4 Run

1. Run the demo project by clicking the  button or pressing the "F8" key.

For details on how to operate the debug screen, refer to the following user's manual, section 5.4.
e2studio User's Manual: Getting Started Guide for V7.0(r20ut4374)

5.6 Connection service or operation device

5.6.1 DALI Master Controller GUI

The operation procedure of the demo project when operating with the DALI Master Controller GUI is shown below.

1. Install necessary software and connect the device referring to Chapter 5.1 to 5.4.
2. Refer to section 5.5, and run this demonstration project.
3. Install the DALI Master Controller GUI.
   Refer to the following user's manual for installation.

   DALI Master Controller GUI User's Manual(r20ut0715)

4. Start the DALI Master Controller GUI.
5. Click [Settings] -> [Serial⋯], select [USB Serial Device (COM x)] and click [OK].
   * The COM port number of the connected USB is displayed in COM x. The COM port number varies depending on the connected PC.
6. When the word “Broadcast” is displayed on the left side of the screen, the connection with the DALI Master Controller GUI is complete.

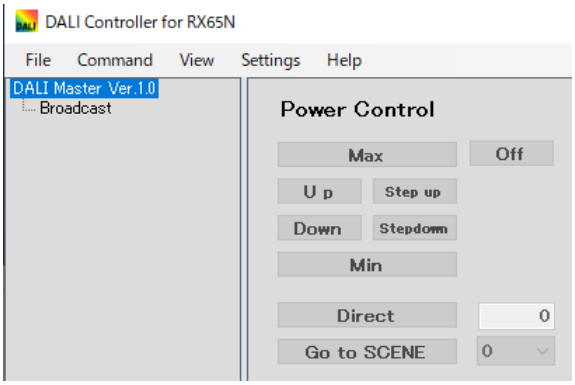

Figure 54 Excerpt from screen for DALI master controller GUI (1/2)

7. Recognize the connected Control Gear.
   - Select [Command] tab -> [Random Address Allocation⋯] and click [Start].

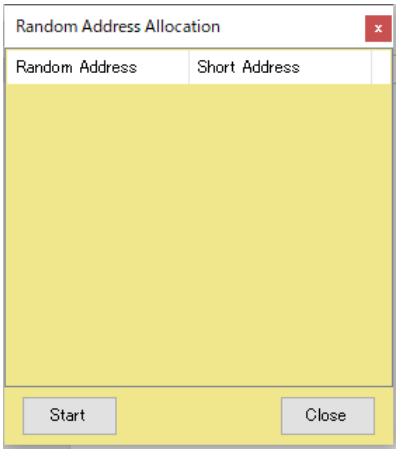

Figure 55 Screen of Random Address Allocation

   - When Address is displayed for the number of connected Control Gears, click [Close].

・When the screen display changes as shown below, the connection with Control Gear is completed.

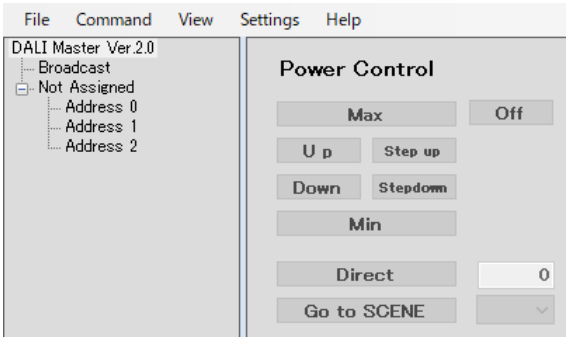

Figure 56 Excerpt from screen for DALI master controller GUI (2/2)

8. Operate the DALI Master Controller GUI.
   ex.1: Turn on all connected lamps with MAX light intensity -> Select Broadcast and press the MAX button
   ex.2: Turn off all connected lamps -> Select Broadcast and press the OFF button
   ex.3: Turn on the connected lamps individually with MIN light intensity -> Select Address X and press the MIN button
   ex.4: Increase the brightness of each connected lamp gradually -> Press the UP button repeatedly from the state of ex.3

For other detailed operation methods, refer to the following user's manual.
DALI Master Controller GUI User's Manual(r20ut0715)

5.6.2 AWS Web application

The operation procedure of the demo project when operating with the AWS Web application is shown below.

1. Install the necessary software and connect the device, referring to Chapters 5
2. Save the AWS web application zip file to a location of your choice.
3. Register and set up to AWS.
   To run AWS web applications and demo projects, you need an AWS account, an IAM user with access to the AWS IoT and Amazon Free-RTOS cloud services.
   In addition, it is necessary to register the information of RX65N Cloud kit + DALI-2 Option board in AWS IoT.

   Refer to the following URL for how to set your AWS account and access permissions.
   Create a user account.
   https://docs.aws.amazon.com/freertos/latest/userguide/freertos-account-and-permissions.html.

   Next, register the board with AWS IoT by referring to the following URL.
   Follow the steps to connect the board manually.
   https://docs.aws.amazon.com/freertos/latest/userguide/get-started-freertos-thing.html.

   Set the source code by referring to the URL below and let the demo project communicate with AWS.
   Describes how to import certificates and configure Wi-Fi.
   https://docs.aws.amazon.com/freertos/latest/userguide/freertos-configure.html.

   The following is an excerpt from the source code change for how to import the certificate and how to set the Wi-Fi.

4. Import certificate and private key （1/4）

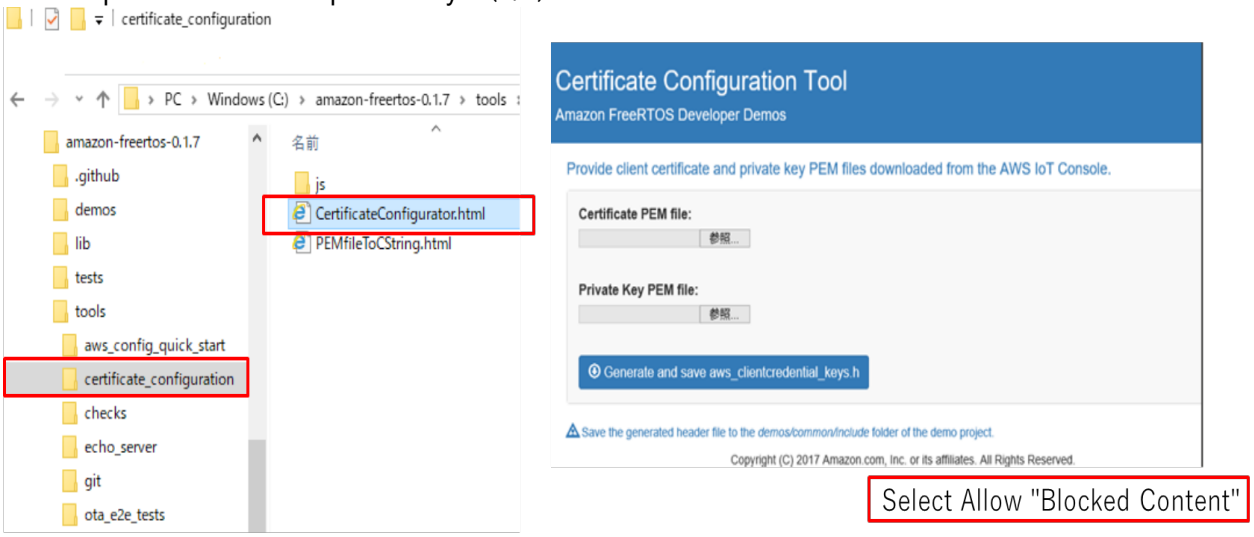

Figure 57 Import certificate and private key (1/4)

-Import device (thing) certificate and private key to source code-Double-click
$ {base_folder} ¥ tools ¥ certificate_configuration ¥ CertificateConfigurator.html to start
(Permit if message indicating that ActiveX is restricted is displayed please)

5. Import certificate and private key （2/4）

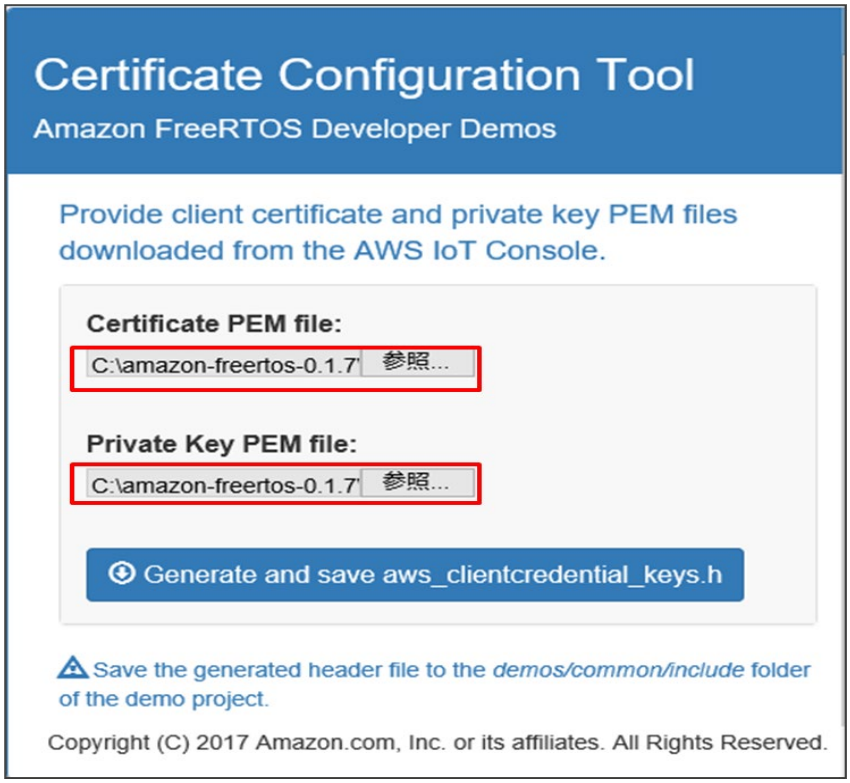

Figure 58 Import certificate and private key (2/4)

・Incorporation of device (thing) certificate and private key into source code
・The certificate of the object and the private key (* 2) generated in
“Register the device in AWS IoT” are each file and specified in CertificateConfigurator.html
・Xxxxxxxxxx-certificate.pem.crt --- Certificate
・Xxxxxxxxxx-private.pem.key --- Private key

6. Import certificate and private key （3/4）

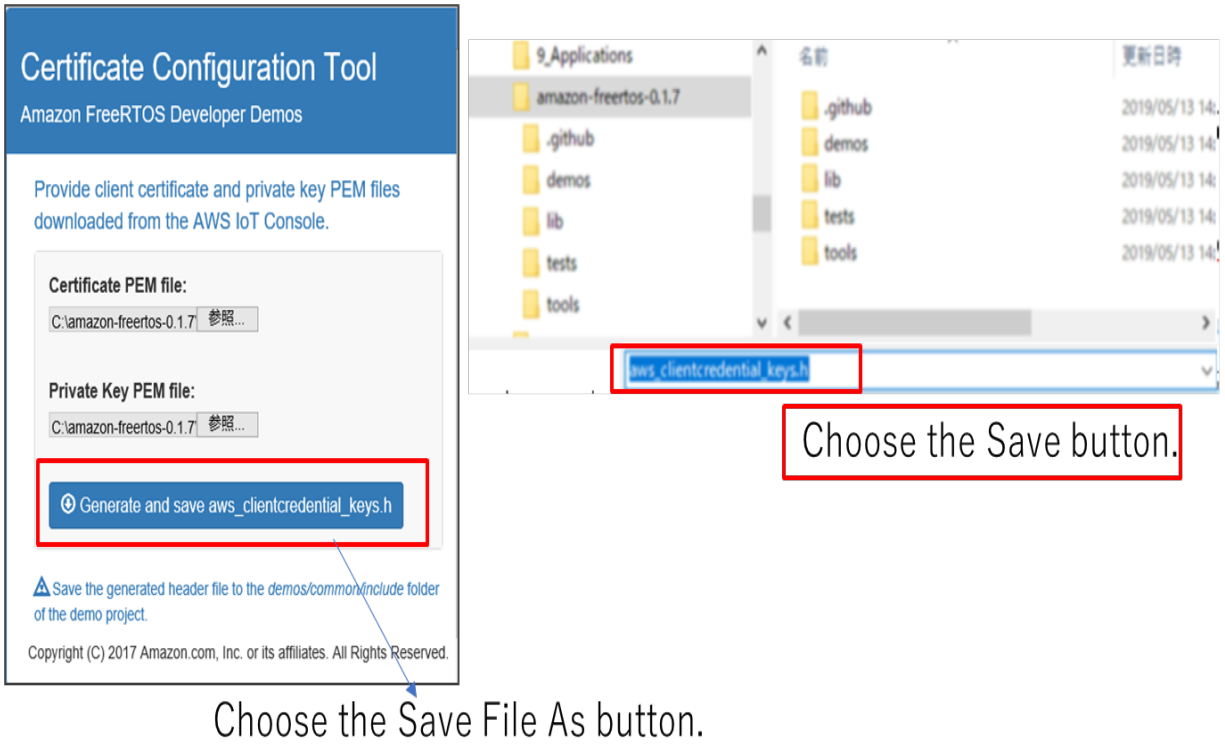

Figure 59 Import certificate and private key (3/4)

- Click the Generate and save aws_clientcredential_keys.h button
- Download aws_clientcredential_keys.h to any location

7. Import certificate and private key （4/4）

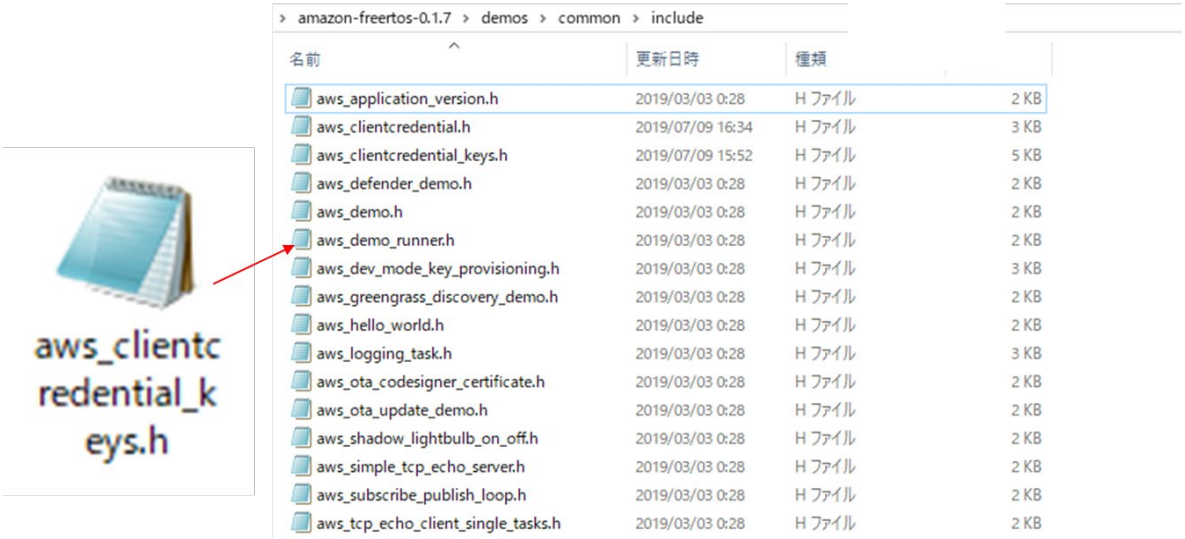

Figure 60 Import certificate and private key (4/4)

- Open $ {base_folder} ¥ demos ¥ common ¥ include ¥ aws_clientcredential_keys.h with Explorer
- Drag and drop the generated “aws_clientcredential_keys.h” to Explorer
- The message "Do you want to replace?" is displayed. Select "Yes".

8. Macro settings

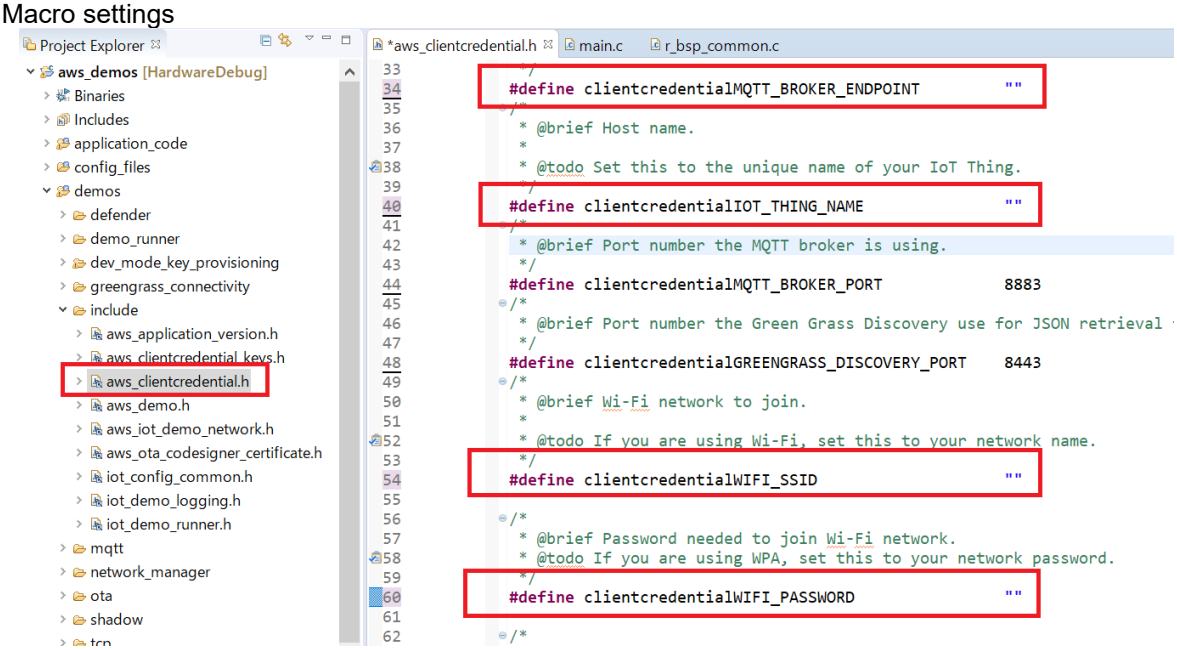

Figure 61 Macro settings

・Project Explorer -> aws_demos -> application_code -> common_demos -> include -> aws_clientcredential.h
・Set the following two macros (input to the contents of "")
  ・clientcredentialMQTT_BROKER_ENDPOINT
   -> Confirmed in "Check AWS IoT Endpoint" Endpoint name (*3)
  ・clientcredentialIOT_THING_NAME
    The name of the thing registered in “Register the device with AWS IoT” (*1)

In the case of WiFi, set the following two types
  (SSID and PASSWORD are already entered in the exercise program)
  ・clientcredentialWIFI_SSID -> SSID with access point to connect
  ・clientcredentialWIFI_PASSWORD-> Password of access point to connect

9. Attach the policy to IAM.

Attach the policy to IAM by referring to the following URL.

https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_manage-attach-detach.html

＜Policy name＞

- AmazonS3FullAccess
- CloudWatchReadOnlyAccess
- AWSIoTFullAccess
- AWSLambdaFullAccess
- AutoScalingFullAccess
- AWSElasticBeanstalkFullAccess
- AmazonAPIGatewayAdministrator
- AmazonCognitoPowerUser
- IAMFullAccess

Note 1: Can register maximum 10 policies per an account.If another policy is already registered and the number of registered policies is 11 or more, please make a group and attach policies via the group.
Note 2: Before attaching policies, please pay attention and understand about contents of permission by each policy.

10. Register user settings in Cognito.
· Log in to the AWS Management Console.
· Choose Cognito selection.

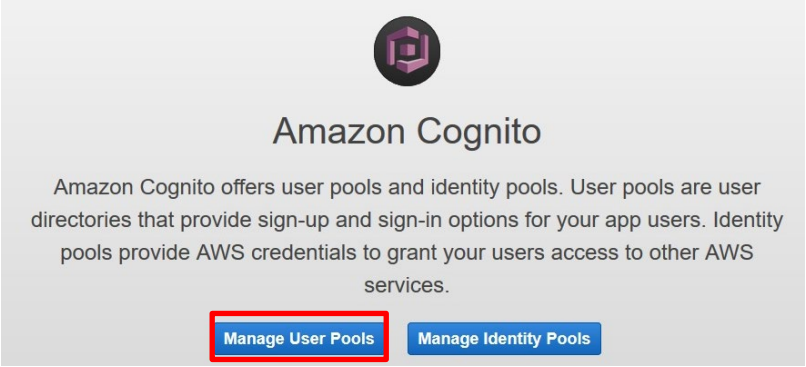

Figure 62 Cognito (1/10)

· Select [Manage user Pools].

Figure 63 Cognito (2/10)

· Select [Create a user pool].

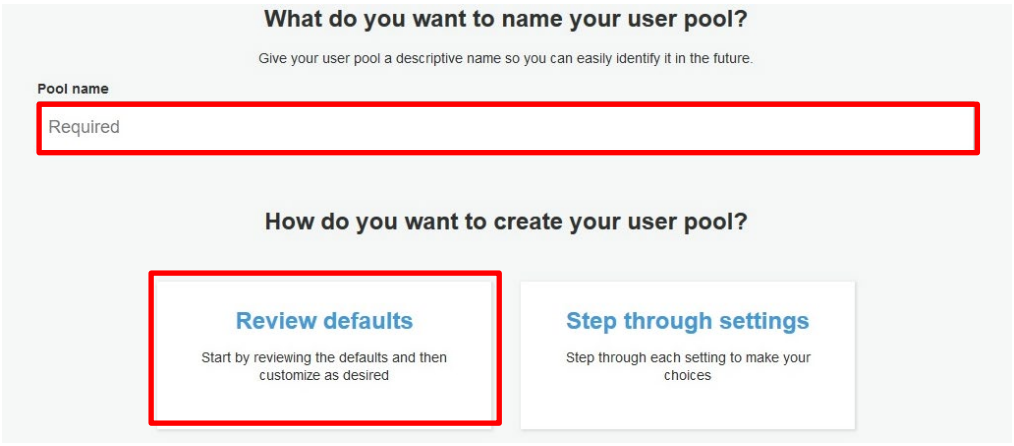

Figure 64 Cognito (3/10)

· Enter [daliweb_userpool] for the Pool name.
· Select [Review defaults].

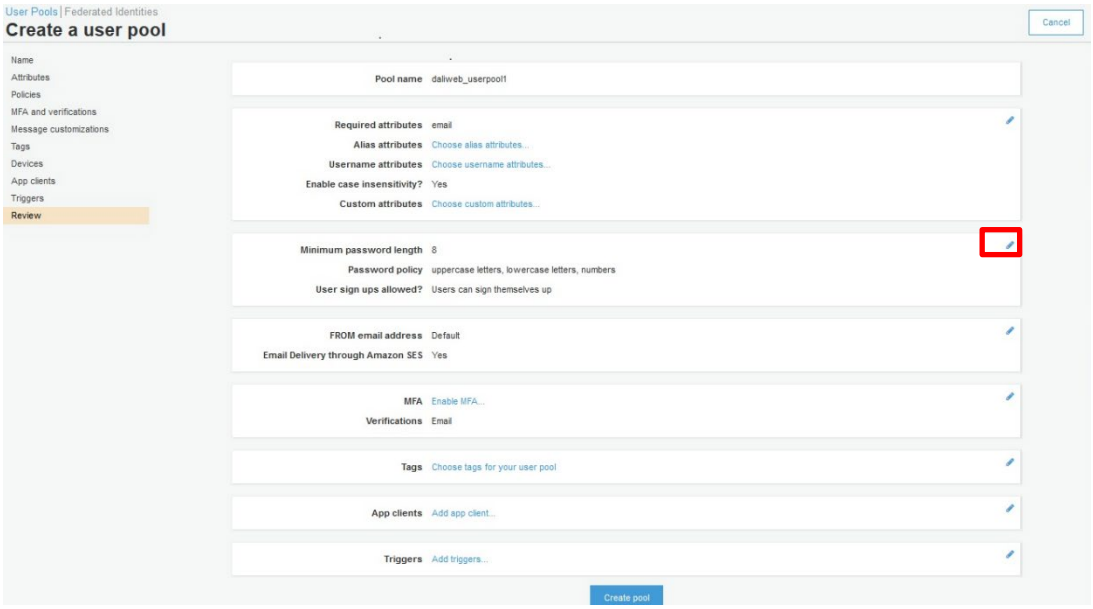

Figure 65 Cognito (4/10)

・Select the location shown in the figure.

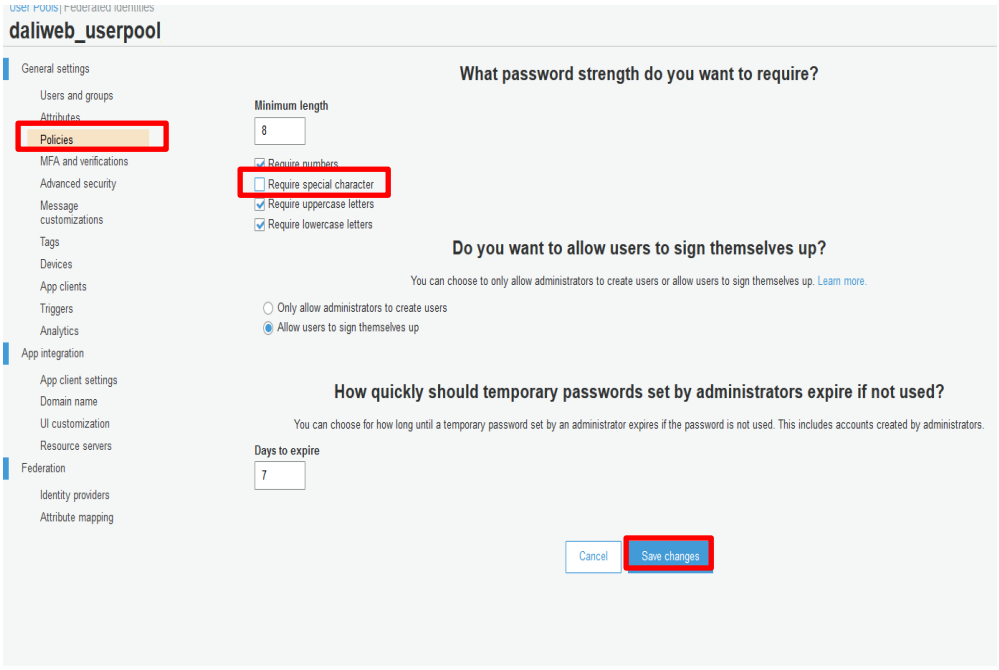

Figure 66 Cognito (5/10)

・Select [Policy] in in the figure.
・Uncheck "Display special characters" in the figure.
・Select Save Changes on the diagram.

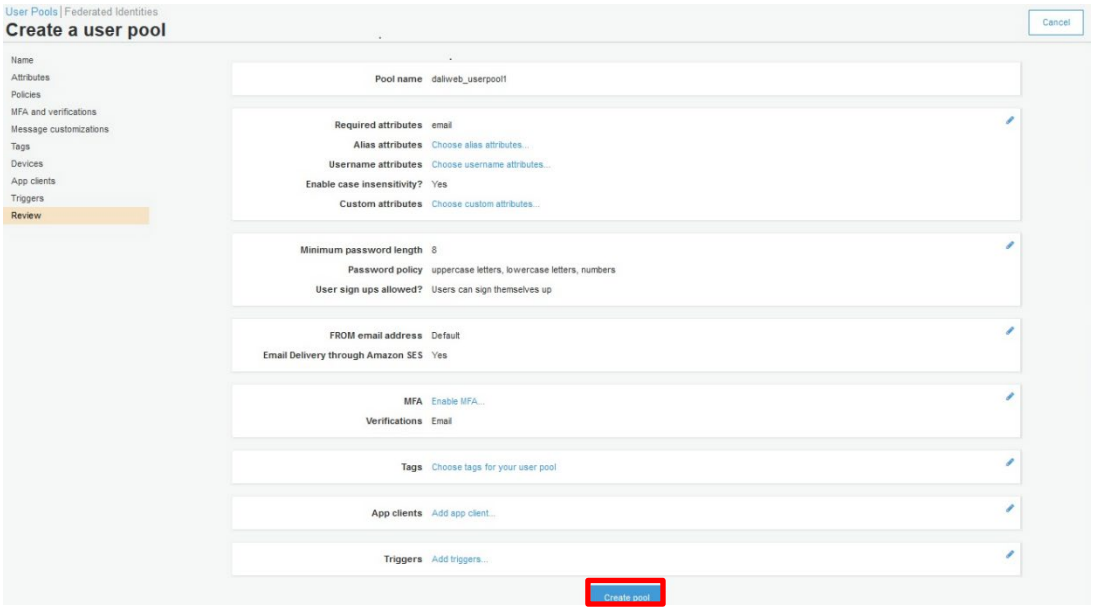

Figure 67 Cognito (6/10)

・Select [Create pool] in in the figure.

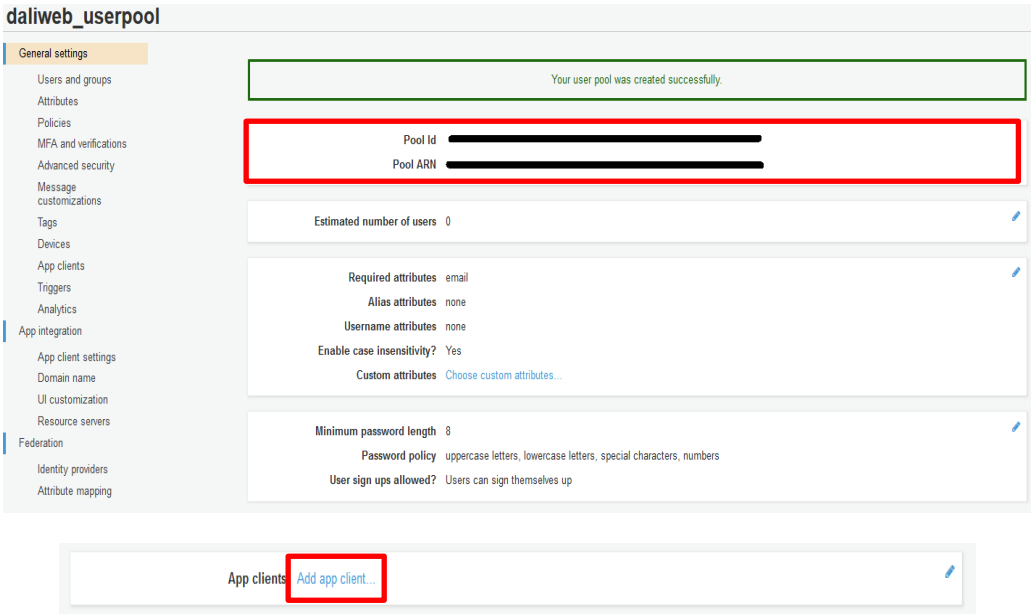

Figure 68 Cognito (7/10)

• Make a note of the Pool ID[UsetPoolId] and Pool ARN in the figure.(*1)
• Move the screen down and select "Add app Client".

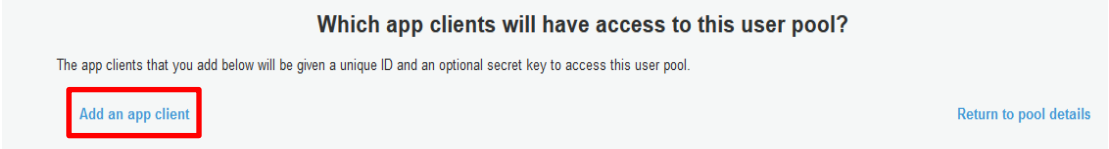

Figure 69 Cognito (8/10)

• After switching the screen, select "Add App Client".

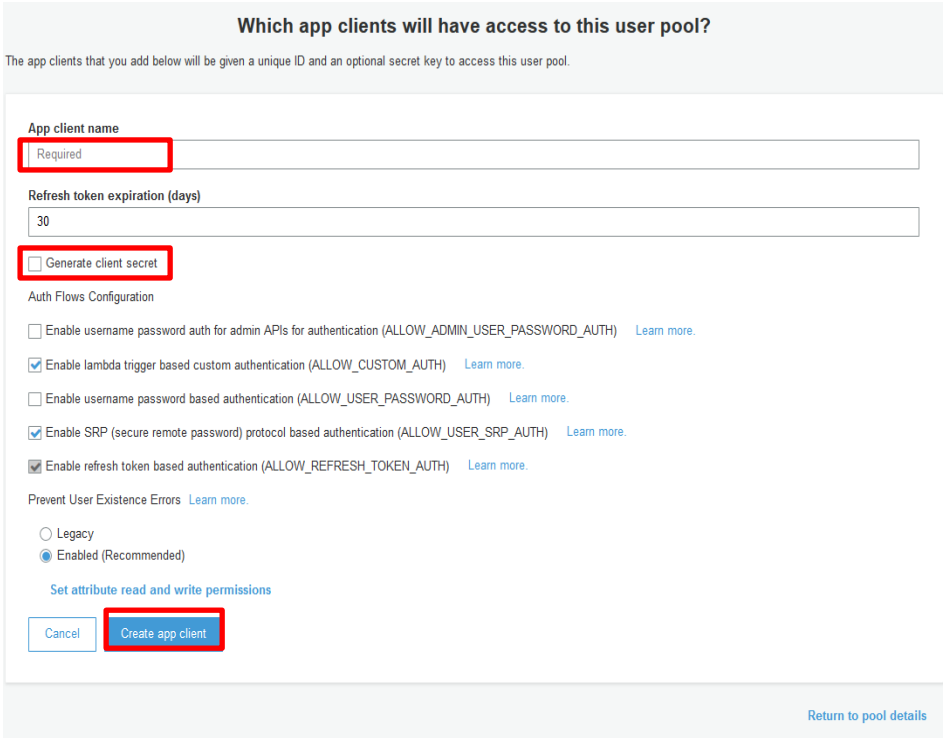

Figure 70 Cognito (9/10)

• Set [daliweb_aplcl] to the app client name in the figure.
• Uncheck Generate Client Secret in the figure.
・Select [Create app client] in in the figure.

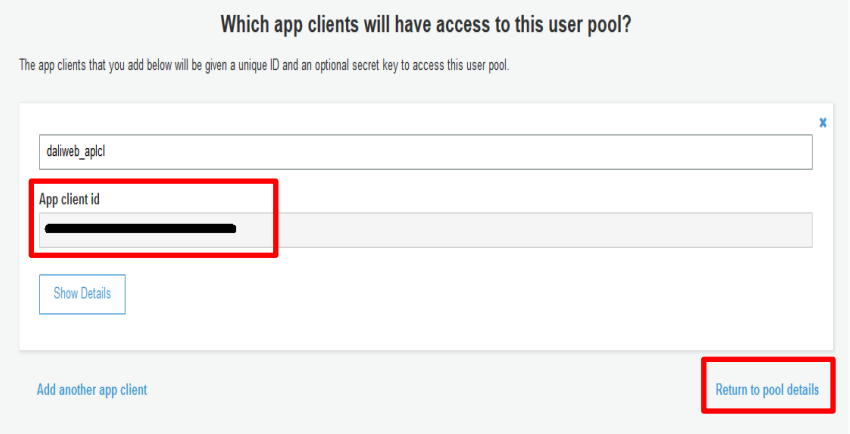

Figure 71 Cognito (10/10)

• Make a note of the App Client ID in the diagram. (ClientId) (*2).
• Select Return to Pool Details in the figure..

```
'use strict'
var poolData = {
    UserPoolId: '[redacted]',
    ClientId: '[redacted]'
}
```

Figure 72 Cognito setting

• Unzip the aws setting file (r_aws_setting.zip).
• Set the following in cognito-auth.js.
  •UserPoolId
  •ClientId

•The user pull screen is finished.

11. From the AWS Management Console Services, select S3.

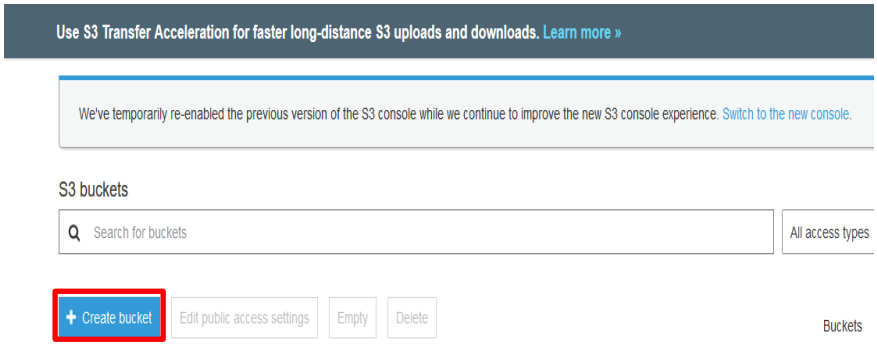

Figure 73 Amazon S3 (1/9)

• After moving the screen, select [Create bucket] in the figure.

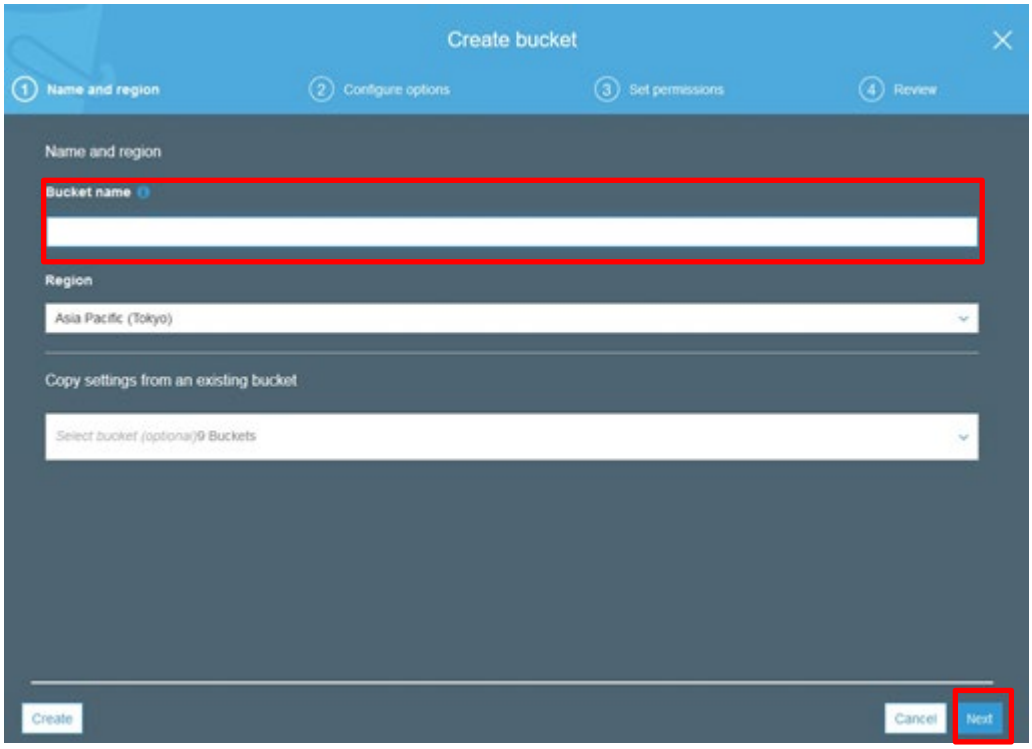

Figure 74 Amazon S3 (2/9)

• Enter a unique name (*1) in aws s3. The name must be unique among other AWS users.
(*1) Bucket name precautions
•Bucket name is limited to 3 to 63 characters.
•Uppercase letters and underscores cannot be included in the bucket name.
•Please refer to the following AWS official guide for details of naming convention.
• Select [Next] in the figure.

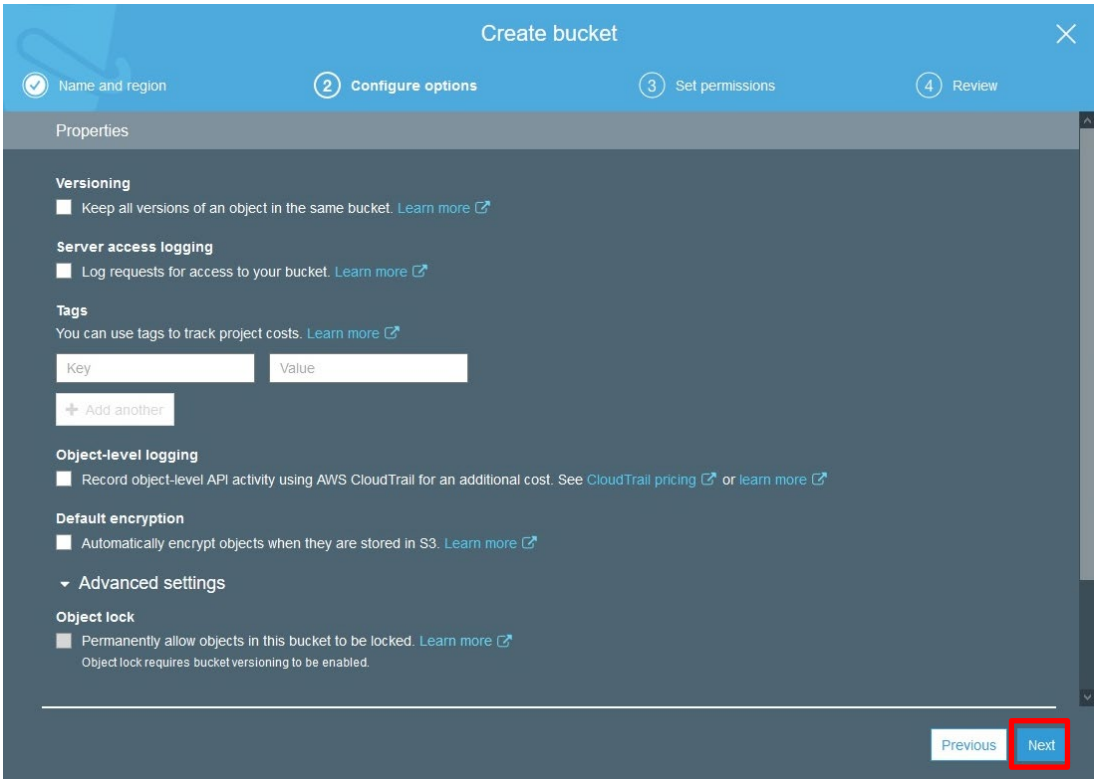

Figure 75 Amazon S3 (3/9)

• Select [Next] in the figure.

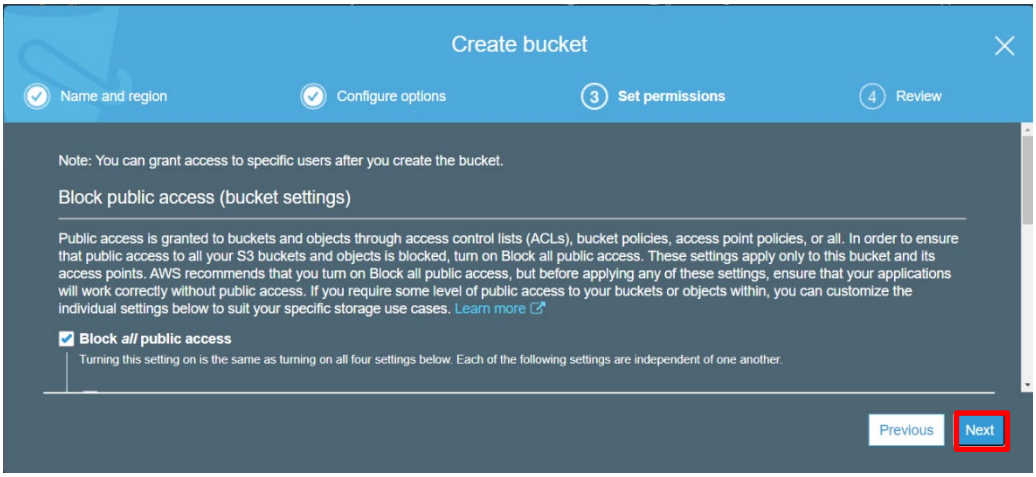

Figure 76 Amazon S3 (4/9)

• Select Next in the figure.

*The public settings here are just examples for testing.
Customers should review this setting during actual operation.

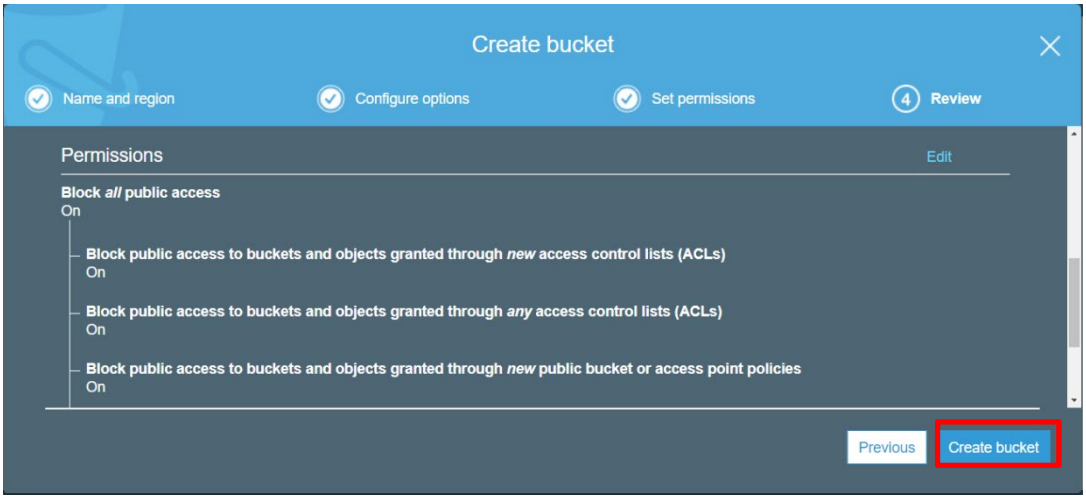

Figure 77 Amazon S3 (5/9)

• Select [Create Bucket] in the figure.

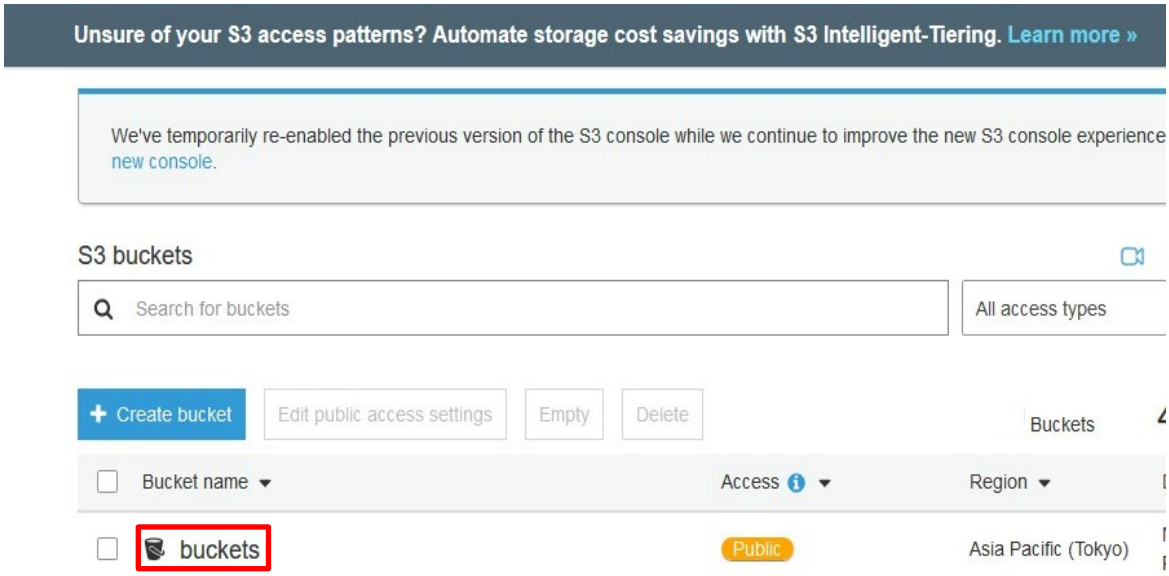

Figure 78 Amazon S3 (6/9)

• Select [buckets] you created n the figure.

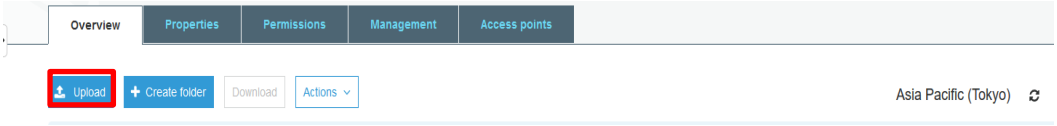

Figure 79 Amazon S3 (7/9)

• Select [Upload] you created n the figure.

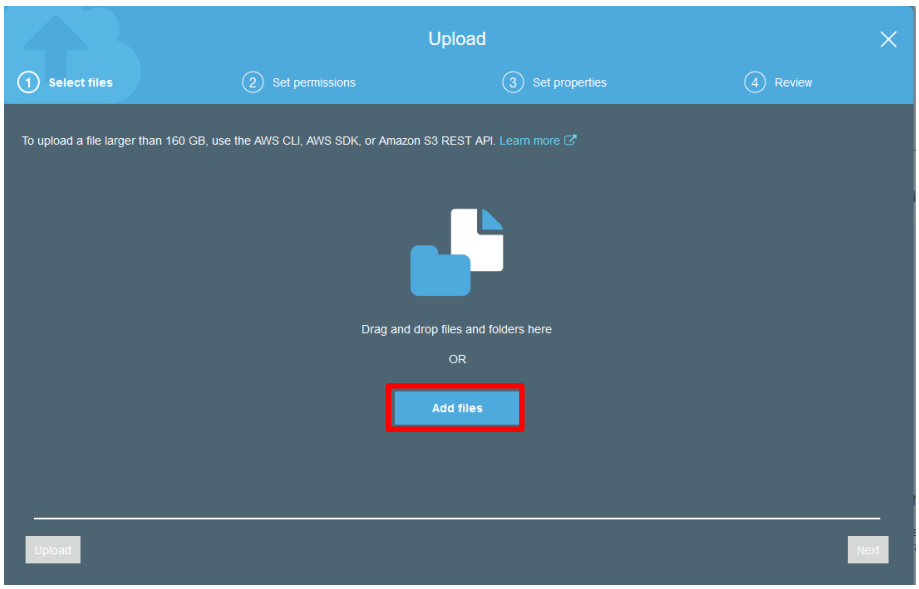

Figure 80 Amazon S3 (8/9)

• Select [Add files] you created n the figure.

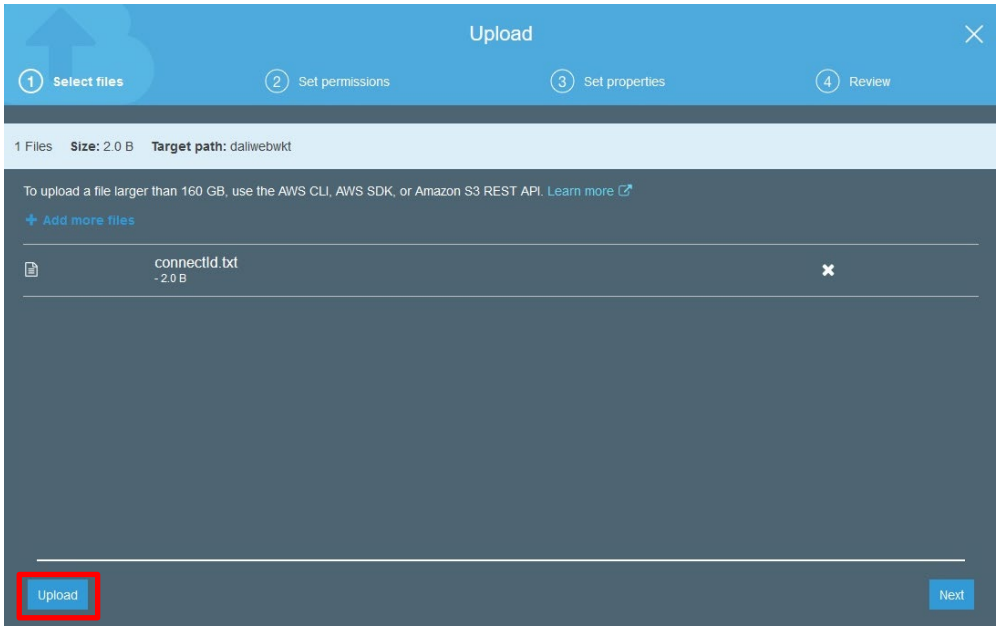

Figure 81 Amazon S3 (9/9)

• Select [s3/connectId.txt] of the aws setting file (aws_setting.zip).
• Select "Upload" in the figure.
• The setting of S3 is completed.

12. From Services in AWS Management Console, choose lambda.

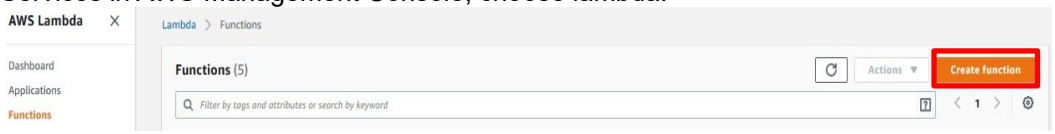

Figure 82 lambda (1/6)

• Create the "daliweb_onconnect" function. Click the [Create Function] button in the figure.

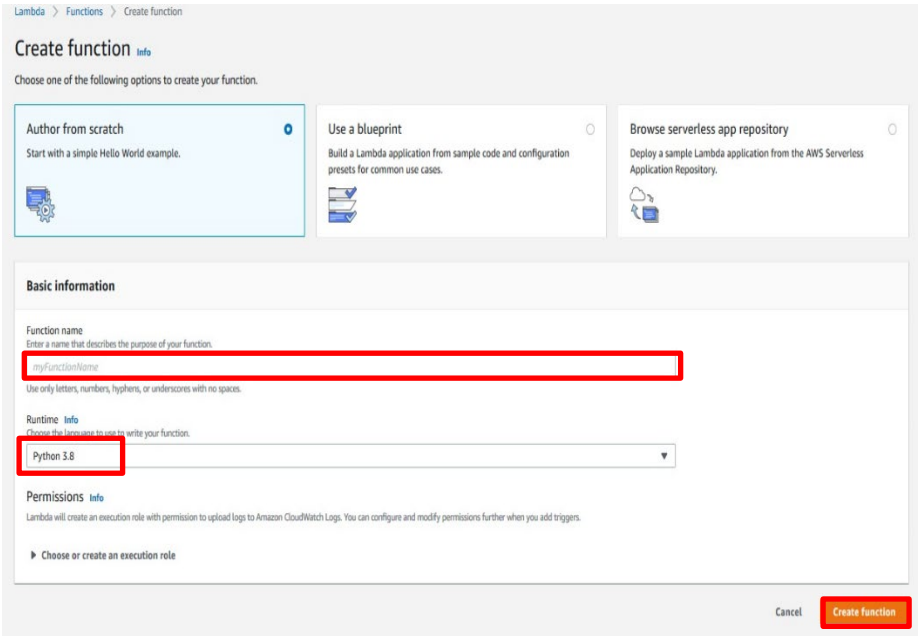

Figure 83 lambda (2/6)

• Set [daliweb_onconnect] to the function name in the figure.
• Select Python 3.8. for the diagram runtime.
• Select [Create Function] in the figure.

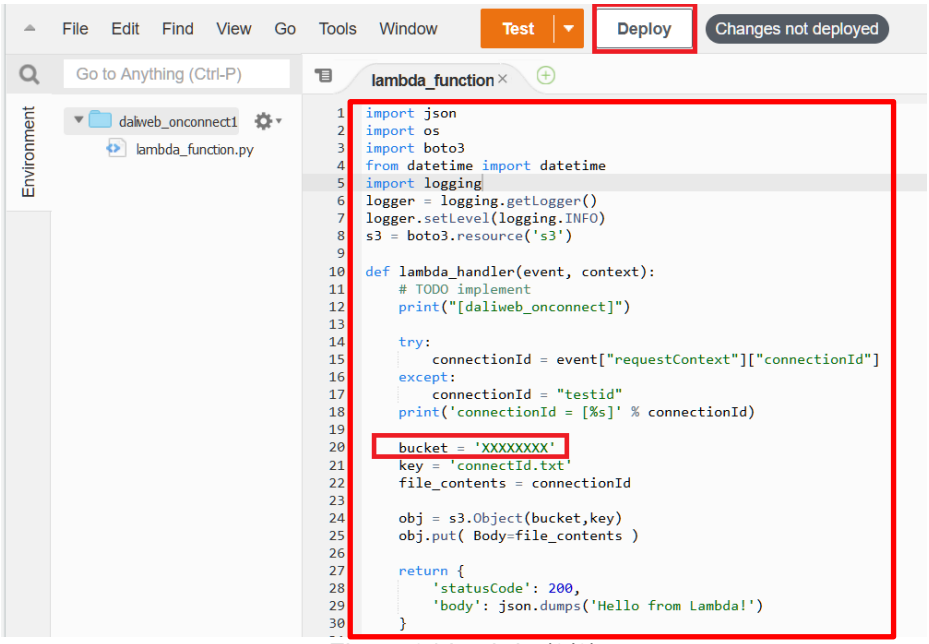

Figure 84 lambda (3/6)

• Open "lambda/daliweb_onconnect/lambda_function.py" in the aws settings file (aws_setting.zip) with an editor.
After that, copy the content and paste it on the function code screen with "CTRL + V".
• Set the "bucket name" created in Amazon S3 in the black line part (bucket ='XXXXXXXX') of the figure.
• Select [Deploy] in the figure.

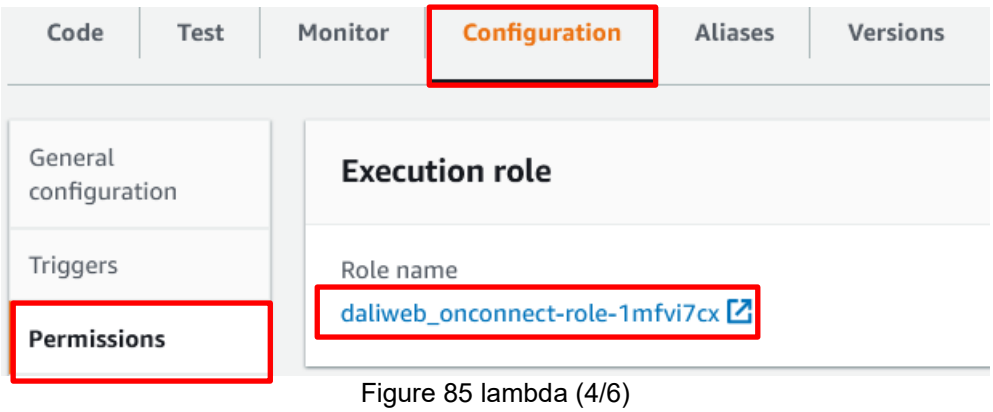

Figure 85 lambda (4/6)

• Select [Permissions] on Configuration tab in the figure.
• Select [Role Name] in the figure.

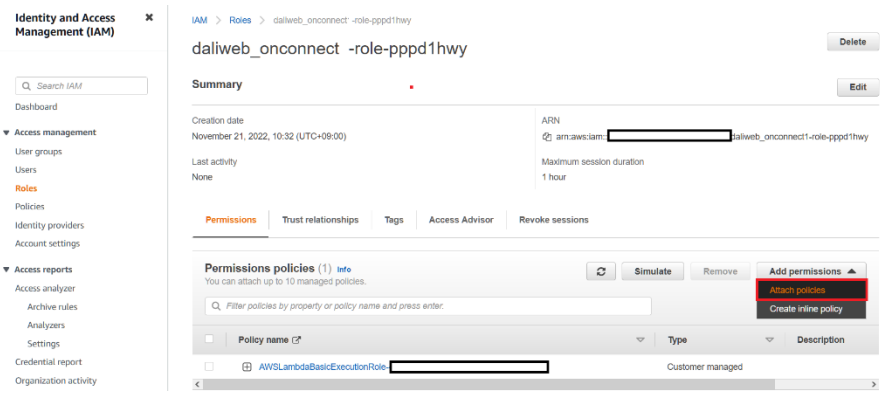

Figure 86 lambda (5/6)

• Select [Attach Policies] in the figure.

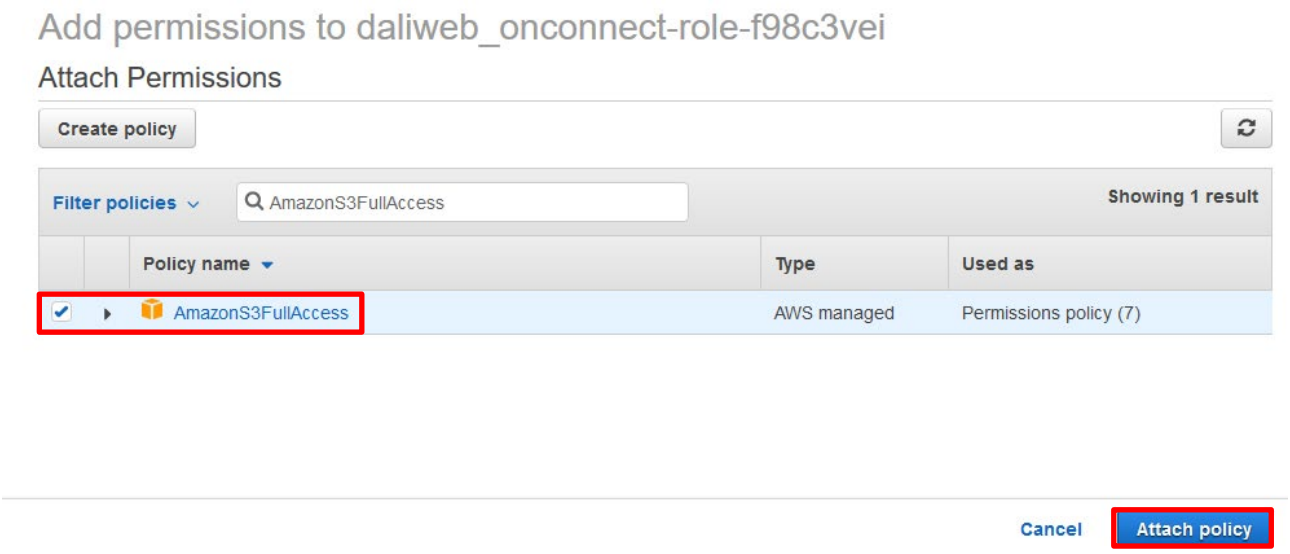

Figure 87 lambda (6/6)

• Select “AmazonS3FullAccess ” in the figure and select “Attach policy”.
• Verify that "Administrator Access" in the figure is added to the policy.
• Similar to “daliweb_onconnect” in this chapter, create a function with the function name “daliweb_send_Iot2web”.

• The aws setting file (aws_setting.zip) uses “lambda/daliweb_send_Iot2web/lambda_function.py”.
• For daliweb_send_Iot2web, please add the following as the policy settings to add.
・AmazonS3ReadOnlyAccess
・AmazonAPIGatewayInvokeFullAccess

12. From Services in the AWS Management Console, choose API Gateway.

Figure 88 API Gateway (1/15)

Figure 89　API Gateway　(2/15)

• Select [Create API] in the figure.

Figure 90　API Gateway　(3/15)

• Select [Build] for the WebSocket API in the figure.

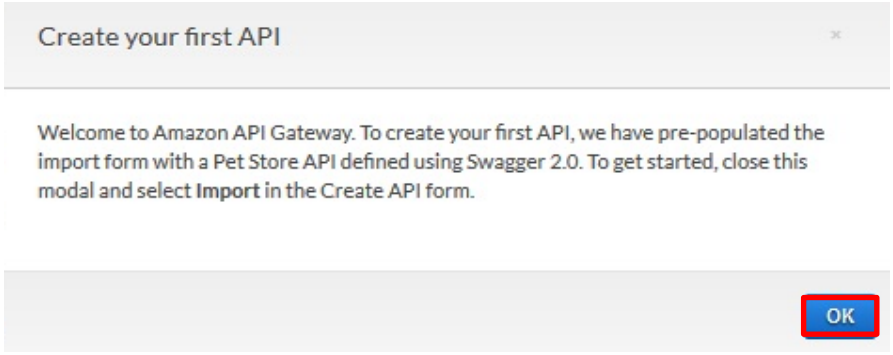

Figure 91　API Gateway　(4/15)

• Select “OK” in the figure.

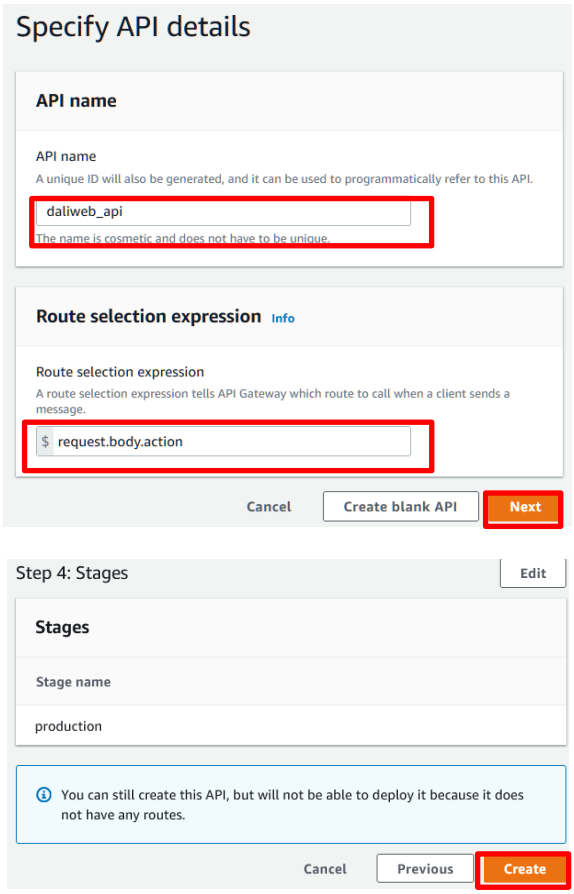

Figure 92 API Gateway (5/15)

•
• Set "daliweb_api" to the API name.
• Set "$request.body.action" in route selection formula.
• Click [Create blank API].
• Click [Create].

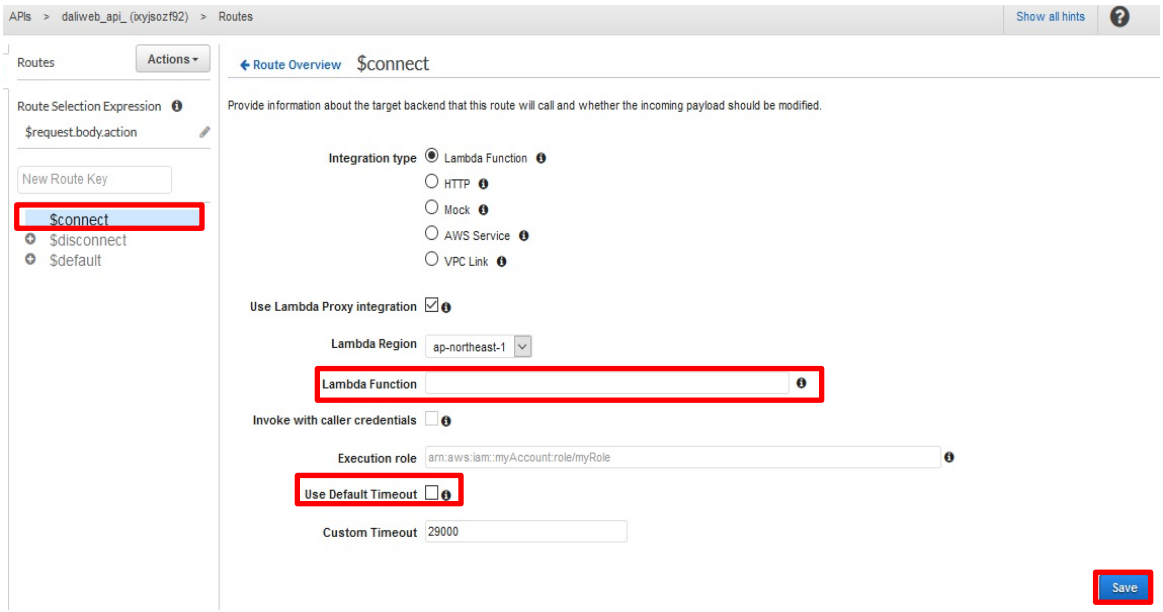

Figure 93 API Gateway (6/15)

•Select "connect" in the figure.
•Set "daliweb_onconnect" to your Lambda function in the figure.
• Uncheck the "Use Default Timeout" specification in the figure.

• Select [Save] in the figure.

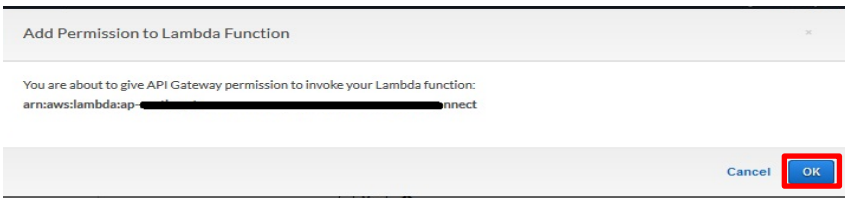

Figure 94 API Gateway (7/15)

• [Add permission to Lambda Function] is displayed,.
• Select [OK] in the figure.

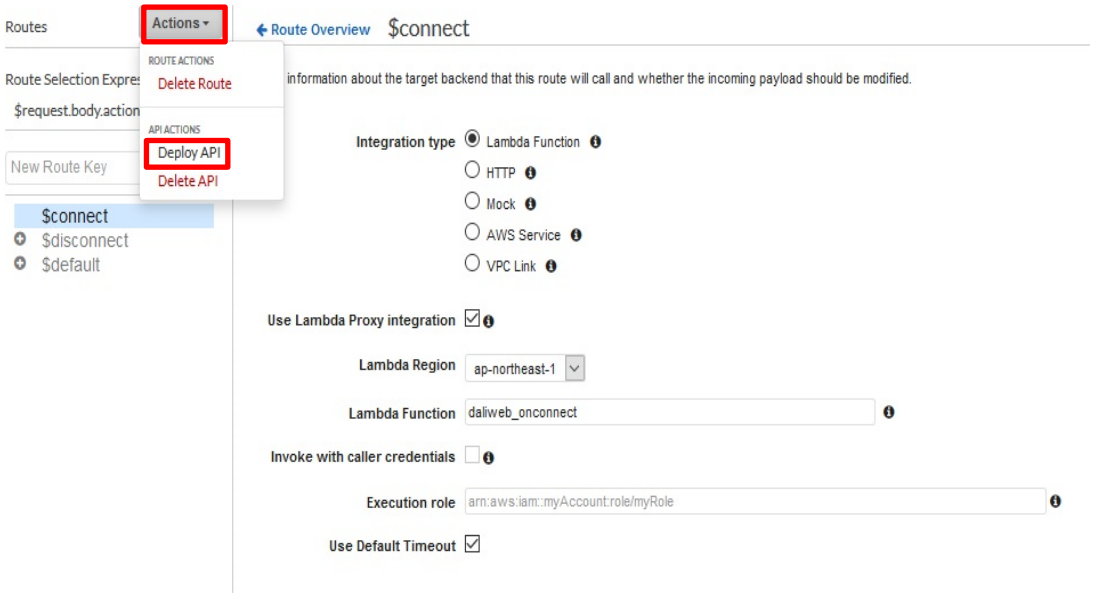

Figure 95 API Gateway (8/15)

• Select [Deploy API] from [Action] in the figure.

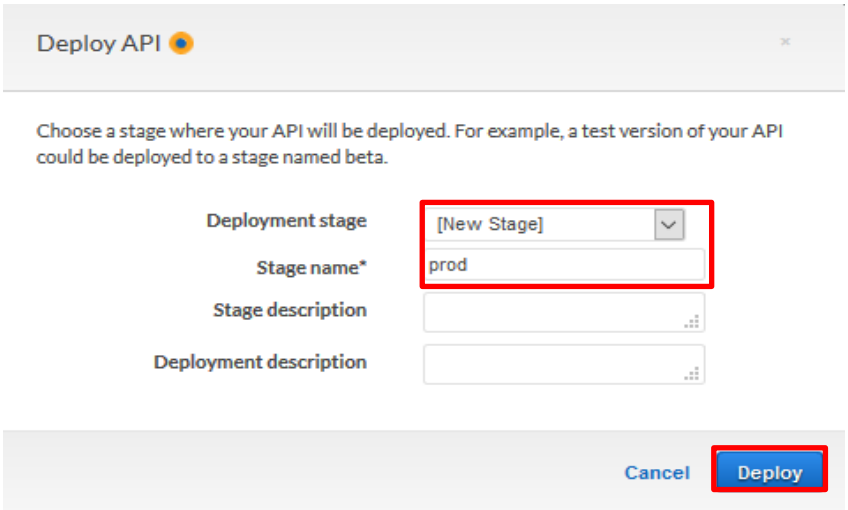

Figure 96 API Gateway (9/15)

• Select [New Stage] for the Deployment stage in the figure.
• Enter [prod] as the stage name in the figure.
• Select [Deploy] in the figure.

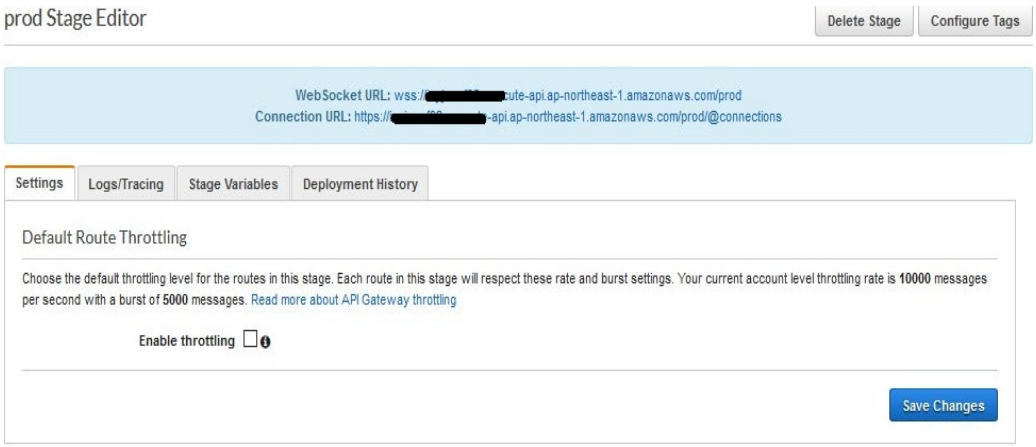

Figure 97 API Gateway (10/15)

• Websocket URL is issued.

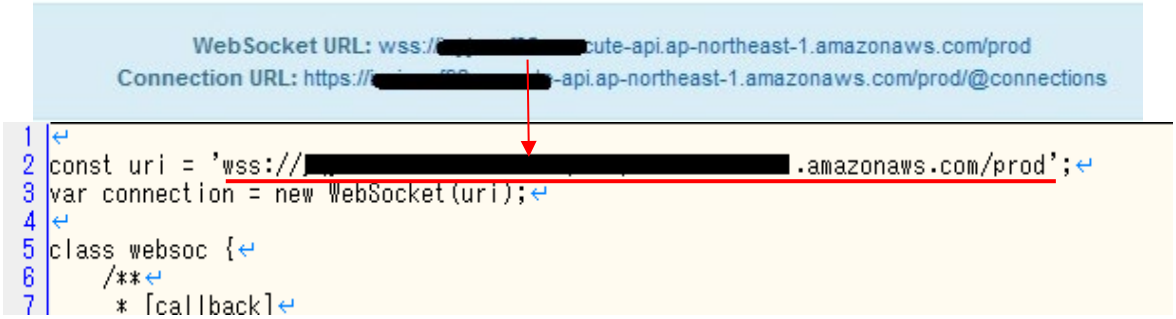

Figure 98 API Gateway (11/15)

• Set the published WebSocket URL to websoc.js.
• Set websoc.js to [const url] of [daliweb/html/websoc.js] of AWS-WEB file (aws-web.zip).

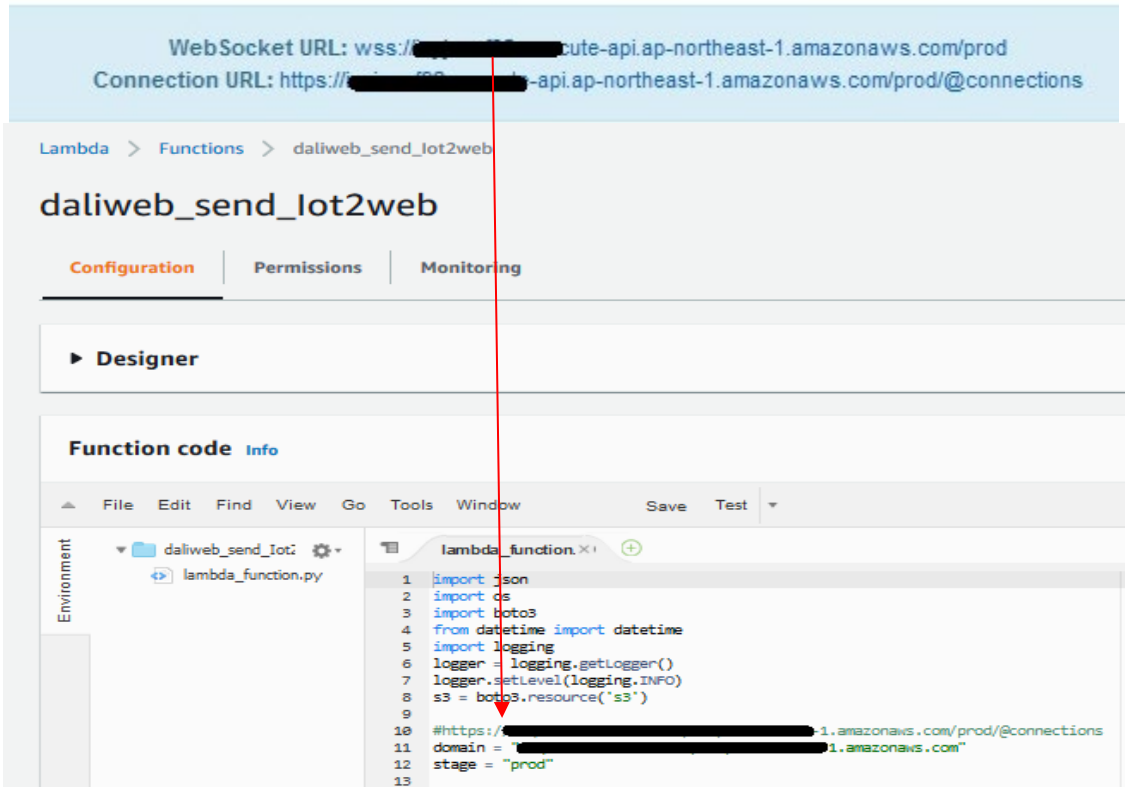

Figure 99 API Gateway (12/15)

• Set the issued connection URL in the lambda function (daliweb_send_Iot2web).

(Lambda->Functions->daliweb_send_Iot2web screen transition)

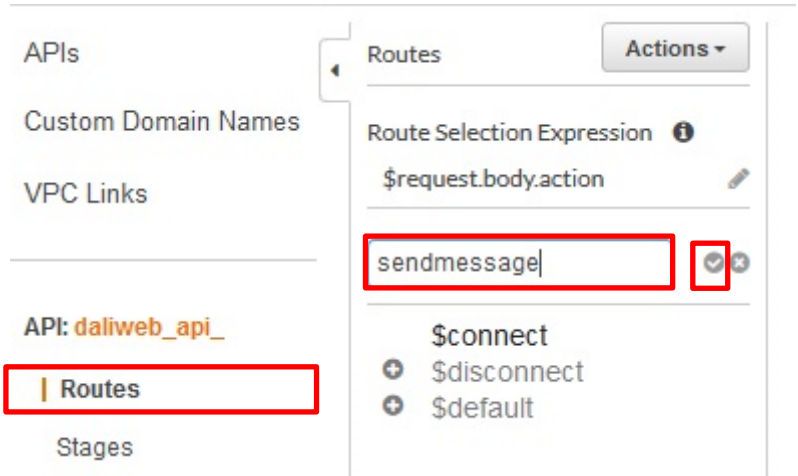

Figure 100 API Gateway (13/15)

•Select [Routes] in the figure.
• Enter [send message] in the [New root key] in the figure.
• Select the check mark in the figure..

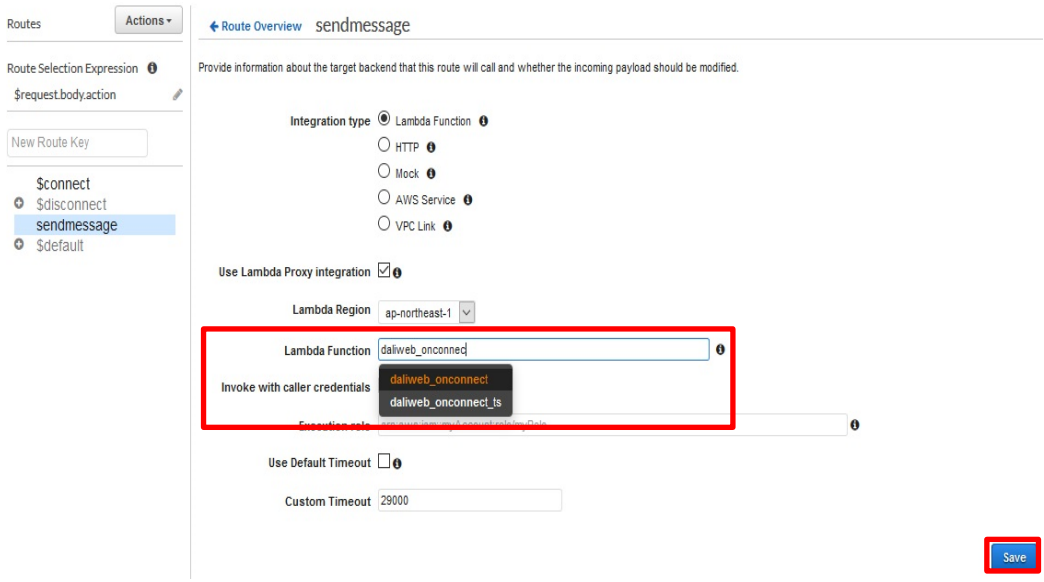

Figure 101 API Gateway (14/15)

• Enter "daliweb" in the figure.
• Select "daliweb_send_Iot2web" in the figure.
• Select “Save in the figure.

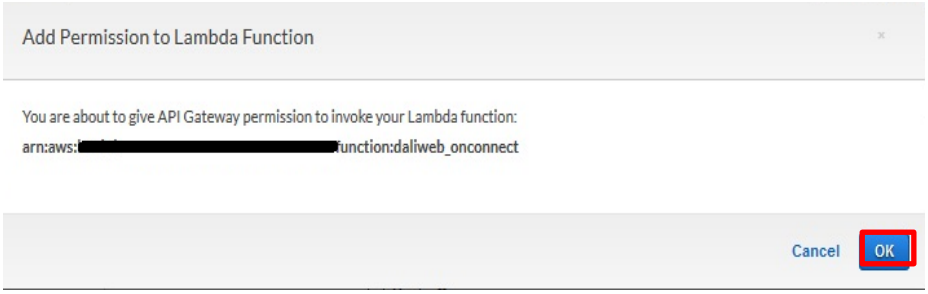

Figure 102 API Gateway (15/15)

• Select “OK” in the figure.
• API Gateway is finished.

14. From Services on the AWS Management Console, select IoT Core.

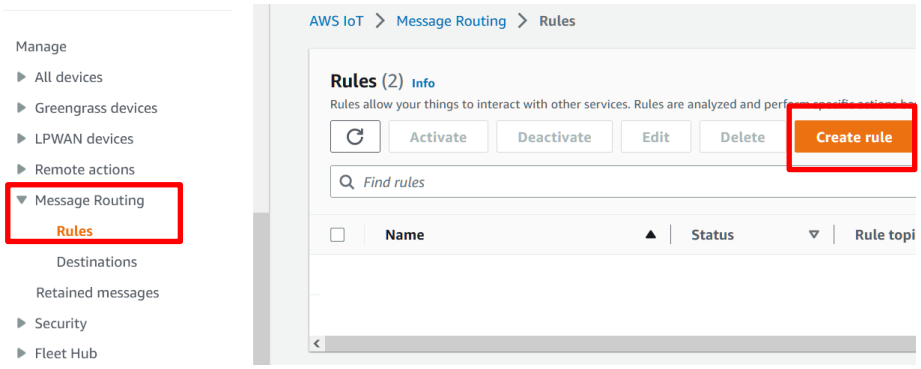

Figure 103 Aws IoT (1/7)

• Select [ACT] and [Rules] in the figure.

• Select [Create] in the figure.

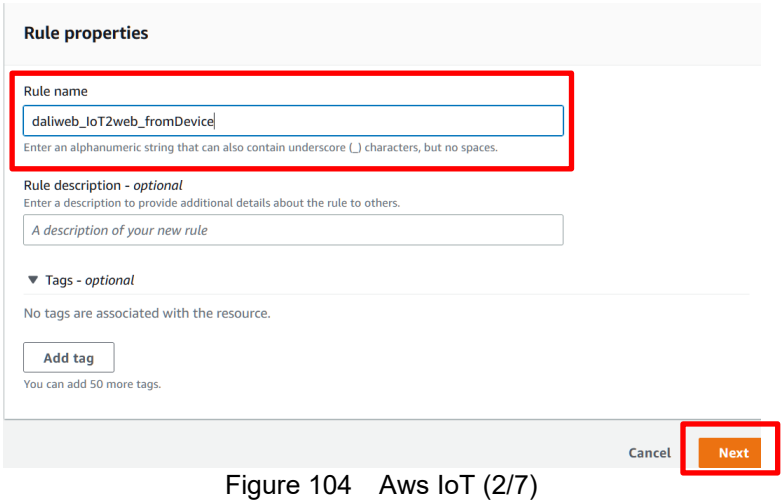

Figure 104 Aws IoT (2/7)

• Set the name to "daliweb_IoT2web_fromDevice" then select “next”.

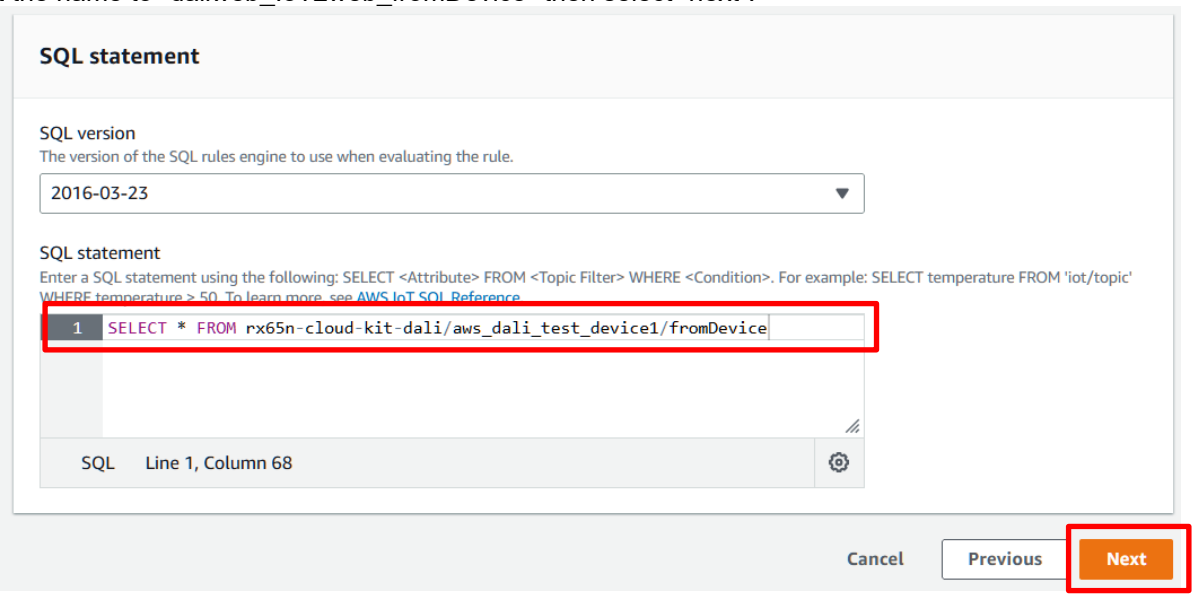

Figure 105 Aws IoT (3/7)

•Set ‘rx65n-cloud-kit-dali/thing name/fromDevice’ in the rule query statement.
In this example, the thing name is ‘aws_dali_test_device1’.
• Select [Next] in the figure.

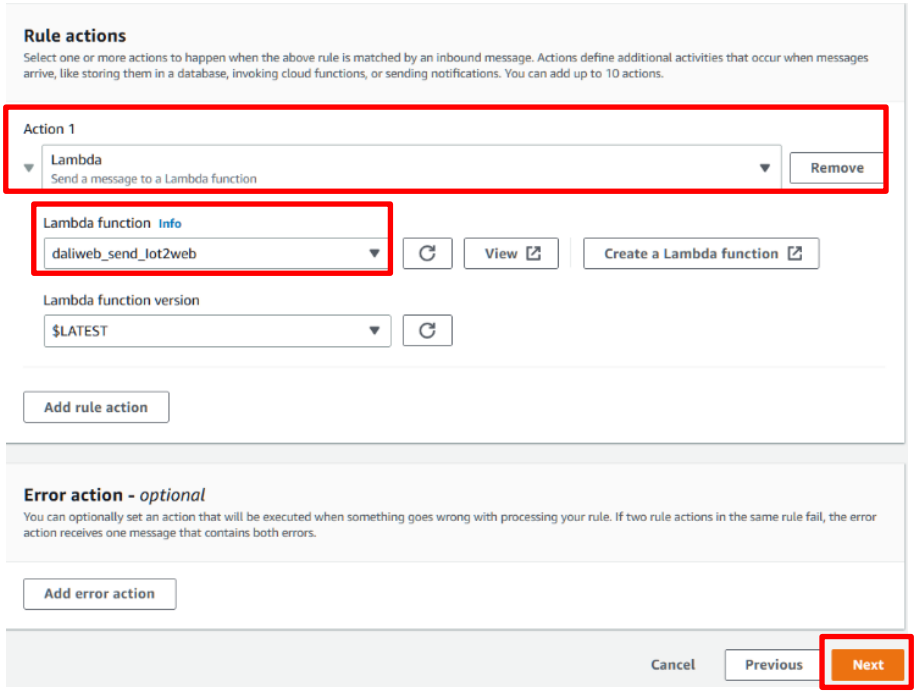

Figure 106 Aws IoT (4/7)

•Select “Lambda” as Action 1 and select “daliweb_send_iot2web” as Lambda function in the figure.
• Select [Next] in the figure.

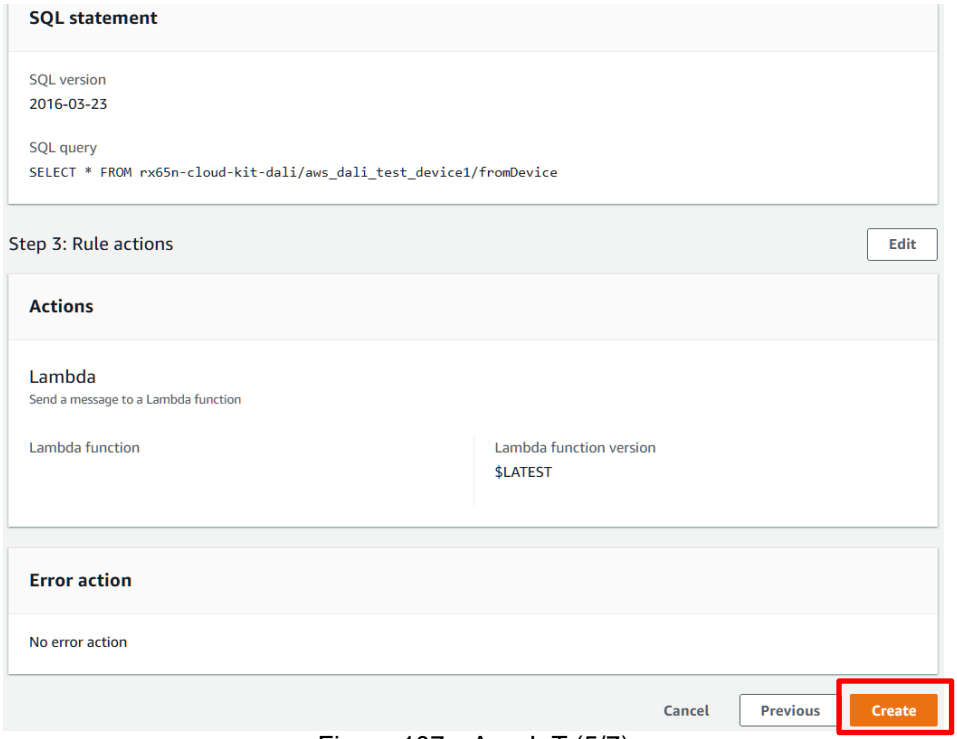

Figure 107 Aws IoT (5/7)

• Check items and select [Create] in the figure.

• Select [Create] again for creating new rules.

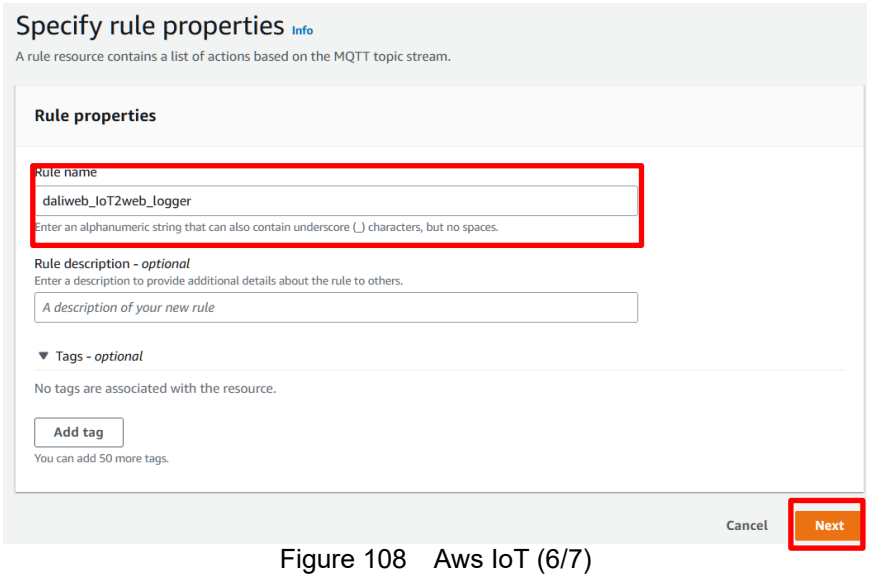

Figure 108 Aws IoT (6/7)

• Set "daliweb_IoT2web_logger" to the name in the figure then select “next”.

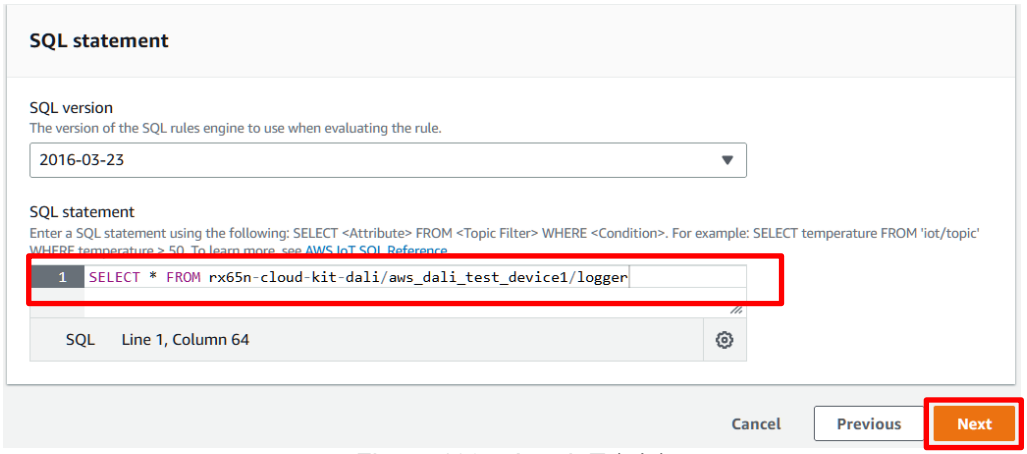

Figure 109 Aws IoT (7/7)

• Set ‘rx65n-cloud-kit-dali/thing name/logger’ in the rule query statement to the name in the figure.
In this example, the thing name is ‘aws_dali_test_device1’.
• Set as "logger" in the same way as "fromDevice".
• Select [Next] in the figure then check items and select “create” in the same way as “from Device”.

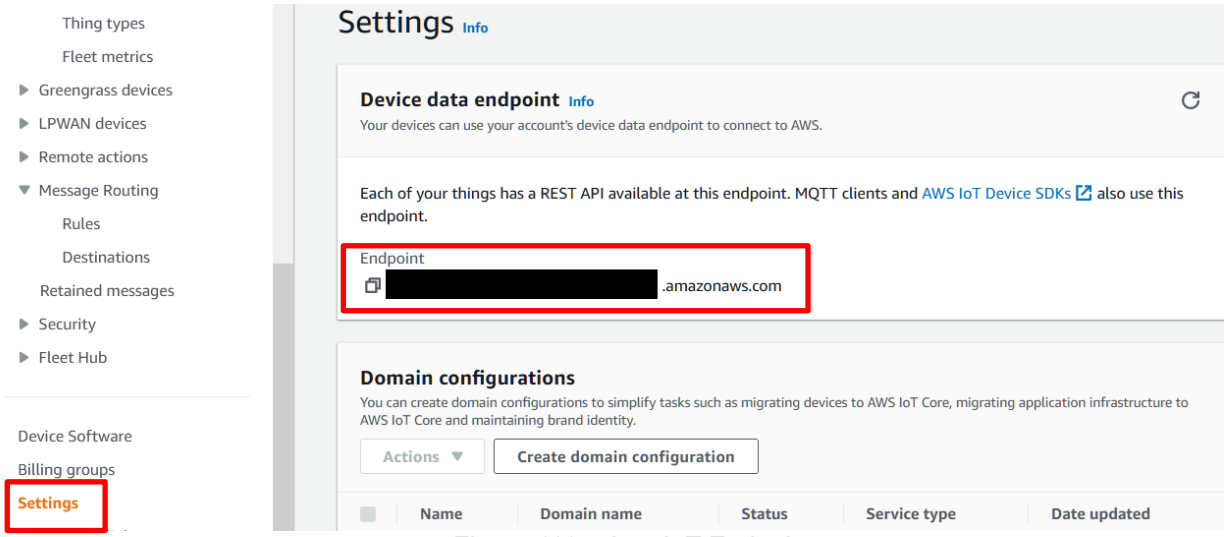

Figure 110 Aws IoT Endpoint

• Get the endpoint from AWS IoT.(IoT core -> Settings)

15. Prepare the AWS web application.

• Unzip the AWS web application (aws-web.zip).
• Place the following certificate file in the daliweb¥html¥assets¥php folder.
• The certificate will be the following file.
・XXXXXXXXXX-certificate.pem.crt
・XXXXXXXXXX-private.pem.key
・AmazonRootCA1.pem

```
1 ###Please change the settings here##################
2 MONO=[redacted]
3 CERT=[redacted]-certificate.pem.crt
4 PRIVATE=[redacted]-private.pem.key
5 CA=AmazonRootCA1.pem
6 URL=[redacted].amazonaws.com
7 ######################################################
```

Figure 111 AWS WEB KEY

• Modify the root/setting.txt file in aws-web.zip.
-MONO: Set the "thing name".
-CERT: Set the "XXXXXX-certificate.pem.crt" file name
・ PRIVATE: Set the file name of "XXXXXX-private.pem.key"
・CA: Set "xxxxxRootCA1.pem" file name
-Set the "thing endpoint URL" confirmed on the previous page
Note:
・Do not include spaces.
・Please do not make a line break.

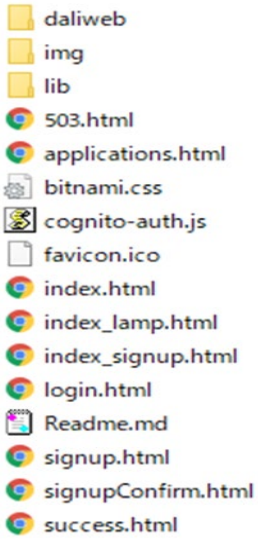

Figure 112 AWS WEB files

• Compress the AWS-WEB source as a zip file with any file name as shown in the figure.
Example: aws-web-XXXX.zip
important point:
Compress it as a zip file while it is placed in Root as shown.

13. From Services in the AWS Management Console, choose AWS Elastic Beanstalk

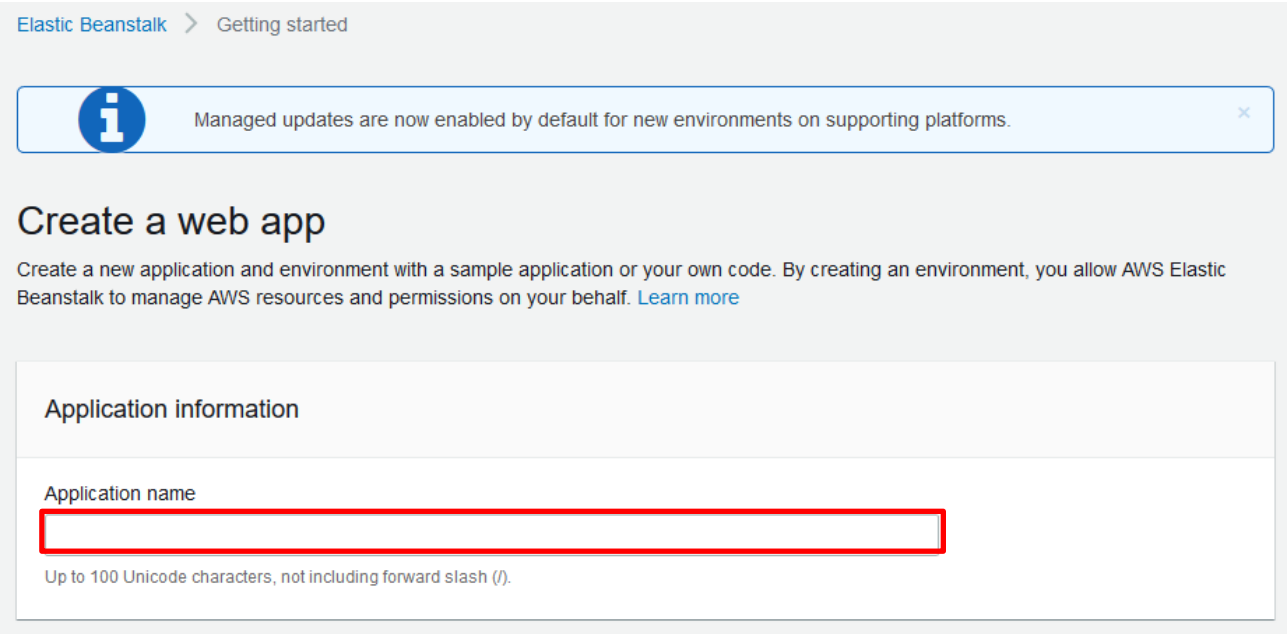

Figure 113 AWS Elastic Beanstalk(1/2)

• Click [Create Application].
• Enter AWS DALI for the application name in the figure then select “create”.
• Select [Create a new environment].
• Select [Web server environment] and Select [Select].

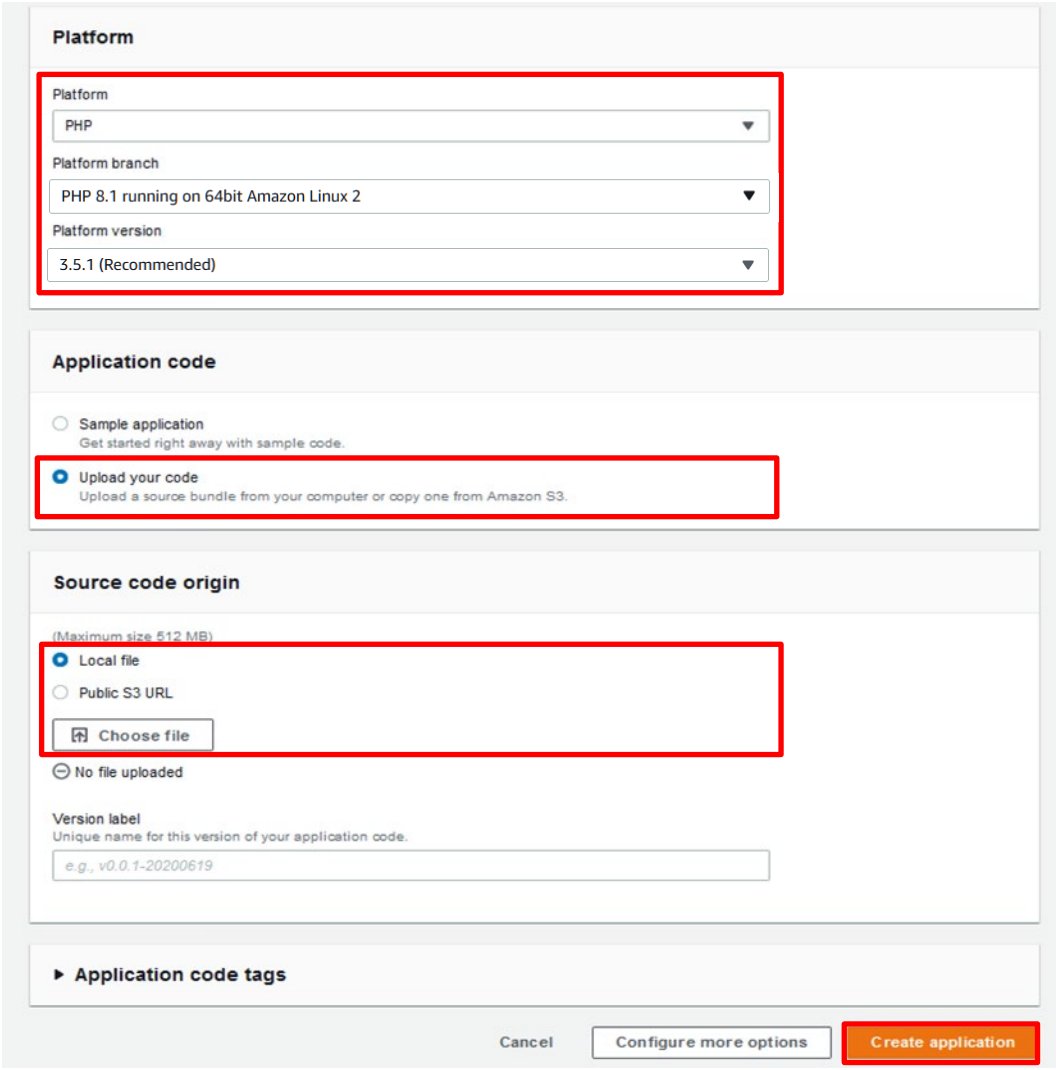

Figure 114 AWS Elastic Beanstalk(2/2)

・Select [PHP] in the figure.
・Select [Upload Code] for the application code in the figure.
・Create a ZIP file of the AWS-WEB application that saved the source code of the figure in the previous chapter.
・Select [Create application] in the figure.

-When the environment information screen is displayed, check the web server environment and click " Platform ". Select PHP from the drop-down list.

-After setting the platform, select "Create environment".
・After switching to the screen below, wait until the process is completed.

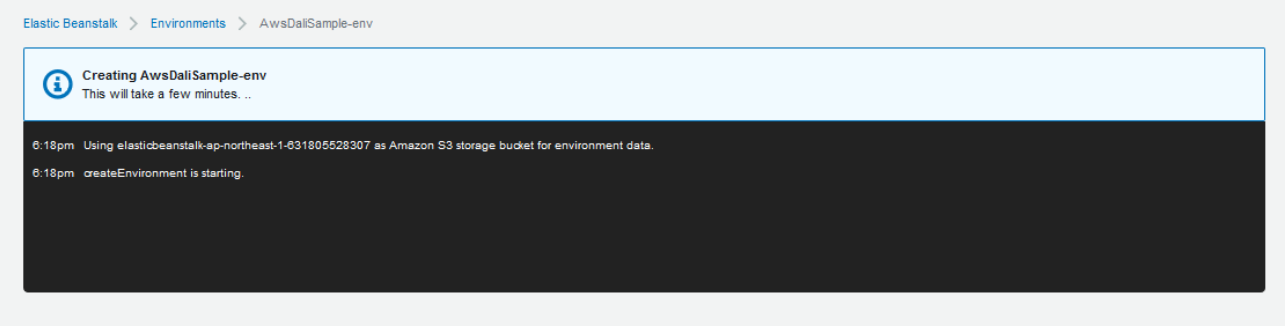

Figure 115 Elastic Beanstalk environment creation screen

-Registration with AWS Elastic Beanstalk is complete. Save URL as text. (*1)

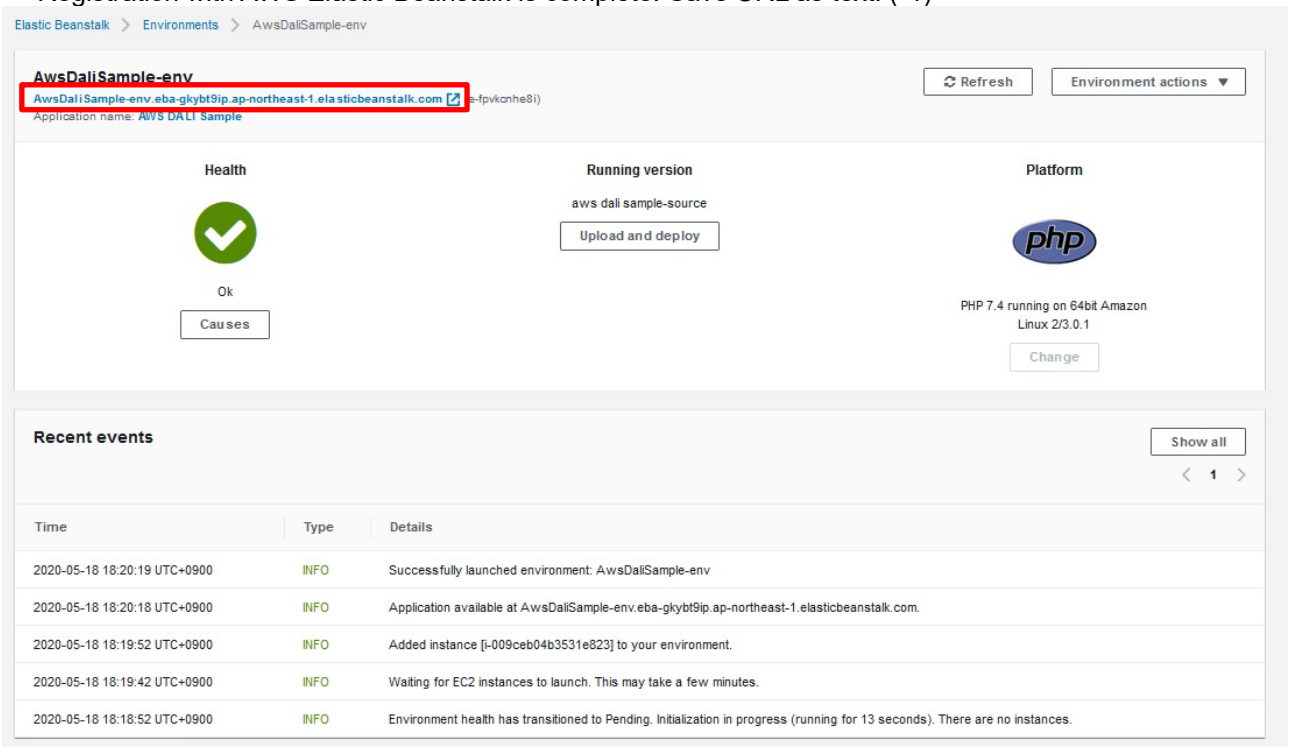

Figure 116 Elastic Beanstalk environment main screen

14. Refer to section 5 (Building the environment) and execute this demo project.
15. Set the URL (* 1) using Google Chrome.

* If you use another browser, it may not work properly.Start Google Chrome by specifying URL/signup.html.

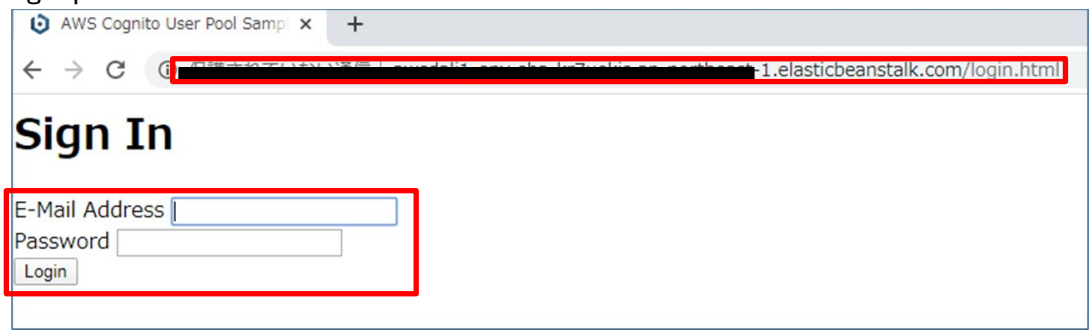

Figure 117 AWS web application sign-in screen

・The Sign In screen will be displayed.
・Specify Mail address and password
Password must be at least 8 characters, using uppercase and lowercase letters and numbers
The verification code will be sent to the email address you set.

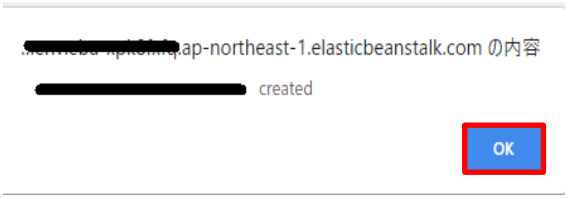

Figure 118 AWS web application sign-in screen

・Select [OK] in the figure.

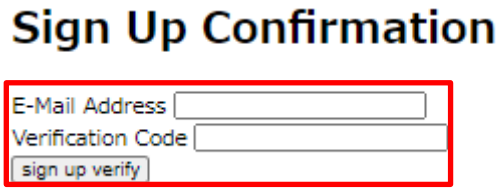

Figure 119 AWS web application sign-in screen

• The verification code will be sent to the registered email.
• Enter your email address and verification code.
• Click “sign up verify”.

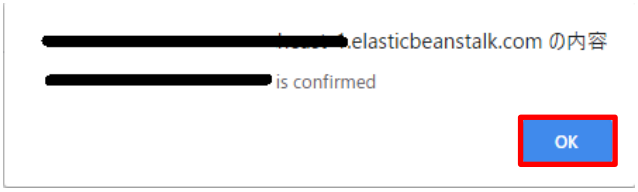

Figure 120 AWS web application sign-in screen

・Select [OK] in the figure.

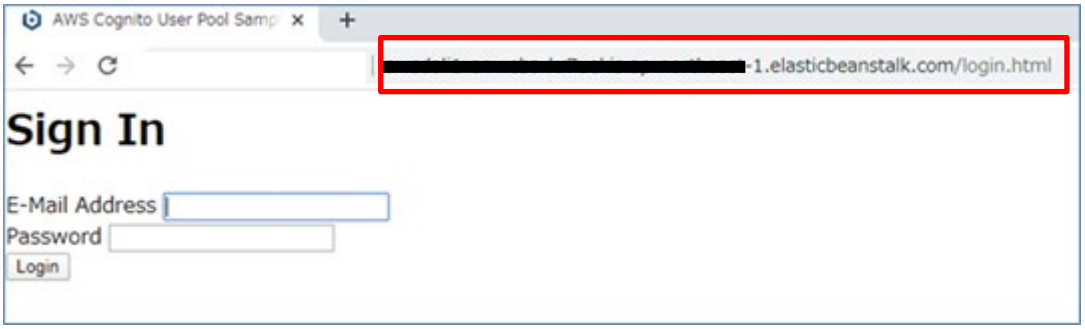

Figure 121 AWS web application sign-in screen

The Sign In screen will be displayed. Please set the email address and password used for the AWS console.
Note: For web applications related to user authentication, please review the entire application as well.

16. Operate the AWS Web application screen.
    Before controlling the lighting of the lamp, prepare on the 5.6.2.1 (1) DALI Control screen.
    - Obtain information on the connected Control Device / Control Gear.
    - Place the acquired Control Gear on the floor.
    - After pressing the Group Select button, select the number you want to light, and then press the “End” button.
      * Since it is "Group", you can select multiple numbers.
17. Operate the lamp on the AWS Web application screen.
    ex.1: Press the ON button in the figure below -> The connected lamp lights up.
    ex.2: Press the OFF button shown below -> The connected lamp goes off.

Figure 122 AWS Web Application System Overview Diagram

18. How to operate screen

Operate the lamp on the AWS Web application screen.

On the AWS Web application screen, an operation screen is prepared in addition to the screen described in the operation procedure.

The detailed operation method and usage of each screen are shown below.

・ DALI Control screen

On this screen, you can get the lamps connected to the DALI Network, lay out the lamps in the office floor image, group the lamps, and issue frames to the grouped lamps.

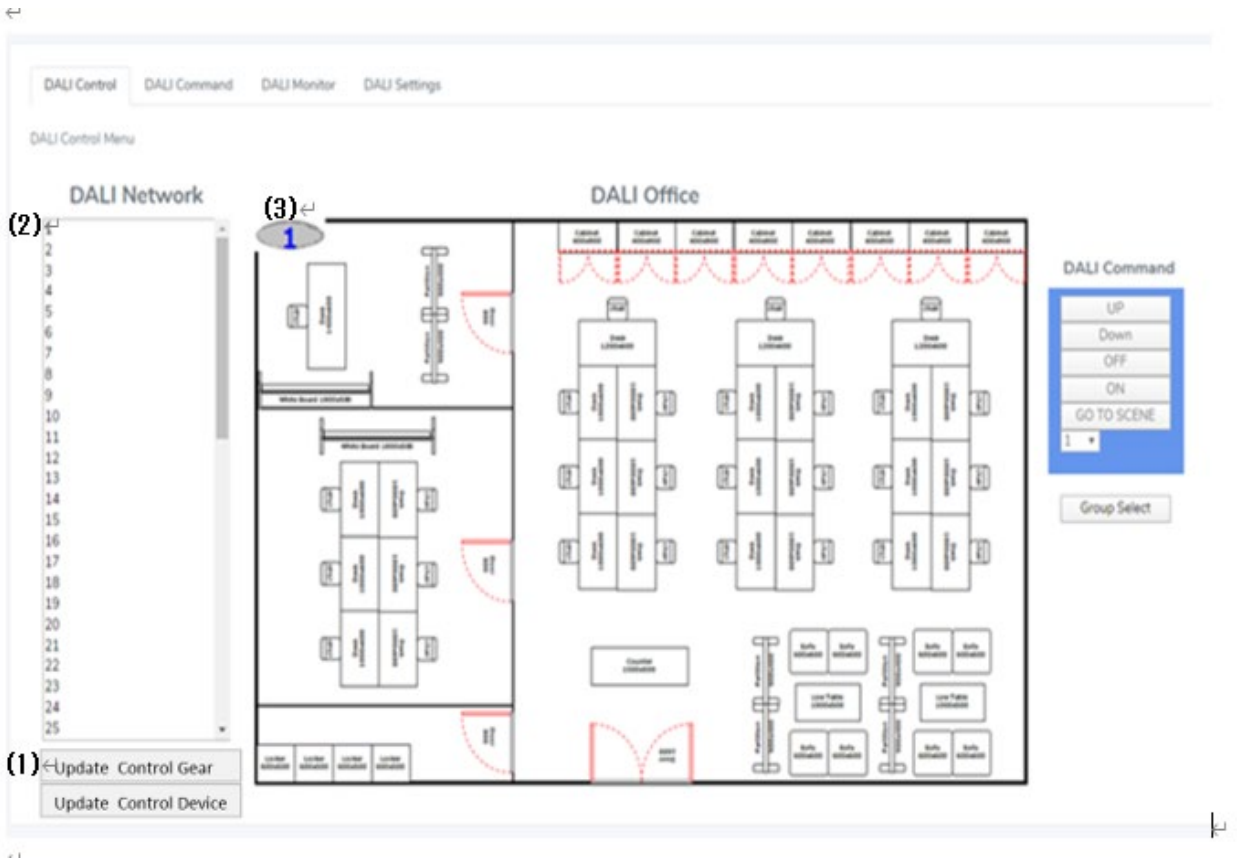

Figure 123 DALI Control

-1. Acquire Control Gear and Control Device

(1) Click [Update Control Gear] to get the short address of Control Gear connected to DALI Network.
Click [Update Control Device] to get the short address of Control Device connected to DALI Network.

(2) List the acquired Control Gear and Control Device in [DALI Network].
(Control Gear-up to 64, Control Device-up to 64)
Control Gear is displayed as "XX" and Control Device is displayed as "A XX".
(XX indicates short address number)

(3) At the same time as listing, the lighting image of Control Gear only is placed on the upper left of the floor. (The young number is the front)
* Control Device images are not displayed.

-2.Control Gear layout

Lay out the upper left lamp at the desired position on the floor with Drag & Drop.
This function is only for Control Gear. Control Device images are not displayed.

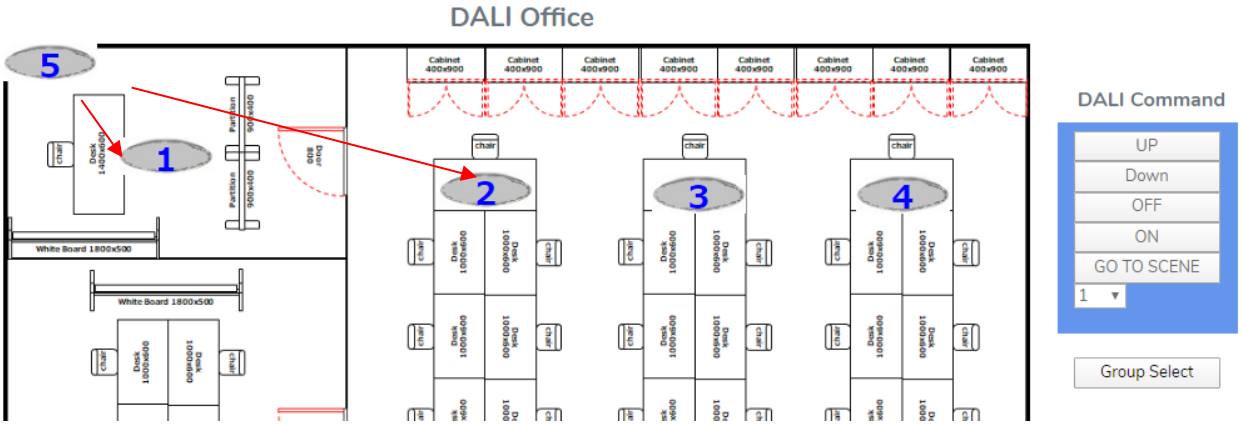

Figure 124 Layout of Control Gear

-3.Group Select mode

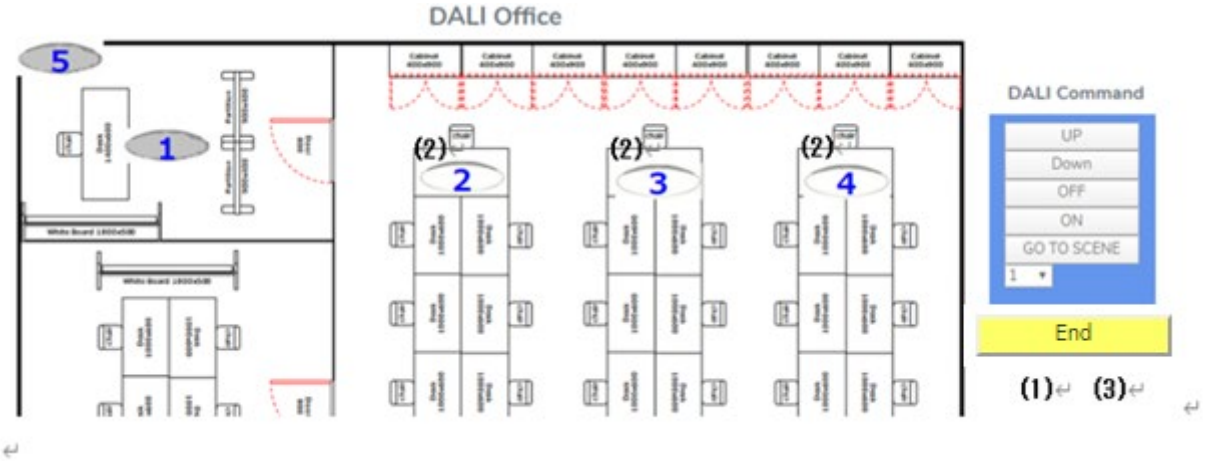

Figure 125 Group Select mode

(1) Click [Group Select] on the right side of the screen to switch to the lamp grouping mode.
[Group select] button (*1) is changed to [End] button (*2)

(2) Select the lamps you want to group by clicking on them. The selected lamp changes to a lighting display.

(3) Click [Exit] to end Group Select mode.

*1 Clicking [Group Select] again clears the previous selection and allows you to redo the grouping.

*2 The position of lamps you put and the setting of grouping is not saved.

-4.DALI Command
It Issues a Frame with the following buttons for the grouped lamps and operates Control Gear.

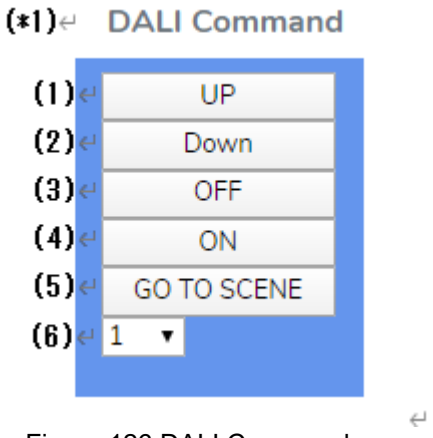

Figure 126 DALI Command

(1) Increase the light intensity of the lamp.
(2) Reduce the light intensity of the lamp.
(3) Turn off the lamp.
(4) Turn on the lamp.
(5) Execute the scene of the scene number selected in the pull-down.
(6) Select the scene number to issue in [GO TO SCENE].

*1 If the Control Gear is not grouped, the buttons are disabled and cannot be operated.
When transmitting a Frame to an ungrouped Control Gear, transmit it from the "DALI Command" screen.
By grouping Control Gear, the buttons are enabled and can be operated.
This function is only for Control Gear.

- DALI Command screen

On this screen, you can get the Control Gear connected to the DALI Network and issue a Frame to the grouped Control Gear.

Only [Update Control Device] button is available for Frame to Control Device. You can get the short address of the Control Device connected to DALI Network.

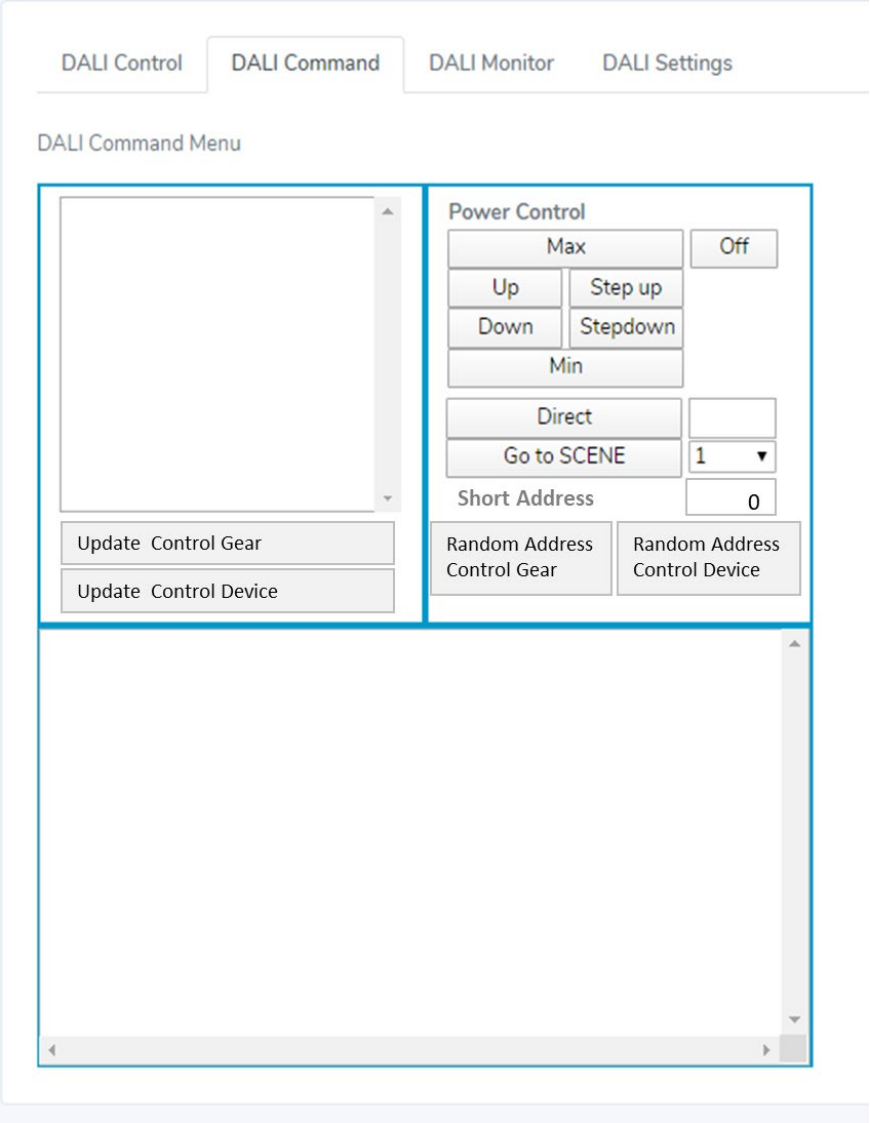

Figure 127 DAIL Command

-1.Acquire Control Gear

On the left side of this screen, you can get the short address of Control Device and Control Gear connected to DALI Network.

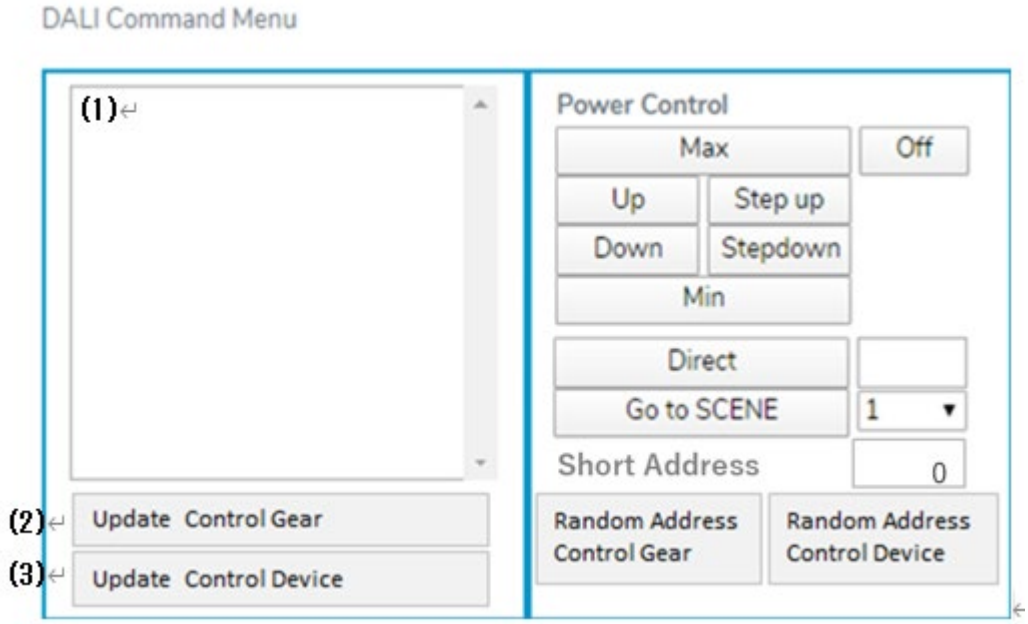

Figure 128 Acquire Control Gear

・Check the Control Gear connected to DALI Network. (* 1)
・By pressing [Update Control Gear] button(2), the acquired Control Gear is listed. (Up to 64)
・You can get the short address of Control Device connected to DALI Network. (* 2)
By pressing [Update Control Device] button(3), the short address of the acquired Control Device is listed on the screen (1). (Up to 64)

*1 This function is the same as the [Update Control Gear] function on the DALI Control tab.
The display in the DALI Control tab list and the display of the Control Gear image are also linked.
*2 This function is the same as the [Update Control Device] function on the DALI Control tab.
The list display on the DALI Control tab is also linked.

-2.Power Control
On the right side of this screen, Frame is issued by each button operation.

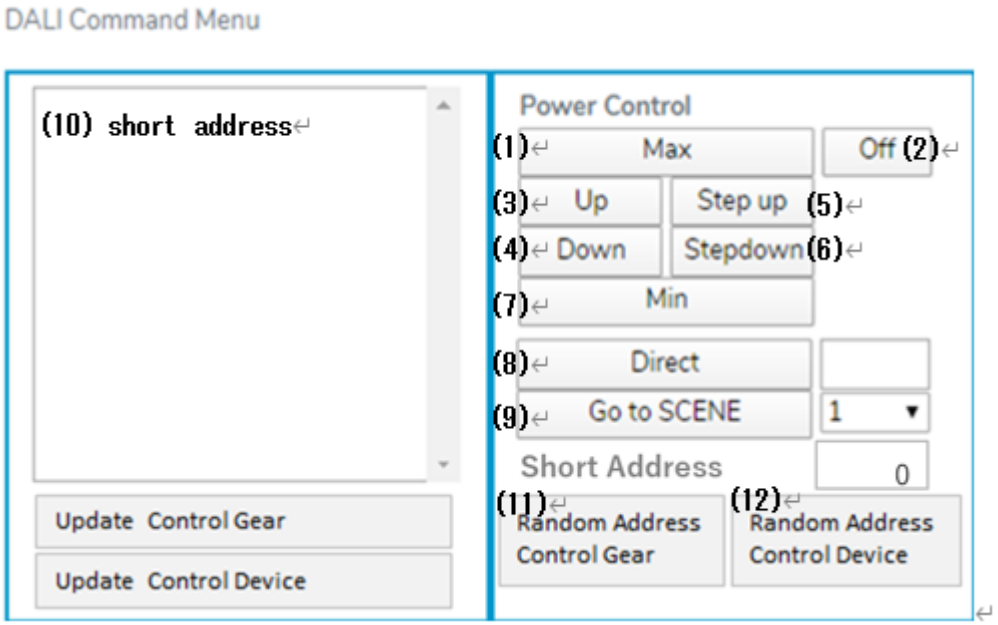

Figure 129 Power Control

Issues a Frame to ControlGear. (*1)
(1) Issue Max Frame
(2) Issue Off Frame
(3) Issue Up Frame
(4) Issue Down Frame
(5) Issue Step up Frame
(6) Issue Step down Frame
(7) Issue Min Frame
(8) Get the right input value and issue Direct Frame
(9) Issue a Go to SCENE Frame for the scene number selected in the right pull-down
(10) Specify short address when issuing Frame

The following Frames can be issued to Control Gear and Control Device connected to DALI Network.
(11) Issue random address allocation to Control Gear
(12) Issue random address allocation to Control Device

*1 Specify the short address of (11) before pressing each frame issue button.
It is not linked with the lighting display of the Control Gear image on the DALI Control tab.

- DALI Monitor screen

  On this screen, you can monitor the DALI communication status and save the results to a file.

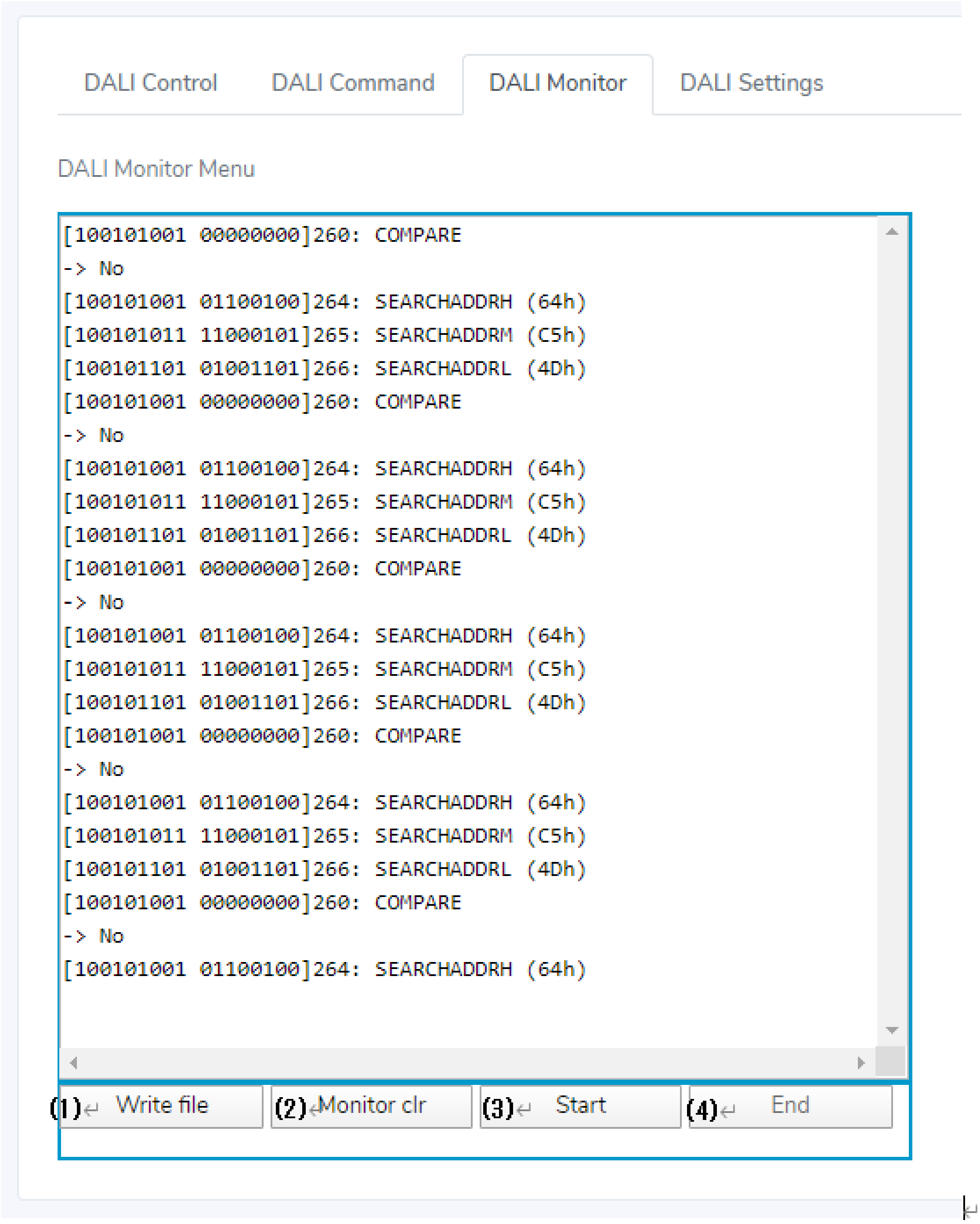

Figure 130 DALI Monitor

(1) Save the display contents to a file
(2) Clear the display content
(3) Start monitoring (Enable Logging Function)
(4) Stop monitoring (Disable Logging Function)

・ DALI Control screen
On this screen, you can allocate / read Frames to 9 matrix buttons on the board and issue frames.

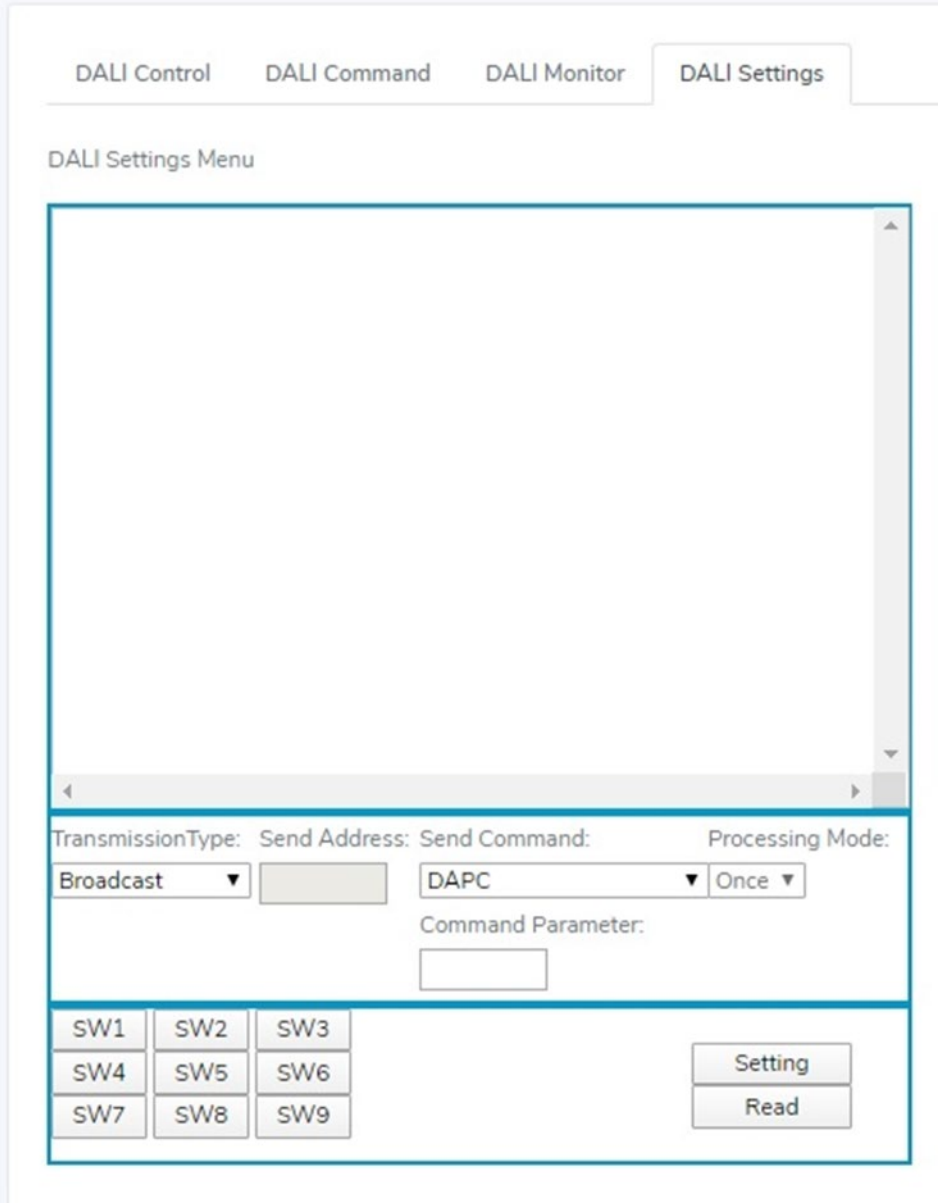

Figure 131 DALI Setting

-1.Allocate Matrix button

In this screen, you can allocate Frame to the specified matrix button.

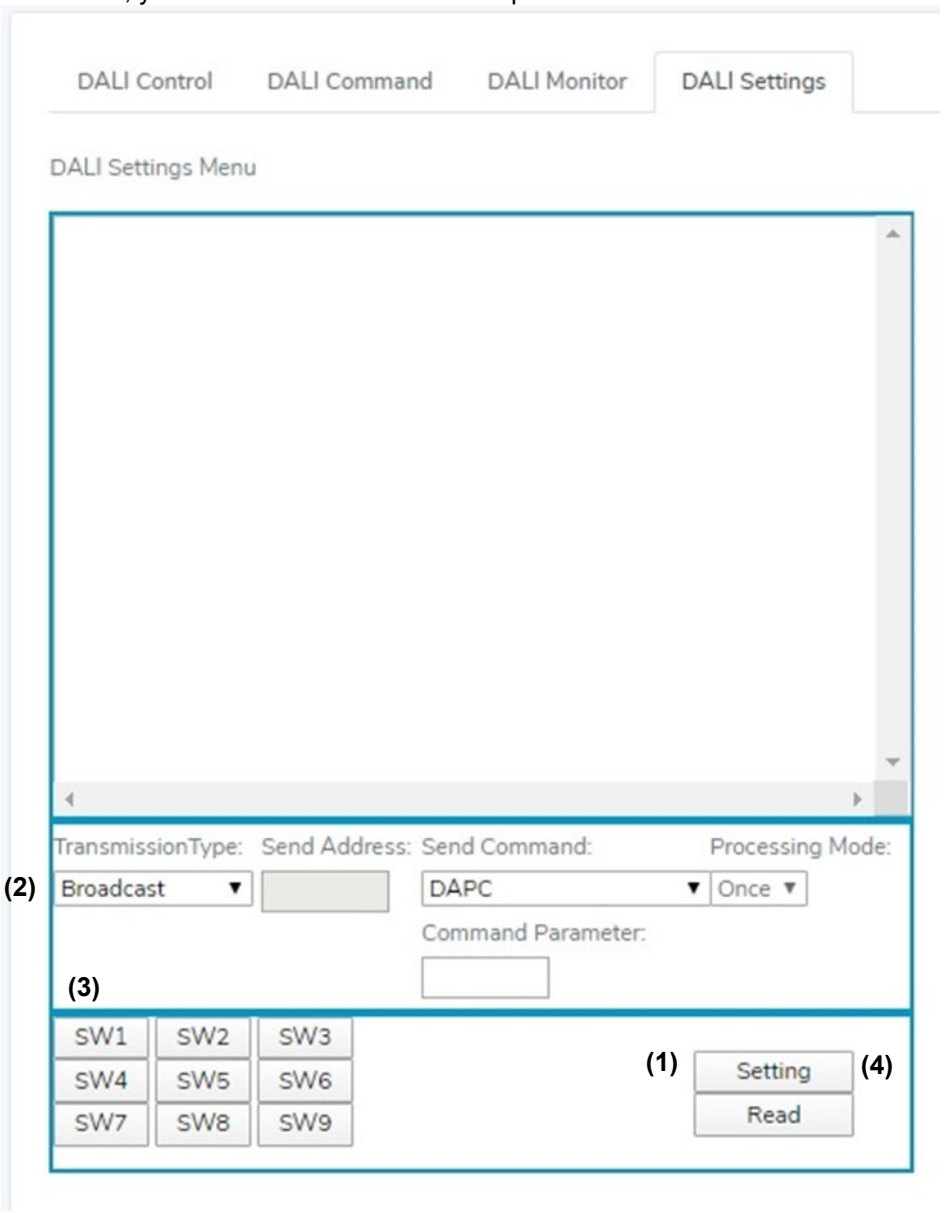

Figure 132 Allocate DALI CODE

(1) Click [Setting] to move to allocation mode. The button switches to [End].
(2) Select the Transmission Type, Send Address, and Send Command to be set on the matrix button.
(3) Click the buttons of SW1 to 9 to allocate the matrix buttons.
(4) Click [End] to complete the allocation.

-2.Read or issue allocation code
On this screen, read the Frame allocated to the specified matrix button or issue the Frame.

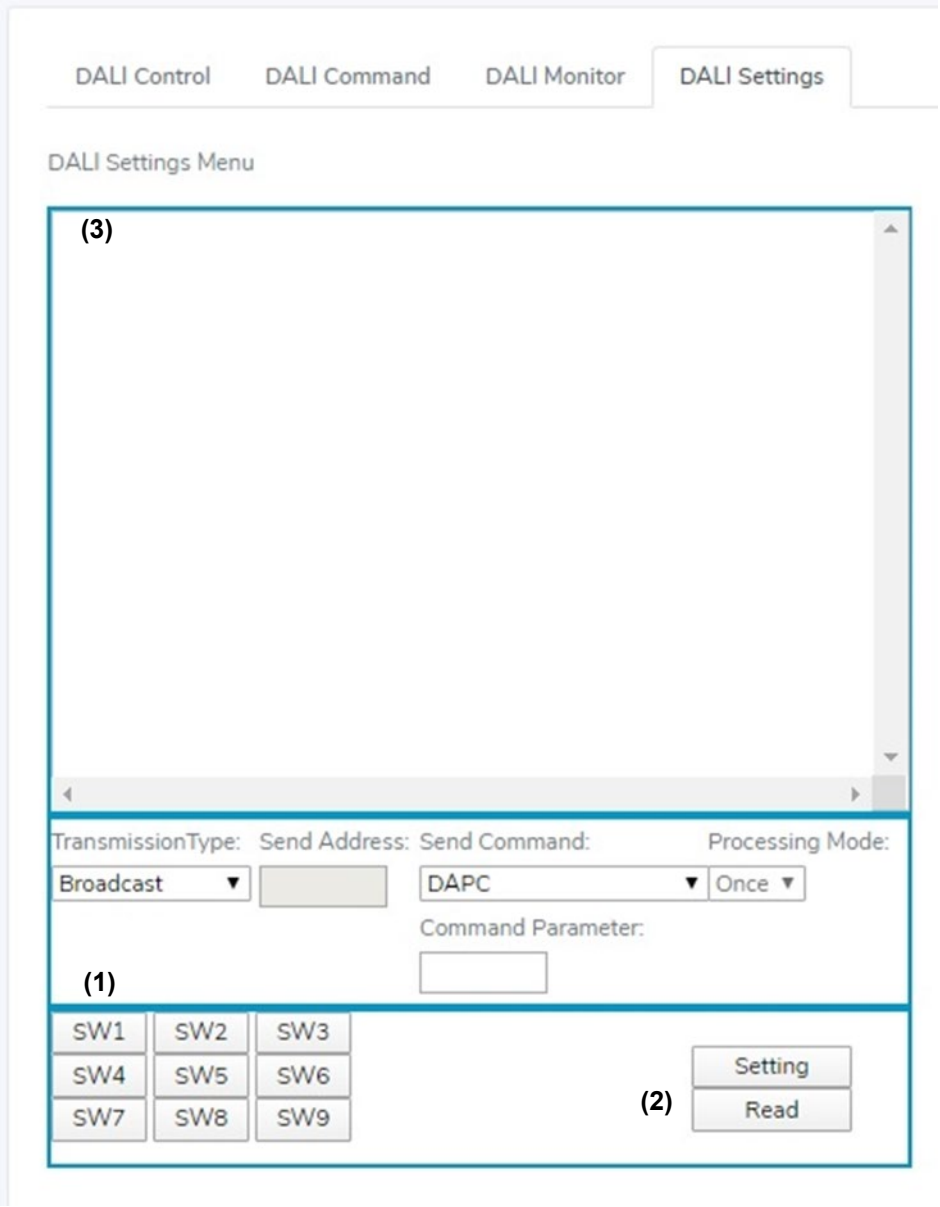

Figure 133 Read and publish

(1) Select the number (SW1 ~ 9) of the matrix button to get
(2) Click the button to read the matrix button
(3) Frame of loaded matrix button is displayed

5.6.3 Operator Matrix button

The operation procedure of the demo project when operating with the matrix button operation is shown below.

1. Install necessary software and connect the device referring to Chapter 5.1 to 5.4.
2. Refer to section 5.5, and run this demonstration project.
3. Operate the matrix buttons.
   ex. 1: Turn on all connected lamps with the maximum light intensity -> Press the SW1 button
   ex. 2: Reduce the light intensity of all connected lamps -> Press and hold SW5 button

Frames shown in the table below are set as initial settings for each matrix button.
Short press of a matrix button sends a frame once, long press repeatedly transmits a frame. If multiple matrix buttons are pressed at the same time, no frame is transmitted.
If you want to set a different Frame from the default Frame, you can change the Frame setting by connecting to the AWS web application. It is also possible to read the currently set Frame.

For how to change / read the Frame setting of the matrix button, refer to the operation method in chapterClick [Setting] to move to allocation mode. The button switches to [End]..

Table 67 Default setting value of matrix button

Switch	Content
SW1	Broadcast RECALL MAX LEVEL
SW2	Broadcast RECALL MIN LEVEL
SW3	Broadcast OFF
SW4	Broadcast UP
SW5	Broadcast DOWN
SW6	Broadcast Scene 0
SW7	Broadcast ON AND STEP UP
SW8	Broadcast STEP DOWN AND OFF
SW9	Broadcast DAPC 229

5.7 Restrictions

1. After starting the demo project, a waiting time of 50ms is required to stabilize the power supply voltage of the RX65N DALI-2 Option board.
2. When connecting to AWS, the authentication process is prioritized during AWS user authentication, so the response to the operation of the matrix button will be delayed.
3. When connecting and communicating with the DALI Master Controller GUI while connecting to the AWS Web Application, do not issue Frames from both sides at the same time. Frame response may not be received correctly.
4. The log display of the AWS web application uses MQTT communication.
   Since the order is not guaranteed in MQTT communication, the order of DALI command display may change.
5. When connecting to AWS, the connection with AWS may be lost. The connection will be restored automatically, but the AWS web application will not respond during that time.
6. In the AWS web application log display, the command name is displayed for the command sent by itself. For commands sent by others, the command name is not displayed.
7. OTA will suspend all features after start and restart after completion.

5.8 OTA

1. OTA procedure

Refer to the "Procedure Summary" at the following URL when trying to update over the air (OTA).
https://github.com/renesas-rx/amazon-freertos/wiki/OTA%E3%81%AE%E6%B4%BB%E7%94%A8
However, it is not necessary to change No. 2 in "Procedure Summary".

2. OTA procedure from AWS console

Upload the OTA update file to AWS and follow the OTA implementation procedure from the AWS console.
Refer to the following URL.
https://docs.aws.amazon.com/freertos/latest/userguide/ota-console-workflow.html

3. For the basic part of OTA, please refer to the following URL.

Refer to the following URL.
https://docs.aws.amazon.com/freertos/latest/userguide/freertos-ota-dev.html

6. Error handling

6.1 Cannot build

6.1.1 The project folder hierarchy is deep

If the folder hierarchy of the placed project is too deep, it may not be possible to build the project.
In that case, lower the folder hierarchy of the project file.

```
For example:
AAAA, XXXX is any folder name created for the project
C:/AAAA/XXXX/Project file
↓
C:/XXXX/Project file
```

6.1.2 FIT module is changed

The following error may occur when you change the FIT module using the software component and build an imported demo project.

```
fatal error: file name: No such file or directory
```

Since the FIT module to be used may have been deleted, register it again if it has been deleted.

6.2 Cannot connect to DALI device

6.2.1 DALI Master Controller GUI

Communication settings such as the COM port and baud rate used may be incorrect.

1. Check the COM port
   Open Device Manager and check the displayed COM port.
2. Communication settings
   The communication settings of the DALI Master Controller GUI are as follows.

Table 68 Communication settings

Item	Settings
baud rate	115200
Data	8bit
Parity	none
Stop bit	1bit
Flow control	none

Note: Do not use it to check Control Device connection.
The DALI Master Controller GUI controls the Control Gear and cannot check the connection with the Control Device. Use the AWS web application to check the connection with the Control Device.

6.2.2 AWS Web Application

1. PC cannot log in to AWS
   Check the contents of Chapter again and confirm that the user registration has been completed.

2. RX65N board cannot connect to Wi-Fi
   The certificate used, the registered SSID and the password may be incorrect.
   Check the contents of Chapter again and check that the settings are correct. If you restart the RX Cloud Option Board while holding down the User SW, the SSID display and the communication log with MQTT are displayed, so you can use it to check the connection with MQTT.

Figure 134 Screen displayed by pressing the user switch

Figure 135 Log display screen when connected to MQTT

* Separate communication log display software such as Tera Tarm is required. Communication settings of Log display software are the same as Table 64.
* When using, DALI Master Controller GUI cannot be connected. When connecting, restart by pressing Reset SW.

7. Reference document

e2studio User's Manual: Getting Started Guide for V7.0(r20ut4374)

RX Family Flash Module Using Firmware Integration Technology(r01an2184)

RX Family GPIO Module Using Firmware Integration Technology(r01an1721)

RX Family SCI Module Using Firmware Integration Technology(r01an1815)

RX Family ADC Module Using Firmware Integration Technology(r01an1666)

RX65N Group, RX651 Group User’ s Manual:Hardware(r01uh0590)

RX Family Cloud Option Board User’s Manual (r12um0039)

User's Manual Target Board for RX65N(r12um0038)

DALI Master Controller GUI User's Manual(r20ut0715)

Appliret EZ for HCD Conttoller Ver.9.00 User's Manual(r20ut0435)

EZ-0012 RL78/I1A DC/DC LED Control Evaluation Board User's Manual(r01uh0363)

RL78/I1A Lighting Communications Using RL78/I1A (Reception)(r01an1115)

RL78/I1A Lighting Communications Using RL78/I1A (Transmission)(r01an3193)

RL78/I1A DALI-2 Control Gear Basic (102) LED (207) Color Control (209Tc) Application Notes(r01an4654)

DALI standard
IEC62386-103ed1.1
IEC62386-101ed2.1

Amazon FreeRTOS User Guide
https://docs.aws.amazon.com/freertos/latest/userguide/what-is-amazon-freertos.html

REVISION HISTORY

Rev.	Date	Description	
		Page	Summary
1.00	June.19.20	All	First edition issued
1.01	Nov.21.22	All	This fixes AWS policy settings to a minimum configuration

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.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.

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

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.

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

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

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

Contact information

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