RX65N グループ　アプリケーションノート

R01AN5396JJ0101
Rev.1.01
Nov.21.22

Cloud kit を使用した DALI-2 照明通信
(Control Device/Application Controller 編)

要旨

本アプリケーションノートは、RX65N Cloud Kit を使用した照明通信規格 DALI の Application Controller 機能を搭載したデモプロジェクトの動作について説明しています。

本アプリケーションノートは、目的別にご使用いただけます。

- デモ環境を動かしたい方
  2 章、3 章、5 章を参照してください。
- デモプロジェクトを使用した開発を行いたい方
  1 章から順にご覧ください。

動作確認デバイス

RX65N

目次

1.　DALI 概説

1.1　DALI とは

DALI (Digital Addressable Lighting Interface)は、国際オープンの照明制御通信プロトコルで、主に複数の蛍光灯や LED 照明などの照明機器の制御を行うための規格です。DALI 規格に則って作られた照明機器であれば、異なるメーカーの製品間でも通信することができます。

規格の概要と Control Gear、Control Device については、以下の各アプリケーションノートを参照してください。

- Control Gear について：RL78/I1A による照明通信(受信編)　アプリケーションノート(r01an1115)
- Control Device について：RL78/I1A による照明通信(送信編)　アプリケーションノート(r01an3193)

本アプリケーションノートでは、上記各アプリケーションノートに記載されていない Control Device で動作させるうえで関連する内容を記載します。

1.2　Receiver 機能と Transmitter 機能

Frame の送受信を行うにあたって、Frame の受信を行う Receiver 機能と Frame の送信を行う Transmitter 機能があります。この Receiver 機能と Transmitter 機能は、使用する BUS Unit によって対応しなければいけない Frame の種類やビットタイミング定義等が異なります。

BUS Unit の Receiver と Transmitter 仕様の要約を下表に示します。

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="2">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></td><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 の Application Controller では Multi-master 送信、Single-master の Application Controller では、Single-master 送信が可能な機能の搭載が必要となります。

1.3　DALI Frame 概要

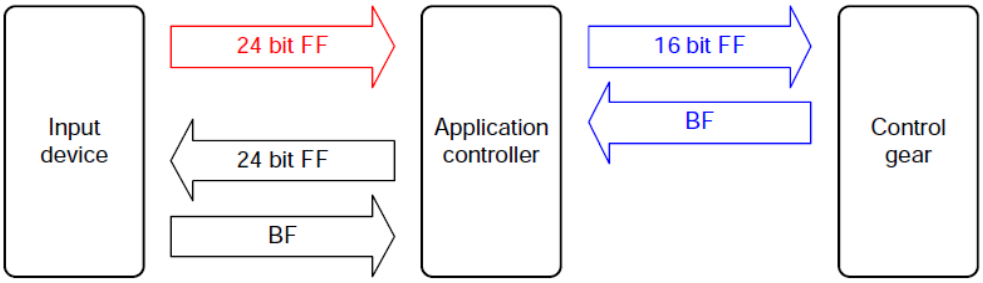

Figure 1 Communication between bus units

規定では Forward Frame(16bit, 24bit)と Backward Frame(8bit)が存在します。その他に Reserved Forward Frame(20bit, 32bit)とユーザ独自実装として Proprietary Forward Frame があります。

Proprietary Forward Frame はデータビット数が規定以外の場合と、開始ビット、データビット数、停止条件のエンコーディングが異なる場合の 2 種類に分類され、各々Proprietary Forward Frame を解釈するように設計されていない BUS Unit で Frame サイズ違反やビットタイミング違反を判定する必要があります。

2.　システム概要

本デモプロジェクトでは、RX65N Cloud kit + DALI-2 オプションボードから複数の接続サービスまたは機器を使用して DALI の通信と Control Gear を介した灯具の調光制御を行います。
接続サービスまたは機器によって 3 通りの動作を行うことができます。

- DALI GUI 操作モード：DALI マスタコントローラ GUI と接続して調光制御を行う
- AWS Web 操作モード：AWS Web アプリケーションと接続して調光制御を行う
- スイッチ操作モード：マトリクスボタンで調光制御を行う

本デモプロジェクトのシステム概要図を下図に示します。

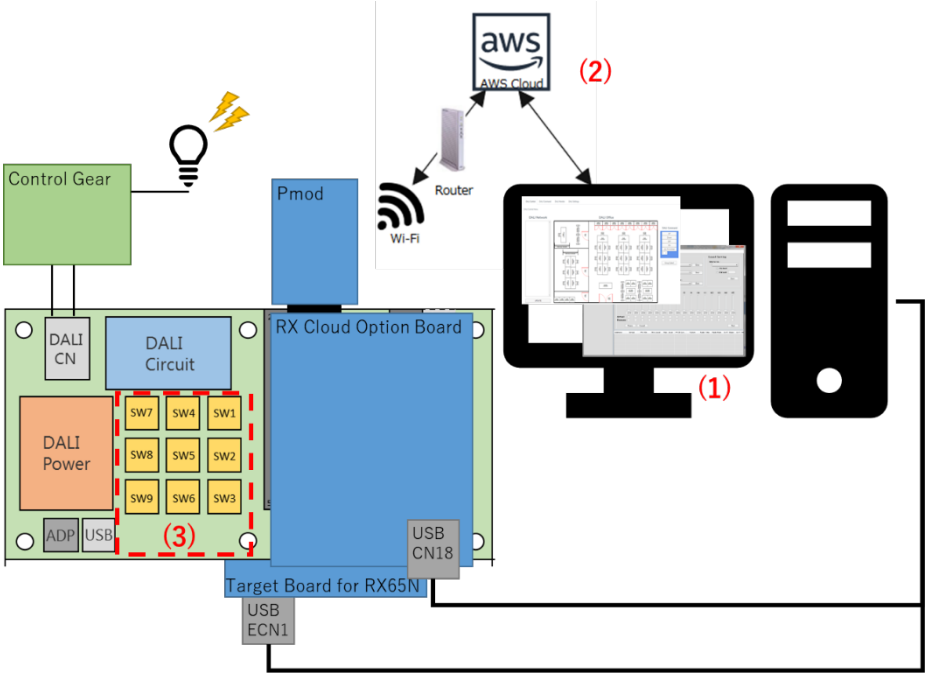

Figure 2 System outline diagram

2.1　DALI GUI 操作モード

RX65N Cloud kit + DALI-2 オプションボードに DALI マスタコントローラ GUI を接続して動作させることによって、Control Gear に接続された灯具の調光制御を行うことが出来ます。DALI マスタコントローラ GUI は PC 上で動作し、DALI 規格に従った通信が行える GUI(Graphical User Interface)です。

2.2　AWS Web 操作モード

AWS Web アプリケーションは AWS Cloud の AWS IoT Core を使用し Google Chrome ブラウザ上で動作します。RX65N Cloud kit＋DALI-2 オプションボード と WEB アプリケーションとの通信には MQTT を使用して Wi-Fi 接続で通信を行います。ブラウザ上からの操作で Control Gear に接続された灯具の簡単な調光制御を行うことが出来ます。

2.3　スイッチ操作モード

接続サービスを使用せず、RX65N Cloud kit＋DALI-2 オプションボード単独で動作します。マトリクスボタンを押すと対応する 16bit Forward Frame の送信を行い、Control Gear に接続された灯具の簡単な調光制御を行うことが出来ます。

2.4　download ファイル構成

ダウンロードでするファイル構成は以下になります。
ファイル名 : r01an5396xx0ZZZ-dali-2-rx-application-controller
ZZZZ:は version により変わります。

フォルダ構成
Workspace/
r_aws_setting　AWS 設定ファイル
r_aws_web　AWS-WEB ソースファイル
rx65b_dali　Rx65N 用プロジェクトファイル
＊rx65b_dali のフォルダは階層を小さくする必要があります。

3.　ハードウェア構成

3.1　ハードウェア環境

本デモプロジェクトで使用するハードウェア環境を下表に示します。

Table 2 Hardware environment list

Item	Content	Provider	Description
使用ボード	Target Board for RX65N	Renesas Electronics Corporation	RX65N MCU 搭載の評価ボード [a]
	RX Cloud Option Board		AWS 接続可能なクラウド通信評価ボード [a]
	Silex Pmod Module		無線 LAN モジュール搭載の通信ボード [a] AWS Web 操作モード時のみ使用します
	EZ-0012		DALI 通信回路を搭載した照明制御用評価ボード [b]
	RX65N DALI-2 オプションボード	TESSERA TECHNOLOGY	RX65N Cloud kit 接続用の DALI 通信回路を搭載した DALI 評価ボード [c]
Wi-Fi	無線ルーター	-	AWS Web 操作モード時のみ、Wi-Fi の接続先として使用します。 無線 LAN 規格：IEEE 802.11b/g/n(2.4GHz) 暗号化方式：なし、AES
PC	Windows10	-	推奨 OS
	Google Chrome	-	使用するブラウザ AWS Web 操作モード時のみ使用します

a Target Board for RX65N と RX65N Cloud Option Board、SILEX UART Pmod は、RX65N Cloud　kit に同梱されています。
ご購入に関してはルネサスエレクトロニクス株式会社へお問い合わせください。
https://www.renesas.com/rx65n-cloud
b EZ-0012 は RL78/I1A を搭載した照明制御用の評価ボードで、本デモプロジェクトでは Control Gear として使用します。ボード搭載 LED にて点消灯を行うため、別途灯具を準備する必要がありません。
※別途 EZ-0012 用 AC 電源アダプタが必要です。
EZ-0012 は別途ツールを使用した設定を行う必要があります。EZ-0012 の設定方法に関しては 5.3 章を参照してください。

c RX65N DALI-2 オプションボードは RX65N Cloud Option kit と組み合わせて Control Device(Application Controller)として使用します。
ボード提供元は TESSERA TECHNOLOGY 社となります。ご購入に関しては TESSERA TECHNOLOGY 社へお問い合わせください。
https://www.tessera.co.jp/

3.2　ボード構成図

3.2.1　RX Cloud Option Board

RX Cloud Option Board の構成図は下記ユーザーズマニュアルを参照してください。

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

3.2.2　Target Board for RX65N

Target Board for RX65N の構成図は下記ユーザーズマニュアルを参照してください。

- RX65N グループ　Target Board for RX65N ユーザーズマニュアル(r12um0038)

3.2.3　RX65N DALI-2 オプションボード

RX65N DALI-2 オプションボードの構成図と外形図を下図に示します。

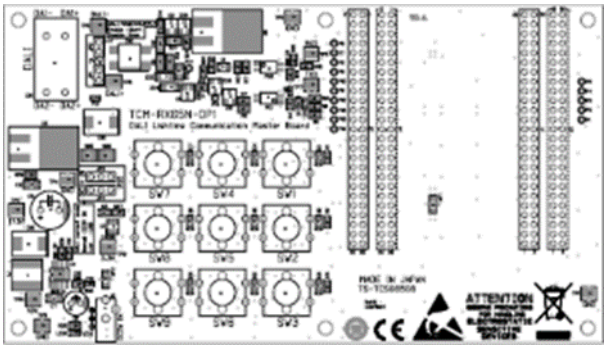

Figure 3 RX65N DALI-2 オプションボード diagram

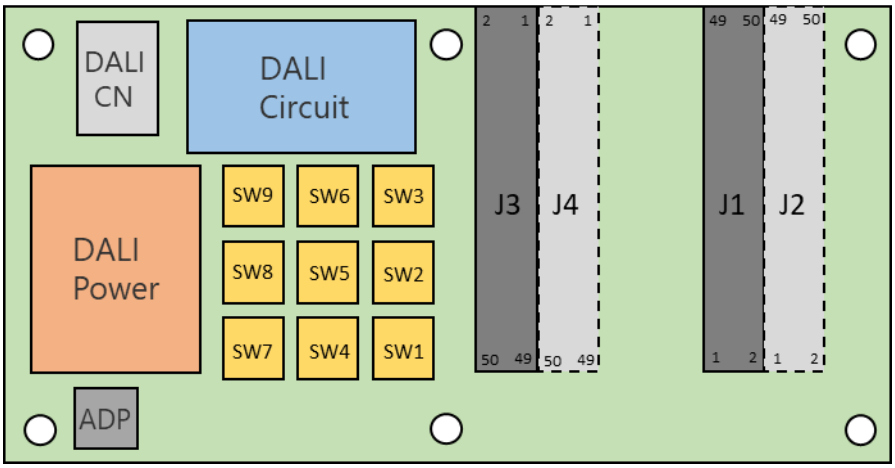

Figure 4 RX65N DALI-2 オプションボード dimensions

使用する各ボードを接続する前に、RX65N DALI-2 オプションボードはボードの設定確認を行う必要があります。ボードの設定は 5 章を参照してください。

4.　ソフトウェア構成

4.1　全体構成

本デモプロジェクトのソフトウェア全体構成を下図に示します。

Amazon Free-RTOS 機能と RX65N の周辺機能、FIT ドライバを使用して、赤枠内の機能を動作させます。

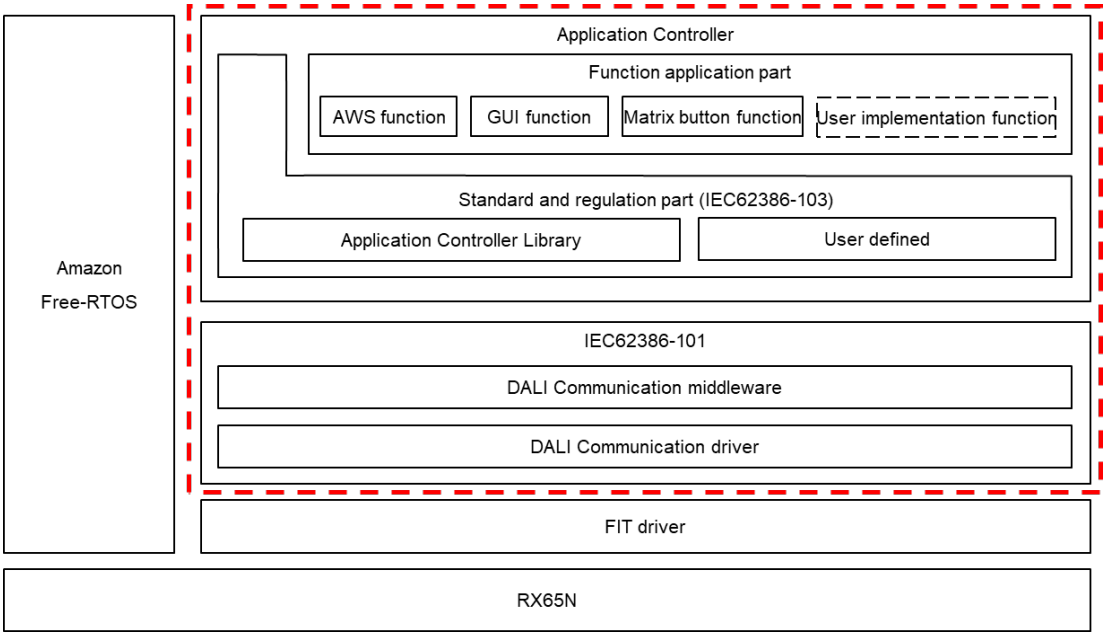

Figure 5 Overall software configuration diagram

- Application Controller
  — 機能アプリケーション部
  各接続サービスまたは機器(AWS Web アプリケーション,DALI マスタコントローラ GUI,マトリクスボタン)を使用して Frame を送受信する際に、各接続サービスまたは機器とのやり取りを行います。
  ユーザ実装機能では、Input Device から発行された Frame に対して処理を行いたい場合など、本デモプロジェクトを使用した独自の動作を実装することが可能な場所です。

  — 規格規定部(IEC62386-103)
  IEC62386-103 規格にて規定されている Application Controller の動作を行います。
  受信した Forward Frame の解釈やそれに対する Backward Frame の発行などを行い、動作の一部に Renesas Electronics 製 Application Controller ライブラリを使用しています。
  ユーザ定義エリアでは、識別動作やメモリバンク、不揮発性メモリへのアクセスなどシステム設計に影響される部分を定義します。

- IEC62386-101
  — DALI 通信ミドルウェア
  DALI 通信ドライバの制御、Application Controller または DALI 通信ドライバとの Frame の通知・管理を行います。

  — DALI 通信ドライバ
  RX65N 周辺機能を使用して DALI BUS 上に発行された Frame の受信と DALI 通信ミドルウェアから受け取った Frame の送信を行います。

4.2　ソフトウェア環境

本デモプロジェクトのソフトウェア環境を下表に示します。

Table 3 Software environment list

Item	Content	Version
言語	C	C99
統合開発環境	ルネサスエレクトロニクス株式会社製 e2studio	7.6.0
コンパイラ	GNU RX GCC	8.3.0.2019.4
FIT ドライバ	RX Driver Package	122
	BSP	5.20
	BYTEQ	1.80
	FLASH	4.20
	ADC (S12AD)	4.20
	SCI	3.20
	GPIO	3.20
	CMT	4.20
	Wi-Fi (SX-ULPGN WIFI FIT Module)	1.00
コード自動生成	CRC	1.7.0
	MTU	1.7.0
	PORT	1.9.0

4.3　RX65N 周辺機能、FIT モジュールの設定

使用する FIT モジュールと rx65n 周辺機能の概要図を下図に示します。

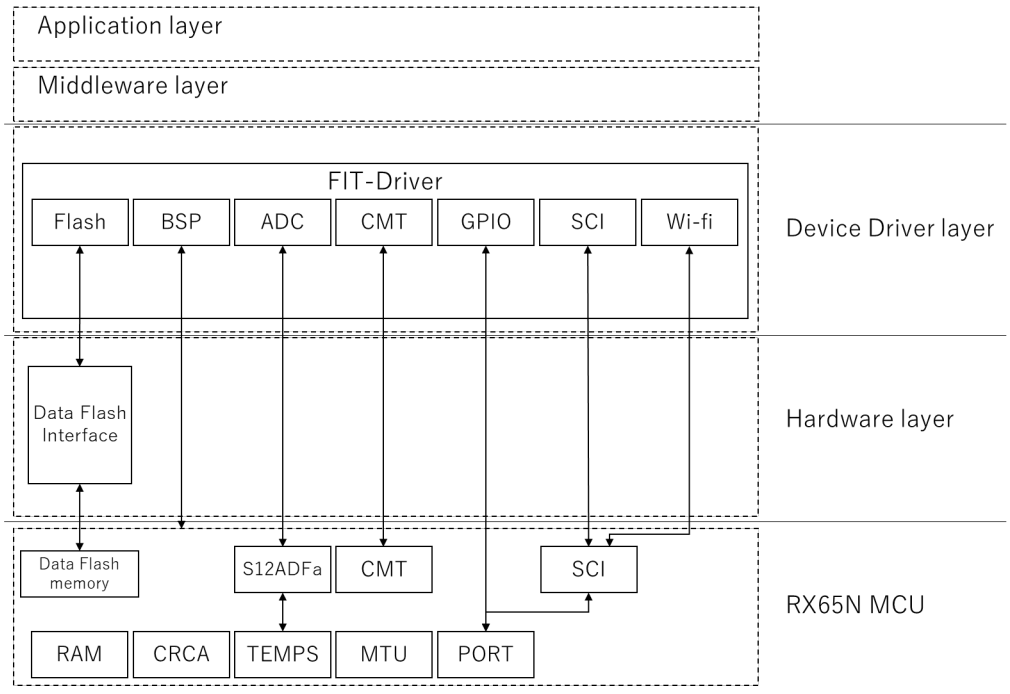

Figure 6 Overview of peripheral functions

4.3.1　周辺機能の設定

Table 4 List of peripheral function settings 1/3

item	Settings
CRC	生成多項式：CRC_32 ビット順：LSB 初期値：0x00000000 演算結果の反転はしない
MTU0	TCNT0 カウンタ設定 　TGRA0 コンペアマッチ/インプットキャプチャ 　(TGRA0 を周期レジスタとして使用) 　カウンタクロック：PLCK/4 　立下りエッジ ジェネラルレジスタの設定 　TGRA0：インプットキャプチャレジスタ 　TGRB0~F0：アウトプットコンペアレジスタ　/　100μs 入出力端子の設定 　MTIOC0A 端子入力の立ち上がりエッジでインプットキャプチャ 　(ノイズフィルタ使用) 割り込み設定 　TGRA インプットキャプチャ/コンペアマッチ割り込み許可 　優先順位：レベル 3
MTU1	TCNT1 カウンタ設定 　TGRA1 コンペアマッチ/インプットキャプチャ 　(TGRA1 を周期レジスタとして使用) 　カウンタクロック：PLCK ジェネラルレジスタの設定 　TGRA1：アウトプットコンペアレジスタ　/　419700ns 　TGRB1：アウトプットコンペアレジスタ　/　396700ns 入出力端子の設定 　無効 割り込み設定 　TGRA インプットキャプチャ/コンペアマッチ割り込み許可 　優先順位：レベル 6 　TGRB インプットキャプチャ/コンペアマッチ割り込み許可 　優先順位：レベル 6

Table 5 List of peripheral function settings 2/3

item	Settings
MTU2	TCNT2 カウンタ設定 　TGRA2 コンペアマッチ/インプットキャプチャ 　(TGRA2 を周期レジスタとして使用) 　カウンタクロック：PLCK/32 　立ち上がりエッジ ジェネラルレジスタの設定 　TGRA2：アウトプットコンペアレジスタ　/　5700μs 　TGRB2：アウトプットコンペアレジスタ　/　10190μs 入出力端子の設定 　無効 割り込み設定 　TGRA インプットキャプチャ/コンペアマッチ割り込み許可 　優先順位：レベル 3 　TGRB インプットキャプチャ/コンペアマッチ割り込み許可 優先順位：レベル 3
MTU3	TCNT3 カウンタ設定 　同期動作をしている他のチャネルのカウンタクリアでクリア 　カウンタクロック：PLCK/256 　立ち上がりエッジ ジェネラルレジスタの設定 　TGRA3：アウトプットコンペアレジスタ　/　4150μs 　TGRB3：アウトプットコンペアレジスタ　/　14000μs 　TGRC3：アウトプットコンペアレジスタ　/　15400μs 　TGRD3：アウトプットコンペアレジスタ　/　16800μs 入出力端子の設定 　無効 割り込み設定 　TGRA～TGRD インプットキャプチャ/コンペアマッチ割り込み許可 　優先順位：レベル 3
MTU4	TCNT4 カウンタ設定 　TGRA4 コンペアマッチ/インプットキャプチャ 　(TGRA4 を周期レジスタとして使用) 　カウンタクロック：PLCK/256 　立ち上がりエッジ ジェネラルレジスタの設定 　TGRA4：アウトプットコンペアレジスタ　/　18500μs 　TGRB4：アウトプットコンペアレジスタ　/　20100μs 　TGRC4：アウトプットコンペアレジスタ　/　97000μs 　TGRD4：アウトプットコンペアレジスタ　/　12800μs 入出力端子の設定 　無効 割り込み設定 　TGRA~TGRD インプットキャプチャ/コンペアマッチ割り込み許可 　優先順位：レベル 3

Table 6 List of peripheral function settings 3/3

item	Settings
PORT	出力端子 　PB1/PB0 　P15/P17/P25/P24 入力端子 　PC4/PC5/PC6/PE0/PE1/PE2 入出力端子 MTIOC0A

4.3.2　FIT モジュールの設定

Table 7 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" 　他 default 設定
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	非同期通信 パラメータチェックなし 使用チャネル：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" 　他 default 設定
ADC	Default 設定
GPIO	Default 設定
CMT	CMT 割り込み優先レベル：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　ソフトウェアコンポーネントの設定

Table 8 List of software component settings

item	Settings
デバイス選択	ボード：CloudKitRX65N (V.1.00) デバイス：R5F565NEDxFP

4.3.4　FreeRTOS Resources

FreeRTOS Resources を以下に記述します。

Table 9 List of FreeRTOS Resources(Task)

item	Name	priority	Stack size	content
Task	r_app_event_control_task	1	150	各アプリケーションタスクへの通知管理を行います。
Task	r_app_1msec_interval_task	3	150	1ms 定期処理を行います
Task	r_app_recv_dali_frame_task	1	150	受信キューの内容から各アプリケーションタスクに通知を行います。
Task	r_gui_recv_com_task	2	150	GUI アプリケーションの SCI 通信処理を行います。
Task	r_gui_report_com_task	2	150	GUI アプリケーションのコマンド処理を行います。
Task	r_btn_sw_input_task	1	150	AWS から指定されたマトリクスボタンに対してコマンドを割り当てます
Task	r_btn_recv_com_task	2	150	マトリクスボタン入力でコマンドを softDALI ミドルウェアの送信キューに設定します。
Task	r_sdmdl_transfer_control_rx_task	3	256	DALI 転送制御でプライオリティ毎に Forward フレームを並び替えて、送信キューにセットします。
Task	r_sdmdl_transfer_control_task	1	256	ソフトウェア DALI ミドルウェア送信処理を行います
Task	r_sdmdl_transfer_control_tx_task	1	256	ソフトウェア DALI ミドルウェア受信処理を行います
Task	prvAWStoDALITask	0	1024	AWS－DALI タスク間を制御します。
Task	prvDALILoggerTask	0	1024	AWS-DALI タスク間でログの制御をします

Table 10 List of FreeRTOS Resources(Message Queue)

item	Name	num	size	content
Message Queue	gs_app_qhandle_rx_frame_to_gui	4	50	受信フレームを GUI に渡すキュー
Message Queue	gs_app_qhandle_rx_frame_to_aws_for_log	5	50	受信フレームを AWS に渡す(log 用)キュー
Message Queue	gs_app_qhandle_rx_frame_to_aws_for_cmd	2	50	受信フレームを AWS に渡す(AWS コマンド用)キュー
Message Queue	gs_app_qhandle_aws_to_btn	2	50	AWS から BTN にフレームデータを渡すキュー
Message Queue	gs_app_qhandle_btn_to_aws	2	50	BTN から AWS にフレームデータを渡すキュー
Message Queue	g_sdmdl_qhandle_tx_forward_frame	10	50	アプリケーションから MDL にフレームデータを渡すキュー
Message Queue	g_sdmdl_qhandle_tx_backward_frame	5	50	アプリケーションから MDL にバックワードデータを渡すキュー
Message Queue	g_sdmdl_qhandle_rx_frame_sdmdl_to_app	10	50	MDL からアプリケーションにフレームデータを渡すキュー
Message Queue	g_sdmdl_qhandle_rx_frame_sddrv_to_sdmdl	5	36	ソフトウェア DALI ドライバから MDL にフレームデータを渡すキュー
Message Queue	g_sdmdl_qhandle_rx_resp_frame_type	5	4	MDL 内部で使用するキュー
Message Queue	g_sdmdl_qhandle_sddrv_rx_event	20	4	ソフトウェア DALI ドライバから MDL に受信イベントを渡すキュー

Table 11 List of FreeRTOS Resources(Event Flag)

item	Name	content
Event Flag	g_app_ehandle_task_information	各 APP タスクイベント管理用ハンドル
Event Flag	g_sdmdl_ehandle_sdmdl_to_sddrv	MDL に送信を通知するイベントフラグ

Table 12 List of FreeRTOS Resources (Timer)

item	Name	Time(ms)	content
Timer	g_sdmdl_thandle_transfer_rx	1000	MDL 用受信タイムアウト
Timer	gs_app_thandle_1msec_timer	1	ソフトウェアタイマ 1ms
Timer	gs_gac_sci_timer_handle	20	UART タイムアウト用
Timer	g_sys_fail_handle	534	Systemfailer 検出用

4.4　フォルダ構成

本デモプロジェクトのファイル構成を下表に示します。

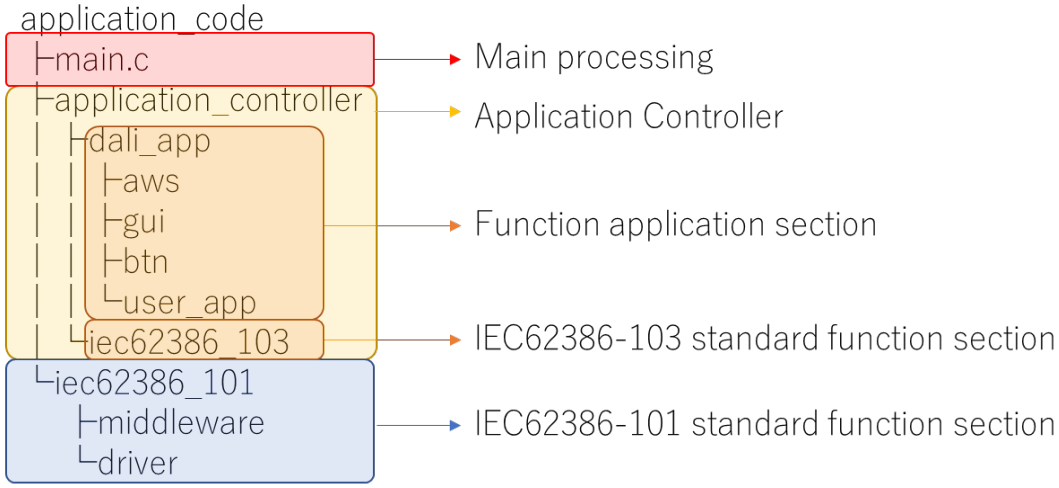

Figure 7 Folder structure

4.5　ファイル構成

本デモプロジェクトで説明する各機能のファイル一覧を示します。

Table 13 File structure 1/3

機能名	ファイル名	説明
Main 処理	main.c	本デモプロジェクトの main ファイルです。
Application Controller	r_app_api_main.c	Application Controller の main ファイルです。
	r_app_api_main.h	Application Controller の main ヘッダファイルです。
	r_app_task.c	Application Controller で使用するタスクのソースファイルです。
	r_app_task.h	Application Controller で使用するタスクのヘッダファイルです。
	r_app_queue.c	Application Controller で使用する Queue のソースファイルです。
	r_app_queue.h	Application Controller で使用する Queue のヘッダファイルです。
	r_app_df_access.c	データフラッシュアクセスのソースファイルです。
	r_app_df_access.h	データフラッシュアクセスのヘッダファイルです。
	r_app_df_access_user.h	データフラッシュアクセスに使用する定義ヘッダファイルです。
機能アプリケーション部	r_aws_control_task.c	AWS との MQTT を介した接続制御を行うソースファイルです。
	r_aws_control_task.h	AWS との MQTT を介した接続制御を行うヘッダファイルです。
	r_aws_dali_task.c	AWS Web アプリケーションで使用するタスクのソースファイルです。
	r_aws_raa_process.c	AWS 経由にてランダムアドレスアロケーションを行うソースファイルです。

	r_aws_raa_process.h	AWS 経由にてランダムアドレスアロケーションを行うヘッダファイルです。

Table 14 File structure 2/3

機能名	ファイル名	説明
機能アプリケーション部	r_gui_api.c	GUI アプリケーションのソースファイルです。
	r_gui_api.h	GUI アプリケーションのヘッダファイルです。
	r_gui_task.c	GUI アプリケーションで使用するタスクのソースファイルです。
	r_gui_task.h	GUI アプリケーションで使用するタスクのヘッダファイルです。
	r_btn_api.c	マトリクスボタンアプリケーションのソースファイルです。
	r_btn_api.h	マトリクスボタンアプリケーションのヘッダファイルです。
	r_btn_task.c	マトリクスボタンアプリケーションで使用するタスクのソースファイルです。
	r_btn_task.h	マトリクスボタンアプリケーションで使用するタスクのヘッダファイルです。
	r_mng_user.c	ユーザ実装を行えるソースファイルです。
	r_mng_user.h	ユーザ実装を行えるヘッダファイルです。
IEC62386-103 規格機能部	r_dali103_appctrl_api.a	Application Controller ライブラリのライブラリファイルです。
	r_dali103_appctrl_api.h	Application Controller ライブラリのヘッダファイルです。
	r_dali103_api.c	Application Controller アプリケーションのソースファイルです。
	r_dali103_api.h	Application Controller アプリケーションのヘッダファイです。
	r_dali103_mbanks_access.c	Application Controller で使用するメモリバンク(外部)のソースファイルです。
	r_dali103_mbanks_access.h	Application Controller で使用するメモリバンク(外部)のヘッダファイルです。
	r_dali103_mbanks_entify.c	Application Controller で使用するメモリバンク(定義)のヘッダファイルです。
	r_dali103_mbanks_entify.h	Application Controller で使用するメモリバンク(定義)のソースファイルです。
	r_dali103_mbank_access.c	Application Controller で使用するメモリバンク(内部)のソースファイルです。
	r_dali103_mbank_access.h	Application Controller で使用するメモリバンク(内部)のソースファイルです。
	r_dali103_random_seed.c	乱数に使用する Seed 値生成を行うソースファイルです。
	r_dali103_random_seed.h	乱数に使用する Seed 値生成を行うヘッダファイルです。

Table 15 File structure 3/3

機能名	ファイル名	説明
IEC62386-101 規格機能部	r_sdmdl_api.c	DALI 通信ミドルウェアのソースファイルです。
	r_sdmdl_api.h	DALI 通信ミドルウェアのヘッダファイルです。
	r_sdmdl_transfer.c	DALI 通信ミドルウェアのエラー処理用ソースファイルです。
	r_sdmdl_transfer.h	DALI 通信ミドルウェアで使用する定義ヘッダファイルです。
	r_sdmdl_transfer_list.c	DALI 通信ミドルウェアで使用するリスト制御のソースファイルです。
	r_sdmdl_transfer_list.h	DALI 通信ミドルウェアで使用するリスト制御のヘッダファイルです。
	r_sdmdl_task_transfer_control.c	DALI 通信ミドルウェアで使用するタスクのソースファイルです。
	r_sdmdl_task_transfer_control_rx.c	DALI 通信ミドルウェア受信処理のソースファイルです。
	r_sdmdl_task_transfer_control_tx.c	DALI 通信ミドルウェア送信処理のソースファイルです。
	r_sddrv_def.h	DALI 通信ドライバの内部定義ヘッダファイルです。
	r_sddrv_user.h	DALI 通信ドライバのユーザ定義関連ヘッダファイルです。
	r_sddrv_api.c	DALI 通信ドライバのソースファイルです。
	r_sddrv_api.h	DALI 通信ドライバのヘッダファイルです。
	r_sddrv_com.c	DALI 通信ドライバのアプリ側通信関連ソースファイルです。
	r_sddrv_com.h	DALI 通信ドライバのアプリ側通信関連ヘッダファイルです。
	r_sddrv_frame.c	DALI 通信ドライバの Frame デコード/エンコード処理ソースファイルです。
	r_sddrv_frame.h	DALI 通信ドライバの Frame デコード/エンコード処理ヘッダファイルです。
	r_sddrv_gpio.c	DALI 通信ドライバのポート関連ソースファイルです。
	r_sddrv_gpio.h	DALI 通信ドライバのポート関連ヘッダファイルです。
	r_sddrv_rx.c	DALI 通信ドライバの受信ソースファイルです。
	r_sddrv_rx.h	DALI 通信ドライバの受信ヘッダファイルです。
	r_sddrv_timer.c	DALI 通信ドライバのタイマ関連ソースファイルです。
	r_sddrv_timer.h	DALI 通信ドライバのタイマ関連ヘッダファイルです。
	r_sddrv_tx.c	DALI 通信ドライバの送信ソースファイルです。
	r_sddrv_tx.h	DALI 通信ドライバの送信ヘッダファイルです。

4.6　機能概要

本デモプロジェクトは、Free-RTOS のタスクのイベント通知と Queue 通知機能を使用して、Application Controller を動作させます。タスクは Application Controller の機能ごとに分かれており、各機能で複数のタスクを使用します。

ただし、DALI 通信ドライバはタスクを使用せずにタイマ割り込みを使用して制御します。

デモプロジェクトの基本動作を下図に示します。

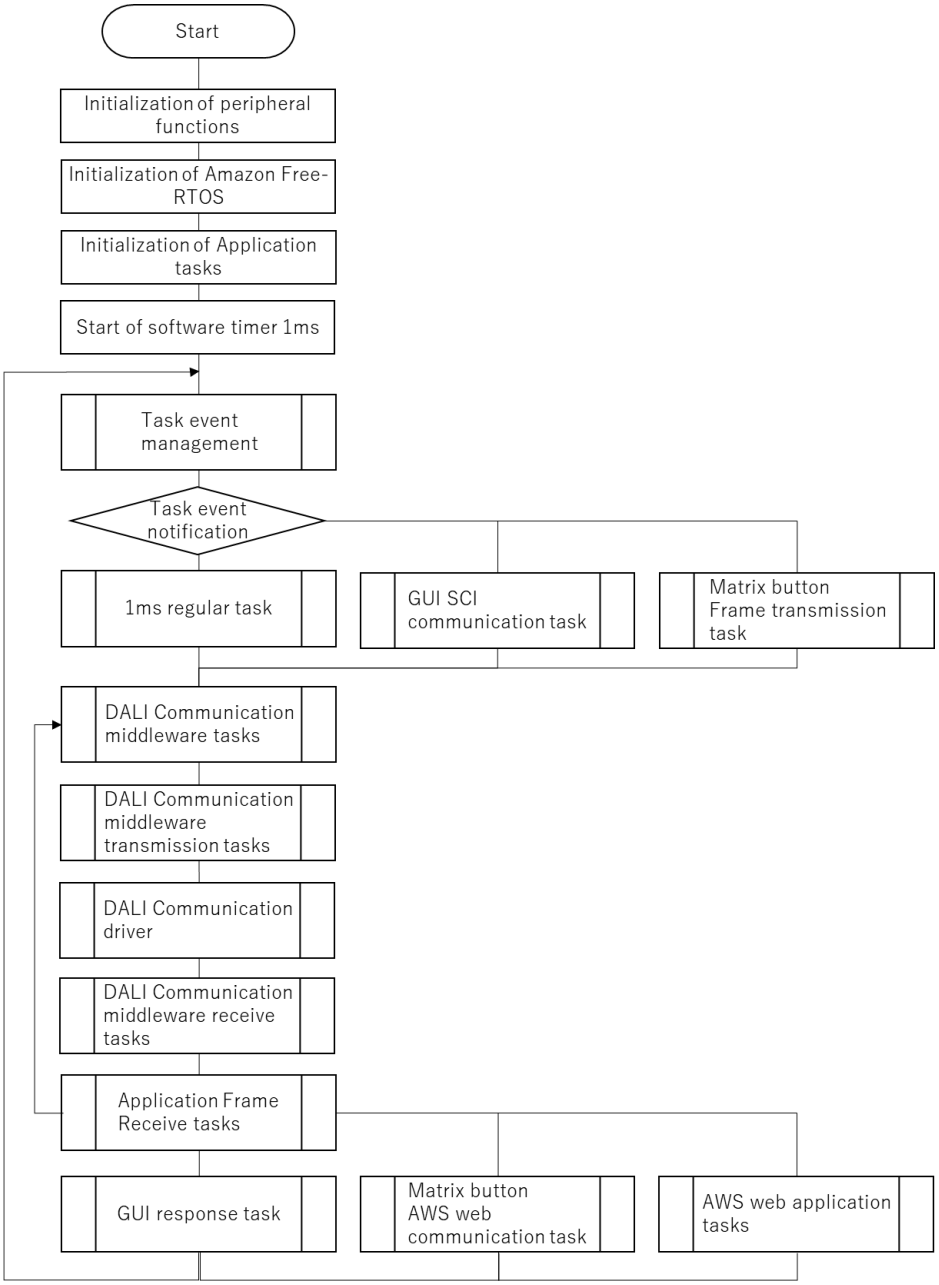

Figure 8 Operation flow diagram of each application

- Application Controller タスク：Application Controller 動作を行うタスクです。
  ー1ms 定期タスク、Frame 受信タスク

- GUI アプリケーションタスク：DALI マスタコントローラ GUI との通信を行うタスクです。
  ーGUI SCI 通信タスク、GUI 応答タスク

- AWS Web アプリケーションタスク：AWS Web アプリケーションとの通信を行うタスクです。
  ーAWS DALI 通信タスク、Logger タスク

- マトリクスボタンアプリケーションタスク：マトリクスボタンに関する動作を行うタスクです。
  ーFrame 送信タスク、AWS Web 通信タスク

- DALI 通信ミドルウェアタスク：DALI 通信ドライバとの制御、Frame 管理と通知を行うタスクです。
  ーDALI 通信ドライバ制御タスク、Frame 送信タスク、Frame 受信タスク

- DALI 通信ドライバ：周辺機能を使用して Frame の送受信を行います。
  ーDALI 通信制御

4.6.1　IEC62386-101 規格規定部

本デモプロジェクトに搭載する DALI 通信の機能は、DALI 規格 IEC62386-101 ed2.1 の Single-master application controller、4.6.5 章 Multi-master application controller の使用を可能とした Receiver/Transmitter 機能になります。

1. DALI 通信ドライバ

本ドライバは、DALI-2 の通信プロトコル(各 Frame 送受信)をソフトウェアで実現するための、DALI 通信プロトコル確認用ドライバ(以降、softDALI ドライバと記載)です。本ドライバの使用については、Control Device の Application controller を対象としています。
受信については、DALI 通信によって受信した Forward Frame が、DALI-2 の Receiver timing 規格に従っていることを確認し、受信 Frame の受け入れ・破棄の判断を行います。
送信については、DALI 通信によって送信する Forward Frame、Backward Frame を Transmitter timing 規格に従って送信します。Multi-master 動作時の Forward Frame 送信については優先度に従って送信を行います。また、他の Control Device からの送信 Frame との衝突の回避、検知、回復の Collision 処理を実装します。
Proprietary 機能を有し、有効ビット上限は 256bit とします。(有効無効指定あり)

Table 16 softDALI driver processing overview list

Soft DALI ドライバ処理概要
Receiver ・8bit　Backward Frame の受信 ・16bit、24bit、32bit　Forward Frame の受信 ・64bit、128bit、256bit　Proprietary Forward Frame の受信(ユーザ変更可能：3 種以内) ・Bit timing Violation(受信 bit タイミング異常)の検出 ・Receiver Frame Size Violation(受信サイズ異常)の検出 ・Stop condition の確認 ・Twice Frame の確認 Transmitter ・8bit　Backward Frame の送信 ・16bit、24bit、32bit　Forward Frame の送信 ・64bit、128bit、256bit　Proprietary Forward Frame の送信 ・送信ビットタイミングでの送信 ・送信時 Collision 回避(Multi-master 時のみ) ・送信時 Collision 検出(Multi-master 時のみ) ・Collision 発生時の復帰動作(Multi-master 時のみ) ・Collision 発生時の再送信(Multi-master 時のみ) ・優先度による送信タイミング調整(Multi-master 時のみ)

本アプリケーションノートでは、主に DALI 通信ドライバの以下の部分について記載します。

- Collision 処理
- Twice Frame
- System failure
- 機能制限

（1） SoftDALIドライバ実現方法

(a)ドライバ内呼称定義

本ドライバ内での使用する呼称を以下に記します。

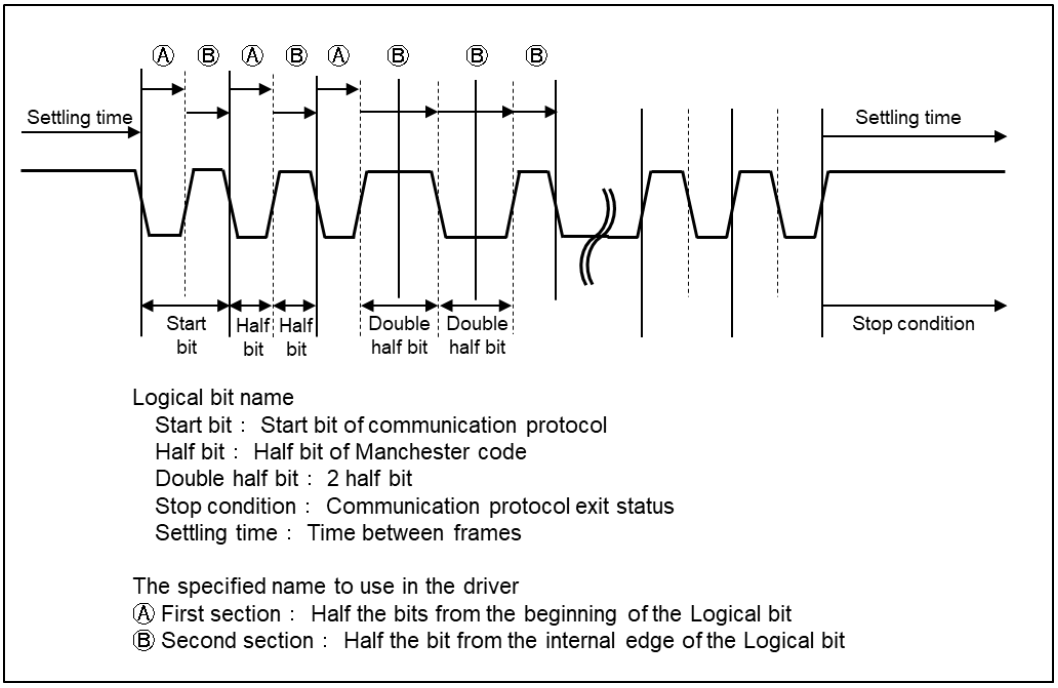

Figure 9 The specified name to use in the driver

(b)受信 Frame データ判断

本ドライバで Frame デコード時に設定するデータの取得方法を以下に示します。

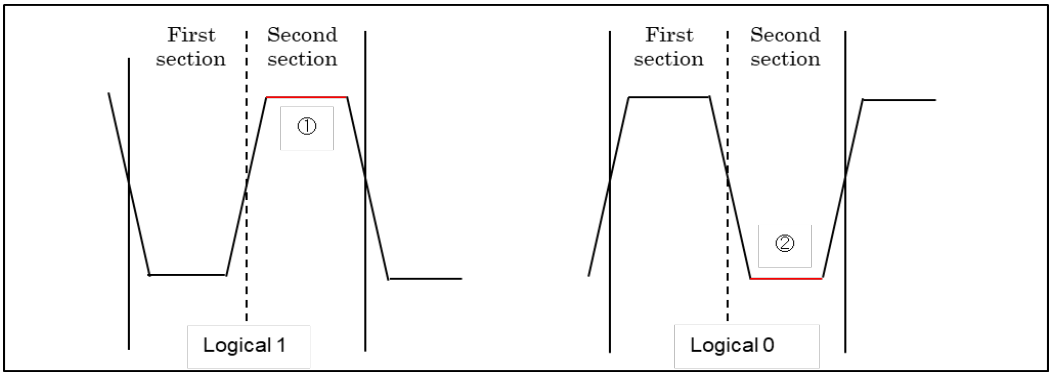

Figure 10 How to determine half bit frame data

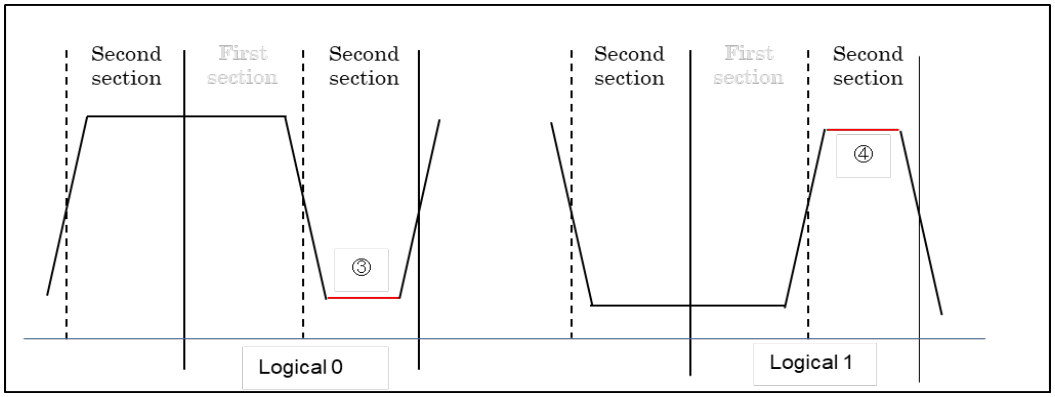

Figure 11 How to determine Double half bit frame data

First section 割り込み発生時のピンの状態を確認する。
High 状態(①)の場合は“1”、Low 状態(②)の場合は“0”として Frame データを作成する。
Double half bit 割り込み発生時の場合は、Double half bit 時のみピンの確認をする。
Low 状態(③)の場合は“0”、High 状態(④)の場合は“1”として Frame データを作成する。

Table 17 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”

その他 Frame デコード条件

- First bit はデータとして保存しない。
- Frame は 8bit、16bit、24bit その他（Reserve の 20bit、32bit）、Proprietary 指定サイズとし、それ以外を受信した場合は、受信サイズエラーとする。
- 受信が上記サイズ以外で Stop condition を受信した場合、受信サイズエラーとする。

(c)送信ビットタイミング

- Single-master Transmitter bit timing 定義

本ドライバ内で使用する Single-master 時の送信ビットタイミング、及び呼称につい以下に記します。

Table 18 Single-master Transmitter bit timing

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

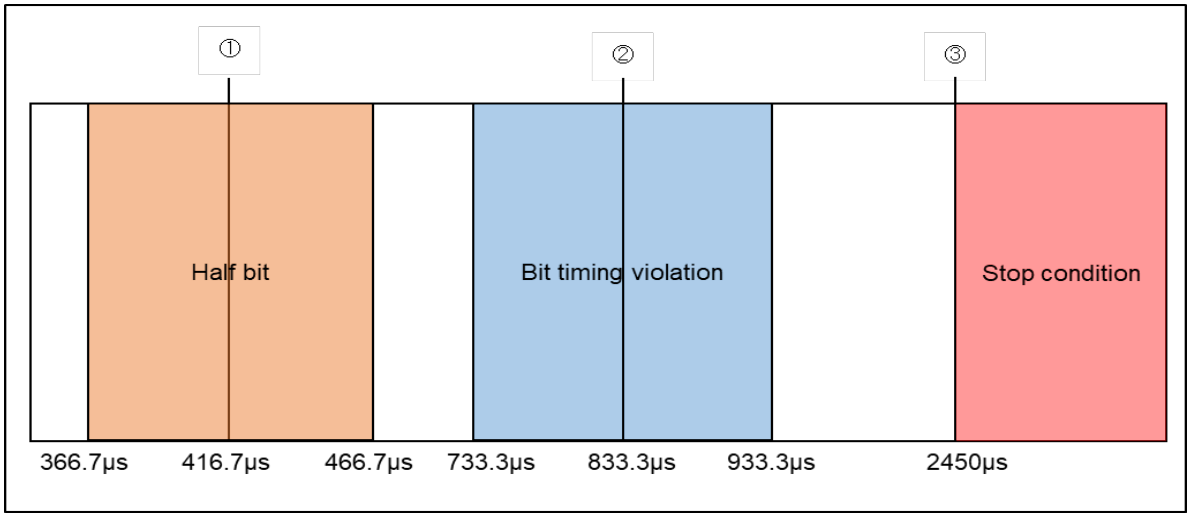

Figure 12 Single-master Transmitter bit timing

- Single-master Transmitter Settling time 定義

本ドライバ内で使用する Single-master 時の Settling time について下表に記します。

Table 19 Single-master Transmitter settling time timing

Minimum	Typical	Maximum	Description
5.5ms		10.5ms	Backward Frame
13.5ms		75.0ms	Forward Frame

Backward Frame は Stop condition 計測開始時からの継続時間となります。BUS State が途中で Active state に変わっても計測は継続します。Backward Frame は Collision の発生は無視して必ず送信しなければなりません。Backward Frame の送信は 5.5ms~10.5ms 内でのみ可能です。

Forward Frame は 13.5ms 以降であれば送信可能です。

- Multi-master Transmitter bit timing 定義

本ドライバ内で使用する Multi-master 時の送信ビットタイミング、及び呼称について以下に記します。

Table 20 Multi-master Transmitter bit timing

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

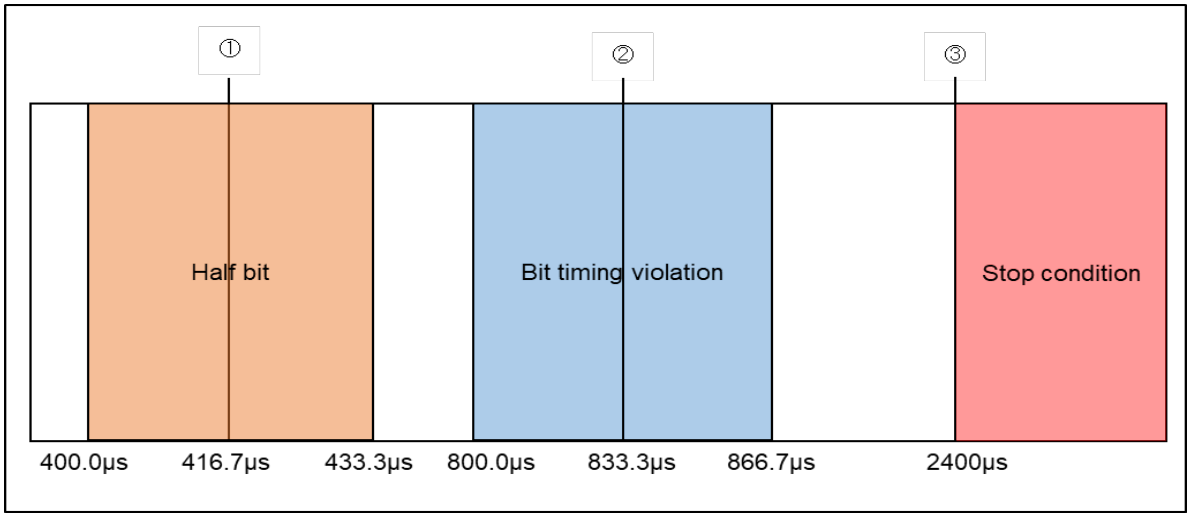

Figure 13 Multi-master Transmitter bit timing

- Multi-master Transmitter Settling time 定義

本ドライバ内で使用する Multi-master 時の Settling time について下表に記します。

Table 21 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)

Backward Frame は Stop condition 計測開始時からの継続時間となります。BUS State が途中で Active state に変わっても計測は継続します（Stop condition 計測済みの場合のみ）。

Backward Frame は Collision の発生は無視して必ず送信しなければなりません。Backward Frame の送信は 5.5ms~10.5ms 内でのみ可能です。

Forward Frame は priority に合わせた Settling time を使用します。

送信は各範囲内の任意のタイミングで行います。固定すると、同じソフトを使った機器同士のタイミングがそろってしまい、Collision 発生の原因になります。

Forward Frame は Minimum 時間以降で送信可能です。

(d)Transmitter bit timing 測定

本ドライバで DALI-2 通信の Transmitter bit timing を測定する方法について以下に記します。

- Transmitter bit timing 測定方法

本ドライバでは送信ビットタイミングを以下の方法を用いて測定します。

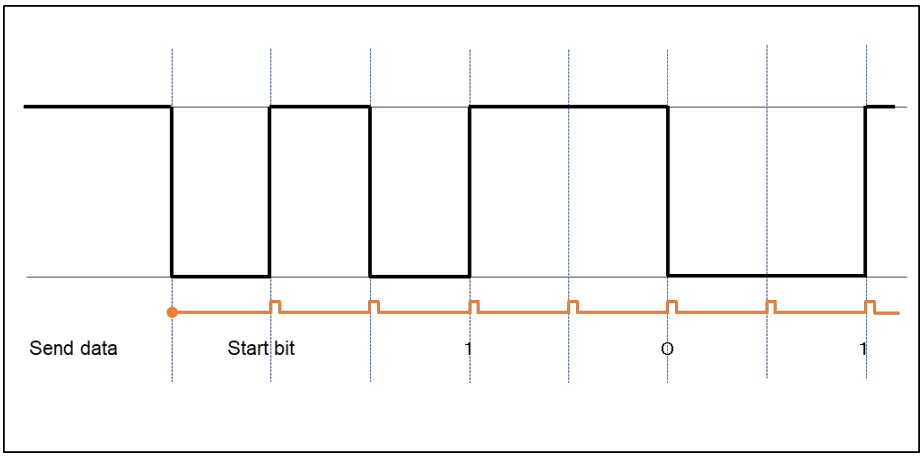

Figure 14 Transmitter bit timing acquisition timing

本ドライバでは、Transmitter bit timing の測定に、以下のタイマを使用します。

- 周期カウント動作タイマを使用し、Half bit 値をカウンタ値として設定します。
  (デバイスごとにタイマは変わります。本仕様は RX65n を対象にしています。)

周期カウントタイマは開始時の割り込みは発生しないので、タイマ開始タイミングで送信ピンに Low 設定します。その後は周期タイマの割り込みを使用して送信ピンに High/Low を設定していきます。

送信周期は Single-master と Multi-master では違うので、マスタのタイプに合わせて送信周期を設定します。

(e)Collision

- Collision 処理

本ドライバは Control device での使用を対象としています。

Control device が Multi master Transmitter である場合、Forward Frame 送信時に Collision 処理を行う必要があります。

Collision 処理には以下に示す 3 点があります。

- Collision 回避
- Collision 検知
- Collision 回復

- Collision 回避

Multi-master transmitter が Forward Frame を送信する前に Collision 回避のための送信タイミングに関する規格があります。

- Multi-master transmitter であること
- 送信する Forward Frame の優先順位で指定された Settling time 以上の時間が経過していること
  (Settling time は priority 毎の指定範囲内で機器毎にランダムのタイミングを作成すること)
- BUS が Idle state であること(Active state になることで Settling time は 0 となる)

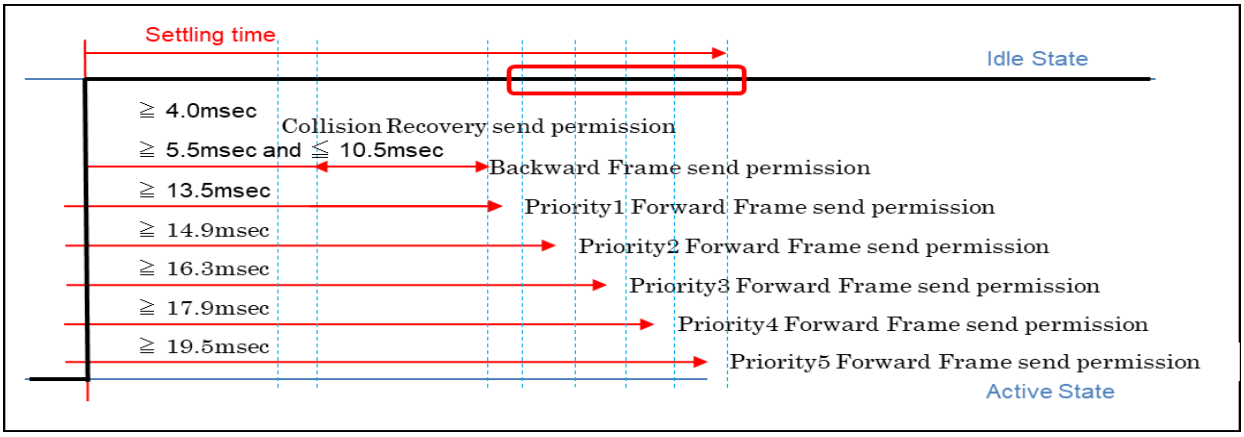

Figure 15 Collision avoidance transmission timing

送信 Forward Frame の指定 Priority に設定された Settling time が経過した時点で、その Forward Frame は送信可能となります。また、送信開始時に BUS の State が Idle State であることを確認する必要があります。本ドライバでは Idle State であるかの確認は、Receiver の受信割り込みで得られた State によって状態を判断します。

Collision Recovery が発生している場合、BUS に繋がる Device は Control Gear も含めて Forward Frame は Collision 発生により Bit violation が発生するため、Backward Frame の送信は発生しないものと考えられます。Backward Frame に関しては Forward Frame より高い優先度で送信をするため、Forward Frame の送信可能 Settling time の経過は発生しません。

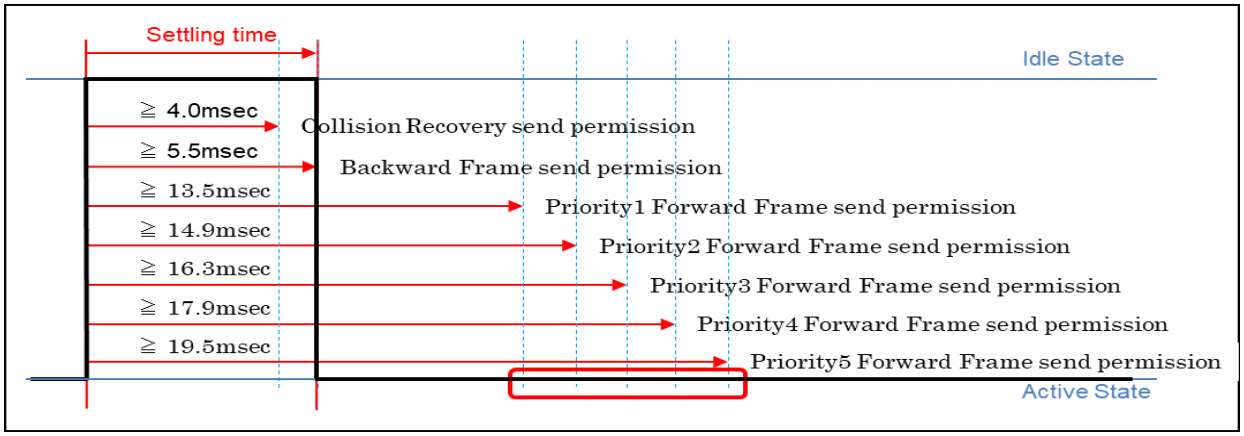

Figure 16 Collision avoidance transmission timing

Collision 回避処理に使用する Settling time のタイマは、Receiver 側で開始します。

送信側では、Settling timer 用のタイマが起動している事を確認し、タイマカウンタ値から経過時間を確認して処理を行ってください。

ただし、Backward Frame は settling time が 5.5ms 以内で終了しても、みなし時間(5.5ms)で送信を行ってください。Backward Frame は他の Application Controller との Collision については検知を行いません。

- Collision 検知

Collision 検知は Forward Frame の送信時に行いますが、検知するのは BUS 上のデータを確認する必要があるため Receiver にて行います。
Receiver 側では Forward Frame の送信時にのみ行うので、送信を行っている事を認識する必要があります。
Collision 検知を行う条件を以下に示します。

- Multi-master transmitter であること
- Forward Frame の送信中であること

以上の条件の場合に Collision 検知を行います。
Collision は送信タイミング等により発生パターンが異なります。また、発生パターンにより回復方法も異なります。
Collision 検知には以下のパターンが存在します。

- BUS State のビット幅が Destroy area ではない場合の Collision 検知
- BUS State のビット幅が Destroy area である場合の Collision 検知

以上のパターンで Collision 検知を行います。

本ドライバで Collision 検知に使用するビットタイミングを以下に示します。

Table 22 First section bit timing

Minimum	Typical	Maximum	Description
		<100µs	Grey area
100µs		356.7µs	Destroy area
>356.7µs	①365µs	400.0µs	Grey area
400.0µs		433.3µs	Valid half bit
>433.3µs	②467µs	<476.7µs	Grey area
476.7µs			Destroy area

First section では①以上③以下の範囲を Half bit と判断し、その他は Destroy area と判断します。
476.7µs 以上は Active state のみ適用と規格書には記載されていますが、Stop condition 経過までの間に受信があった場合、本ドライバでは Idle state でも Destroy area と判断します。

Table 23 Second section bit timing

Minimum	Typical	Maximum	Description
		<100µs	Grey area
100µs		356.7µs	Destroy area
>356.7µs	①365µs	400.0µs	Grey area
400.0µs		433.3µs	Valid half bit
>433.3µs	②467µs	<476.7µs	Grey area
476.7µs		723.3µs	Destroy area
>723.3µs	③762µs	<800.0µs	Grey area

800.0µs		866.7µs	2 valid half bits
>866.7µs	④905µs	943.3µs	Grey area
943.3µs			Destroy area

Second section では①以上③以下の範囲を Half bit、③以上④以下の範囲を Double half bit と判断し、その他は Destroy area と判断します。

943.3µs 以上は Active state のみ適用と規格書には記載されていますが、Stop condition 経過までの間に受信があった場合、本ドライバでは Idle state でも Destroy area と判断します。

- BUS State のビット幅が Destroy area ではない場合の Collision 検知

BUS State のビット幅に Destroy area は存在しないが、送信している State と Bus State に相違がある場合、Collision が発生したと判断します。
このパターンでは、受信割り込みによるビット幅測定だけでは Collision を検知できません。Collision 発生を検知するためには State の比較を行います。

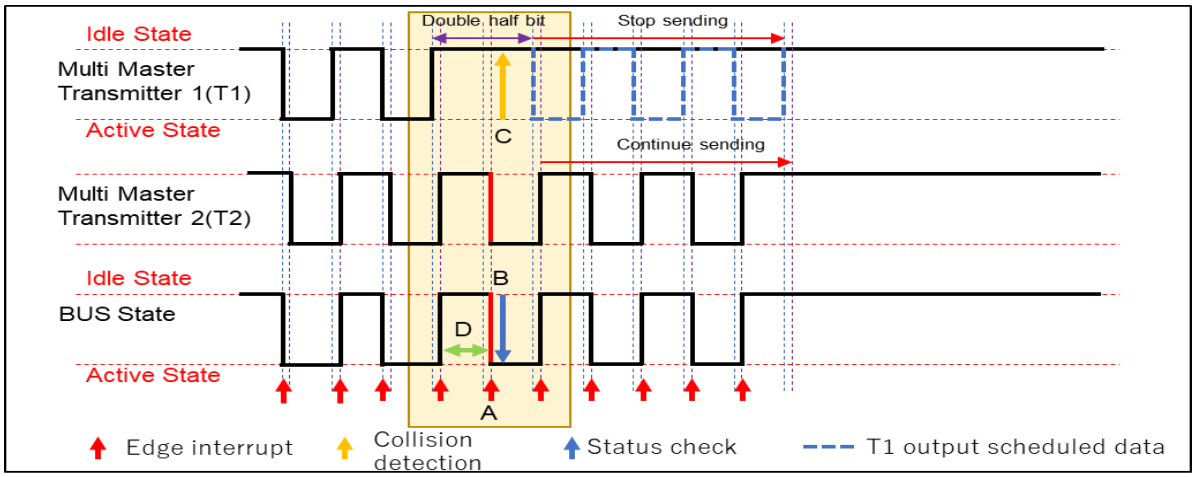

Figure 17 Collision detection when not in Destroy area

上記「Figure 17 Collision detection when not in Destroy area」では、Destroy area ではない場合の Collision 検知を示しています。
Transmitter1(以下 T1)を自機とし、T1 と Transmitter2(以下 T2)で同時にデータ送信を開始したと想定します。

ある時点で T1 は Double half bit で、T2 は Half bit で送信を行います。
同時に Idle State と Active State を送信した場合 Active State が優先されるので、BUS 上では T2 が送信した波形が流れます。そのため、T1、T2 ともに受信エッジ割り込み(A)が発生します。
T1 では、割り込み(A)が発生した時点での送信 State は Idle State、一方 BUS State(B)は Active State となっており、State の相違を認識し T1 は Collision の発生を検知(C)します。

この様なパターンでは受信割り込み(A)で検出したビット幅(D)は Half bit の正常範囲となり、bit 幅からの Collision 検知は行えません。

T1 は Collision を検知した時点で送信を停止しますが、T2 は送信 State と BUS State は同じであるため、Collision 発生とは認識せず、送信を継続します。

- BUS State のビット幅が Destroy area の場合の Collision 検知

BUS State のビット幅に Destroy area が存在した場合、Collision が発生したと判断します。このパターンでは、受信割り込みによるビット幅測定で Collision を検知します。

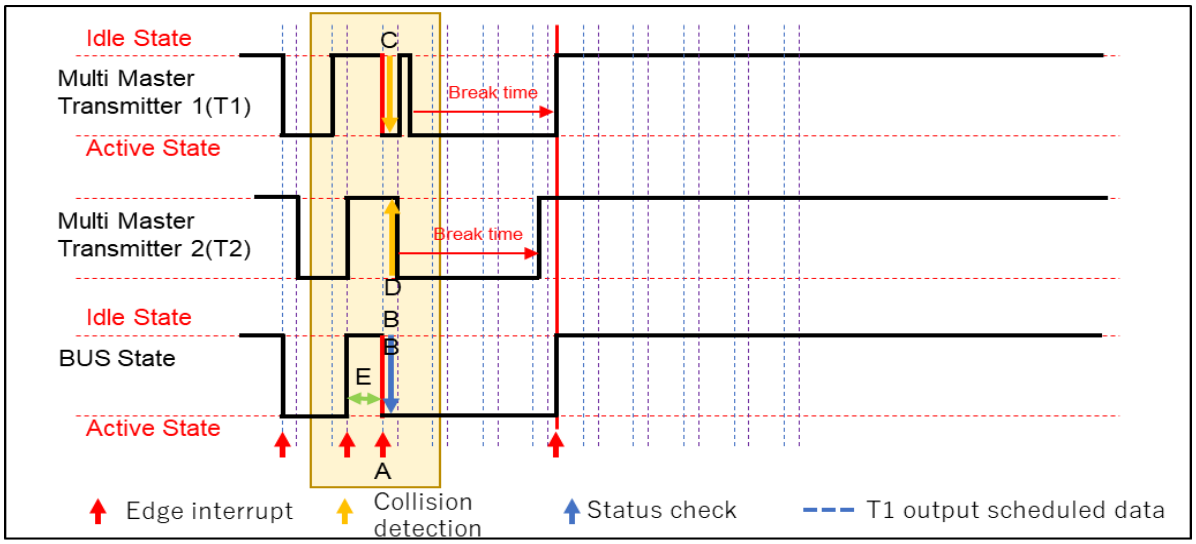

Figure 18 Collision detection in case of Destroy area

上記「Figure 18 Collision detection in case of Destroy area」では、Destroy area が発生した場合の Collision 検知を示しています。
T1 と T2 で若干の時差をもってデータ送信を開始したと想定します。

ある時点で T1 は Active State に切り替え送信を行います。Active State が優先されるので、BUS 上では T1 が送信した波形が流れます。そのため、T1、T2 ともに受信エッジ割り込み(A)が発生します。
T1 では、割り込み(A)が発生した時点での送信 State は Active State、BUS State(B)は Active State で State の相違はありません。受信割り込み(A)で検出したビット幅(E)は Destroy area となっているため T1 は Collision の発生を検知(C)します。
T2 は受信割り込み(A)の State の相違、及びビット幅(E)の Destroy area の両方で Collision の発生を検知できます。
T1、T2 とも Collision を検知した時点で送信を停止します。

上記「Figure 18 Collision detection in case of Destroy area」の検知とは異なり、割り込みが発生しない場合が存在します。
Idle State での割り込み未発生は Stop condition とみなされますが、Active State での未発生は Destroy area を検知しなくてはなりません。

割り込みが発生しないパターンについて、以下に示します。

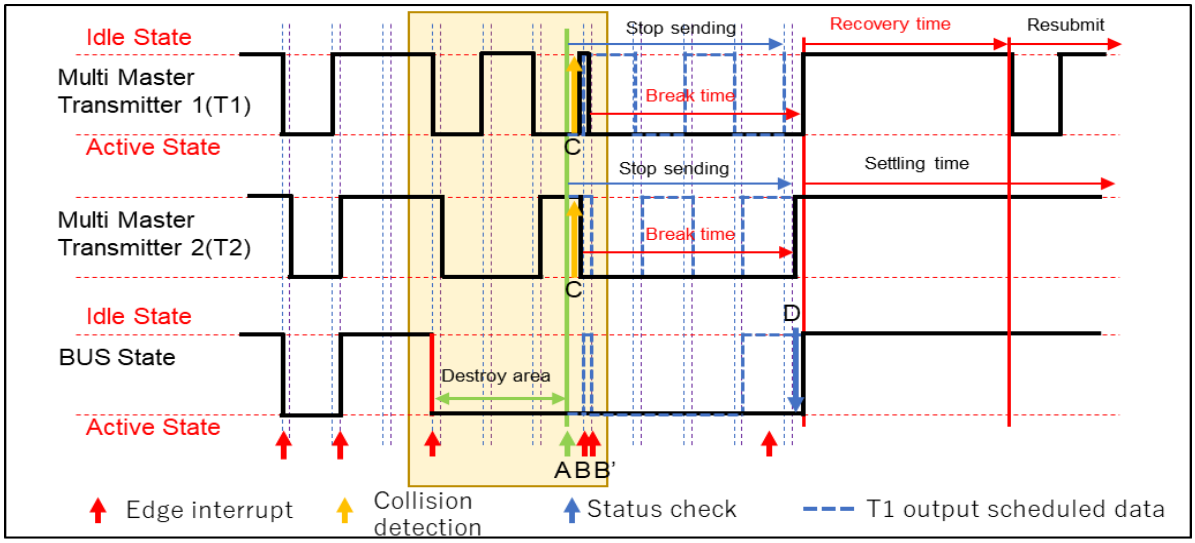

Figure 19 Collision detection in case of Destroy area

上記「Figure 19 Collision detection in case of Destroy area」では、受信割り込みが発生しない条件での Destroy area 発生 Collision 検知を示しています。この方法は BUS State が Active State 時にのみ有効です。

上記の様に T1 と T2 で Active State が交互に続くと、BUS State が Destroy area 以上に Active State が保持される場合があります。この場合、受信割り込みが発生しないため T1、T2 で Collision の検知ができません。
Transmitter では立ち下げ時の受信割り込み時（Active State 移行時）に Destroy area 検知用タイマを起動させ、タイマの割り込み(A)が発生し、(C)にて Collision の発生を検知します。
この検知の方法は BUS State が Active State の時のみ有効で、Idle State 時は Stop condition の検知用として別タイマを使用します。

タイマで使用する Destroy area 検知用タイマ値は Second section 用の最大の Destroy area 開始値(943.3µs)を使用することとします。これは Active State で受信割り込みの最大の待機時間を示します。これにより、複数 Transmitter による BUS State の Active State の継続を検知します。
また、Transmitter は Destroy area による Collision 発生を検知した場合、必ず 1.2ms の Active State を発生させなければなりません。もし初めで検知できなかった場合にも、この Break time で Collision を検知できます。

- Collision 回復

Collision 回復は Collision 検知に Destroy area の発生が含まれているかいないかで回復手順が異なります。
Collision 検知には以下のパターンが存在していました。

- BUS State のビット幅が Destroy area ではない場合の Collision 検知
- BUS State のビット幅が Destroy area である場合の Collision 検知

Destroy area の発生は、BUS 上にある全ての Transmitter が Collision 発生を検知しています。また、Destroy area が発生していない場合は、Collision 発生を検知していない Transmitter が存在します。

- Destroy area ではない場合の Collision 回復

BUS State のビット幅に Destroy area は存在しない場合の回復方法を以下に示します。

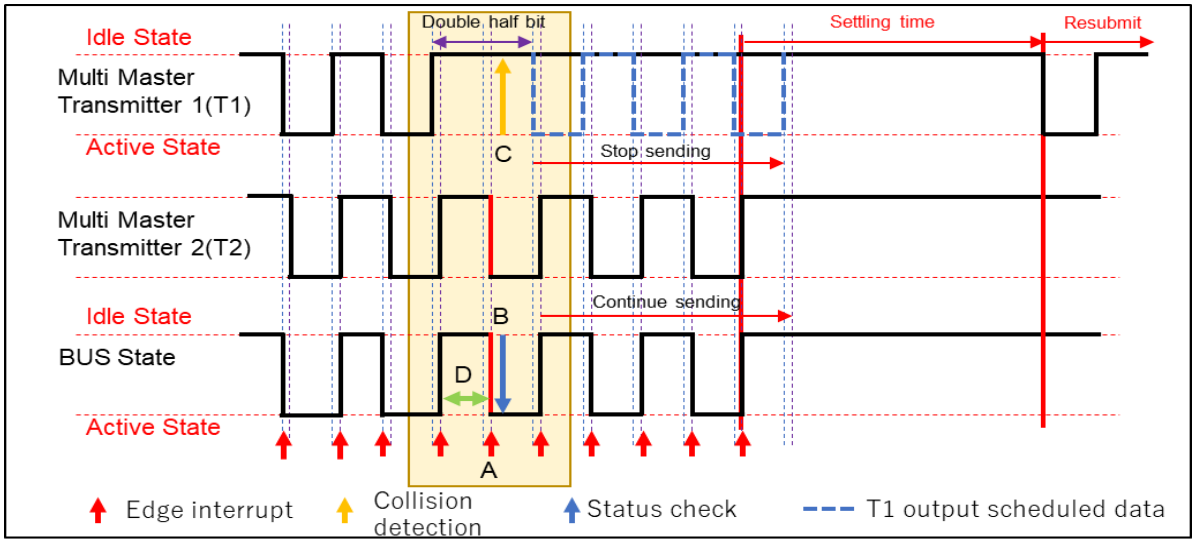

Figure 20 Collision recovery when not in Destroy area

上図「Figure 20 Collision recovery when not in Destroy area」では、Destroy area が発生していない場合の Collision 回復処理を示しています。
上図の場合、T1 は(C)にて State の異常を確認し Collision を検知、送信を停止します。T2 は State の異常は発生していないため Collision 発生を検知せず、送信を継続します。

Destroy area が発生していない場合の Collision 発生時の処理を以下に示します。

- 送信を停止する。
- 受信を継続する。(正常データと同じ受信処理を行う)
- 正常データ受信後の送信手順と同じ方法で再送信を行う。

T1 は(C)にて Collision を検知し送信を停止します。T2 は Collision 発生を検知していないので、送信を継続しています。そのため、BUS 上では正常に T2 の送信処理が行われていることになります。T1 は送信を停止することにより、BUS を T2 に渡し、通常受信動作に移行します。

T1 は送信を停止しているので Collision 検知は行はず、通常受信動作を行い、Stop condition の検知後 Settling time の経過を待ちます。

Settling time の経過を確認後、送信停止したデータの再送信を開始します。再送信は途中からでなく先頭から開始します。

- Destroy area の場合の Collision 回復

Destroy area を検知した場合の Collision 回復方法を以下に示します。

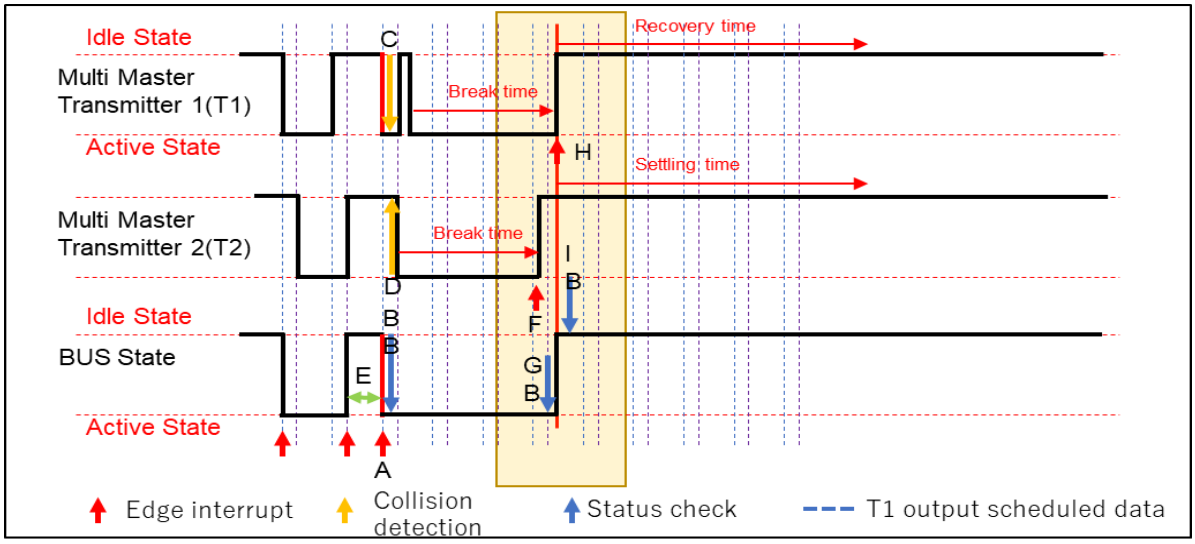

Figure 21 Collision recovery in Destroy area

上記「Figure 21 Collision recovery in Destroy area」では、Destroy area が発生した場合の Collision 検知を示しています。

Destroy area 発生により T1 と T2 で Collision を検知します。

Destroy area を含む Collision を検知した場合、Transmitter は 450μs 以内に Break time として BUS を強制的に Active State にしなければなりません。これは、Collision を検知できていない Transmitter に Collision を検知させるために行います。

上図では、T2 が先に Break time を発生させており、Break time の終了も T2 が若干早いのが分かります。Transmitter は自分の作った Break time が終了すると BUS State を確認します。T2 の場合、(G)で BUS State を確認し、Active State であることを確認します。T1 の場合は(I)で確認し Idle State であることを確認します。

Break time 終了時に確認した BUS State が Idle State であった T1 に Collision Recovery による再送信の優先権が与えられます。

- T1 の再送 settling time は Recovery time を使用します。
- T2 の再送 settling time は送信データの優先順位に定められた時間を使用します(Figure 13 参照)

Transmitter はこのルールに従って Collision の回復を行います。

Table 24 Collision Recovery Timing

Minimum	Typical	Maximum	Description
1.2ms		1.4ms	Break time
4.0ms		4.6ms	Recovery time
Transmitter は Collision 回避のため Recovery time の最小、最大時間の間の任意の時間で送信を始めてください。			

(f) Twice Frame

Twice Frame は、同データを続けて規定時間内に送信することにより成立します。
ドライバ内では、送信については通常通りの送信方法で規定時間内での送信は可能となるので、特別な処理は行いません。受信については前回受信 Frame との比較、及び下記の規定 Settling time 内での受信により Twice Frame であると判断します。
ただし、本ドライバ内では Frame 内容の確認は行わないため、Twice Frame ではないものも Twice と判断し、最終判断はアプリケーション側で行うものとします。
本ドライバで使用する Twice Frame の受信規定時間を以下に示します。

Table 25 Twice Frame Timing

Minimum	Typical	Maximum	Description
	①Stop condition 値	2.4ms	Grey area
2.4ms		94ms	Twice Forward Frame
>94ms	②100ms	<105ms	Grey area
本ドライバでは①Stop condition 値から②100ms までの Settling time を Twice Frame 有効期間とします。			

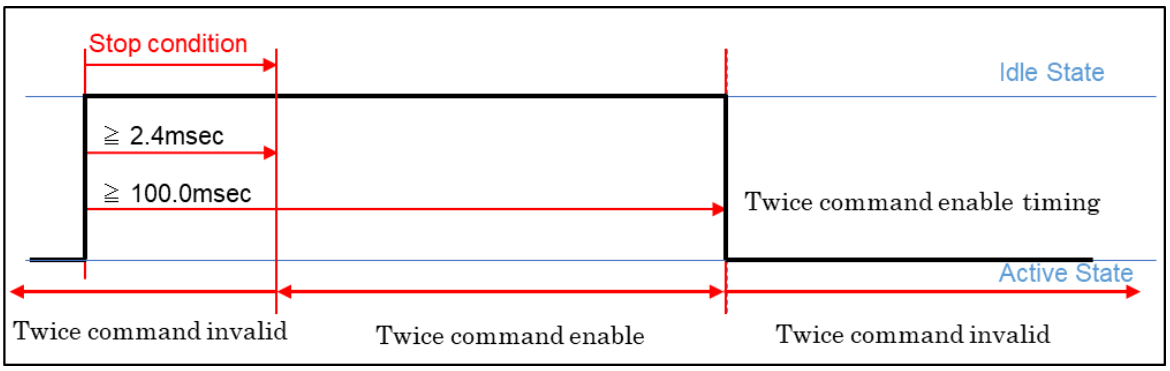

Figure 22 Twice frame effective timing

(g)System failure

適用ボードでは、DALI 回路の確認により System failure の検知を行います。
System failure は、500ms 以上の Active State 継続で発生となります。
計測は System failure 用タイマを立下りエッジ受信時に開始し、Active state 期間を計測します。
System failure タイマ割り込みにより発生を検知し、アプリ側に通知します。

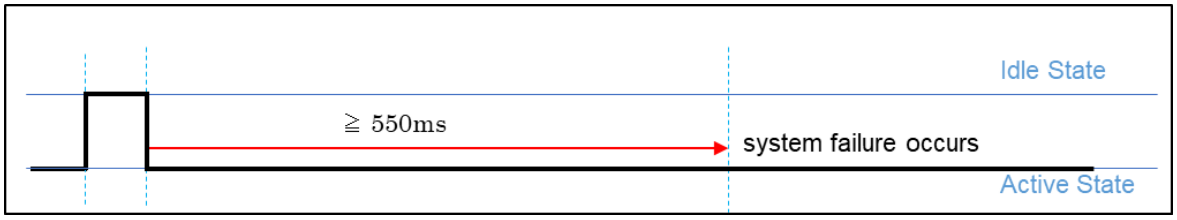

Figure 23 System failure timing

Table 26 System failure Timing

Minimum	Typical	Maximum	Description
>550ms			System failure

(h)機能制限

本ドライバは、一部の機能を制限しております。制限内容を以下に示します。

- Proprietary Frame 定義数

DALI 規格では、Proprietary Frame の指定数に制限はありませんが、本ドライバでは最大 3 種の Frame size に制限します。

Default 値は以下の通り

- 64bit
- 128bit
- 256bit

Proprietary Frame の取り扱いサイズについては、3 種類以下であれば 1 種類のみ、2 種類、または 3 種類というように種類数の変更を可能とします。
また Proprietary Frame サイズを上記 Default 値から変更することも可能とします。

変更はユーザ開示のヘッダファイル内の Define 値を修正して行います。

(2) SoftDALI ドライバ

(a)ステータス遷移

本ドライバの内部状態遷移の遷移図を以下に示します。

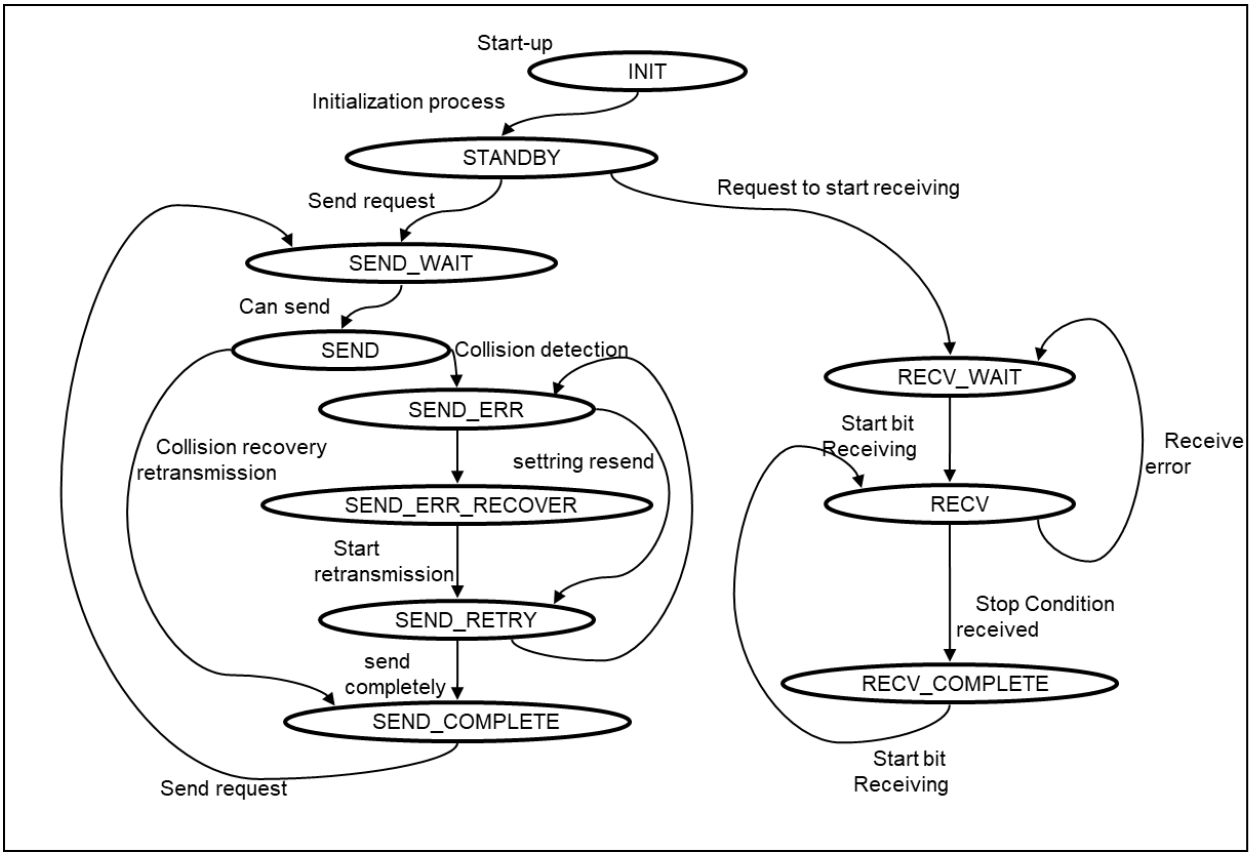

Figure 24 SoftDALI state transition diagram

- SDDRV_STATE_INIT
  本ドライバ起動時の状態です。

- SDDRV_STATE_STANDBY
  本ドライバの初期化処理が正常終了し、送受信のペリフェラルの準備が完了した状態です。
  送信要求、受信要求を受け付けることが可能です。本状態時に受信開始要求を行うことにより受信処理が開始されます。受信開始処理要求により、受信側の状態遷移が開始され、受信待ち状態(SDDRV_STATE_RECV_WAIT)となります。

- SDDRV_STATE_SEND_WAIT
  送信要求を正常に受け付け、送信開始条件が整うのを待っている状態です。
  Settling time、BUS State を確認し送信可能になるのを待ちます。
  送信開始条件が整った場合、送信中状態(SDDRV_STATE_SEND)に遷移します。

- SDDRV_STATE_SEND
  送信可能状態となり、送信を開始し送信を行っている状態です。
  受信側で Collision を検知するか、正常に Stop condition を受信するまで本状態は継続します。

- SDDRV_SEND_ERR
  送信中に Collision 等のエラーを検知した状態です。
  本状態中に遷移後、即座に送信を停止し、回復動作の選択を行います。Destroy area を検知した場合は Collision 回復状態(SDDRV_STATE_SEND_ERR_RECOVERY)に遷移し、Destroy area を検地していない場合は通常の priority に準じた Settling time を設定し再送信状態(SDDRV_STATE_SEND_RETRY)に遷移します。

- SDDRV_STATE_SEND_ERR_RECOVERY
  Destroy area を検知した場合の回復動作を行う状態です。
  遷移後、即座に Break time を発生させます。Break time 終了時に BUS State を確認し Active State の場合は通常の priority に準じた Settling time を設定し、Idle State の場合は Collision Recovery time を Settling time に設定したうえで、再送信状態(SDDERV_STATE_SEND_RETRY)に遷移します。

- SDDRV_STATE_SEND_RETRY
  再送処理を行っている状態です。
  設定された Settling time の経過、BUS State を確認し送信を開始します。受信側で Collision を検知するか、正常に Stop condition を受信するまで本状態は継続します。

- SDDRV_STATE_SEND_COMPLETE
  送信が完了した状態です。
  受信側で正常に Stop condition を受信した場合に本状態に遷移します。本状態で送信完了がアプリケーションに返されます。

- SDDRV_STATE_RECV_WAIT
  受信の待ち状態です。

受信開始要求、受信データエラー発生時に本状態に遷移します。本状態では受信データが存在しないことを意味しています。

- SDDRV_STATE_RECV
  Start bit を受信し、受信が開始されている状態です。

- SDDRV_STATE_RECV_COMPLETE
  Stop condition を受信し、受信が正常に終了した状態です。
  本状態で受信完了がアプリケーションに返されます。

(b)タイマ

本ドライバで使用するタイマの説明を以下に示します。

Table 27 List of timers used by SoftDALI driver

Timer name	Timer type	Timer value	Description
送信タイミングタイマ	周期カウント (TIMER:MTU1)	416μs	ビット送信タイミング用タイマです。 一定周期で割り込みを発生させ、ビット送信タイミングを作成します。 Start bit 設定時にタイマを開始、送信終了時、Collision 検知時に停止します。
受信割り込みタイマ	インプットキャプチャ (TIMER:MTU0)	ー	受信開始要求時に開始します。 停止操作はありません。
タイミング計測用タイマ	コンペアマッチ (TIMER:CMT)	2400μs 944μs 1200μs	Stop condition 計測用タイマ Active State 計測用タイマ Break time 計測用タイマ 3 種類のタイマに使用します。 受信割り込みの立ち上がり時に Stop condition 計測用として開始、立下り時に Active state 計測用として開始します。 Collision 発生時は立下り時に Break time 計測用として開始します。 ※Active state、Break time 計測用タイマは送信有時のみ使用します。
Settling time 計測タイマ (Backward 用)	コンペアマッチ (TIMER:MTU2)	5.5ms 10.5ms	Backward settling time 計測 送信開始タイミング確認処理 として使用します。
Settling time 計測タイマ	コンペアマッチ (TIMER:MTU3) (TIMER:MTU4)	4.0ms 13.5ms 14.9ms 16.3ms 17.9ms 19.5ms 100.0ms	Settling time 計測(Recovery time、Twice 含む) 送信開始タイミング確認処理 Twice Frame 無効化確認処理 として使用します。

- 送信タイミングタイマ

本ドライバのビット送信タイミング用タイマです。
送信タイミングタイマの使用方法を以下に示します。

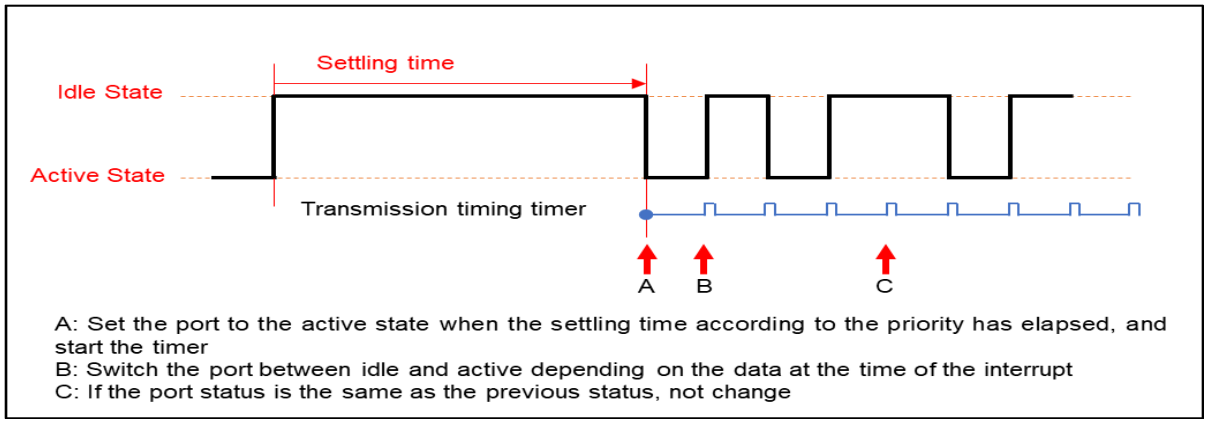

Figure 25 Start of transmission timing timer

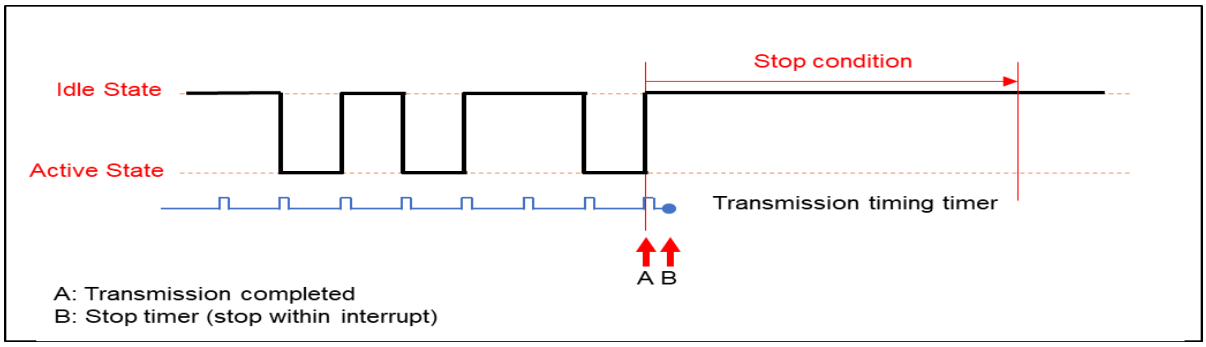

Figure 26 Stop of transmission timing timer

本タイマを使用して DALI 通信のビット送信タイミングを作成します。
規格で指定された Half bit のタイミングで周期動作させビット送信タイミングを作成し、その割り込み処理の中で送信ピンを操作することにより DALI 通信を行います。
送信データが終了した場合、タイマを停止します。

- 受信割り込みタイマ

本ドライバの受信ビット幅の計測用タイマです。
受信時の両エッジで割り込みを発生させるインプットキャプチャタイマを使用します。

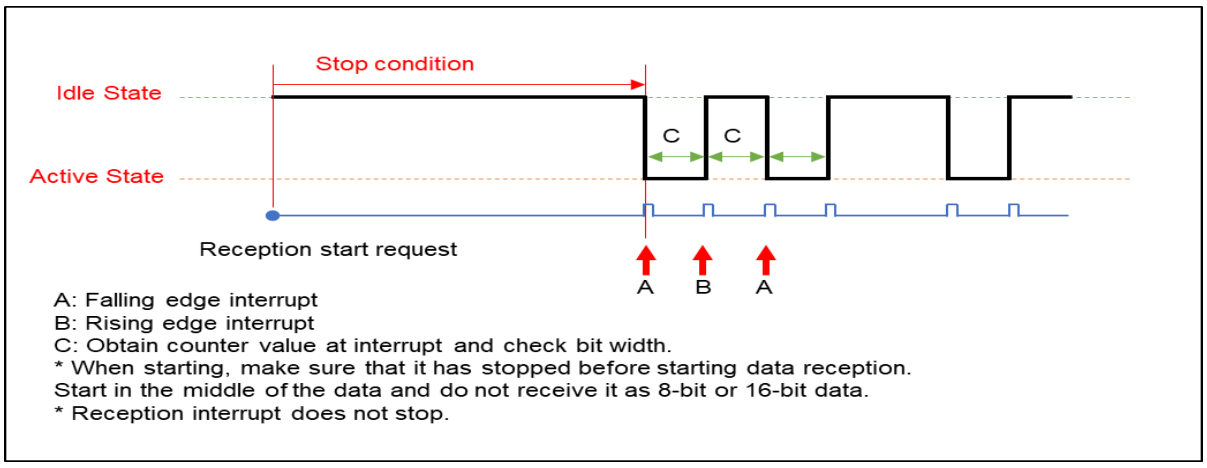

Figure 27 Start reception interrupt timer

- タイミング計測用タイマ

本ドライバのビットタイミング計測用タイマです。
カウンタマッチタイマを使用し、タイマを 2 種類のビット幅の計測用に使用します。

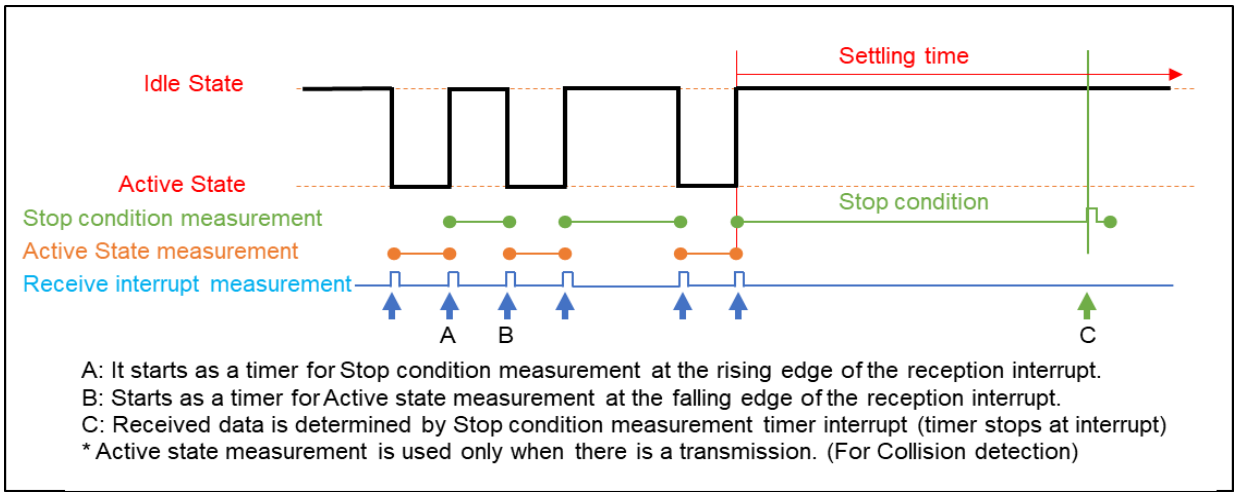

Figure 28 Start / stop of timer for timing measurement (Stop condition)

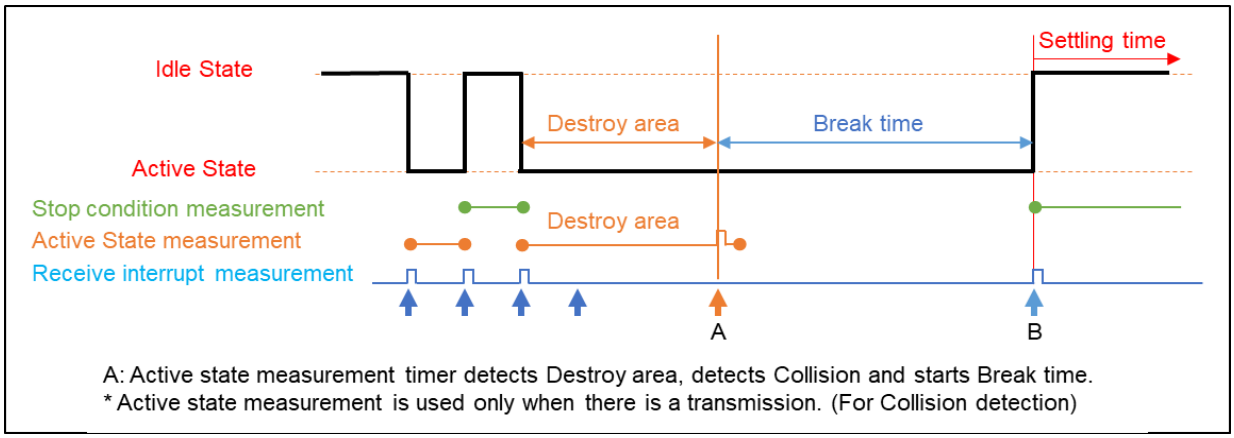

Figure 29 Start / stop of timer for timing measurement (Destroy area)

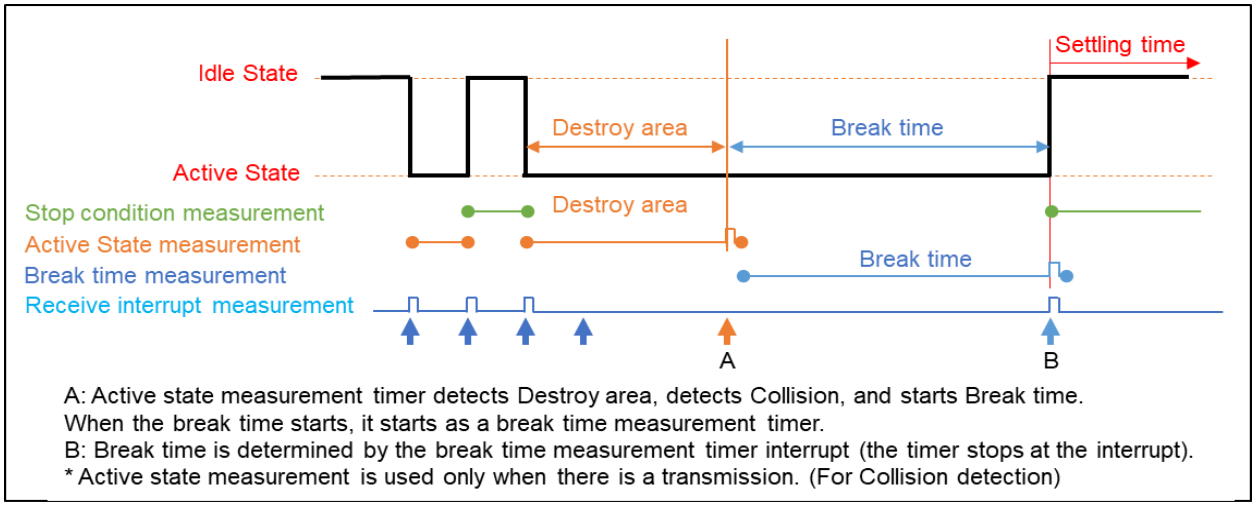

Figure 30 Start / stop of timer for timing measurement (Break time)

- Settling time 計測タイマ

本ドライバ内で使用する Settling time（Backward 用含）計測用のタイマです。
受信データの最終立ち上げ割り込みから始まる各種タイミング計測に使用します。

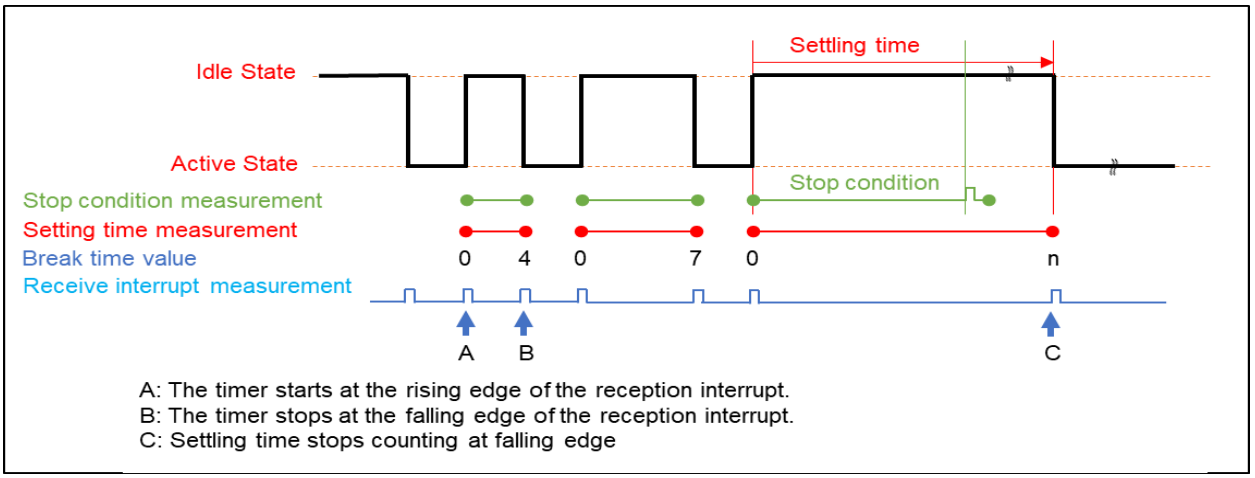

Figure 31 How to use the Settling time timer

Settling time 計測は送信開始のタイミングを計るために行います。
全ての Device において必要になるタイミングです。
Settling time の定義は「Single-master Transmitter Settling time 定義」「Multi-master Transmitter Settling time 定義」を参照してください。
計測時間が Settling time を経過した場合、送信可能 priority を更新していきます。

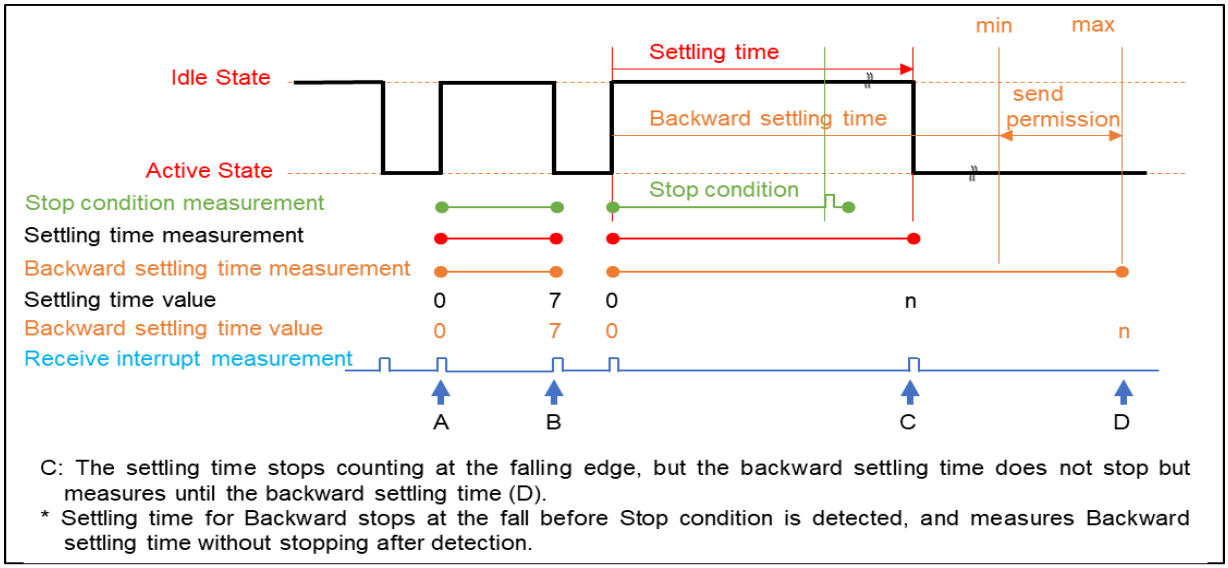

Figure 32 How to use Settling time timer for Backward (Backward settling time)

Backward Frame に使用する Settling time は、前回受信データの最終立ち上がりエッジからの経過時間になります。
通常の Settling time の計測は BUS State の Active state を検知した時点で計測を停止しますが、Backward 用 Settling time は経過時間を計測するため停止することはありません。ただし、Stop condition を受信し、前回データの受信が正常に完了したことが前提になります。Stop

condition を検知する前に BUS state が Active state になった場合は、計測を停止します。Backward Frame の計測は送信規定時間の Maximum 時間で停止し、それ以降は送信禁止となります。

Backward Frame の送信は、確実に Collision が発生することが分かっていても規定時間(5.5~10.5ms)で送信しなくてはならないことに注意してください。

(c)割り込み処理

本ドライバ内で使用する割り込み発生時の処理を以下に示します。

Table 28 SoftDALI driver interrupt list

割り込み名	タイマ発生条件	備考
送信タイミングタイマ割り込み	周期カウンタ経過時	416.7μs 毎
受信割り込み	DALI BUS 両エッジ発生時	
タイミング計測用タイマ割り込み	タイマ経過時 Stop condition タイマ経過時 Active state タイマ経過時 Break time タイマ経過時	同タイマを条件により使い分ける 2400μs 944μs 1200μs
Backward settling time 計測用タイマ割り込み	タイマ経過時	5500μs 10500μs
Settling time 計測用タイマ割り込み	タイマ経過時	4000μs 13500μs 14900μs 16300μs 17900μs 19500μs 100000μs

- 送信タイミングタイマ割り込み

送信タイミングタイマ割り込み発生時の処理について以下に示します。

Table 29 List of interrupt processing of transmission timing timer

条件	処理
最終送信データではない	・送信データに合わせて送信ピンを設定する 0：Low(Active state) 1：High(Idle state) ・送信 state の設定 送信したステートを内部変数に設定
最終送信データ	・送信データに合わせて送信ピンを設定する 0：Low 1：High ・送信 state の設定 送信したステートを内部変数に設定 ・送信タイミングタイマの停止

- 受信割り込みタイマ割り込み

受信タイマ割り込み発生時の処理について以下に示します。

Table 30 List of interrupt processing of reception timing timer

条件	処理
立ち下がりエッジ割り込み	・BUS state 取得 ・Collision 検知 ・ビット幅の確認 ・Stop condition タイマ停止 ・Active state タイマ起動(送信有時のみ) ・送信タイミングタイマの停止(Collision 発生時のみ) ・送信ピン Low(Active state 設定) (Collision 発生時のみ) ・Break time タイマ起動(Collision 発生時のみ) ・Settling time 計測停止 ・受信データの保存(正常受信時) ・状態に合わせたステータス設定(送受信とも)
立ち上がりエッジ割り込み	・BUS state 取得 ・Collision 検知 ・ビット幅の確認 ・Active state タイマ停止(送信有時のみ) ・Stop condition タイマ起動(Collision 発生時以外) ・送信タイミングタイマの停止(Collision 発生時のみ) ・送信ピン Low(Active state 設定) (Collision 発生時のみ) ・Break time タイマ起動(Collision 発生時のみ) ・Settling time 計測開始(Collision 発生時以外) ・Backward settling tine 計測開始(Collision 発生時以外) ・受信データの保存(正常受信時) ・状態に合わせたステータス設定(送受信とも)

- タイミング計測用タイマ割り込み

タイミング計測用タイマ割り込み発生時の処理について以下に示します。

Table 31 List of timer interrupt processing for timing measurement

条件	処理
Stop condition 割り込み	・送信完了設定(送信有時のみ) ・受信データの確定(データ変換) ・受信完了設定 ・状態に合わせたステータス設定(送受信とも) ・前回受信データとの比較 ・Twice 確認設定
Active state 割り込み	・Collision 検知設定(内部変数) ・送信タイミングタイマの停止(Collision 発生時のみ) ・送信ピン Low(Active state 設定) (Collision 発生時のみ) ・Break time タイマ起動(Collision 発生時のみ) ・状態に合わせたステータス設定(送受信とも)
Break time 割り込み	・BUS state 取得 ・Settling time の設定(Recovery time、通常 Settling time) ・状態に合わせたステータス設定(送受信とも)

- フリーランタイマ割り込み

フリーランタイマ割り込み発生時の処理について以下に示します。

Table 32 List of free-run timer interrupt processing

条件	処理
Backward settling time 計測用タイマ割り込み	・Minimum time の場合 Backward 送信可能設定 ・Maximum time の場合 Backward 送信禁止設定
Settling time 計測用タイマ割り込み	・Collision Recovery の場合 Collision 発生時 Frame の再送処理 ・Priority1 の場合 Priority1 の送信処理 ・Priority2 の場合 Priority2 以上の送信処理 ・Priority3 の場合 Priority3 以上の送信処理 ・Priority4 の場合 Priority4 以上の送信処理 ・Priority5 の場合 Priority5 以上の送信処理 ・Twice の場合 Twice Frame 無効化

(d)送受信データ変換

本ドライバで使用する送受信データの扱いについて以下に示します。

- 送信データ

送信データはアプリケーションから設定されます。
送信データの配置の仕方について以下に示します。

Figure 33 Example of transmission data arrangement

上記のような 24bit データがアプリケーション上で作成された場合を想定します。
DALI 通信では、通常 bit23~bit0 の順で送信します。
本ドライバでは、Proprietary データを対象とするため、256bit(32byte)までのデータを許可しています。通常の DALI 用データは 32bit までなので uint32_t で対応できますが、Proprietary の場合は uint8_t の配列とする必要があります。
本ドライバでは、送信データを uint8_t のポインタで扱うため、データの変換が必要となります。
24bit データを本ドライバに設定する場合のデータ配列は以下の様になります。

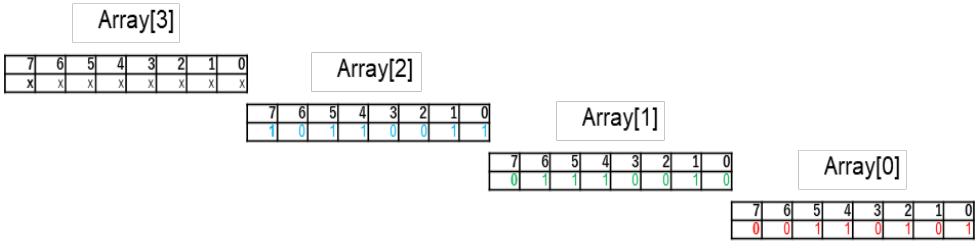

Figure 34 How to store in 8-bit data array

本ドライバには上記のデータ配列の先頭ポインタにてデータの受け渡しを行います。

- 送信データのエンコード

本ドライバ内では、上記の様に設定されたデータを DALI 通信で使用するマンチェスタ符号に変換して使用します。
DALI 送信で使用するマンチェスタ符号への変換方法を以下に示します。

Table 33 Manchester code

	Logical 0		Logical 1	
データビット	0		1	
マンチェスタ符号	1	0	0	1

また、送信での処理カウンタの使用を分かりやすくするために、データ順の入れ替えを行います。

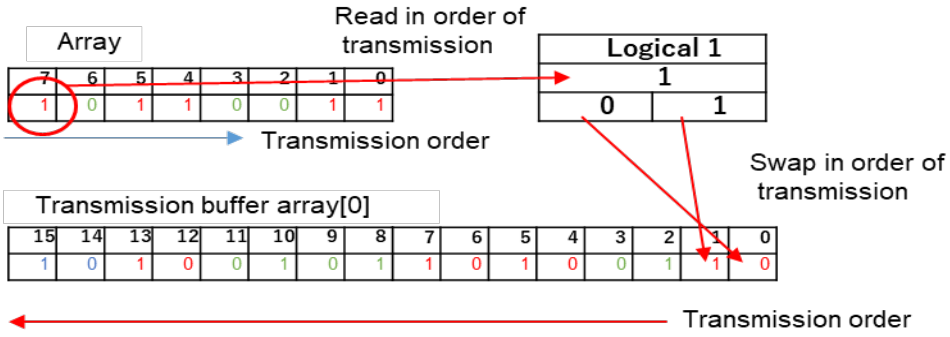

Figure 35 Conversion to Manchester code

- 受信データのデコード

受信データは DALI BUS から取得するマンチェスタ符号をデコードして設定されます。
受信データの配置の仕方について以下に示します。

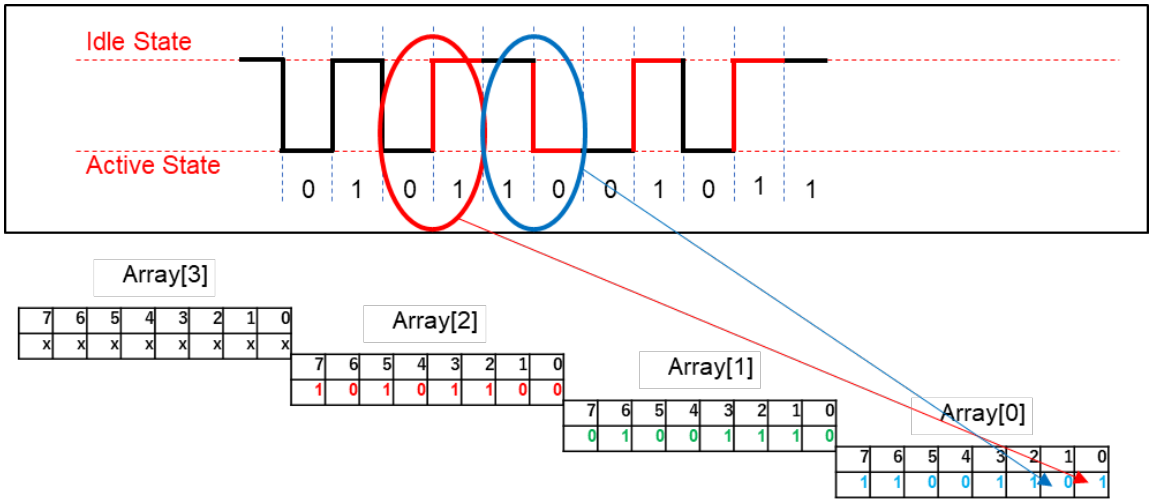

Figure 36 Decoding received data

受信データが何ビットで送られてきているのかが分からないため、受信データは配列の先頭から設定していきます。Stop condition 受信時に上記の様に 24bit データが送られてきたことが分かります。
この時点で規定されていないビット数のデータの場合は受信エラーとなります。
ビット数が分かることにより、アプリケーションに渡すためのデータの作成が行えます。

- 受信データの反転

受信が完了し、保存されているデータは通常の DALI データのビットの並びとは逆になっています。
アプリケーションに渡す場合、ビットの並びを通常の DALI データのものに並び替える必要があります。

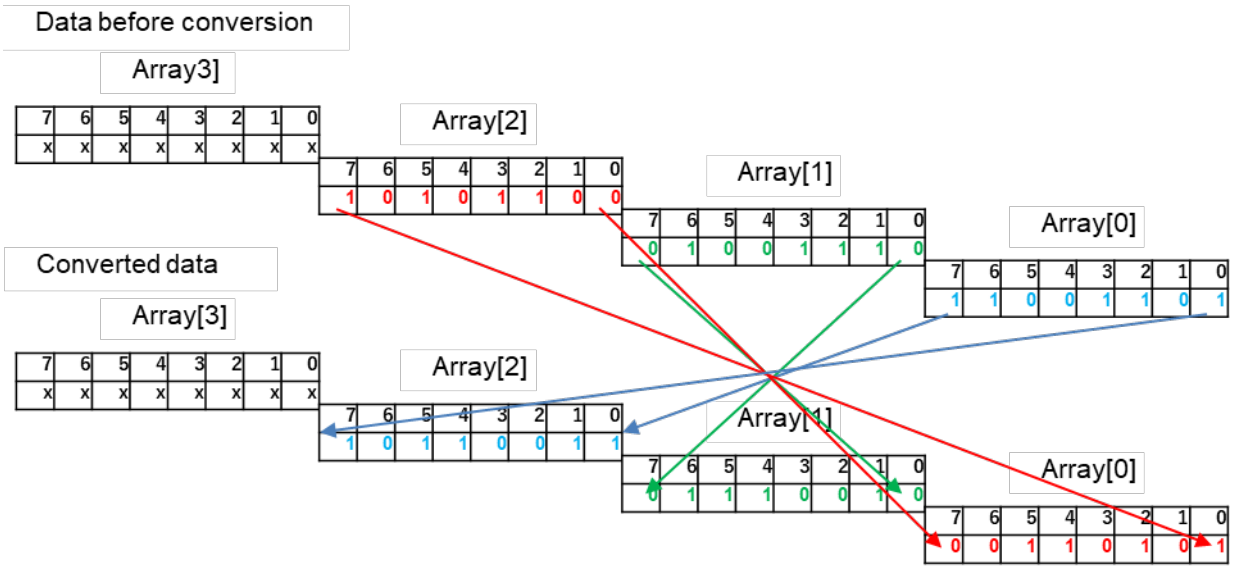

Figure 37 Inversion of received data

Stop condition 受信割り込み時に受信データを上記のように変換します。

(3) SoftDALIドライバ構成

ドライバ構成を以下に記します。

(a)ファイル構成

本ドライバのファイル構成を以下に示します。

Table 34 SoftDALI driver configuration file list

File name	Description
r_sddrv_api.c	SoftDALI API 部ソースファイル
r_sddrv_api.h	SoftDALI API 部ヘッダファイル
r_sddrv_tx.c	SoftDALI 送信部ソースファイル
r_sddrv_tx.h	SoftDALI 送信部ヘッダファイル
r_sddrv_rx.c	SoftDALI 受信部ソースファイル
r_sddrv_rx.h	SoftDALI 受信部ヘッダファイル
r_sddrv_timer.c	SoftDALI タイマ関連ソースファイル
r_sddrv_timer.h	SoftDALI タイマ関連ヘッダファイル
r_sddrv_frame.c	SoftDALI Frame デコードエンコード処理ソースファイル
r_sddrv_frame.h	SoftDALI Frame デコードエンコード処理ヘッダファイル
r_sddrv_com.c	SoftDALI アプリ側通信関連ソースファイル
r_sddrv_com.h	SoftDALI アプリ側通信関連ヘッダファイル
r_sddrv_gpio.c	SoftDALI ピン関連ソースファイル
r_sddrv_gpio.h	SoftDALI ピン関連ヘッダファイル
r_sddrv_def.h	SoftDALI ドライバ内部定義ヘッダファイル
r_sddrv_user.h	SoftDALI ユーザ定義関連ヘッダファイル

(b)公開マクロ定義

本ドライバが提供するマクロを以下に示します。

Table 35 SoftDALI driver function return value list

macro	value	Description
SDDRV_RET_OK	(0)	正常完了
SDDRV_RET_WAIT	(1)	送信完了待ち、受信完了待ち
SDDRV_RET_NG	(-1)	異常終了
SDDRV_RET_ERR_PARAM	(-2)	パラメータエラー
SDDRV_RET_BUFF_FULL	(-3)	送信バッファ使用中
SDDRV_RET_SEND_NONE	(-4)	送信データなし
SDDRV_RET_ERR_SEND_TIMING	(-5)	送信タイミングエラー(Backward のみ)

※int16_t

Table 36 SoftDALI driver buffer usage types

macro	value	Description
SDDRV_FRAME_BUFF_USED_NONE	(0)	全バッファ未使用
SDDRV_FRAME_BUFF_USED_BF	(1)	Backward Frame バッファ使用中
SDDRV_FRAME_BUFF_USED_FF	(2)	Forward Frame バッファ使用中

※Backward、Forward の両バッファ使用中の場合、

```
SDDRV_FRAME_BUFF_USED_BF | SDDRV_FRAME_BUFF_USED_FF
```

のように OR して返すことを想定しています。

(c)構造体・共用体・列挙型

本ドライバが提供する構造体・共用体・列挙型を以下に示します。

Table 37 App communication structure (st_sddrv_com_param_t)

Member variable type	Member variable name	Description
EventGroupHandle_t	sendCompleteEventHandl	受送信完了イベントハンドル
EventBits_t	sendCompleteBFEventBit	Backward Frame 送信完了イベントビット
EventBits_t	sendCompleteFFEventBit	Forward Frame 送信完了イベントビット
EventBits_t	sendErrorCompleteEventBit	送信エラー時送信完了イベントビット
QueueHandle_t	recvCompleteQueueHandl	受信完了データ Queue ハンドル
QueueHandle_t	recvCompleteEventQueueHandl	受信完了イベント Queue ハンドル
EventBits_t	recvCompleteEventBit	受信完了イベントビット
EventBits_t	recvISendCompleteEventBit	受信完了イベントビット（自送信データ）
EventBits_t	systemFailureEventBit	システムフェイラー検知イベントビット
EventBits_t	backwardTimeoutEventBit	Backward Frame タイムアウトイベントビット

※各タイミングで Queue とイベントに設定します。

Table 38 Transmission Frame structure (st_sddrv_send_param_t)

Member variable type	Member variable name	Description
uint8_t	priority	送信 Frame 優先度(1~5)
uint8_t	frameType	送信 Frame タイプ SDDRV_FRAME_BACKWARD SDDRV_FRAME_FORWARD SDDRV_FRAME_COLLAPSED
uint8_t*	pframe_data	送信 Frame データ先頭ポインタ
uint16_t	frame_bit_num	送信 Frame サイズ(ビットサイズ指定)

※SDDRV_FRAME_BACKWARD 指定時の送信 Frame 優先度は無視します。

Table 39 Receive Frame structure (st_sddrv_recv_param_t)

Member variable type	Member variable name	Description
uint16_t	frame_bit_size	受信 Frame サイズ(ビットサイズ)
bool	is_twice	Twice タイミング(true：twice false：no twice)
bool	is_error	データ状態(true：error false：normal)
uint8_t	frame_data[SDDRVE_DATA_SIZE_MAX]	受信 Frame データ先頭ポインタ

※isTwice：前回と同じ Frame を Twice Frame タイミング以内に受け取った場合 True とします。
※isError：受信データが正常データの場合は false、エラーデータの場合は true を設定します。

Table 40 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 単体時 / Multi master：複数 Application controller

Table 41 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 42 Operation status enumeration (e_sddrv_state_t)

macro	value	Description
SDDRV_STATE_INIT	0	初期状態
SDDRV_STATE_STANDBY	1	初期化完了待機状態
SDDRV_STATE_SEND_WAIT	2	送信待機中
SDDRV_STATE_SEND	3	送信中
SDDRV_STATE_SEND_ERR	4	送信エラー(コリジョン発生)
SDDRV_STATE_SEND_ERR_RECOVER	5	送信エラー回復中(コリジョン回復中)
SDDRV_STATE_SEND_RETRY	6	再送中(Settling time 開始：Recovery time 含)
SDDRV_STATE_SEND_COMPLETE	7	送信完了
SDDRV_STATE_RECV_WAIT	8	受信待ち(未受信、異常データ受信)
SDDRV_STATE_RECV	9	受信中
SDDRV_STATE_RECV_COMPLETE	10	受信完了(施錠データ受信完了)

(d)ユーザ設定動作定義ビルド用マクロ

本ドライバをユーザが使用する場合の動作に関する定義を設定します。
ビルド時に必ず設定してください。

Table 43 Symbol for operation setting build

macro	Description
SDDRV_MODE_PROPRIETARY	拡張 Frame 設定 0：無効 / 1：有効 デフォルト　0：無効

※ビルド時に本ドライバの動作を設定するためのシンボルです。ビルド時に必ず設定します。

Table 44 Macro for operation setting build

macro	default	Description
SDDRV_PROPRIETARY_NUM	3	使用する Proprietary Frame の種類(1-3)
SDDRV_PROPRIETARY_BIT_1	64	ユーザ指定ビット数 1
SDDRV_PROPRIETARY_BIT_2	128	ユーザ指定ビット数 2
SDDRV_PROPRIETARY_BIT_3	256	ユーザ指定ビット数 3

※ビルド時に本ドライバの Proprietary の許可ビットを定義するためのマクロです。変更する場合はビルド時に必ず設定します。

※SDDRV_MODE_PROPRIETARY が有効(1) の時のみ有効になります。

Table 45 Build macro for timer clock

macro	default	Description
INPUT_CAPTURE_CLOCK_MHZ	30	受信インプットキャプチャタイマに使用するクロック(MHz)

Table 46 Build macro for timer adjustment

macro	default	Description
SDDRV_BREAK_TIME_INTERVAL_OFFSET	(40)	Stop condition、Active state、Break time 用の One shot タイマ調整用マクロです。
SDDRV_SYSTEM_FAILURE_INTERVAL_OFFSET	(-16)	System failure 用タイマ調整用マクロ

(e)内部定義用マクロ

本ドライバ内で使用するビットタイミングに関して定義を行っているマクロを以下に示します。

Table 47 Macro for internal operation setting build

macro	default Description
SDDRV_STOP_CONDITION_TIMER_ BASE_INTERVAL	2150 Stop condition 計測用タイマベース値(ns)
SDDRV_ACTIVE_STATTE_TIMER_ BASE_INTERVAL	944 Active state 計測用タイマベース値(ns)
SDDRV_BREAK_TIME_TIMER_ BASE_INTERVAL	1200 Break time 計測用タイマベース値(ns)
SDDRV_SYSTEM_FAILURE_ BASE_INTERVAL	550 System failure 計測用タイマベース値(ns)
SDDRV_STOP_CONDITION_ TIMER_INTERVAL	SDDRV_STOP_CONDITION_TIMER_BASE_INTERVAL + SDDRV_STOP_CONDITION_INTERVAL_OFFSET Stop condition 計測用タイマベース値に offset 値を加算した値(ns) この値を使用します。
SDDRV_ACTIVE_STATTE_ TIMER_INTERVAL	SDDRV_ACTIVE_STATE_TIMER_BASE_INTERVAL + SDDRV_ACTIVE_STATE_INTERVAL_OFFSET Active state 計測用タイマベース値に offset 値を加算した値(ns) この値を使用します。
SDDRV_BREAK_TIME_ TIMER_INTERVAL	SDDRV_BREAK_TIME_TIMER_BASE_INTERVAL + SDDRV_BREAK_TIME_INTERVAL_OFFSET Break time 計測用タイマベース値に offset 値を加算した値(ns) この値を使用します。
SDDRV_SYSTEM_ FAILURE_INTERVAL	SDDRV_SYSTEM_FAILURE_BASE_INTERVAL + SDDRV_SYSTEM_FAILURE_INTERVAL_OFFSET System failure 計測用タイマベース値に offset 値を加算した値(ns) この値を使用します。
SDDRV_RISING_TIME_ TIMER_INTERVAL	SDDRV_RISING_TIME_VALUE + SDDRV_INTERRUPT_SYSTEM_DELAY_VALUE ピン立ち上がり時間に起動遅延時間を加算した値(us)

Table 48 Bit width threshold macro (without Collision measurement)

macro	default	Description
SDDRV_BIT_WIDE_1st_HALF_MIN	300000	First section の half bit 最小値(ns)
SDDRV_BIT_WIDE_1st_HALF_MAX	625000	First section の half bit 最大値(ns)
SDDRV_BIT_WIDE_2nd_HALF_MIN	300000	Second section の half bit 最小値(ns)
SDDRV_BIT_WIDE_2nd_HALF_MAX	580000	Second section の half bit 最大値(ns)
SDDRV_BIT_WIDE_2nd_D_HALF_MIN	580000	Second section の Double half bit 最小値(ns)
SDDRV_BIT_WIDE_2nd_D_HALF_MAX	1100000	Second section の Double half bit 最大値(ns)

Table 49 Bit width threshold macro (when there is Collision measurement)

macro	default	Description
SDDRV_BIT_WIDE_1st_HALF_COL_MIN	365000	First section の half bit 最小値(ns)
SDDRV_BIT_WIDE_1st_HALF_COL_MAX	476000	First section の half bit 最大値(ns)
SDDRV_BIT_WIDE_2nd_HALF_COL_MIN	365000	Second section の half bit 最小値(ns)
SDDRV_BIT_WIDE_2nd_HALF_COL_MAX	467000	Second section の half bit 最大値(ns)

SDDRV_BIT_WIDE_2nd_D_HALF_COL_MIN	762000	Second section の Double half bit 最小値(ns)
SDDRV_BIT_WIDE_2nd_D_HALF_COL_MAX	905000	Second section の Double half bit 最大値(ns)

(f) API 関数一覧

本ドライバが提供する API 関数一覧を以下に示します。

Table 50 API function list

関数名	Description
R_SDDRV_Init	SoftDALI ドライバの初期化を行います。
R_SDDRV_RecvStart	受信処理を開始します。
R_SDDRV_CheckSendBuff	送信データバッファの使用状況を確認します。
R_SDDRV_Send	送信開始要求を行います。 送信が完了するまで待機せず即時復帰します。
R_SDDRV_SendCancel	送信を停止し、送信されていない場合はキャンセルします。
R_SDDRV_GetVersion	SoftDALI ドライバのバージョン番号を取得します。

(g)API 関数仕様

- R_SDDRV_Init

関数名	void R_SDDRV_Init (st_sddrv_com_param_t* p_param, e_device_type_t dev_type)
引数	st_sddrv_com_param_t* p_param アプリ通信構造体の先頭ポインタ e_device_type_t dev_type アプリ通信構造体 SDDRV_DEV_SINGLE_APP SDDRV_DEV_MULTI_APP SDDRV_DEV_INPUT_DEVICE
戻り値	void
説明	SoftDALI ドライバの初期設定を行います。 デバイスタイプの設定 各コンポーネントの初期化

- R_SDDRV_RecvStart

関数名	int16_t R_SDDRV_RecvStart (void)
引数	なし
戻り値	Int16_t SDDRV_RET_OK：正常終了 SDDRV_RET_NG：異常終了
説明	SoftDALI ドライバでの受信処理を開始します。 受信用両エッジ割り込みインプットキャプチャ割り込みを許可します。

- R_SDDRV_CheckSendBuff

関数名	uint8_t R_SDDRV_CheckSendBuff (void)
引数	なし
戻り値	uint16_t bit0：Backward frame buffer 使用フラグ　0：not used　　1：used bit1：Forward frame buffer 使用フラグ　0：not used　　1：used
説明	送信データバッファの使用状況を確認します。

- R_SDDRV_Send

関数名	int16_t R_SDDRV_Send (const st_sddrv_send_param_t* p_send)
引数	const st_sddrv_send_param_t* p_send 送信 Frame 構造体の先頭ポインタ
戻り値	int16_t SDDRV_RET_OK：正常終了 SDDRV_RET_ERR_PARAM：パラメータエラー SDDRV_RET_BUFF_FULL：前データ送信中 SDDRV_RET_ERR_SEND_TIMING：送信タイミングエラー発生
説明	送信するデータを設定し送信を開始します。 送信完了を待たずに即時復帰します。 前回設定されたデータが送信完了していない場合「SDDRV_RET_BUFF_FULL」を返します。 Backward 送信要求時に規定時間を既に超えていた場合 「SDDRV_RET_ERR_SEND_TIMING」を返します。 Forward Frame が送信待ち状態にある場合に Backward Frame を追加で設定することが可能です。この場合 Backward Frame が先に送信されます。 Backward Frame が送信待ち状態にある場合に Forward Frame を追加で設定することはできません。この場合「SDDRV_RET_BUFF_FULL」を返します。

- R_SDDRV_SendCancel

関数名	uint8_t R_SDDRV_SendCancel (void)
引数	なし
戻り値	Int16_t SDDRV_STATE_OK：正常終了 SDDRV_RET_SEND_NONE：送信なし
説明	送信処理を停止します。 送信待ち中、送信中に限らず送信処理を停止します。

- R_SDDRV_GetVersion

関数名	uint16_t R_SDDRV_GetVersion (void)
引数	なし
戻り値	uint16_t 上位 8bit：メジャーバージョン 下位 8bit：マイナーバージョン
説明	SoftDALI ドライバのバージョンを返します。

(h)内部グローバル変数一覧

本ドライバ内部で使用するグローバル変数を以下に示します。

Table 51 Internal Global List 1/7

Member variable type	Member variable name	Description
e_sddrv_state_t	g_transmitter_status	送信処理ステータス 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 設定タイミング 処理状態変化時 初期値 SDDRV_STATE_INIT
e_sddrv_state_t	g_receiver_status	受信処理ステータス SDDRV_STATE_INIT SDDRV_STATE_STANDBY SDDRV_STATE_RECV_WAIT SDDRV_STATE_RECV SDDRV_STATE_RECV_COMPLETE 設定タイミング 処理状態変化時 初期値 SDDRV_STATE_INIT
e_device_type_t	g_device_type	指定デバイスタイプ SDDRV_DEV_SINGLE_APP SDDRV_DEV_MULTI_APP SDDRV_DEV_INPUT_DEVICE 設定タイミング 初期化処理時 初期値 SDDRV_DEV_SINGLE_APP
st_sddrv_com_param_t	g_com_info	アプリ通信構造体保存領域 設定タイミング 初期化処理時 初期値 ALL 0

Table 52 Internal Global List 2/7

Member variable type	Member variable name	Description
uint8_t	g_recv_data_buff[]	受信バッファ(受信データ) バッファサイズ SDDRVE_DATA_SIZE_MAX 設定タイミング 受信割り込み時 デコードタイミング 削除タイミング デコードデータ作成完了時 使用方法 受信完了時のデコード元データとして使用
uint8_t	g_prev_data_buff[]	前回受信バッファ(受信データ) バッファサイズ SDDRVE_DATA_SIZE_MAX 設定タイミング StopCondition 割り込み時 前回データ比較後 削除タイミング なし 使用方法 前回受信データと現受信データの比較、 Twice 判断に使用します。
uint16_t	g_cnt_recv_bit	受信ビット数 設定タイミング 受信割り込み時 デコードタイミング 削除タイミング 受信開始時 使用方法 受信ビット数の確認、通知
bool	g_stop_condition_flg	Stop condition 受信フラグ false：未受信 true：受信済み true 設定タイミング Stop condition 割り込み時 false 設定タイミング 受信開始時 使用方法 Stop condition 受信の確認

Table 53 Internal Global List 3/7

Member variable type	Member variable name	Description
bool	g_recv_section_flg	Stop condition 受信フラグ false：first section true：second section true 設定タイミング 受信割り込み時 false 設定タイミング なし 使用方法 デコードタイミングの判断
uint8_t	g_cnt_start_bit	Start bit 送信数 0：初期値 1：start bit first section 送信 2：start bit second section 送信 設定タイミング 受信割り込み時 削除タイミング 受信開始時 1 設定 使用方法 受信中のデータ受信開始タイミング判断
bool	g_collision_flg	Collision 検知フラグ false：Collision 未検知 true：Collision 検知 true 設定タイミング Receiver 側受信割り込み時 送信 state と BUS state 比較で相違発生時 && Destroy area 発生時（bit violation） Destroy area 計測タイマ割り込み時 false 設定タイミング Break time 計測割り込み受信時 使用方法 Stop condition 計測の開始可不可判断 初期値 False 送信 state と BUS state 比較で相違が発生したが、Destroy area は発生しなかった場合は、通常の送信エラー処理として受信は継続。

Table 54 Internal Global List 4/7

Member variable type	Member variable name	Description
bool	g_recv_err_flg	受信データエラーフラグ false：受信データエラー無 true：受信データエラー有 true 設定タイミング 受信割り込み時 Stop condition 割り込み時 Active state 割り込み時 false 設定タイミング 受信開始時 使用方法 受信データの Bit violation、ビット数、Destroy area 検知の判断
bool	g_collision_retry_flg	Collision 回復判断フラグ false：通常回復 true：Collision 回復 true 設定タイミング Break time 割り込み時 false 設定タイミング 受信開始時 使用方法 再送に使用する Settling time の確認
bool	g_recv_start_flg	受信開始フラグ false：未受信 true：受信中 true 設定タイミング 受信開始時 false 設定タイミング Stop condition 割り込み時 Active state タイマ割り込み時 使用方法 受信データのデコード判断、Stop condition 割り込み時データデコード判断。
bool	g_twice_flg	Twice 判断フラグ false：twice 未受信 true：twice 受信 true 設定タイミング Stop condition 割り込み時 false 設定タイミング twice タイムアウト割り込み時 使用方法 アプリへの Twice Frame 受信通知

Table 55 Internal Global List 5/7

Member variable type	Member variable name	Description
uint16_t	g_ff_data_buff[]	Forward Frame 送信バッファ(エンコード) バッファサイズ SDDRVE_DATA_SIZE_MAX 設定タイミング 送信要求時 削除タイミング 送信完了確認時(Stop condition 割り込み時) 使用方法 送信タイミングタイマ割り込み時の送信データとして使用。
uint16_t	g_bf_data_buff	Backward Frame 送信バッファ(エンコード) 設定タイミング 送信要求時 削除タイミング 送信完了確認時(Stop condition 割り込み時) 使用方法 送信タイミングタイマ割り込み時の送信データとして使用。
bool	g_is_ffset	Forward Frame 送信バッファ設定フラグ true 設定タイミング 送信要求時 false 設定タイミング 送信完了確認時(Stop condition 割り込み時) 使用方法 Forward Frame 設定確認に使用
bool	g_is_bfset	Backward Frame 送信バッファ設定フラグ true 設定タイミング 送信要求時 false 設定タイミング 送信完了確認時(Stop condition 割り込み時) 使用方法 Backward Frame 設定確認に使用
uint8_t	g_ff_priority	Forward Frame 送信優先度 設定タイミング 送信要求時 削除タイミング 送信完了確認時(Stop condition 割り込み時) 使用方法 使用 Settling time の判断に使用

Table 56 Internal Global List 6/7

Member variable type	Member variable name	Description
uint16_t	g_ff_size	Forward Frame サイズ 設定タイミング 送信要求時 削除タイミング 送信完了確認時(Stop condition 割り込み時) 使用方法 送信データのサイズ判断に使用
uint16_t	g_encode_size	送信 Frame マンチェスタサイズ 設定タイミング 送信要求時 削除タイミング 送信完了確認時(Stop condition 割り込み時) 使用方法 送信データの送信完了判断に使用
uint16_t	g_transmitted_num	送信ビット数 設定タイミング 送信割り込み時 削除タイミング 送信開始時 使用方法 送信済みデータカウンタとして送信完了判断に使用
uint8_t	g_send_state	送信中 Frame 設定 SDDRV_FRAME_SENDING_NONE：未送信 SDDRV_FRAME_SENDING_BACKWARD SDDRV_FRAME_SENDING_FORWARD 設定タイミング 送信開始時 削除タイミング 送信完了時 使用方法 送信中の Frame 判断に使用
uint8_t	g_bf_permit_state	Backward Frame 送信許可判断 SDDRV_BACKWARD_SEND_WAIT SDDRV_BACKWARD_SEND_POSSIBLE SDDRV_BACKWARD_SEND_IMPOSSIBLE 設定タイミング タイミング計測カウンタ割り込み時 Backward settling time 計測にて 5.5ms 経過 Backward settling time 計測にて 10.5ms 経過 使用方法 送信設定、及び送信開始判断に使用

Table 57 Internal Global List 7/7

Member variable type	Member variable name	Description
uint8_t	g_ff_permit_priority_num	Forward Frame 送信許可優先度指定 0：送信禁止 1：priority 1 のみ送信許可 2：priority 2 以上送信許可 3：priority 3 以上送信許可 4：priority 4 以上送信許可 5：priority 5 以上送信許可 6：Collision recovery 設定タイミング 0：送信禁止時 Start bit 受信時 1-5：priority 送信許可 各 Settling time 計測割り込み時 Collision recovery 送信許可 Settling time 計測割り込み時 使用方法 Forward Frame の送信開始許可判断
bool	g_tx_send_short_int_flg	High レベル送信フラグ False：送信未送信 True：送信済み 設定タイミング 送信立ち上がり用割り込み時 送信開始時 使用方法 通常送信タイミングに送信を行うかの判断に使用

2. DALI 通信ミドルウェア

DALI 通信ミドルウェアは、DALI 通信ドライバのミドルウェア機能です。Application Controller から要求された Frame を DALI 通信ドライバへ通知し、DALI 通信ドライバから受信した Frame を Application Controller へ通知を行うなどの処理を行います。

(1) 機能

DALI 通信ミドルウェアは、以下の機能を実装します。

Table 58 Function list of DALI communication middleware

Function	Description
送信処理	送信 Queue 制御、送信 Frame priority 制御、トランザクション制御を行います。
受信処理	受信 Queue 制御を行います。
Soft DALI ドライバ通信	Soft DALI ドライバとの Frame 送受信を行います。

(2) DALI 通信ミドルウェアタスク:送信処理

DALI 通信ミドルウェアタスクでの送信処理内容を記載します。

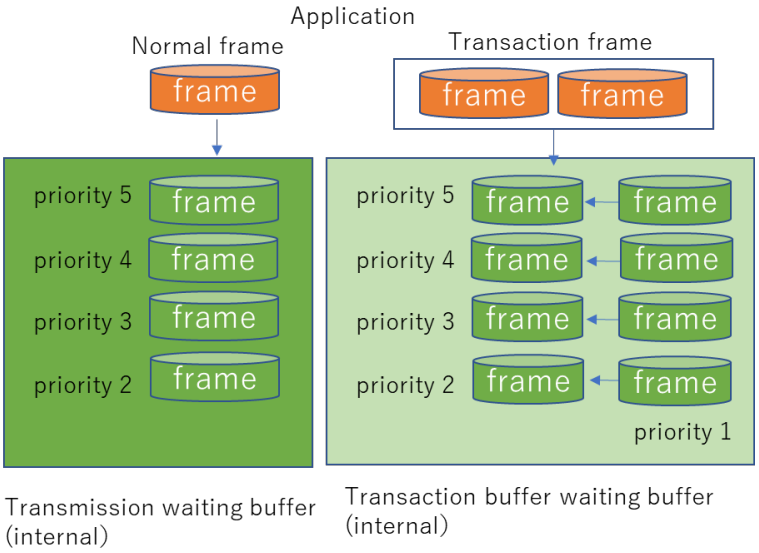

Figure 38 Overview of transmission processing

[制御内容]

1. アプリケーションからセットされた Queue から Frame を抽出
2. 抽出した Frame がトランザクションか判別し、ミドルウェアのバッファに登録
   トランザクションの場合はトランザクションバッファ、通常 Frame の場合は通常のバッファに登録
3. 登録後の内部処理として priority でソートする
4. Queue に Frame があれば 1 に戻る

5. 次に送る予定となる Frame があるかどうか、ミドルウェアの送信 Queue の状態を監視
   送信 Queue に送信 Frame があり、DALI 通信ドライバのバッファが空き状態であれば DALI 通信ドライバに送信 Frame を渡す
6. DALI 通信ドライバが送信可能であれば送信開始を通知、送信不可であれば何もしない
7. DALI 通信ドライバの送信完了待ち(タイムアウトの時は1へ戻る)
8. トランザクションフレーム時は再度 6 へ戻る
9. Backward Frame を送信する場合は、内部のバッファ(Priority による並び替え) を使用せずに DALI 通信ドライバにデータを渡す

(3) DALI 通信ミドルウェアタスク：受信処理
DALI 通信ミドルウェアタスクでの受信処理内容を記載します。

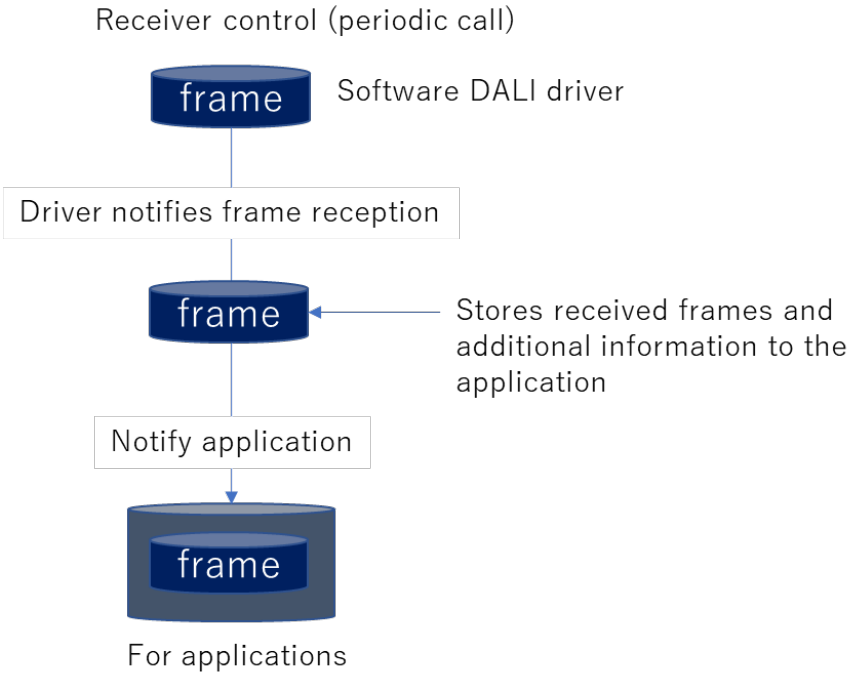

Figure 39 Overview of reception processing

[制御内容]

1. DALI 通信ドライバから受信通知を受け取る
2. Frame 受信の場合
   Application Controller に Queue を使用して受信 Frame を通知する
3. Backward タイムアウトなどの通知の場合
   タイムアウト情報を格納し、Application Controller に Queue を使用して通知する

4.6.2　IEC62386-103 規格規定部

IEC62386-103 規格規定部は Application Controller ライブラリと Application Controller ライブラリを動作させる Application Controller アプリケーション、周辺アプリケーションによって構成されます。Application Controller ライブラリに関しては、ユーザーズマニュアルを参照してください。

1. Application Controller アプリケーションタスク

Application Controller アプリケーションタスクは Application Controller 動作を行うタスクです。1ms 定期で動作するタスクと Frame 受信で動作するタスクがあり、Application Controller ライブラリから提供されている API 関数を使用して Application Controller 動作や Frame の処理を行います。

Application Controller ライブラリの API 関数は下記図の黒枠が該当します。

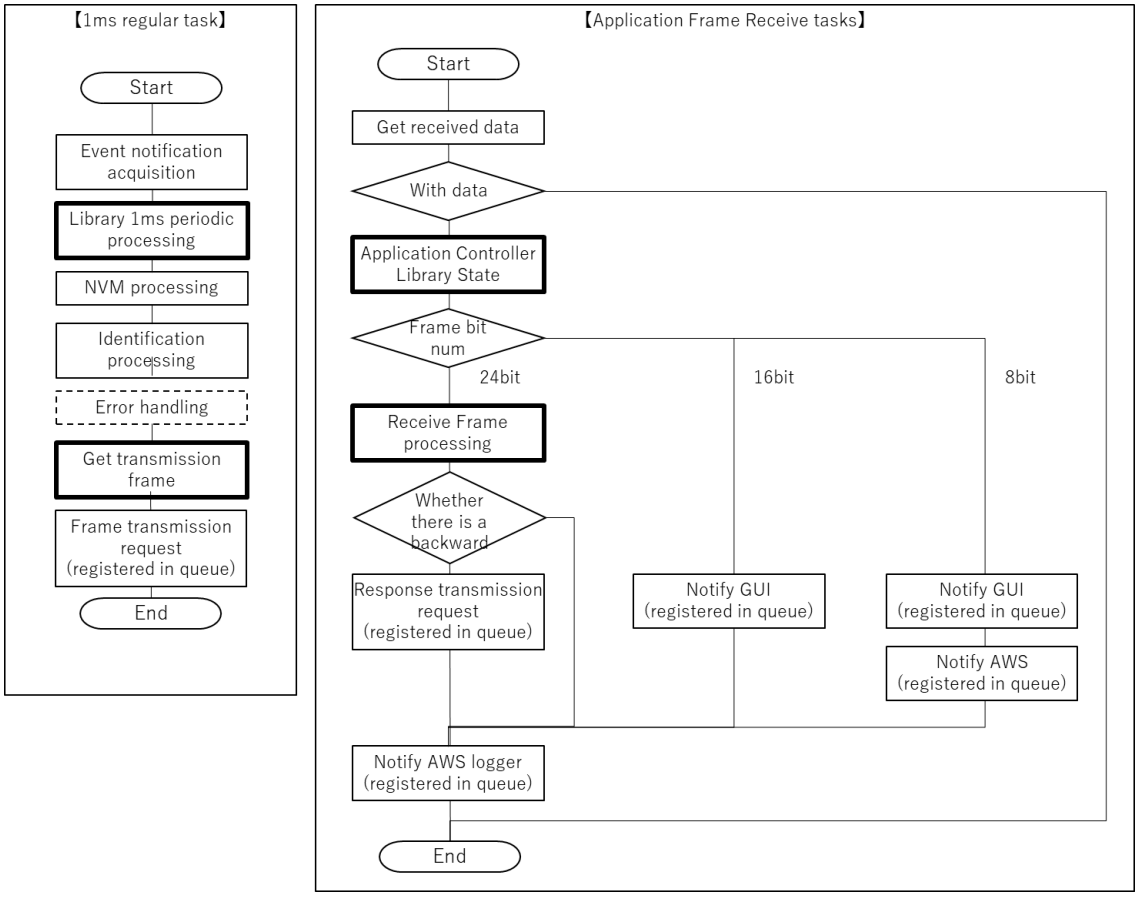

Figure 40 Application controller task flow diagram

- 1ms 定期タスク
  Free-RTOS 機能ソフトウェアタイマによる 1ms 定期イベントで定期的にタスク処理を行います。
  Application Controller ライブラリで指定されているライブラリ 1ms 定期処理を実施します。
  本デモプロジェクトでは合わせて送信 Frame の取得処理も 1ms 定期タスク内で実施しています。
  ※破線のエラー処理はユーザ実装の処理となります。エラー状態をライブラリへ通知するサンプルコードがありますので、エラー処理を行う際は 4.6.2(4)項のサンプルコードを参考に実装してく

ださい。

- Frame 受信タスク
  受信したフレームのサイズによって、対象となる Application Controller タスクへ Frame の通知または Frame の処理を行います。

（1） 関数仕様

- r_app_1msec_interval_task

関数名	void r_app_1msec_interval_task (void * pvParameters)
引　数	void * pvParameters Free-RTOS のタスクハンドラ
戻り値	void
説　明	タスク通知を受け取った際、使用する Logical Unit ごとに 1ms 定期処理を行います。 使用する Logical Unit 毎に以下の処理を行います。 • Application Controller ライブラリの 1ms 定期処理を行います。 • NVM 定期処理を行います。 • 識別処理を行います。 • Forward Frame が発生している場合、Forward Frame の取得を行います。 • Forward Frame を DALI 通信ミドルウェアタスクの送信 Queue へ登録します。 • マトリクスボタン用データフラッシュ書き込み処理を行います。

- r_app_recv_dali_frame_task

関数名	void r_app_recv_dali_frame_task (void * pvParameters)
引　数	void * pvParameters Free-RTOS のタスクハンドラ
戻り値	void
説　明	DALI 通信ミドルウェアタスクの受信 Queue に対象のデータが格納された際に、Frame のビットサイズまたは受信状態と Logical Unit の状態によって Application Controller タスクへの queue による Frame の受け渡しと通知を行います。 また、AWS Web アプリケーションの Logger 機能が有効となっている場合、Logger 機能への通知も合わせて行います。 □Frame サイズによる処理又は通知 24bit frame • 受信データから Application Controller ライブラリが Frame 処理の実行対象か判断します。 • 使用する Logical Unit ごとに Frame 処理を行います。 • Frame 処理の応答結果(Backward) がある場合、DALI 通信ミドルウェアタスクの Backward 用 Queue に Backward を登録します。

	※複数 Logical Unit 使用時は Backward Frame の破損要求有無を合わせて判断・Queue へ登録します。 16bit frame • 自発行 Frame の場合、GUI アプリケーションタスクへ通知します。 8bit frame • GUI アプリケーション、AWS Web アプリケーションへ通知します。 口受信状態による処理または通知 Backward 受信 TIMEOUT GUI アプリケーション、AWS Web アプリケーションへ通知します。

2. ユーザ定義

Application Controller の動作の一部はユーザによって定義を行う必要があります。

本デモプロジェクトではユーザ依存に関する部分をシンボルで定義するほか、一部サンプルコードを用意しております。使用目的に合わせて変更・使用してください。

(1) 各シンボルの説明

各シンボルの内容を下表に示します。

Table 59 List of contents of each symbol

Symbol name	Default value	Description
IS_SINGLE_MODE	False (Multi-master mode)	使用する Logical Unit の Multi/Single master 動作設定
LOGI_UNIT_NUM	1	使用する Logical Unit 数
IDENTIFY_LED_PORT	GPIO_PORT_B_PIN_0	識別に使用する LED 点消灯ピン番号
DEFAULT_OPERATING_MODE	0x00	使用する Logical Unit の Operating Mode の default 値
USE_PROPRIETARY_FRAME_SIZE_NUM	3	使用する Proprietary Frame 数
PROPRIETARY_FRAME_SIZE_1	64	送信する Proprietary Frame の byte 数
PROPRIETARY_FRAME_SIZE_2	128	送信する Proprietary Frame の byte 数
PROPRIETARY_FRAME_SIZE_3	256	送信する Proprietary Frame の byte 数

(2) シンボルの設定変更方法

「プロパティ」→「一般」→「パスおよびシンボル」→「シンボル」タブ　から指定シンボルの値を変更してください。

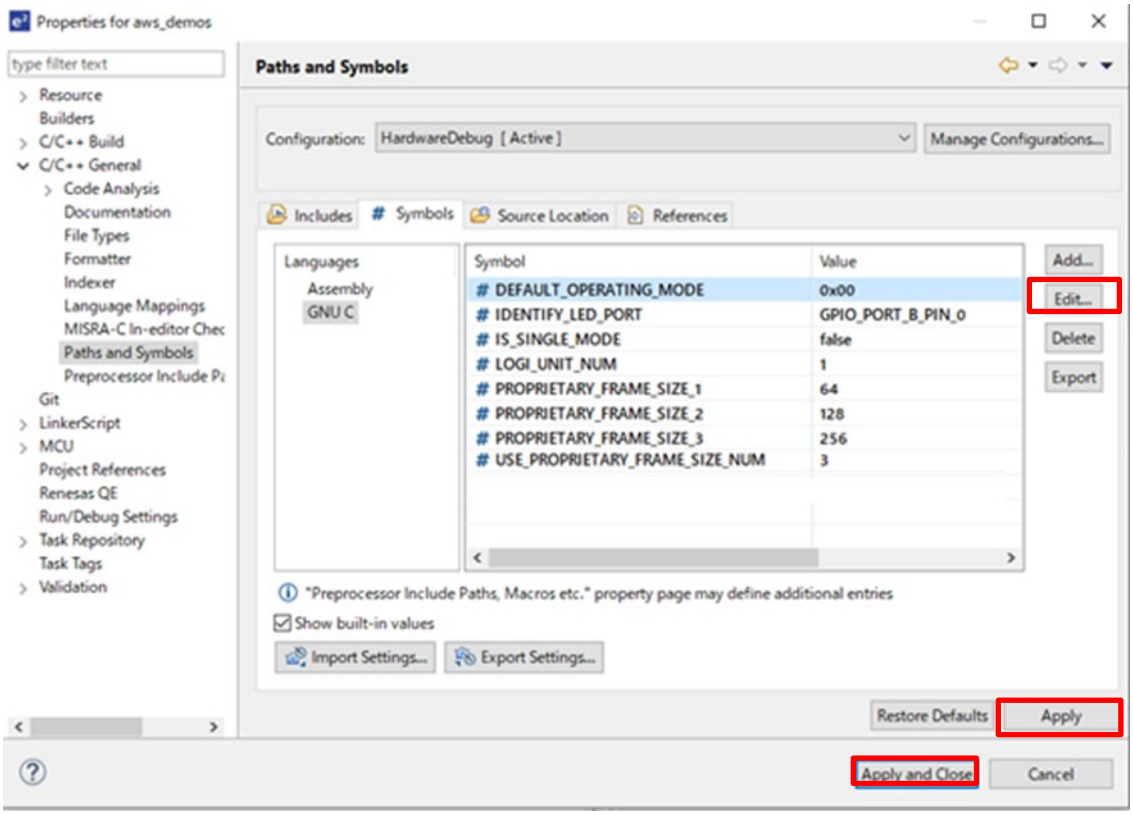

Figure 41 How to change the setting of each symbol

(3) 複数 Logical Unit へ変更する場合

本デモプロジェクトでは、Logical Unit を単体で定義しています。

複数の Logical Unit で使用する場合は、Logical Unit 使用数のシンボル値と実体の初期化定義を追加する必要があります。

以下に本デモプロジェクトの Logical Unit の定義サンプルコードを示します。

```
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,
        },
};
```

複数 Logical Unit に変更する場合の例を以下に示しますので、目的に合わせて変更・使用してください。

- Logical Unit を 3 つ定義して使用する場合の変更手順

1. 「パスおよびシンボル」画面で LOGI_UNIT_NUM を 3 に変更

2. static st_appctrl_unit_t gs_logi_unit[LOGI_UNIT_NUM]の初期化定義を以下のように追加

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,
        },
};
```

注：複数 Logical Unit の場合、Logical Unit の Active 設定に注意が必要です。Active 設定に関しては、IEC62386-103 規格書をご確認ください。

(4) エラー処理のサンプルコード

DALI の規格では、Application Controller の状態を公開することが規定されており、その中にエラー状態が含まれています。エラー状態はエラーの発生有無とエラーの詳細があり、エラーの詳細内容の定義はユーザ依存となります。
本デモプロジェクトでは Application Controller ライブラリを使用するため、エラー情報を通知・登録する必要があります。

Application Controller ライブラリへのエラー通知用として、以下のサンプルコードがありますので目的に合わせて変更・使用してください。

```
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) 識別処理のサンプルコード

DALI の規格には接続された機器の識別を行う「IDENTIFY」動作があります。この動作は、使用する Control Device や Control Gear のハードウェア環境によって動作は異なるため、ユーザが実装する必要があります。

本デモプロジェクトは以下のサンプルコードがありますので、ユーザの目的に合わせて変更・使用してください。

```
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) Operating Mode 別動作のサンプルコード

DALI の規格には「Operating Mode」があります。規格の基本動作のほかに、ユーザの独自動作を行いたい場合に Operating Mode を別途設定することで動作させることができます。

以下のサンプルコードは、Frame を受信した際に Operating Mode 別に Frame を解釈するなどの際にユーザの目的に合わせて変更・使用してください。

```
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)
   {

      /*****中略*****/

   }
#if 0
  }
  else
  {
   /* user-defined processing */
  }
#endif

  return retval;
}
```

3. メモリバンクの実装

(1) メモリバンクの設定値

Application Controller が保有する仕様の一つとしてメモリバンクと呼ばれるものがあり、IEC62386-103 規格書「9.10 Memory banks」に規定されています。メモリバンクを構成する一つ一つのデータ値はユーザ依存となっており、Logical Unit ごとに定義する必要があります。

本デモプロジェクトでは、実装が必要なメモリバンク 0、メモリバンク 1 を下表のように設定しております。

Table 60 Macro definition

Macro	Value	Description
NO_CHANGE	(-1)	リセットされないロケーションの resetValue 値
UNUSED_VALUE	(0x00)	未実装のロケーションの defaultValue 値
MBANK_NUM	(2)	メモリバンクの数
MBANK0_SIZE	(0x1B)	メモリバンク 0 の使用するサイズ
MBANK1_SIZE	(0x11)	メモリバンク 1 の使用するサイズ

Table 61 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 62 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) メモリバンクの操作関数

本デモプロジェクトで実装する Application Controller では Application Controller ライブラリを使用しているため、メモリバンクに対する操作関数を実装して Application Controller ライブラリに登録を行う必要があります。

本デモプロジェクトでは、メモリバンクを操作する関数を以下のように実装を行い、ライブラリに登録しています。

(a)コールバック関数型構造体

Application Controller ライブラリへ登録する際に使用する構造体を以下に示します。

Table 63 メモリバンクアクセスコールバック関数型構造体(st_appctrl_mbank_callback_t)の定義

```
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 64 ライブラリに登録する関数一覧

登録先	関数名	説明
*p_Reset	r_callback_mbank_reset	メモリバンクの RESET 対象データを RESET 値に設定
*p_Read	r_callback_mbank_read	メモリバンクを読み込みます。
*p_Write	r_callback_mbank_write	メモリバンクに書き込みます。
*p_UnlatchRead	NULL ※	データラッチ(保持)を解除します。
*p_CancelWrite	NULL ※	マルチバイトデータ書き込みキャンセルします、
※本デモプロジェクトではマルチバイトデータを使用していないため、実装していない関数となります。		

(b)関数仕様

- r_callback_mbank_reset

引　数	uint8_t unit 　logical unit の番号 uint8_t bank 　リセット対象とするメモリバンク番号
戻り値	void
説　明	引数にて指定したメモリバンク番号の RESET 対象データを RESET 値に設定します。 メモリバンクの MEMORY TYPE によって RESET 処理は異なります。 ROM の場合：何もしない RAM 場合：Lock バイトの状態により RESET 値を設定します。 NVM の場合：Lock バイトの状態により RESET 値を設定後、NVM 保存要求を ON します。

- r_callback_mbank_read

引　数	uint8_t unit 　logical unit の番号 uint8_t bank 　読み出し対象とするメモリバンク番号 uint8_t location 　読み出すメモリバンクの指定アドレス
戻り値	int16_t 　0x00 - 0xFF：読み出し値 　DALI103_APPCTRL_MBANK_BANK_IS_NOT_IMPLEMENT：指定したメモリバンクが未実装 　DALI103_APPCTRL_MBANK_LOCATION_IS_NOT_IMPLEMENT：指定したロケーションが未実装
説　明	引数にて指定したメモリバンク番号の指定アドレスのデータを取得します。 メモリバンクの MEMORY TYPE によって READ 処理は異なります。 ROM の場合：Default 値を返します。 RAM,NVM の場合：読み出した値を返します。 以下の場合はエラーを返します。 ● 指定したメモリバンクが未実装の場合： 　DALI103_APPCTRL_MBANK_BANK_IS_NOT_IMPLEMENT を返します。 ● 指定したメモリバンクのアドレスが未実装の場合： 　DALI103_APPCTRL_MBANK_LOCATION_IS_NOT_IMPLEMENT を返します。

- r_callback_mbank_write

引　数	uint8_t unit 　logical unit の番号 uint8_t bank 　書き込み対象とするメモリバンク番号 uint8_t location 　書き込むメモリバンクの指定アドレス uint8_t data 　書き込みデータ
戻り値	int16_t 　0x00 - 0xFF：書き込んだデータ値 　DALI103_APPCTRL_MBANK_BANK_IS_NOT_IMPLEMENT：指定したメモリバンクが未実装 　DALI103_APPCTRL_MBANK_LOCATION_IS_NOT_IMPLEMENT：指定したロケーションが未実装 　DALI103_APPCTRL_MBANK_EXECUTE_ERROR：実行エラー
説　明	引数にて指定したメモリバンク番号の指定アドレスに対し、データを書き込みます。 メモリバンクの MEMORY TYPE が RAM または NVM の時のみ書き込みを行います。 また、NVM の場合は書き込み後に NVM 保存要求を ON します。 以下の場合はエラーを返します。 ● 指定したメモリバンクが未実装の場合： 　DALI103_APPCTRL_MBANK_BANK_IS_NOT_IMPLEMENT を返します。 ● 指定したメモリバンクのアドレスが未実装の場合： 　DALI103_APPCTRL_MBANK_LOCATION_IS_NOT_IMPLEMENT を返します。 ● 指定したメモリバンクの MEMORY TYPE が ROM の場合： ● 指定したメモリバンクが読み取り専用状態の場合： 　DALI103_APPCTRL_MBANK_EXECUTE_ERROR を返します。

4.6.3　機能アプリケーション部

1. GUI アプリケーションタスク

GUI アプリケーションタスクは DALI マスタコントローラ GUI と Frame の送受信を行うタスクで、SCI 通信タスクと応答タスクの 2 つのタスクがあります。

SCI 通信には対応 FIT モジュール「RX ファミリ SCI モジュール FIT 」を使用します。

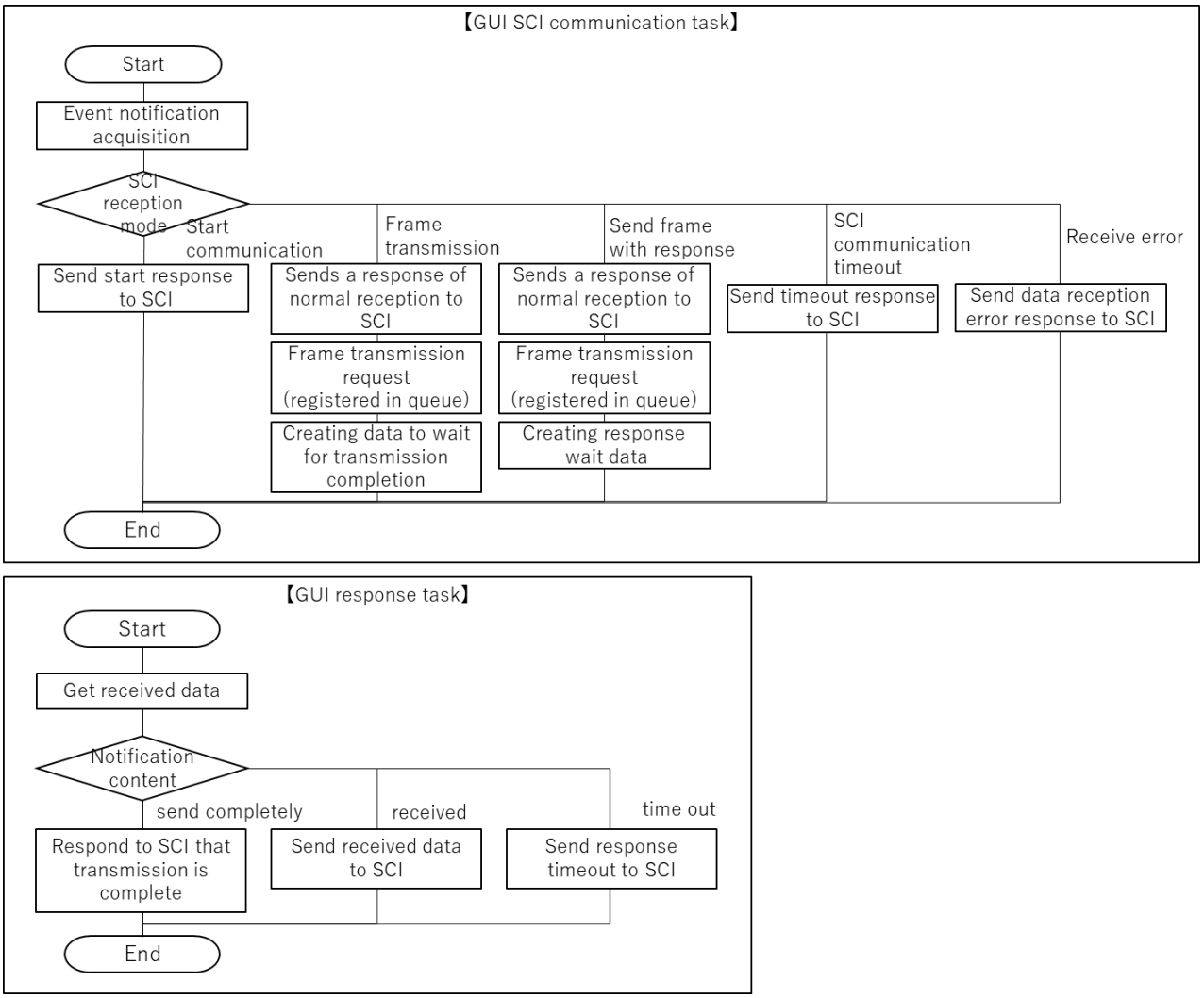

Figure 42 GUI application task flow diagram

- SCI 通信タスク
  DALI マスタコントローラ GUI と SCI 通信を行い、Frame の受信と DALI 通信ミドルウェアタスクへの送信要求を行います。送信要求を行う際、Frame の送信完了または Backward の応答待ち状態を作成します。

- 応答タスク
  SCI 通信タスクで作成された「送信完了待ち」「Backward 応答待ち」の状態と Frame の送信結果から、DALI マスタコントローラ GUI に SCI 通信にて応答を返します。

Frame の送信結果の取得は Application Controller アプリケーションタスクの Frame 受信処理から Queue による通知で行われます。

(1) 関数仕様

- r_gui_recv_com_task

関数名	void r_gui_recv_com_task (void * pvParameters)
引　数	void * pvParameters Free-RTOS のタスクハンドラ
戻り値	void
説　明	タスク通知を受け取った際、DALI マスタコントローラ GUI への応答と DALI 通信ミドルウェアタスクの送信 Queue に SCI 通信で取得した Frame を登録します。 • GUI_MESSAGE_START 応答としてバージョンを DALI マスタコントローラ GUI へ返します。 • GUI_MESSAGE_REQ_SEND_ONCE • GUI_MESSAGE_REQ_SEND_TWICE 応答として NORMAL を DALI マスタコントローラ GUI へ返した後、SCI 受信データを送信 Queue へ登録します。 ※送信 Queue への登録が行えなかった場合は再度 NORMAL を返します。 • GUI_MESSAGE_REQ_NEED_ANSWER 応答として NORMAL を DALI マスタコントローラ GUI へ返した後、SCI 受信データを送信 Queue へ登録します。 ※送信 Queue への登録が行えなかった場合は、再度 NORMAL を返します。 • GUI_MESSAGE_REQ_DATA_ANSWER 応答として NORMAL を DALI マスタコントローラ GUI へ返した後、SCI 受信データを送信 Queue へ登録します。 ※送信 Queue への登録が行えなかった場合は、何も返さずタイムアウトを発生させます。 • GUI_MESSAGE_REPLY_ERROR_TIMEOUT SCI による Request 受信が 3byte 以下で受信タイムアウトが発生した場合、 タイムアウトエラー(0xE2) を DALI マスタコントローラ GUI へ返します。 • 上記以外の場合：受信データ異常となるため、受信エラー(0xE1) を応答として DALI マスタコントローラ GUI へ返します。

- r_gui_report_com_task

関数名	void r_gui_report_com_task (void * pvParameters)
引　数	void * pvParameters Free-RTOS のタスクハンドラ
戻り値	void
説　明	受信 Queue に対象のデータが格納された際に、DALI マスタコントローラ GUI へ受信データを返します。 • 送信完了待ちの状態で通知が来た場合： 送信した Frame が GUI_MESSAGE_REQ_SEND_ONCE 又は GUI_MESSAGE_REQ_SEND_TWICE の時、送信完了したタイミングで DALI マスタコントローラ GUI へ NORMAL を返します。 • Frame 応答待ちの状態でタイムアウト通知が来た場合： 送信した Frame が GUI_MESSAGE_REQ_NEED_ANSWER のとき、DALI マスタコントローラ GUI へ NORMAL を返します。 • Frame 結果待ちが GUI_MESSAGE_REQ_NEED_ANSWER、 GUI_MESSAGE_REQ_DATA_ANSWER の場合： 受信 Queue から取得したデータを DALI マスタコントローラ GUI へ返し、Frame の結果待ち状態をクリアします。

2. AWS Web アプリケーションタスク

AWS Web アプリケーションタスクは AWS Web アプリケーションと Frame の送受信を行うタスクで、AWS-DALI 通信タスクと Logger タスクの 2 つのタスクがあります。

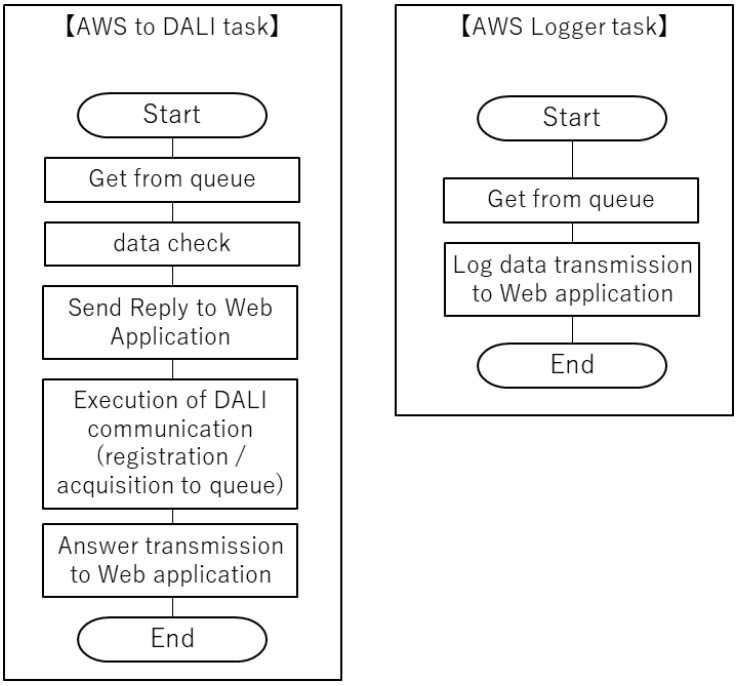

Figure 43 AWS Web application task flow diagram

- AWStoDALI タスク
  MQTT からの Queue による通知によって動作を開始します。Frame のチェック結果を元に REPLY メッセージを生成して、MQTT に対し Publish します。その後 DALI 通信を実行し、Application Controller アプリケーションによる受信 Queue の通知内容を元に ANSWER メッセージを生成して MQTT に対し Publish します。この後、再度 MQTT からの Queue による通知を待ちます。

- Logger タスク
  DALI BUS 上に出力されている Frame を取得し、MQTT に対し Publish します。

(1) トピック

MQTT の通信にはトピック(データの送信先、及び Subscribe 登録先）が必要となります。

トピック一覧を下表に示します。

Table 65 Topic List

Publisher	Subscriber	トピック名
WEB アプリ	RX65N	rx65n-cloud-kit-dali/<thingName>/toDevice
RX65N	WEB アプリ	rx65n-cloud-kit-dali/<thingName>/fromDevice

RX65N	WEB アプリ	rx65n-cloud-kit-dali/<thingName>/logger

※thingName は、AWS 上で rx65n-cloud-kit を識別するための名前（モノの名前）です。

(2) 関数仕様

- prvAWStoDALITask

関数名	static void prvAWStoDALITask(void * pvParameters)
引　数	void * pvParameters 　Free-RTOS のタスクハンドラ
戻り値	void
説　明	MQTT から Queue による通知を受け取った際、MQTT 経由で AWS Web アプリケーションへ受信できたかの応答と DALI 通信ミドルウェアタスクの送信 Queue に取得した Frame を登録します。 Application Controller アプリケーションの Frame 受信処理から Queue による通知を受けた際、応答データを作成して MQTT 経由で AWS Web アプリケーションへ返します。 通知された Frame の内容が異常値の場合は DALI 通信は行わずにエラーを MQTT 経由で AWS Web アプリケーションへ返します。

- prvDALILoggerTask

関数名	static void prvDALILoggerTask(void * pvParameters)
引　数	void * pvParameters 　Free-RTOS のタスクハンドラ
戻り値	void
説　明	Application Controller アプリケーションの Frame 受信処理から Queue による通知で受け取った Frame を MQTT 経由で AWS Web アプリケーションに送信します。 Logger 機能が有効設定となっていない場合、Queue による通知は行われません。

4. マトリクスボタンアプリケーションタスク

マトリクスボタンアプリケーションタスクは、マトリクスボタン入力で Frame 送信(16bit Forward Frame のみ)を行うタスクと、AWS Web アプリケーションからマトリクスボタンに割り当てられている Frame を変更または読み出しをするタスクの 2 つのタスクがあります。
割り当てられた Frame はデータフラッシュへ保存される為、起動ごとに設定変更を行う必要はありません。
対応 FIT モジュール「RX ファミリ　GPIO モジュール FIT」と「RX ファミリ　フラッシュモジュール FIT」を使用します。

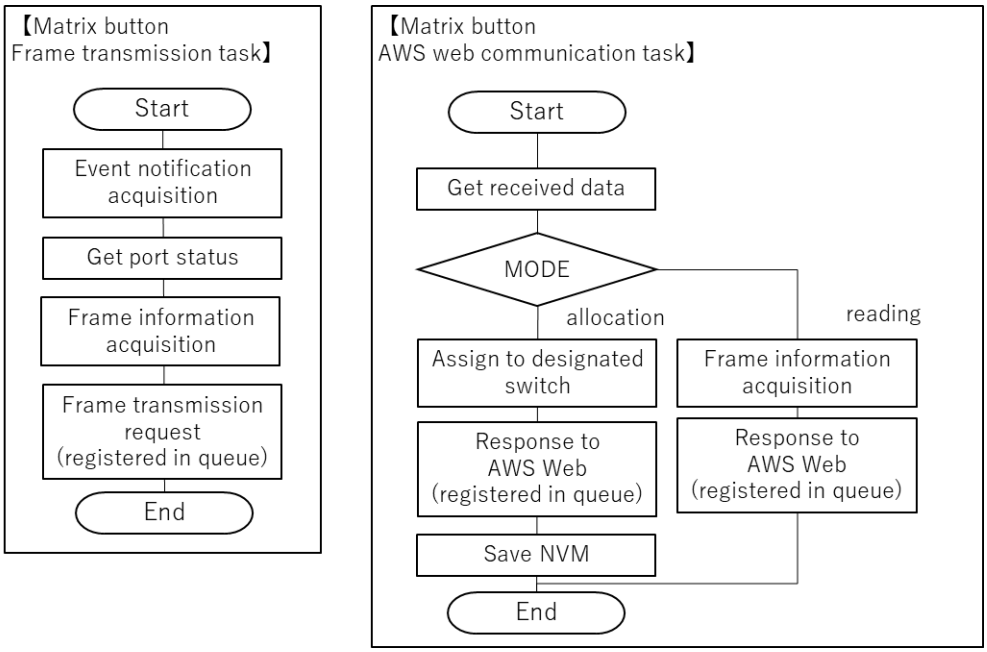

Figure 44 Matrix button application task flow diagram

- Frame 送信タスク
  マトリクスボタンが押されていたら、そのマトリクスボタンに割り当てられている Frame の送信要求を行います。

- AWS Web 通信タスク
  AWS Web アプリケーションから受け取った MODE によって処理を行います。
  割り当て：指定された Frame をマトリクスボタンに割り当てます。
  読み出し：指定されたマトリクスボタンに割り当てられている Frame を AWS Web アプリケーションに返します。

(1) 関数仕様

- r_btn_recv_com_task

関数名	void r_btn_recv_com_task (void * pvParameters)
引　数	void * pvParameters Free-RTOS のタスクハンドラ
戻り値	void
説　明	AWS Web アプリケーションから指定されたマトリクスボタンに対して、Frame の設定または Frame 情報を返します。 Frame の設定 • マトリクスボタンに Frame を設定した結果(OK/NG)を AWS Web アプリケーション通知用 Queue に登録します。 • NVM 書き込みフラグを ON にして、NVM 書き込み処理を行います。 Frame の読み出し • マトリクスボタンに割り当てられている Frame を取得し、AWS Web アプリケーション通知用 Queue に登録します。

- r_btn_sw_input_task

関数名	void r_btn_sw_input_task (void * pvParameters)
引　数	void * pvParameters Free-RTOS のタスクハンドラ
戻り値	void
説　明	10ms 定期のイベント通知を受け取った際、マトリクスボタンの入力状態を各ピンから判断します。 マトリクスボタンの入力があった場合、該当する番号から Frame を取得して、DALI 通信ミドルウェアの送信 Queue に Frame を登録します。 マトリクスボタンの入力状態により、Frame の送信回数が異なります。 ・短押し：Frame を 1 回送信 ・長押し：Frame を繰り返し送信 ・複数押し：無効

5. ユーザ実装による動作

本デモプロジェクトでは、以下にユーザが独自に実装できる空のソースファイル、ヘッダファイルを用意しています。

Table 66 File name list

File	Description
application_code¥application_controller¥dali_app¥user_app¥	
r_user_app.c	ユーザ実装を行えるソースファイルです。
r_user_app.h	ユーザ実装を行えるヘッダファイルです。

「Input Device から受信した Event や別の Application Controller から受信した Frame を解析することで DALI ネットワーク環境を把握し、同ネットワークを管理するために送信すべき Forward Frame を決定する」などの DALI 規格で規定されていないアプリケーションを実装する場合に使用してください。

4.6.4　メイン・初期化

Application Controller の初期化処理を下図に示します。

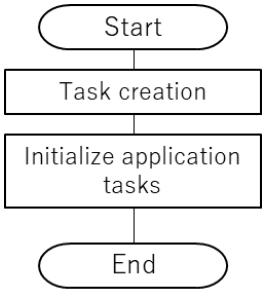

Figure 45 Initialization flow chart

1. 関数仕様

- r_rxdali_app_task_create

関数名	void r_rxdali_app_task_create (void)
引　数	void
戻り値	bool [out]: true - task 生成完了 false - task 生成失敗
説　明	Application Controller で使用するタスクの生成を行います。 Application Controller タスクと DALI 通信ミドルウェアで使用するタスクの生成をします。 以下の場合、エラーとして false を返します。 • タスク生成の戻り値が psPASS 以外

- r_rxdali_app_init

関数名	bool r_rxdali_app_init (void)
引　数	void
戻り値	bool [out]: true -初期化完了 false - 初期化失敗
説　明	Application Controller の初期化処理を行います。

	• データフラッシュの ID データサイズを指定します。 • データフラッシュの初期化を行います。 • アプリケーション層で使用する Queue を生成します。 • タスク管理用イベントハンドラを生成します。 • Application Controller アプリケーションと Application Controller ライブラリの初期化処理を行います。 • GUI アプリケーションタスクの初期化処理を行います。 • マトリクスボタンアプリケーションタスクの初期化処理を行います。 • Free-RTOS 機能のソフトウェアタイマ 1ms の生成を行います。 以下の場合、エラーとして false を返します。 • イベントハンドラの生成ができなかった場合 • 各 APP の初期化処理が失敗した場合 • 1ms ソフトウェアタイマハンドラの生成ができなかった場合

- r_rxdali_app_main

関数名	void r_rxdali_app_main (void)
引　数	void
戻り値	void
説　明	Application Controller の main 関数です。 タスク生成処理と Application Controller の初期化処理を呼び出します。 初期化処理にて生成した 1ms 定期ソフトウェアタイマを開始します。 以下の場合、エラー状態となり無限ループへ移行します。 • 初期化異常 • task 生成異常

4.6.5　タスクとタスク管理

Application Controller を動作させるタスクは、Free-RTOS のイベント通知機能を使用します。
ソフトウェアタイマのコールバックで作成する定期イベントの発生と、発生したイベントを対象のタスクに通知する通知用タスクを以下に示します。

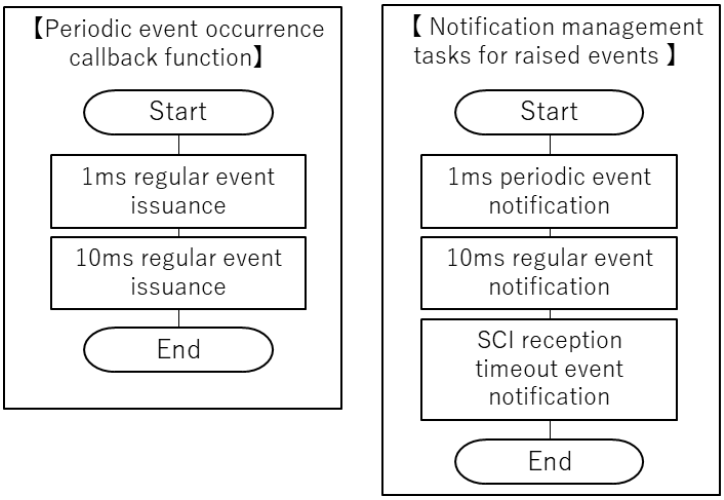

Figure 46 Task management flow diagram

1. 関数仕様

- r_app_1msec_timer_interrupt

関数名	void r_app_1msec_timer_interrupt (TimerHandle_t xTimer)
引　数	TimerHandle_t xTimer Free-RTOS ソフトウェアタイマ機能のタイマハンドラ
戻り値	void
説　明	1ms が経過した際にソフトウェアタイマのコールバック関数で 1ms 処理を行います。 1ms 割り込みは定期的に発生します。 • 1ms 経過時に r_app_1msec_interval_task へ通知を行います。 • 10ms 経過時にタスク通知用イベントを発行します。

- r_app_event_control_task

関数名	void r_app_event_control_task (void * pvParameters)
引　数	void * pvParameters Free-RTOS のタスクハンドラ
戻り値	void
説　明	タスク用通知イベントを発行された際、対象のタスクへ通知を行った後にイベントをクリアします。 • APP_EVENT_TASK_10MS_SW：r_btn_sw_input_task へ通知します。 • APP_EVENT_TASK_GUI_SCI：r_gui_recive_com_task へ通知します。 ※SCI によるイベントは SCI 通信割り込み処理内で発行します。

4.6.6　不揮発性メモリへのアクセス

Application Controller アプリケーションとマトリクスボタンアプリケーションは不揮発性メモリへのアクセスを行います。

不揮発性メモリへのアクセスは対応 FIT モジュール「RX ファミリ フラッシュモジュール FIT 」を使用します。

1. 関数仕様

- r_df_init

関数名	bool r_df_init(uint8_t id_num, callback_write_finish_t p_callback)
引　数	uint8_t id_num [in] ：使用する ID callback_write_finish_t p_callback [in]：書き込み完了時に呼び出すコールバック関数ポインタ
戻り値	bool [out]　初期化結果 true：初期化正常終了 false：初期化異常終了
説　明	データフラッシュにて使用するエリアの初期化処理を行います。

- r_df_finish

関数名	void r_df_finish(void)
引　数	void
戻り値	void
説　明	データフラッシュへのアクセスを終了します。

- r_df_read

関数名	bool r_df_read(uint8_t id, void* p_data)
引　数	e_df_id_t id [in] data id　　：読み出し対象 ID void* p_data　：読み出し格納先ポインタ
戻り値	bool [out]　読み込み結果 true：読み込み成功 false：読み込み失敗
説　明	指定された ID データをデータフラッシュから読み取ります。

- r_df_write_with_bgo

関数名	bool r_df_write_with_bgo(uint8_t id, void* p_data)
引　数	e_df_id_t id [in] data id　　：書き込み対象 ID void* p_data　：書き込みデータ格納先ポインタ
戻り値	bool [out]　書き込み結果 true：書き込み成功 false：書き込み失敗
説　明	bgo（バックグラウンド操作）を使用して、指定された ID データをデータフラッシュに書き込みます。

- r_df_handler

関数名	e_df_status_t r_df_handler(void)
引　数	void
戻り値	e_df_status_t [out]　動作状態 DF_STATUS_OK：動作可能 DF_STATUS_BUSY：ビジー状態 DF_STATUS_NG：異常発生
説　明	データフラッシュのアクセスステータスをチェックし、操作を開始します。

- r_df_is_write_process_idle

関数名	bool r_df_is_write_process_idle (void)
引　数	void
戻り値	bool true -フラッシュ書き込み処理がアイドル状態(制御可能) false-フラッシュ書き込み処理がビジー状態
説　明	データフラッシュの書き込み処理が動作中かどうか確認します。

5.　環境の構築

本デモプロジェクトで使用するソフトのインストール方法やボードの接続方法を記載します。
5.1 章～5.5 章までは操作モードによらず共通で使用します。

5.1　e2studio インストール方法

本デモプロジェクトは e2studio にて動作するため、e2studio のインストール方法を記載します。

1. e2studio インストーラをダブルクリックして e2studio インストールウィザードページを開き「Next」をクリック
2. Install Folder
   インストール先を指定する
3. Device Families
   RX デバイス・サポートを選択後、「Next」をクリック
4. Extra Components
   言語パック、RTOS サポートを選択後、「Next」をクリック
5. Components
   オプション・コンポーネントで GCC for Renesas RX Build Support にチェックが入っていることを確認し、「Next」をクリック
6. Additional Software
   GCC for Renesas RX 8.3.0 201904 コンパイラを選択し「Next」をクリック
   ※表示されない場合は以下 URL にてユーザ登録を行い、コンパイラをインストールしてください。

   GNU ツールチェーンのダウンロードサイト:
   https://gcc-renesas.com/ja/

7. License Agreement
   ライセンス契約を読んで同意した後、「Next」をクリック
8. Shortcuts
   スタートメニューに表示するショートカット名を選択し、「Next」をクリック
9. Summary
   インストールされるコンポーネントの一覧が表示されるので、内容を確認ののち「Install」をクリック
10. Installing…
    インストールの実行中には、追加ソフトウェアで選択した項目に応じてソフトウェアをインストールするためのダイアログボックスが開く為、画面の指示に従って操作を行う
11. Results
    インストールを行った結果が表示されるのでエラー等が出ていないかを確認後「Finish」ボタンをクリック

インストール、その他基本操作についての詳細は以下ユーザーズマニュアルを参照してください。

- 統合開発環境 e2studio ユーザーズマニュアル　入門ガイド(R20UT4204)

5.2　e2studio プロジェクトのインポート方法

本デモプロジェクトのインポート方法を記載します。

1. 提供プロジェクトファイルを解凍、任意の場所(フォルダ)へ配置する。
   ※パス階層が深くなるとビルドが通らなくなる可能性がありますので、配置箇所に注意してください。¥workspace¥rx65n_dali フォルダを C ドライブ直下などに移動することを推奨します。
2. e2studio を起動し、プロジェクト・エクスプローラで右クリックまたは「ファイル」タブ→「インポート」を選択
3. 「一般」から「既存プロジェクトをワークスペースへ」を選択し、「次へ」をクリック
4. 「ルート・ディレクトリーの選択」を選択し、「参照」から 1.で配置したプロジェクトファイルを選択
5. 「/projects/renesas/rx65n-cloud-kit-uart-sx-ulpgn/e2studio-gcc」のみを選択
6. Aws_demos と Boot の両方が選択されていることを確認
7. 「ワークスペースにプロジェクトをコピー」のチェックを外す
8. 「終了」をクリック

5.3　EZ-0012 の設定方法

EZ-0012 は DALI 通信の他に DMX512 通信、赤外線通信機能を搭載した照明制御用評価ボードです。
ソフトウェア自動生成ツール Applilet EZ for HCD Controller、EZ-0012 用 DALI-2 拡張ボードを使う場合は Applilet EZ for DALI Control Gear を使用して、DALI 通信を行えるコードの生成/書き込みを行った後、Control Gear として動作させることが可能です。本章では以下のユーザーズマニュアルを使用して設定方法の説明を行います。

- Applilet EZ for HCD Controller　ユーザーズマニュアル(r20ut0435)
- Applilet EZ for DALI Control Gear　ユーザーズマニュアル(r11ut0078)
-

注意 : EZ-0012 と EZ-0012 用 DALI-2 拡張ボード，またはそれ以外のボードを使用する場合は、それぞれのユーザーズマニュアルを参照ください。

5.3.1　Applilet EZ for HCD Controller のインストール

Applilet EZ for HCD Controller のインストールにはユーザーズマニュアルの 2 章を参照してください。

5.3.2　Applilet EZ for HCD Controller によるコードの生成/書き込み

Applilet EZ for HCD Controller ユーザーズマニュアルの内容をもとに、DALI 通信を行うための操作・設定方法を記載します。

1. Applilet EZ for HCD Controller を起動し、初回起動設定を行います。(ユーザーズマニュアル 3 章)
2. 評価ボードの設定を行います。(ユーザーズマニュアル 4.2 章)
   評価ボードの設定内容例を下図に示します。

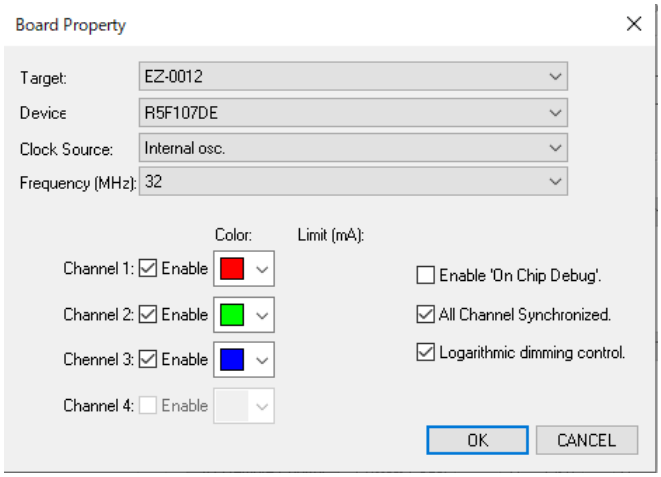

Figure 47 Board Property

3. モードの設定を行います。(ユーザーズマニュアル 4.3.6 章)
   モード設定内容例を下図に示します。

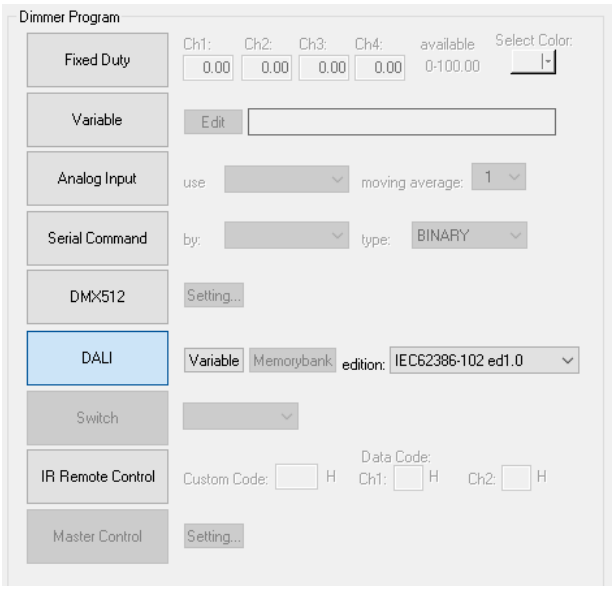

Figure 48 Dimmer Program

4. 生成と書き込みを行います。(ユーザーズマニュアル 4.4 章)

5.3.3　EZ-0012 の動作モード設定

DALI 通信を行うには、設定スイッチ(SW1,SW2)の状態を初期設定時から変更する必要があります。

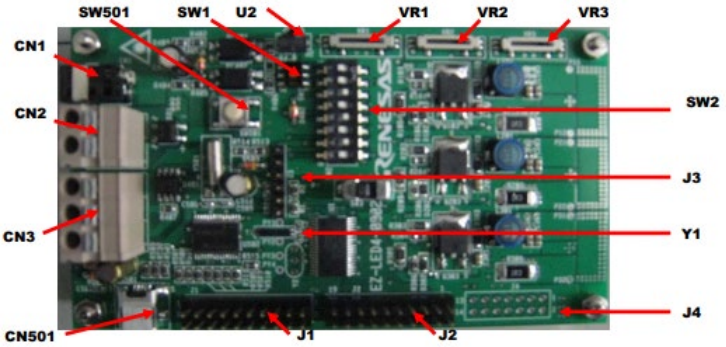

Figure 49 RL78 / I1A DC / DC LED Control Evaluation Board Components

1. 設定スイッチの設定
   SW1 は下図の「DALI」側へ変更、SW2 は下表の設定に変更してください。

Figure 50 Setting switch SW1

Table 67 Setting switch SW2

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

その他、EZ-0012 について詳細を知りたい方は、以下ユーザーズマニュアルを参照してください。

- EZ-0012　RL78/I1A DC/DC LED 制御評価ボード　ユーザーズマニュアル(r01uh0363)

注意 ：EZ-0012 と EZ-0012 用 DALI-2 拡張ボード，またはそれ以外のボードを使用する場合は、それぞれのユーザーズマニュアルを参照ください。

5.4　ボード接続方法

本デモプロジェクトで使用するボードの接続方法を下図に示します。

RX65N Cloud kit + DALI-2 オプションボードの電源供給は USB 接続(ECN1)、EZ-0012 の電源供給は AC アダプタ電源接続で行うものとし、操作モードによらず配線接続方法は共通となります。

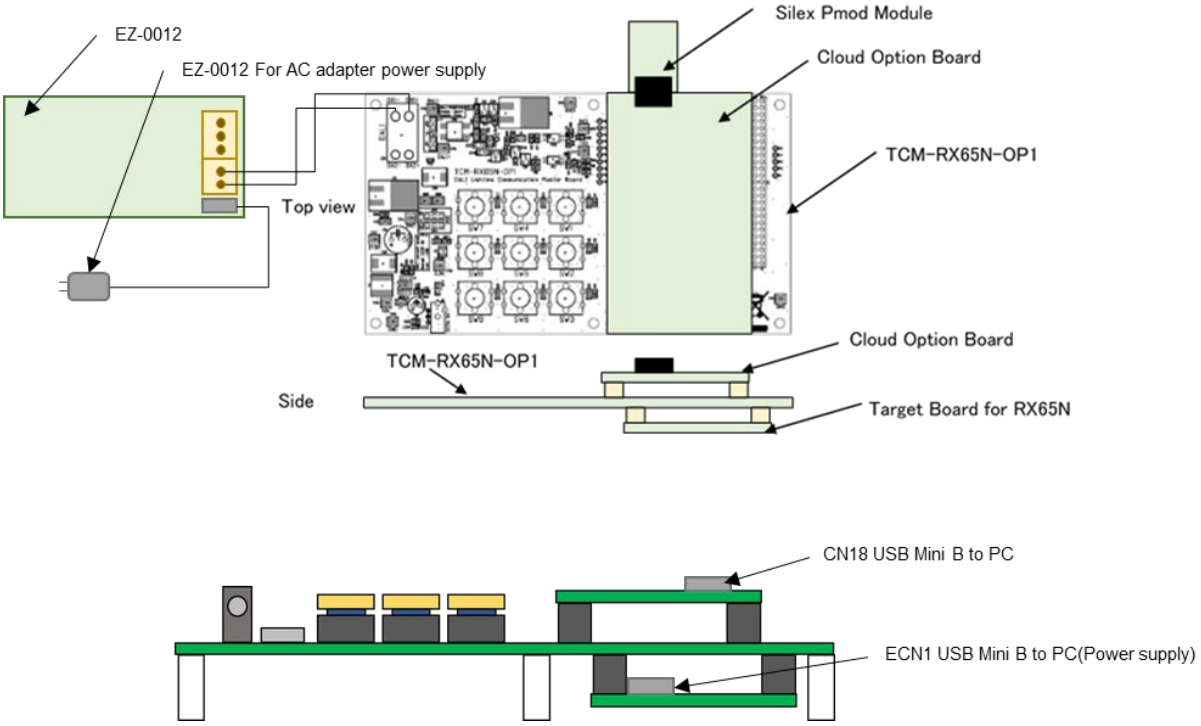

Figure 51 Overall hardware configuration diagram

USB 接続による電源供給設定とする為、RX65N DALI-2 オプションボードのジャンパ接続を以下の状態に変更してください。

- JP1 と JP2 が各々ジャンパ接続
- JP3 と JP4 が各々USB 側へジャンパ接続

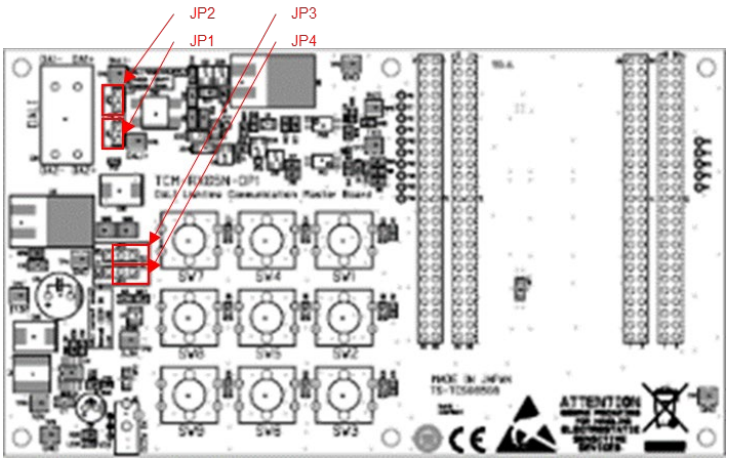

Figure 52 RX65N DALI-2 オプションボード jumper connection

5.5　デモプロジェクトの実行

5.5.1　ビルドオプションの設定

1. プロジェクト名を右クリックし、メニュー表示→「プロパティ」を選択します。
2. 「C/C++ビルド」→「設定」の「Toolchain」タブをクリックして、ツールチェーンとバージョンを確認してください。

   - ツールチェーン：GCC for Renesas RX
   - バージョン　　：8.3.0.201904

5.5.2　プロジェクトのビルド

1. プロジェクト・エクスプローラでプロジェクトを右クリックし、「プロジェクトのビルド」を選択します。
2. ビルドが開始され、「コンソール」にビルドの状況が表示されるので”Build Finished”というメッセージが表示されたらビルド完了です。

5.5.3　デバッグ

1.  ボタンをクリックしてマイクロコントローラへプログラムをダウンロードします。
2. 「実行」→「デバッグの構成…」を選択し、「デバッグ構成」ウィンドウを開きます。
3. 「デバッグ構成」ウィンドウで“aws_demos HardwareDebug” デバッグ構成の表示を展開し、既存のデバッグ構成をクリックします。
4. 「debugger」→「Connection Settings」タブに切り替え、下図の設定となっていることを確認してください。

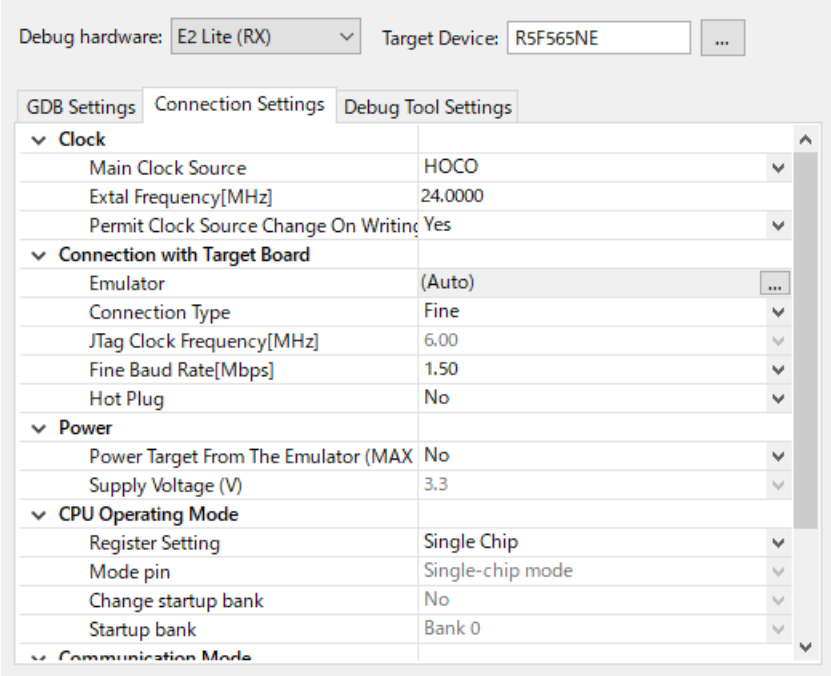

Figure 53 Debug screen settings pickup

5. 「デバッグの開始」を選択し、「デバッグ」ビュー画面が表示されたらデバッグの準備が完了です。

5.5.4　実行

1.  ボタンをクリック、または「F8」キー入力でデモプロジェクトを実行します。

その他、デバッグ画面の操作方法は以下ユーザーズマニュアルの 5.4 章を参照してください。

- 統合開発環境 e2studio ユーザーズマニュアル　入門ガイド (R20UT4204)

5.6　接続サービスまたは機器別の操作方法

5.6.1　DALI マスタコントローラ GUI

DALI マスタコントローラ GUI と動作させる場合のデモプロジェクトの動作手順を以下に示します。

1. 5.1 章～5.4 章を参照し、必要ソフトのインストールと機器の接続を行います。
2. 5.5 章を参照し、本デモプロジェクトを実行します。
3. DALI マスタコントローラ GUI をインストールします。
   インストールには以下ユーザーズマニュアルを参照してください。

   DALI マスタコントローラ GUI ユーザーズマニュアル(r20ut0715)

4. DALI マスタコントローラ GUI を起動します。
5. 「Settings」→「Serial…」をクリックし、「USB シリアルデバイス(COM x)」を選択後「OK」をクリックします。
   ※COM x には接続した USB の COM ポート番号が表示されます。COM ポート番号は接続する PC により異なります。
6. 画面左側に「Broadcast」の文字が表示されたら DALI マスタコントローラ GUI との接続は完了です。

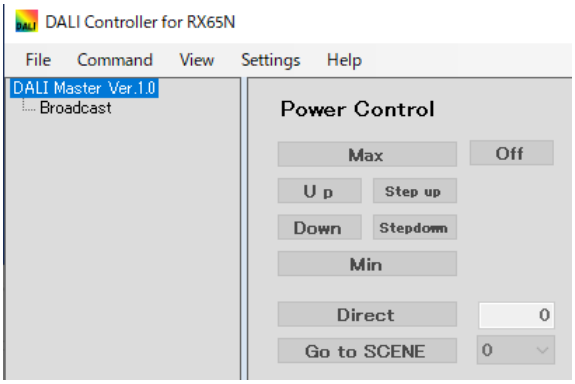

Figure 54 Excerpt from screen for DALI master controller GUI (1/2)

7. 接続されている Control Gear の認識を行います。
   ・「Command」タブ→「Random Address Allocation…」を選択後、「Start」をクリック
   ※注意画面が出ますが、「OK」を選択して続行してください。

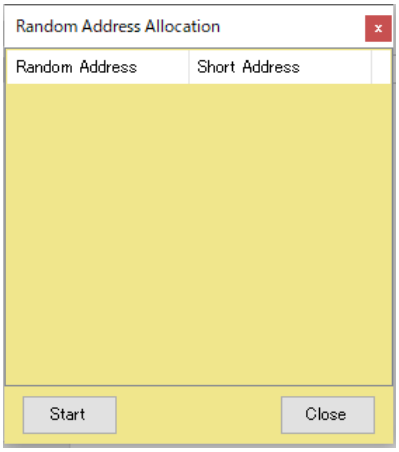

Figure 55 Screen of Random Address Allocation

・接続されている Control Gear の数分 Address が表示されたら「Close」をクリック
・下図のように画面表示が変更されたら Control Gear との接続は完了です。

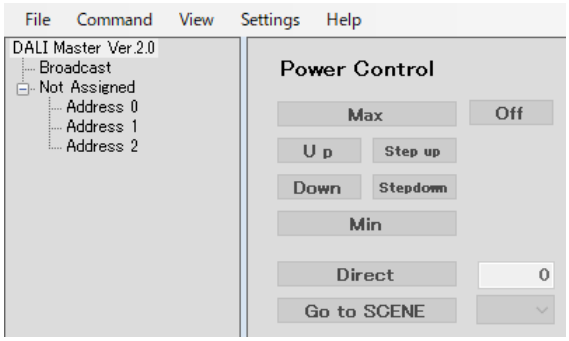

Figure 56 Excerpt from screen for DALI master controller GUI (2/2)

8. DALI マスタコントローラ GUI を操作します。
例 1：接続されている灯具全てを MAX の光量で点灯させる→Broadcast を選択後、MAX ボタンを押す
例 2：接続されている灯具全てを消灯させる→Broadcast を選択後、OFF ボタンを押す
例 3：接続されている灯具を個別に MIN の光量で点灯させる→Address X を選択後、MIN ボタンを押す
例 4：接続されている灯具を個別に徐々に明るくさせる→例 3 の状態から UP ボタンを繰り返し押す

その他、詳細な操作方法は以下ユーザーズマニュアルを参照してください。

- DALI マスタコントローラ GUI(r20ut0715)

5.6.2　AWS Web アプリケーション

・環境構築手順

AWS Web アプリケーションと動作させる場合のデモプロジェクトの動作手順を以下に示します。

1. 5.1 章～5.4 章を参照し、必要ソフトのインストールと機器の接続を行います。
2. AWS Web アプリケーションの zip ファイルを任意の場所に保存します。
3. AWS への登録・設定を行います。
   AWS Web アプリケーションとデモプロジェクトを動作させる場合、AWS アカウント、AWS IoT および Amazon Free-RTOS クラウドサービスへのアクセス権限を持つ IAM ユーザが必要です。
   また、AWS IoT に RX65N Cloud kit + DALI-2 オプションボードの情報を登録する必要があります。
   AWS アカウント及びアクセス許可の設定方法については下記 URL を参照してください。
   ユーザアカウントの作成を行います。
   https://docs.aws.amazon.com/ja_jp/freertos/latest/userguide/freertos-account-and-permissions.html
   次に、以下 URL を参照して AWS IoT にボードを登録します。
   手動でボードを接続する手順に従ってください。
   https://docs.aws.amazon.com/ja_jp/freertos/latest/userguide/freertos-prereqs.html#get-started-freertos-thing.
   以下 URL を参照してソースコードを設定し、デモプロジェクトを AWS と通信させます。
   証明書の取り込み方法や Wi-Fi 設定方法を記載しています。
   https://docs.aws.amazon.com/freertos/latest/userguide/freertos-configure.html.
   証明書の取り込み方法・Wi-Fi 設定方法に関してソースコード変更を行う箇所を抜粋した内容を以下に示します。

4.　証明書とプライベートキーの取込み(1/4)

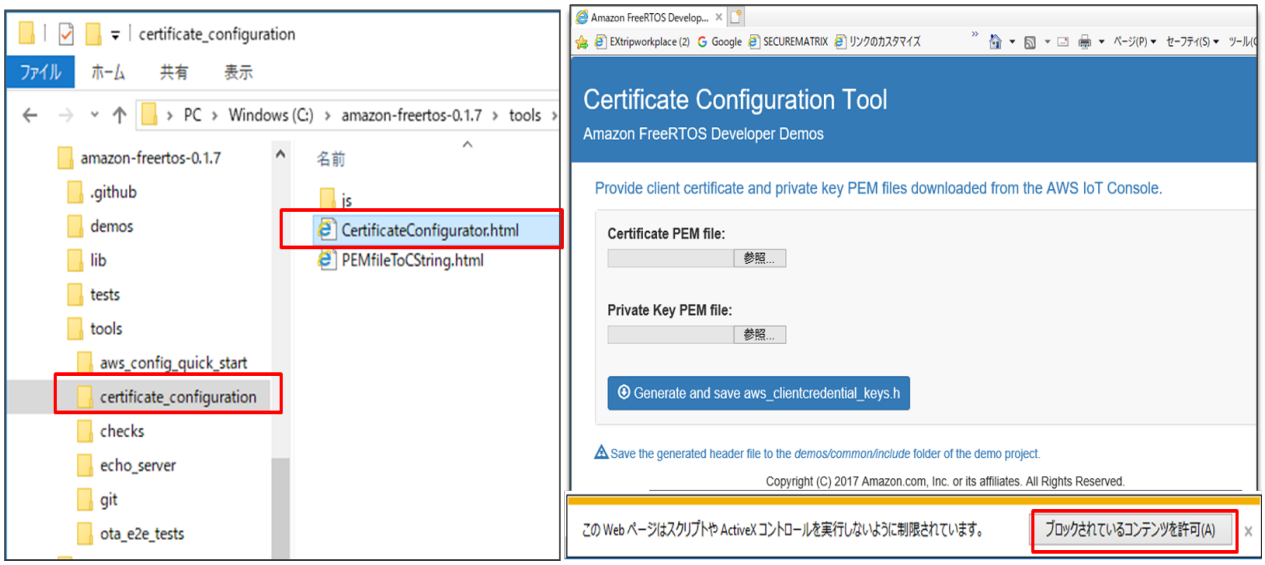

Figure 57 Import certificate and private key (1/4)

・ ${base_folder}¥tools¥certificate_configuration¥CertificateConfigurator.html をダブルクリックで起動します。
(ActiveX が制限されている旨のメッセージが表示されたら許可してください)

5.　証明書とプライベートキーの取込み(2/4)

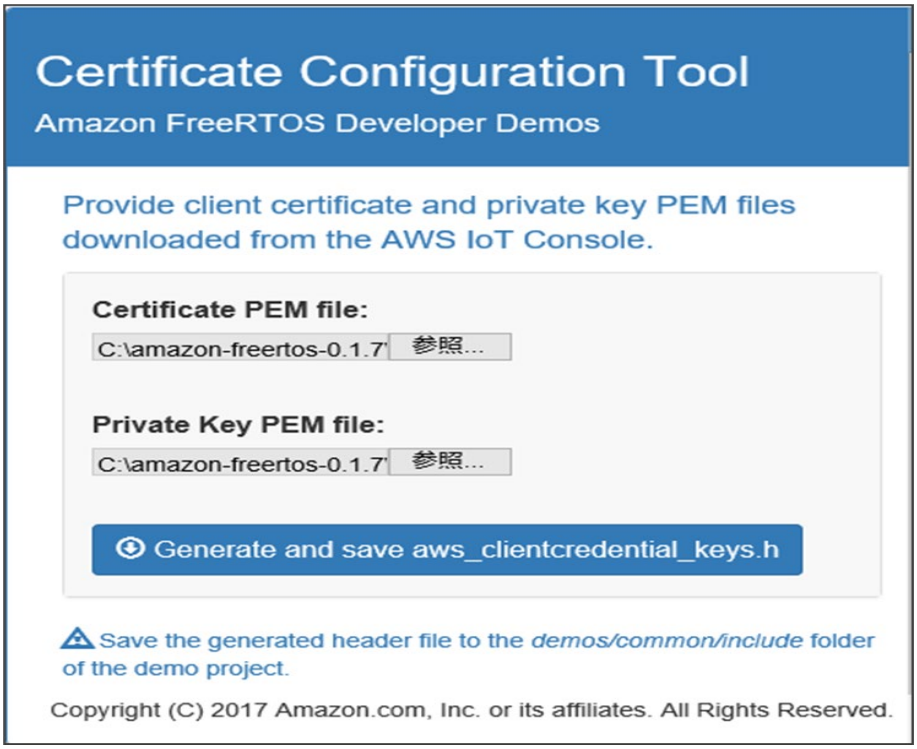

Figure 58 Import certificate and private key (2/4)

・「デバイスを AWS IoT に登録」で生成した モノ の証明書とプライベートキー ☆2 をそれぞれファイルとし、CertificateConfigurator.html　に指定します
　・xxxxxxxxxx-certificate.pem.crt　---　証明書

・xxxxxxxxxx-private.pem.key　---　プライベートキー

6.　証明書とプライベートキーの取込み(3/4)

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

・Generate and save aws_clientcredential_keys.h ボタンを押下します。
・aws_clientcredential_keys.h を任意の場所にダウンロードします

7.　証明書とプライベートキーの取込み(4/4)

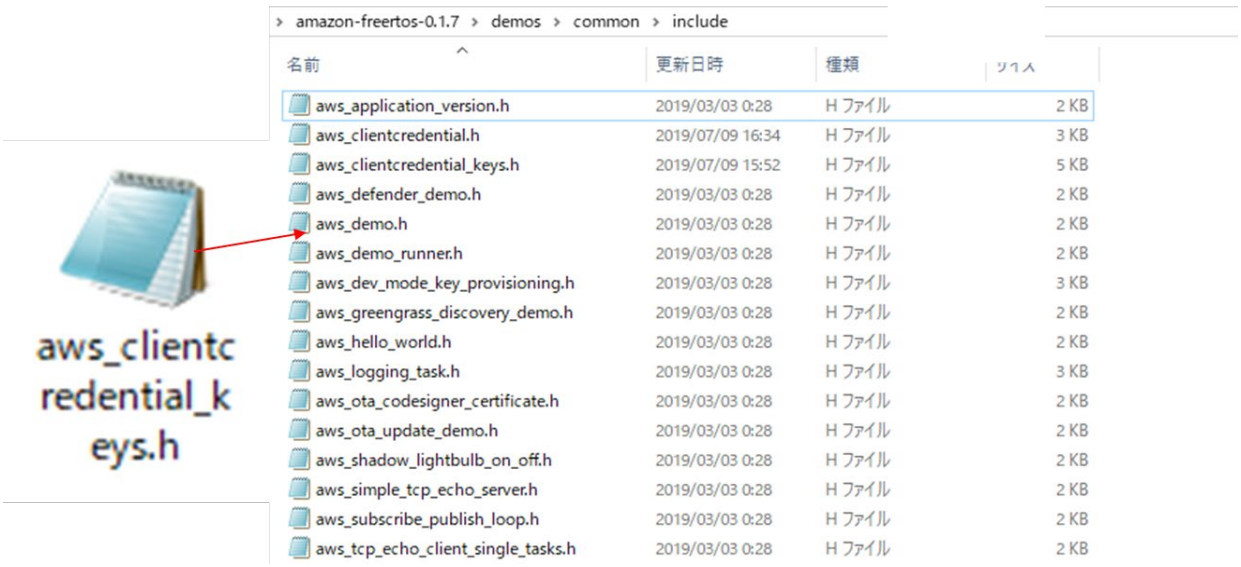

Figure 60 Import certificate and private key (4/4)

・${base_folder}¥demos¥ include¥aws_clientcredential_keys.h をエクスプローラーで開きます。
・生成された aws_clientcredential_keys.h をエクスプローラーにドラッグ&ドロップします。
・「置き換えますか?」とメッセージが表示されるので「はい」を選択します。

8.　マクロの設定を行います。

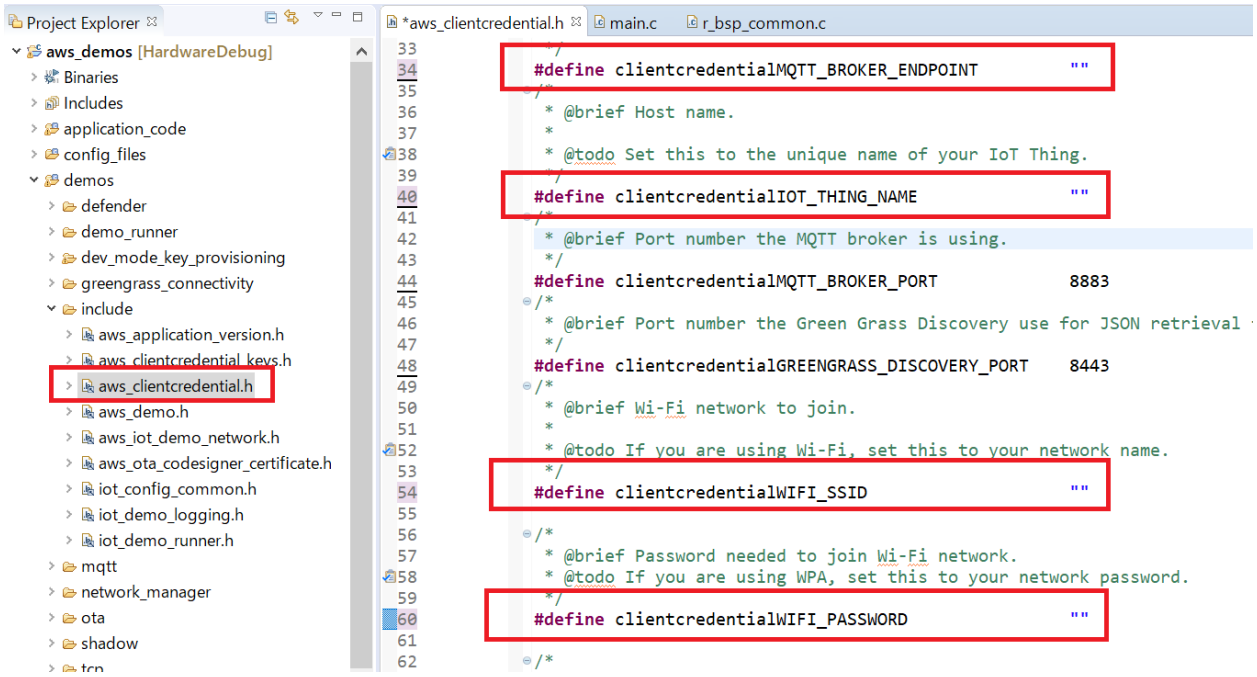

Figure 61 Macro settings

・プロジェクト・エクスプローラ -> aws_demos -> demos -> include -> aws_clientcredential.h
以下 2 種マクロに設定（“ ”の中身に入力）します。
-1.clientcredentialMQTT_BROKER_ENDPOINT
「AWS IoT エンドポイントを確認」 で確認したエンドポイント の名前☆3 を設定します。
-2.clientcredentialIOT_THING_NAME
「デバイスを AWS IoT に登録」 で登録したモノの名前☆1 を設定します。
-3.WiFi 用に以下の 2 種も設定します。
・clientcredentialWIFI_SSID
接続するアクセスポイントとの SSID
・clientcredentialWIFI_PASSWORD
接続するアクセスポイントのパスワード

9. IAM に以下のポリシーをアタッチします。
以下 URL を参照して、IAM にポリシーをアタッチします。
https://docs.aws.amazon.com/ja_jp/IAM/latest/UserGuide/access_policies_manage-attach-detach.html

＜Policy name＞

- AmazonS3FullAccess
- CloudWatchReadOnlyAccess
- AWSIoTFullAccess
- AWSLambdaFullAccess
- AutoScalingFullAccess
- AdministratorAccess-AWSElasticBeanstalk
- AmazonAPIGatewayAdministrator
- AmazonCognitoPowerUser
- IAMFullAccess

注意 1. 1 つのアカウントに対して 10 個までのポリシーが登録可能です。すでに別のポリシーが登録されており、ポリシーの登録数が 11 以上の場合はグループを作成してグループ経由でアタッチしてください。
注意 2. ポリシーのアタッチについては、与えられる権限等を確認して十分にご注意ください。

10.　AWS Web アプリケーションを使用する為に、Cognito にユーザ設定を登録します。

・AWS マネジメントコンソールにログインし、Cognito 選択を選択します。

Figure 62 Cognito　(1/10)

・ユーザープルの管理を選択します。

Figure 63 Cognito　(2/10)

・”ユーザープールを作成する”を選択します。

Figure 64 Cognito　(3/10)

・プール名に「daliweb_userpool」を入力します。

・「デフォルトを確認する」を選択します。

Figure 65 Cognito　(4/10)

・図の箇所を選択します。

Figure 66 Cognito　(5/10)

- 図の「ポリシー」を選択します。
- 図の「特殊文字を表示する」の「チェック」を解除します。
- 図の「変更の保存」を選択します。

Figure 67 Cognito　(6/10)

- 図の「プールの作成」を選択します。

Figure 68 Cognito　(7/10)

- 図のプール ID (UsetPoolId),プール ARN をメモします。(*1)
- 画面を下へ移行し、「アプリクライアント」の追加を選択します。

Figure 69 Cognito　(8/10)

- 画面が切り替わり後、「アプリクライアントの追加」を選択します。

Figure 70 Cognito　(9/10)

- 図のアプリクライアント名に「daliweb_aplcl」設定します。
- 図の「クライアントシークレットを生成」のチェックを解除します。
- 図の「アプリクライアントの作成」を選択します。

Figure 71 Cognito (10/10)

- 図のアプリクライアント ID をメモします。(ClientId) (*2)
- 図の「プールの詳細に戻る」を選択します。

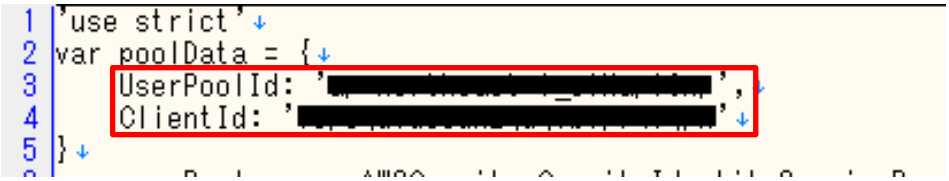

Figure 72 cognite setting

- AWS-WEB ファイル(r_aws_web.zip)を解凍します。
- cognito-auth.js をエディタで開き、以下を設定します。

・ユーザープル ID (*1)
・アプリクライアント ID(ClientId) (*2)

- ユーザープルの画面は終了です。

11. AWS マネジメントコンソールのサービスから、S3 を選択します。

Figure 73 Amazon S3 (1/9)

- 画面が移行後、図の「バケットを作成する」を選択します。

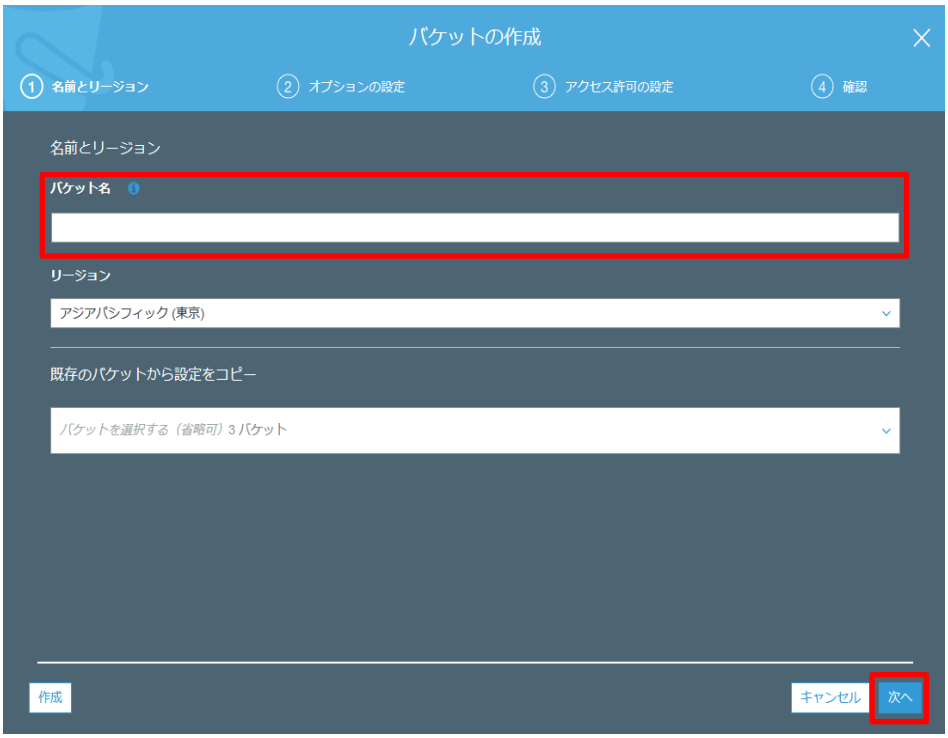

Figure 74 Amazon S3 (2/9)

- 図の一意の名称(*1)を入力します。他の AWS ユーザ含めて一意の名前でなければなりません。
  ・*1 : バケット名注意点
  ・バケット名の文字数は 3～63 文字以内です
  ・大文字、アンダースコアをバケット名に含める事はできません。
  ・命名規則の詳細は、以下の AWS 公式ガイドをご参照下さい。
- 図の次へを選択します。

Figure 75 Amazon S3 (3/9)

- 図の次へを選択します。

Figure 76 Amazon S3 (4/9)

- 図の次へを選択します。

＊ここでのパブリック設定は、あくまでテストするための例になります。
　実運用時は、お客様でこの設定は見直しをしてください。

Figure 77 Amazon S3 (5/9)

- 図の「バケット作成」を選択します。

Figure 78 Amazon S3 (6/9)

- 図の作成した「buckets」を再度選択します。

Figure 79 Amazon S3 (7/9)

- 図の「アップロード」を選択します。

Figure 80 Amazon S3 (8/9)

- 図の「ファイルを追加」を選択します。

Figure 81 Amazon S3 (9/9)

- aws 設定ファイル(aws_setting.zip)の”s3/connectId.txt”を選択します。
- 図の「アップロード」を選択します。
- S3 の設定は完了です。

12. AWS マネジメントコンソールのサービスから、lambda を選択します。

Figure 82 lambda (1/6)

- 「daliweb_onconnect」と言う関数を作成します。図の「関数の作成」ボタンをクリックします。

Figure 83 lambda (2/6)

- 図の関数名に daliweb_onconnect を入力します。
- 図のランタイムに「Python3.8.」を選択します。
- 図の「関数の作成」を選択します。

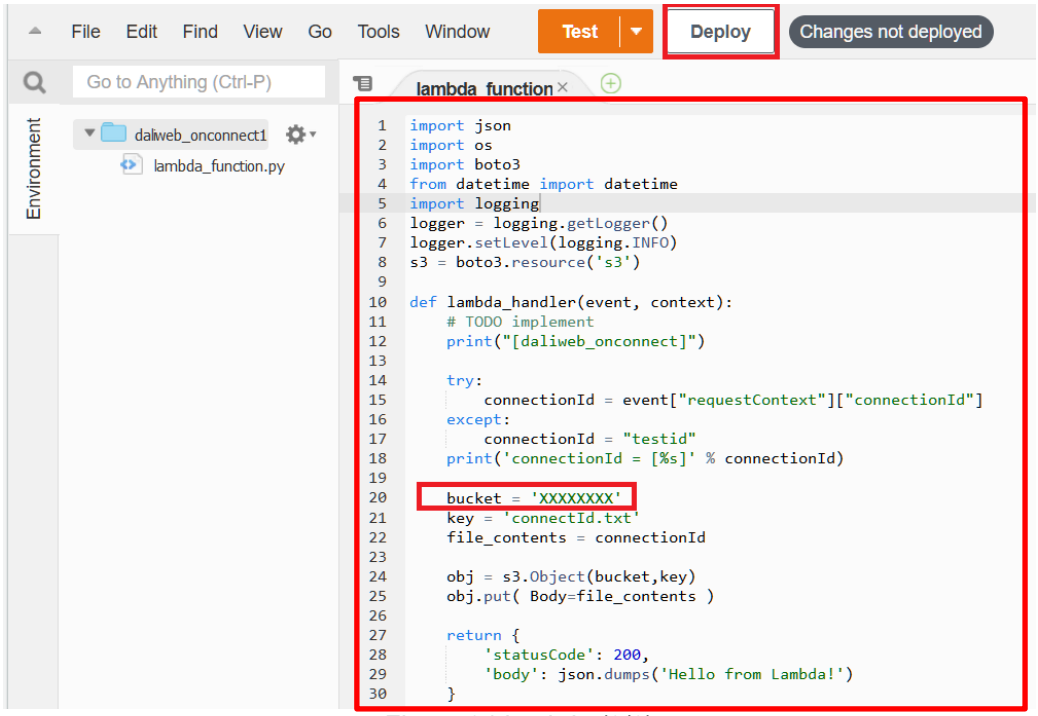

Figure 84 lambda (3/6)

- aws 設定ファイル(aws_setting.zip)の”lambda/daliweb_onconnect/lambda_function.py”をエディタで開きます。その後、内容をコピーし、関数コードの画面上に”CTRL＋V”にて、張り付けます。
- 図の黒線部分（bucket = 'XXXXXXXX'） の箇所に Amazon S3 で作成した”バケット名”を設定します。
- 図の「Deploy」を選択します。

Figure 85 lambda (4/6)

- 図の「アクセス権」を選択します。
- 図の「ロール名」を選択します。

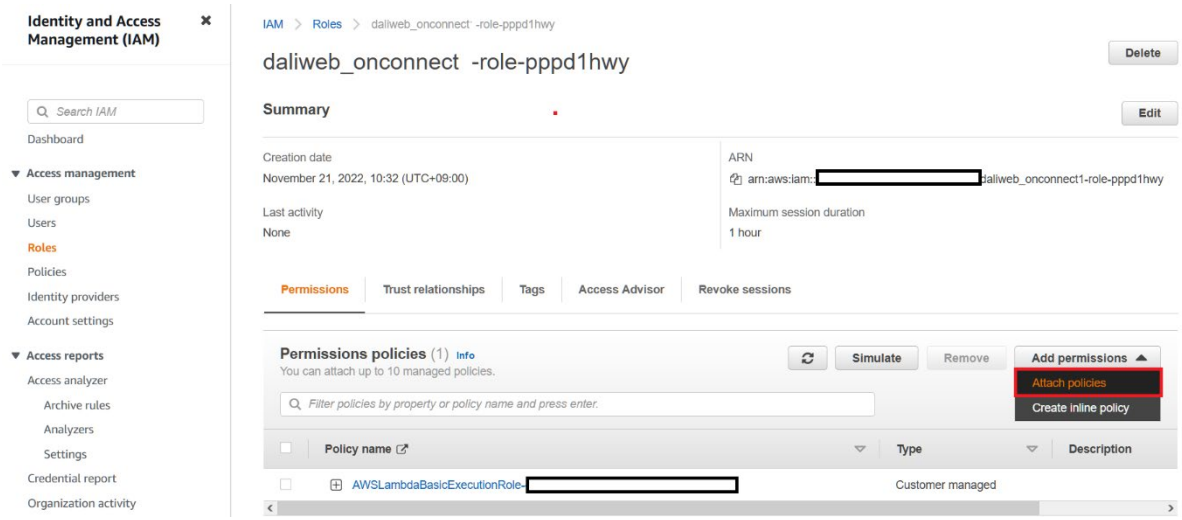

Figure 86 lambda (5/6)

- 図の「ポリシーをアタッチします」を選択します。

Figure 87 lambda (6/6)

- 図の「AmazonS3FullAccess」を選択し、「ポリシーのアタッチ」を選択します。
- 図の「AmazonS3FullAccess」がポリシーに追加されたことを確認します。

- 本章「daliweb_onconnect」と同様に、関数名「daliweb_send_Iot2web」として関数を作成します。
  aws 設定ファイル(aws_setting.zip)は、” lambda/ daliweb_send_Iot2web /lambda_function.py”を使用します。
  daliweb_send_Iot2web の場合、追加するポリシー設定は、以下を追加してください。
  ・AmazonS3ReadOnlyAccess
  ・AmazonAPIGatewayInvokeFullAccess

13. AWS マネジメントコンソールのサービスから、API Gateway を選択します。

Figure 88　API Gateway　(1/15)

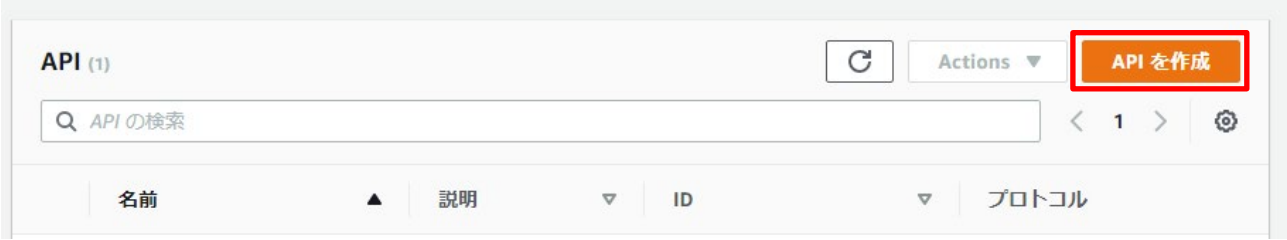

Figure 89　API Gateway　(2/15)

- 図の「API を作成」を選択します。

Figure 90　API Gateway　(3/15)

- 図の WebSocket API「構築」を選択します。

Figure 91　API Gateway　(4/15)

- 図の「OK」を選択します。

Figure 92　API Gateway　(5/15)

- API 名に「daliweb_api」を設定します。
- ルート選択式に”$request.body.action”を設定します。
- 「空の API を作成」をクリックします。
- 「作成」をクリックします。

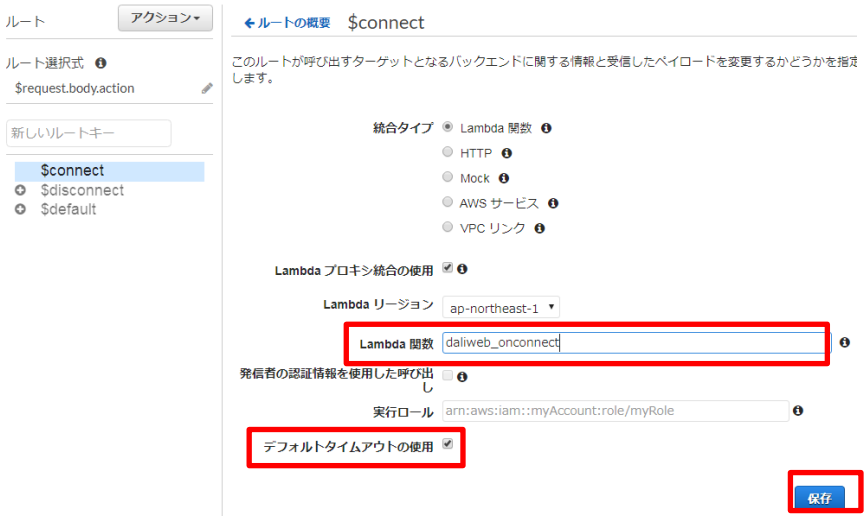

Figure 93　API Gateway　(6/15)

- 図の「connect」を選択します。
- Lambda 関数に「daliweb_onconnect」を設定します。
- 「デフォルトタイムアウト」の仕様のチェックを外します。
- 「保存」を選択します。

Figure 94　API Gateway　(7/15)

- 「Lambda 関数に権限を追加する」と表示されますので、「OK」を選択します

Figure 95　API Gateway　(8/15)

- 図の「アクション」の「API のデプロイ」を選択します。

Figure 96　API Gateway　(9/15)

- 図のデプロイされるステージを「新しいステージ」を選択します。
- 図のステージ名を「prod」を入力します。
- 図の「デプロイ」を選択します。

Figure 97　API Gateway　(10/15)

- Websocket の URL が発行されます。

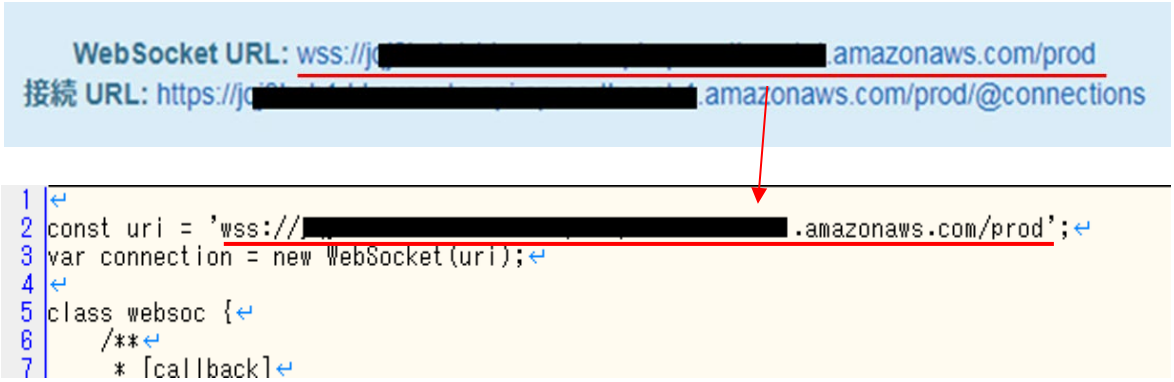

Figure 98　API Gateway　(11/15)

- 発行された WebSocket URL を websoc.js に設定します。
- websoc.js は、AWS-WEB ファイル(aws-web.zip)の“daliweb/html/websoc.js”の”const url”に設定します。

Figure 99　API Gateway　(12/15)

- 発行された接続 URL を lambda ファンクション(daliweb_send_Iot2web)に 設定します。
  (Lambda->Functions->daliweb_send_Iot2web で画面移行)

^

Figure 100　API Gateway　(13/15)

- 図の「ルート」選択します。
- 図の「新しいルートキー」に「sendmessage」を入力します。
- 「レ点」を選択します。

Figure 101　API Gateway　(14/15)

- 図の「daliweb」を入力します。
- 図の「daliweb_send_Iot2web」を選択します。
- 図の「保存」を選択します。

Figure 102　API Gateway　(15/15)

- 図の「OK」を選択します。
- API Gateway は終了です。

14. AWS マネジメントコンソールのサービスから、IoT Core を選択します。

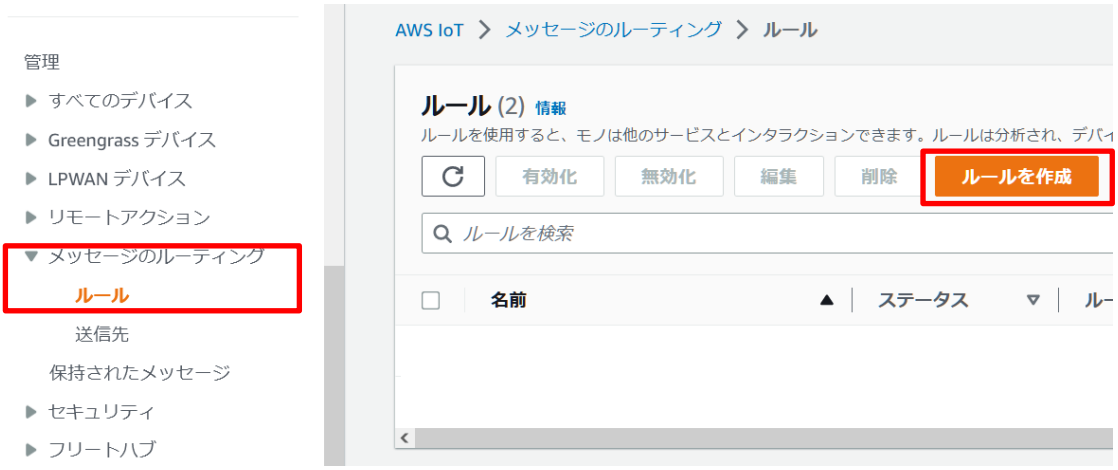

Figure 103 Aws IoT　(1/7)

- 「メッセージのルーティング」の「ルール」を選択します.
- 「作成」を選択します。

Figure 104 Aws IoT　(2/7)

- 名前に「daliweb_IoT2web_fromDevice」を設定し、「次へ」を選択します。

Figure 105 Aws IoT　(3/7)

- ルールクエリステートメントに’rx65n-cloud-kit-dali/モノ名/fromDevice’を設定します。
  この例では、モノ名は’aws_dali_test_device1’としています。

Figure 106 Aws IoT　(4/7)

- アクション 1 に ”Lambda” を選択し、Lambda 関数として”daliweb_send_iot2web”を選択します。
  その後、[次へ]を選択します。

Figure 107 Aws IoT　(5/7)

- 各項目を確認した後、図の「作成」を選びます。

- 新しいルールを作成するため、再び「作成」を選びます。

Figure 108 Aws IoT　(6/7)

- 図の名称に「daliweb_IoT2web_logger」を設定します。

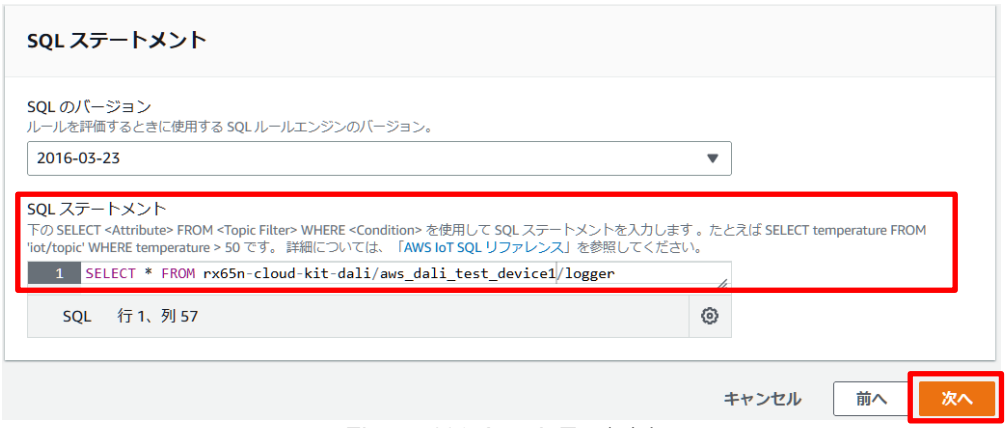

Figure 109 Aws IoT　(7/7)

- 図の名称にルールクエリステートメントに’rx65n-cloud-kit-dali/モノ名/logger’を設定します。
  この例では、モノ名は’aws_dali_test_device1’としています。
- 「fromDevice」と同様に「logger」として設定を行います。

Figure 110 Aws IoT Endpoint

- AWS IoT で、「設定」からエンドポイントを取得します。

15. AWS Web アプリケーションを準備します。

- AWS Web アプリケーション(aws-web.zip)を解凍します。
- 以下の証明書ファイルを daliweb¥html¥assets¥php フォルダに配置します。
- 証明書は以下のファイルになります。

・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

- aws-web.zip の root /setting.txt ファイルを修正します。

・MONO:「モノ名」を設定します。

・CERT:「XXXXXX-certificate.pem.crt」ファイル名を設定

・PRIVATE:「XXXXXX-private.pem.key」ファイル名を設定

・CA:「xxxxxRootCA1.pem」ファイル名を設定

・前ページで確認した「モノのエンドポイント URL」を設定

注意:

・スペースは含めないでください。

・改行しないでください。

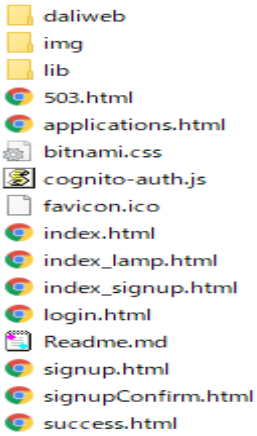

Figure 112 AWS WEB ファイル

- 図の状態で AWS-WEB ソースを任意のファイル名で zip ファイルとして圧縮します。

例:aws-web-XXXX.zip

注意点:

図のように Root に配置された状態で zip ファイルとして圧縮してください。

16. AWS マネジメントコンソールのサービスから、AWS Elastic Beanstalk を選択します。

- 「新しいアプリケーションの作成」を選択します。

Figure 113 AWS Elastic Beanstalk(1/2)

- 図のアプリケーション名に「AWS DALI」と入力し、「作成」を選択します。
- 続いて「新しい環境の作成」を選択します。
- 「ウェブサーバ環境」を選択し、「選択」を押下します。

Figure 114 AWS Elastic Beanstalk(2/2)

- 図のプラットフォームを「PHP」を選択します。
- 図のアプリケーションコードを「コードのアップロード」を選択します。

- 図のソースコード元を前章で保存した AWS-WEB アプリケーションの ZIP ファイルを選択します。
- 図の環境の作成を選択します。

・以下の画面に切り替わったら処理が完了するまで待ちます。

Figure 115 Screen while creating Elastic Beanstalk application

・以下の画面に切り替わったら AWS Elastic Beanstalk への登録は完了です。
URL をテキストに保存してください。

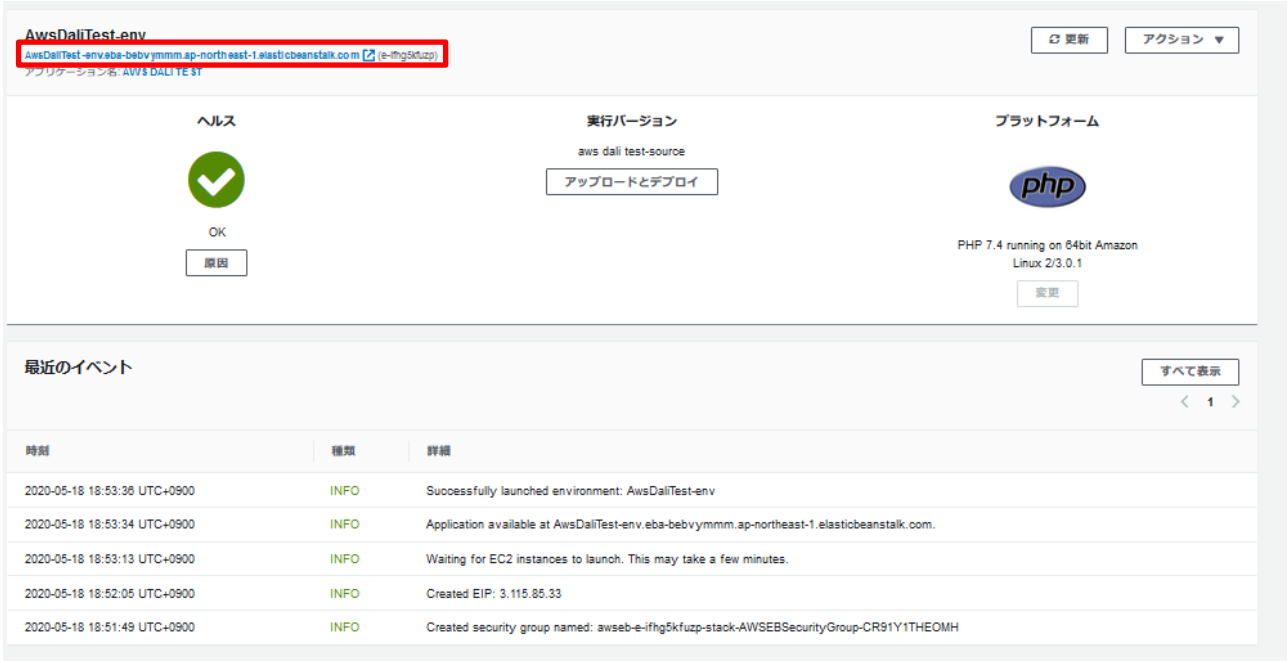

Figure 116 Screen of application registered in Elastic Beanstalk

1. 5.5 章を参照し、本デモプロジェクトを実行します。
2. Google Chrome にて、手順 4.で保存した URL を設定し AWS Web アプリケーションを開きます。Google Chrome に、URL/signup.html を指定して起動します。
   Sign In 画面が表示されるので、AWS コンソールに使用しているメールアドレスとパスワードを設定してください。

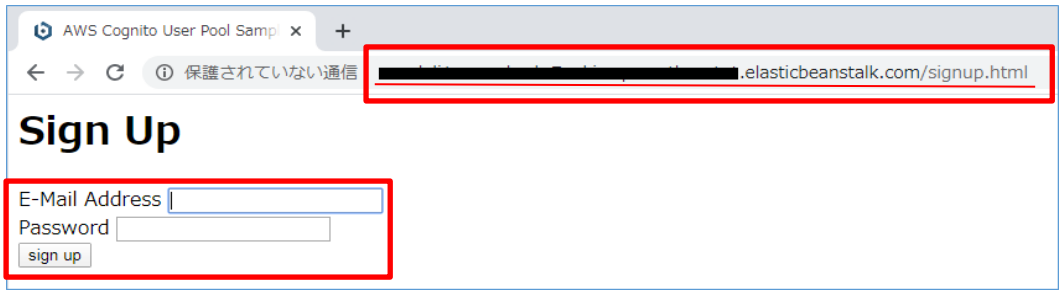

Figure 117 AWS web application sign-in screen

- Mail アドレスとパスワードを指定
  パスワードはアルファベットの大文字、小文字、数字を使用して 8 文字以上必要
  設定したメールアドレスに認証コードが送られます。

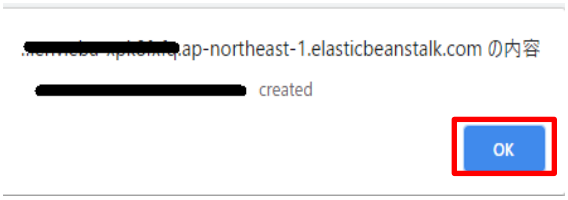

Figure 118 AWS web application sign-in screen

- 図の OK を選択します。

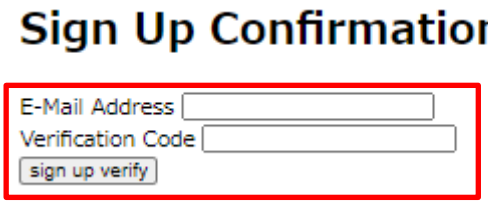

Figure 119 AWS web application sign-in screen

- 認証コードが登録したメールに届きます。
- メールアドレス、認証コードを入力します。
- “sign up verify”をクリックします。

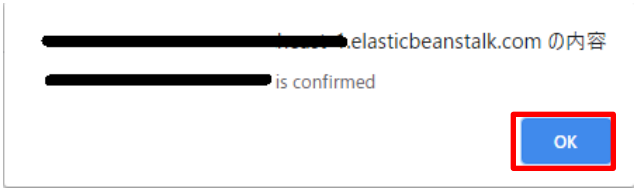

Figure 120 AWS web application sign-in screen

- 図の OK を選択します。

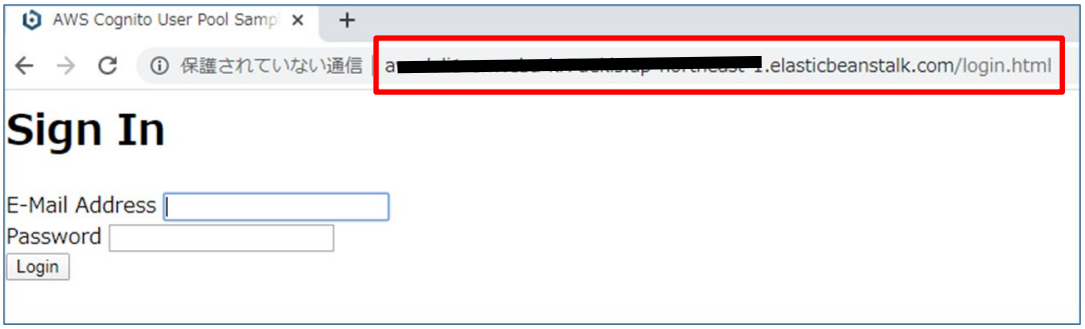

Figure 121 AWS web application sign-in screen

注意：ユーザ認証を含めた WEB アプリケーションについては、アプリケーション全体を含めて見直してください。

3. AWS Web アプリケーション画面を操作します。
灯具の照明制御を行う前に、5.6.2.1(1) DALI Control 画面にて前準備を行います。
- 接続されている Control Device/Control Gear の情報を取得します。
- 取得した Control Gear をフロア内に配置します。
- Group Select ボタンを押した後、点灯させたい番号を選択した後、「終了」ボタンを押します。
※「Group」なので番号は複数選択が可能です。

4. AWS Web アプリケーション画面で灯具を操作します。
例 1：下図の ON ボタンを押す→接続されている灯具が点灯する
例 2：下図の OFF ボタンを押す→接続されている灯具が消灯する

Figure 122 AWS Web Application System Overview Diagram

・画面操作方法

AWS Web アプリケーション画面には、動作手順で記載した画面の他に操作画面が用意されています。

各画面の詳細な操作方法と用途を以下に示します。

(1) DALI Control 画面

この画面では DALI Network に接続された灯具の取得、Office フロア画像内への灯具のレイアウト、灯具グループ化、及びグループ化された灯具へ Frame の発行を行います。

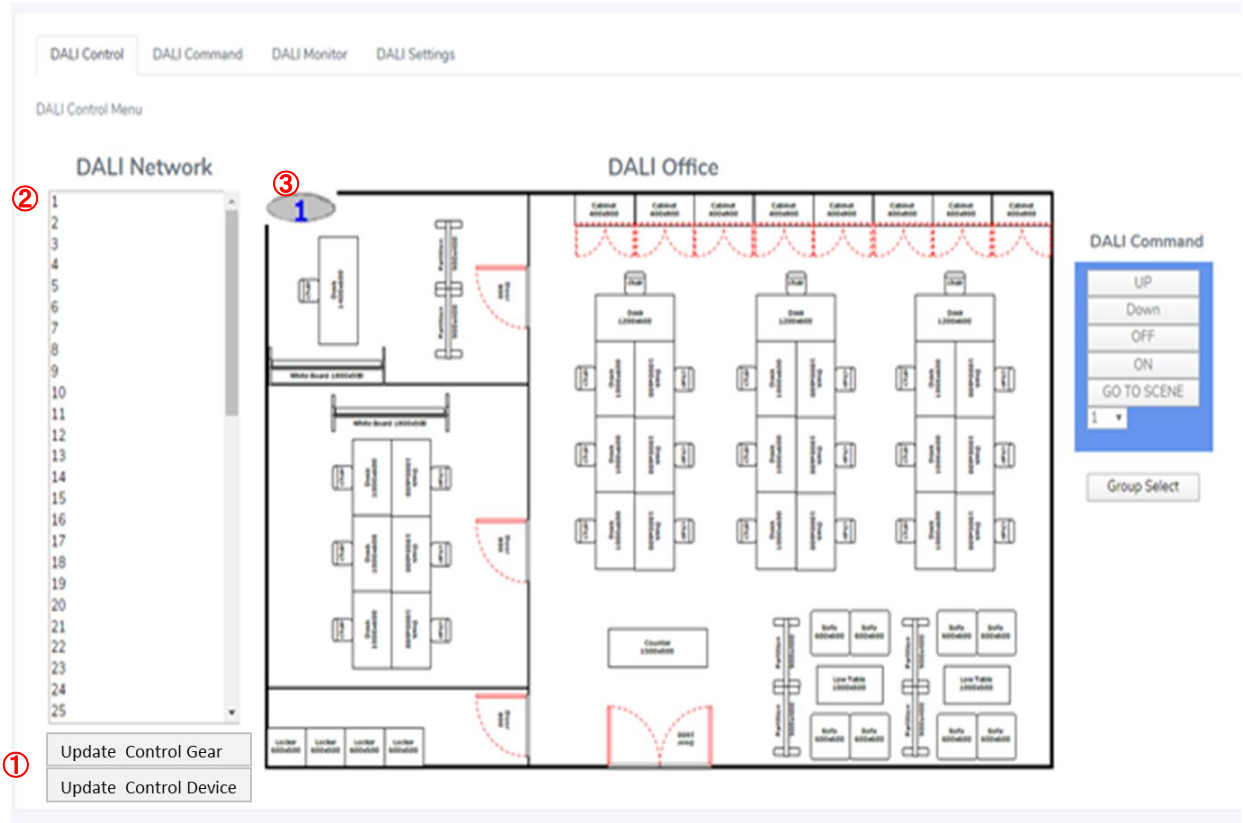

Figure 123 DALI Control

(a)Control Gear、Control Device 取得

① [Update Control Gear]ボタンをクリックし DALI Network に接続された Control Gear の
ショートアドレスを取得。
[Update Control Device]ボタンをクリックし DALI Network に接続された Control Device の
ショートアドレスを取得。

② 取得した Control Gear、Control Device を[DALI Network]にリストアップ。
(Control Gear – 最大 64 台、Control Device – 最大 64 台)
Control Gear は「XX」、Control Device は「A XX」と表示されます。
(XX はショートアドレス番号表示)

③ リストアップと同時にフロア左上に Control Gear のみ灯具画像を配置。(若い番号が前面)
＊Control Device の画像は、表示されません。

(b)Control Gear レイアウト

左上の灯具を Drag & Drop でフロア内の希望する位置にレイアウトします。

この機能は、Control Gear のみの機能になります。Control Device の画像は、表示されません。

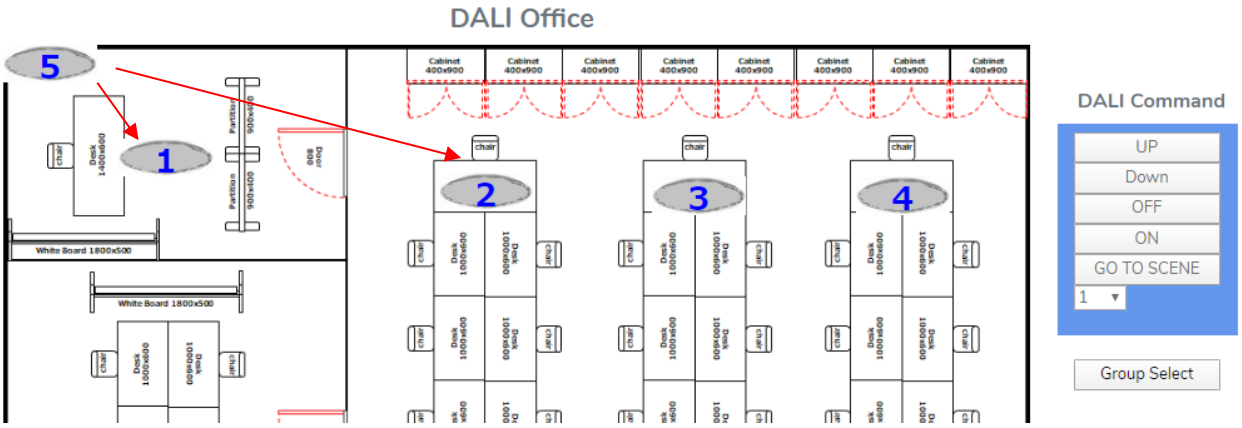

Figure 124 Layout of Control Gear

(c)Group Select モード

①画面右の[Group Select] ボタンをクリックして灯具のグルーピングモードに移行。
[Group Select]ボタン(*1)は[終了]に切り替わります。(*2)
②グループ化したい灯具をクリックで選択。選択された灯具は点灯表示に変化する。
③[終了]ボタンをクリックし Group Select モードを終了します。

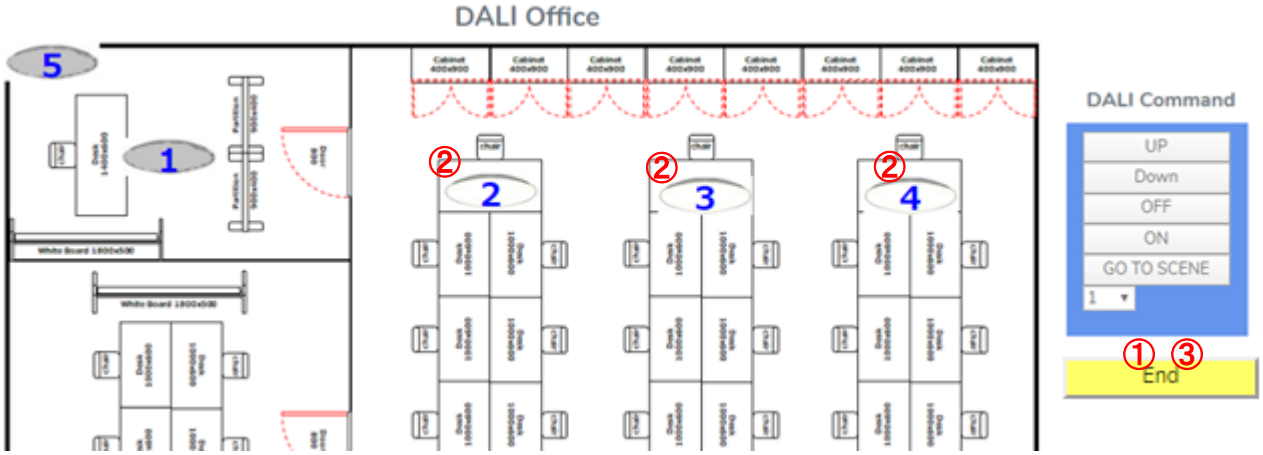

Figure 125 Group Select mode

*1	再度 [Group Select]をクリックすると以前の選択状態はクリアされ、グループ化をやり直す事ができます。
*2	配置した灯具位置、グルーピングされた内容は保存されません。

(d)DALI Command

グループ化された灯具に対して以下の各ボタンで Frame を発行し Control Gear を操作します。

(*1)

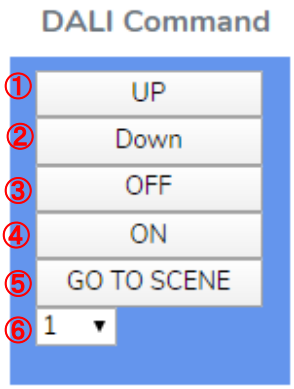

Figure 126 DALI Command

①灯具の光量を増加させる。
②灯具の光量を減少させる。
③灯具消灯させる。
④灯具点灯させる。
⑤プルダウンで選択されているシーン番号のシーンを実施する。
⑥[GO TO SCENE]で、発行するシーン番号を選択する。

*1　Control Gear がグループ化されていない状態では各ボタンは無効状態となり操作はできません。
グループ化されていない Control Gear に Frame を送る場合は、「DALI Command」画面から送ってください。
Control Gear をグループ化することでボタン有効状態となり操作可能となります。
この機能は、Control Gear のみの機能になります。

(2) DALI Command 画面

この画面では DALI Network に接続された Control Gear の取得と、グループ化された Control Gear への Frame の発行を行います。

Control Device への Frame は、[Update Control Device]ボタンのみ可能です。DALI Network に接続された Control Device のショートアドレスを取得できます。

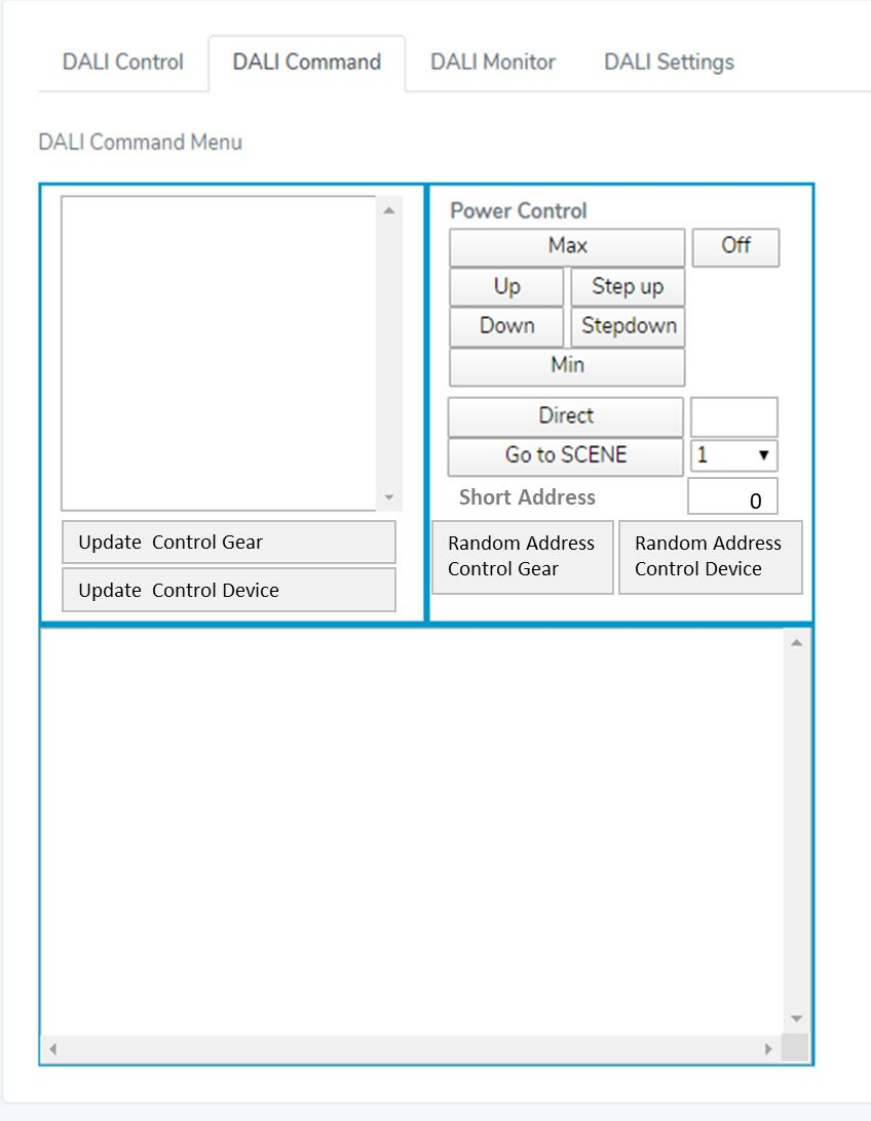

Figure 127 DAIL Command

(a)Control Gear 取得

この画面左側では、DALI Network に接続された Control Device と Control Gear のショートアドレス取得を行います。

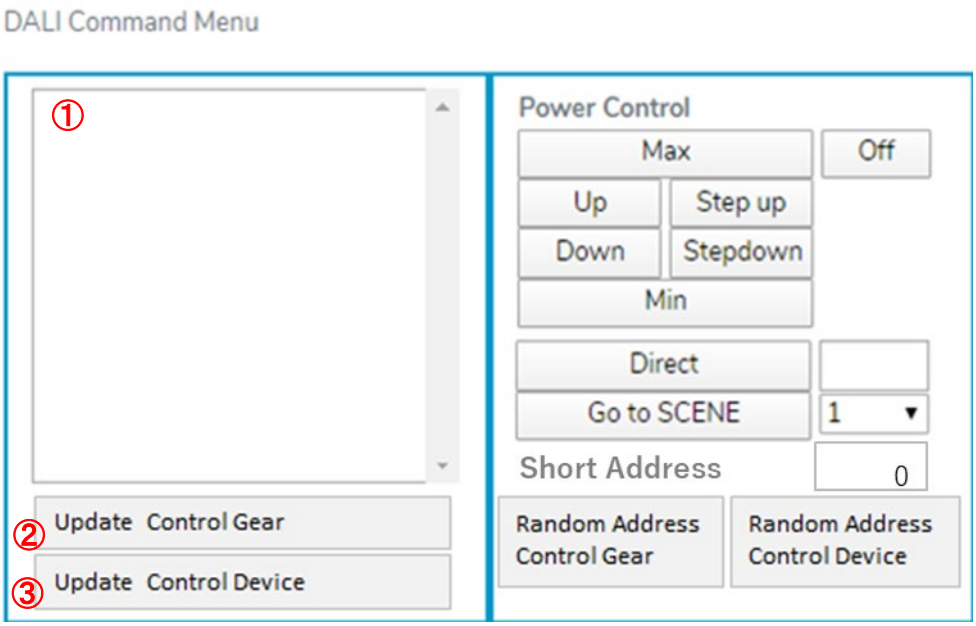

Figure 128 Control Gear 取得

① DALI Network に接続された Control Gear を確認する。(*1)
② ボタン押下により、取得した Control Gear がリストアップされる。(最大 64 台)
③ DALI Network に接続された Control Device のショートアドレスを取得できます。(*2)
ボタン押下により、取得した Control Device のショートアドレスが②画面
リストアップされる。(最大 64 台)

*1	この機能は DALI Control タブの[UPDATE]機能と同一機能です。 DALI Control タブのリストへの表示と、Control Gear 画像の表示も連動して実施されます。
*2	この機能は DALI Control タブの[UPDATE]機能と同一機能です。 DALI Control タブのリスト表示も連動して実施されます。

(b)Power Control

この画面右側では、各ボタン操作による Frame の発行を行います。

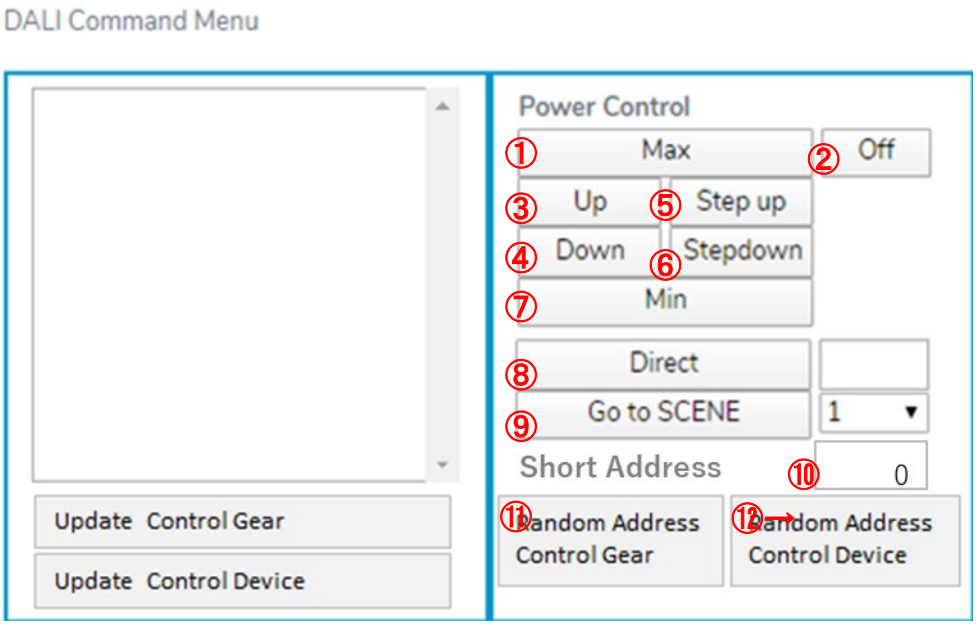

Figure 129 Power Control

ControlGear へ Frame を発行します。(*1)

① Max の Frame を発行
② Off の Frame を発行
③ Up の Frame を発行
④ Down の Frame を発行
⑤ Step up の Frame を発行
⑥ Step down の Frame を発行
⑦ Min の Frame を発行
⑧ 右の入力値を取得し Direct の Frame を発行
⑨ 右のプルダウンで選択したシーン番号に対する Go to SCENE の Frame を発行
⑩ Frame を発行する際のショートアドレスを指定

DALI Network に接続された Control Gear、Control Device に対して
以下の Frame を発行できます。

⑪ Control Gear へランダムアドレスアロケーションを発行
⑫ Control Device へランダムアドレスアロケーションを発行

*1　各 Frame 発行ボタンを押す前に⑪のショートアドレスを指定してください DALI Control タブの Control Gear 画像の点灯表示とは連動しません。

(3) DALI Monitor 画面

この画面では DALI 通信状態のモニタリングと結果のファイルへの保存を行います。

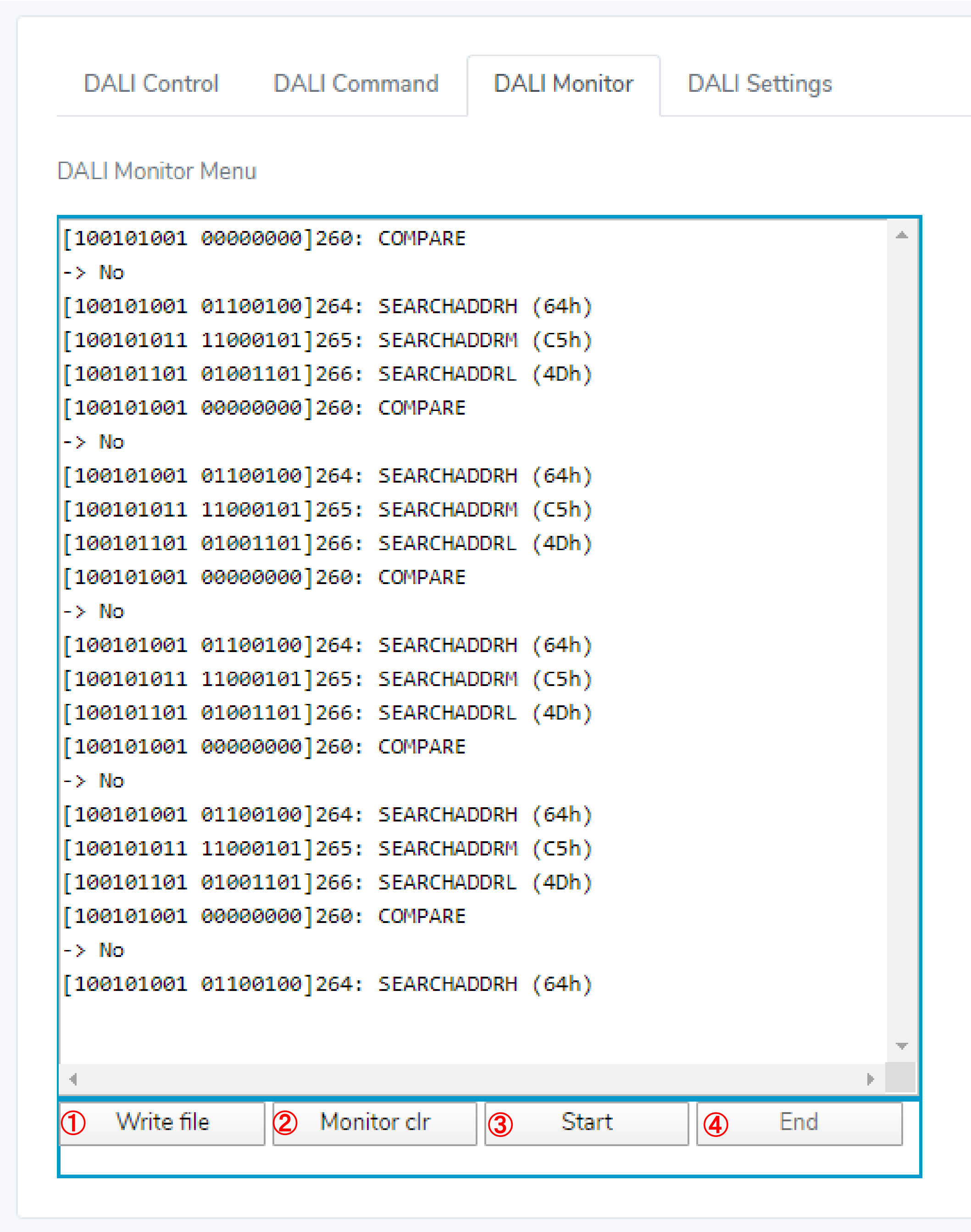

Figure 130 DALI Monitor

① 表示内容をファイルに保存
② 表示内容を消去
③ モニターを開始(Logging Function を有効にします)
④ モニターを停止(Logging Function を無効にします)

(4) DALI Settings 画面

この画面ではボード上の 9 つあるマトリクスボタンへ Frame の割り当て/読み込みと、Frame の発行を行います。

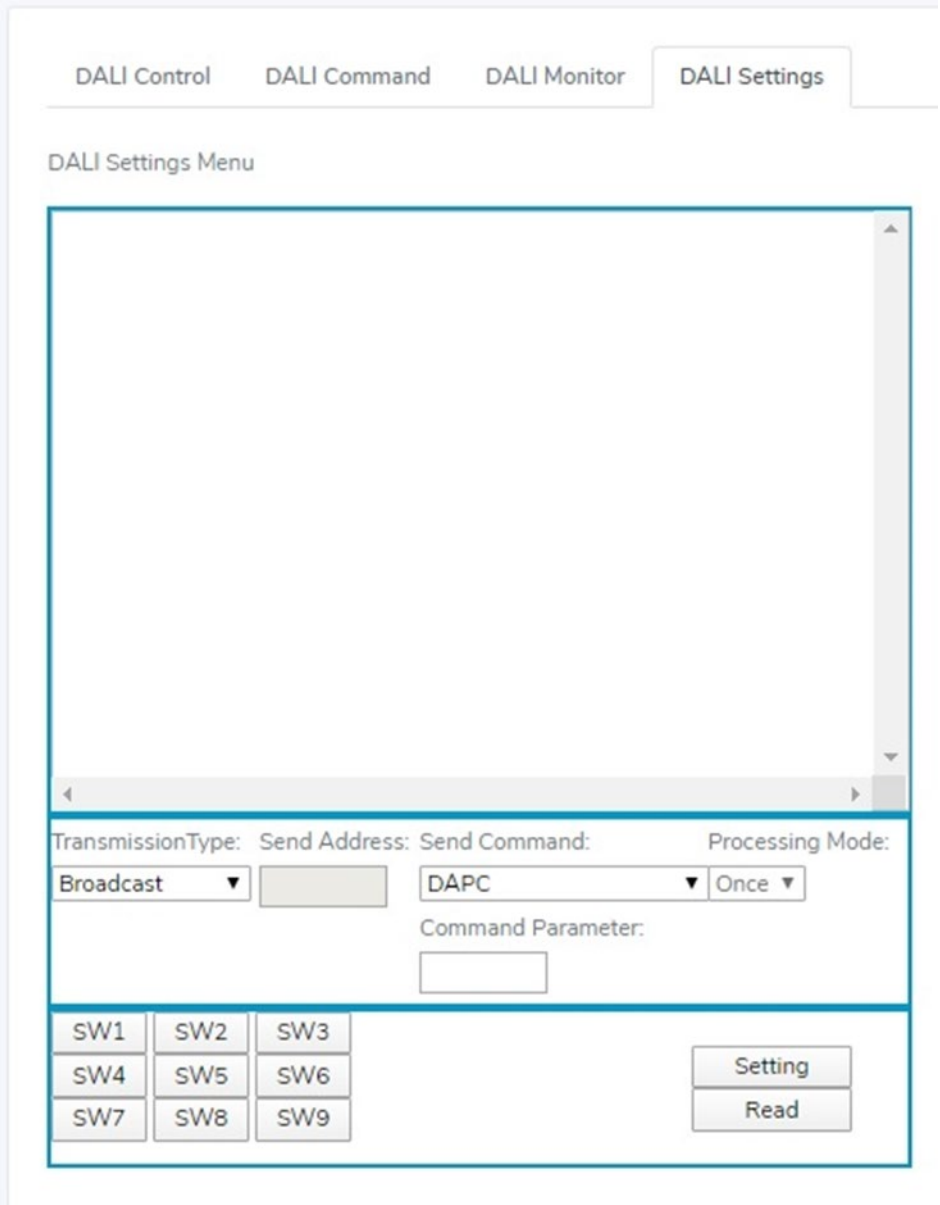

Figure 131 DALI Setting

(a)マトリクスボタン割り当て

この画面では指定したマトリクスボタンに Frame の割り当てを行います。

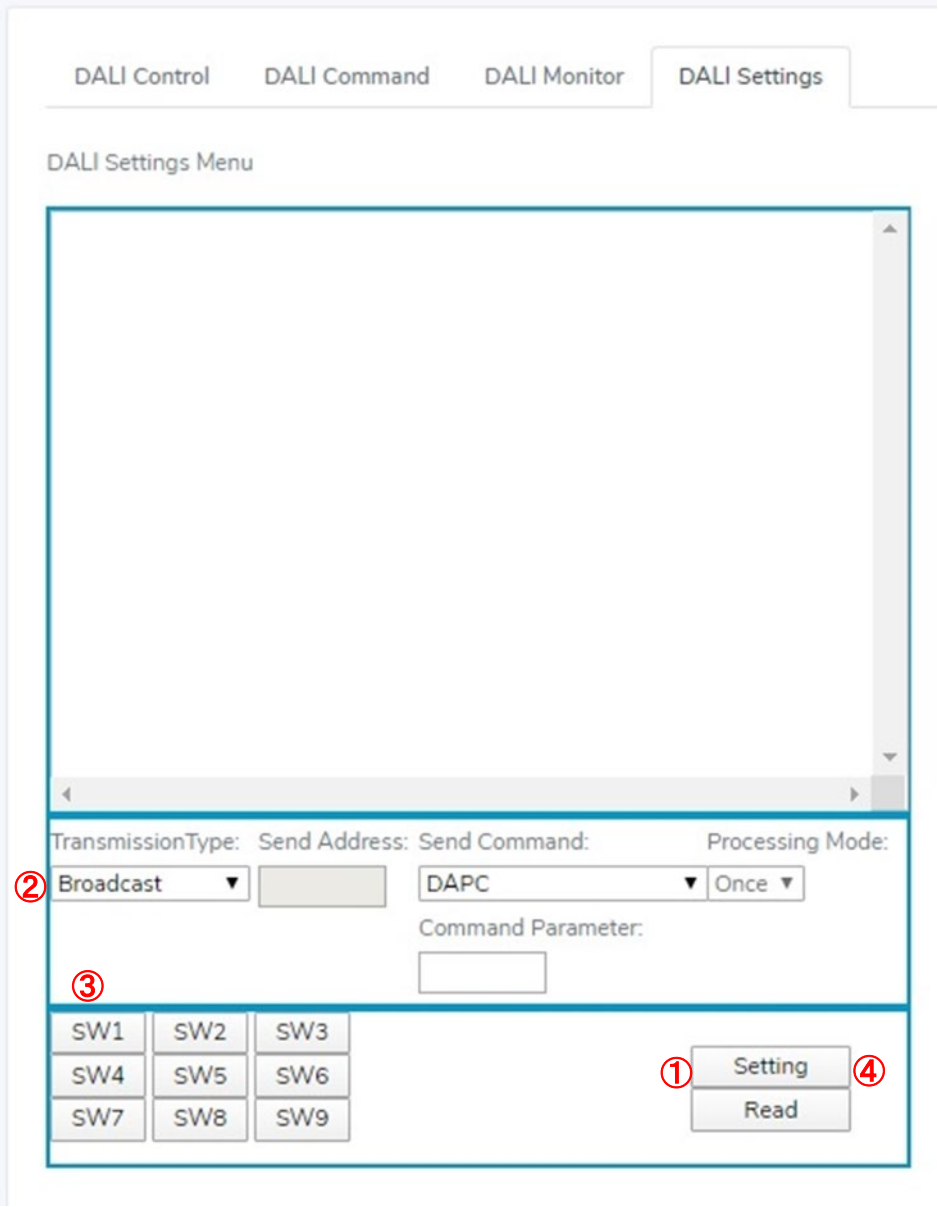

Figure 132 DALI CODE 割り当て

① [Settinh]をクリックして割り当てモードに移行。ボタンは[End]に切り替わる。
② マトリクスボタンに設定する Transmission Type、Send Address、Send Command を選択。
③ SW1～9 のボタンをクリックしてマトリクスボタンを割り当てる。
④ [End]をクリックして割り当てを終了する。

(b)割り当てコード読み込みまたは発行

この画面では指定したマトリクスボタンに割当たっている Frame の読み出し、または Frame の発行を行います。

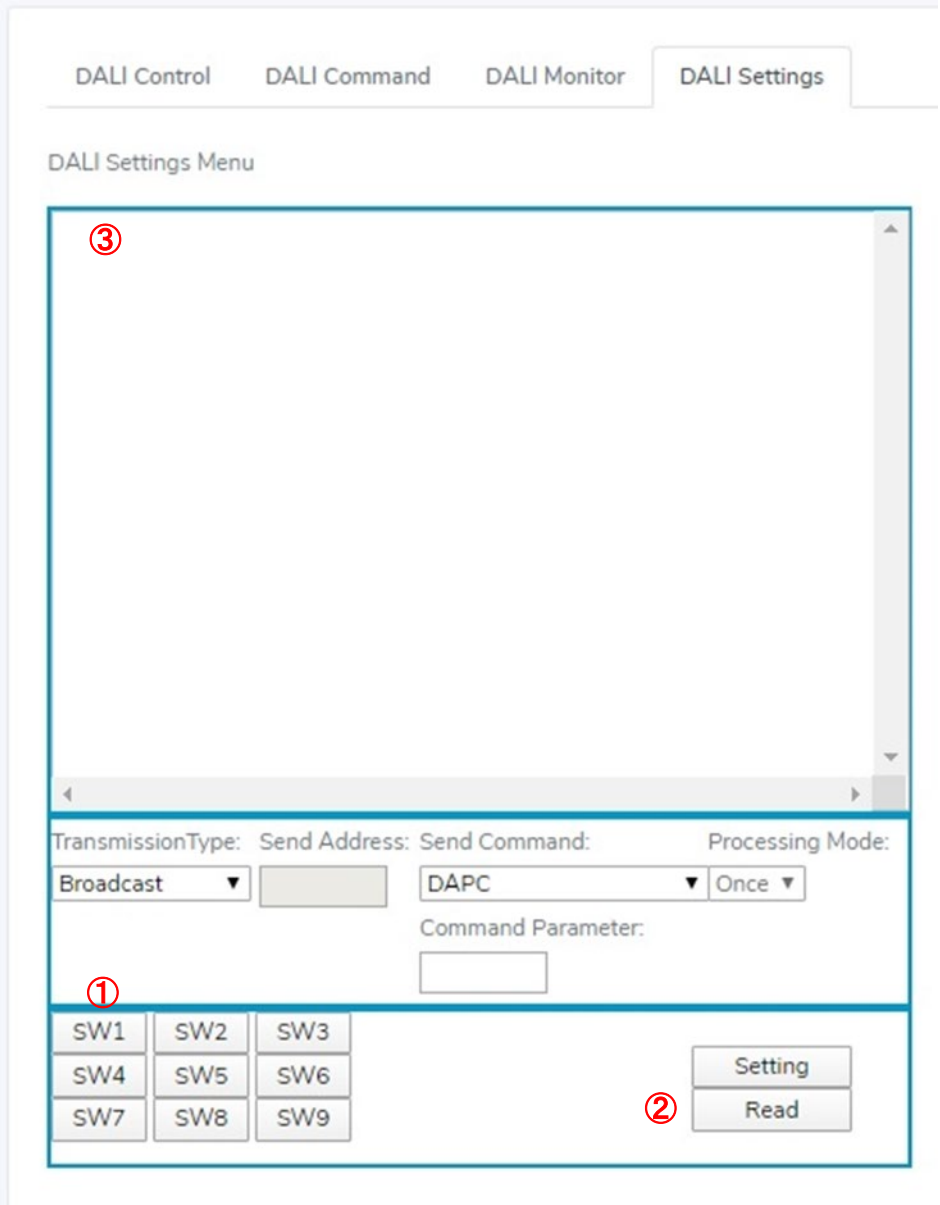

Figure 133 Read and publish

- Frame の読み出し
  ① 取得するマトリクスボタンの番号(SW1~9)を選択
  ② ボタンクリックしてマトリクスボタン読み込み
  ③ 読み込んだマトリクスボタンの Frame が表示される

5.6.3　マトリクスボタン操作

マトリクスボタン操作で動作させる場合のデモプロジェクトの動作手順を以下に示します。

1. 5.1 章～5.4 章を参照し、必要ソフトのインストールと機器の接続を行います。
2. 5.5 章を参照し、本デモプロジェクトを実行します。
3. マトリクスボタンを操作します。
   例 1：接続されている灯具全てを MAX の光量で点灯させる→SW1 ボタンを押す
   例 2：接続されている灯具全ての光量を下げる→SW5 ボタンを長押し

各マトリクスボタンには、初期設定として下表に示す Frame が設定されています。
マトリクスボタンの短押しで Frame を 1 回送信、長押しで Frame の繰り返し送信を行い、複数のマトリクスボタンを同時に押した場合は Frame 送信を行いません。
初期設定の Frame とは異なる Frame を設定したい場合に AWS Web アプリケーションとの接続を行うことで、Frame の設定変更が行えます。また、現在設定されている Frame の読み出しを行うことも可能です。

マトリクスボタンの Frame 設定変更・読み出しについては、0 章の操作方法を参照してください。
また、AWS Web アプリケーション章を参照し、AWS への登録・設定等を行ってください。

Table 68 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　制限事項

1. デモプロジェクト起動後、RX65N DALI-2 オプションボードの電源電圧を安定させる為に 50ms の待ち時間が必要です。
2. AWS 接続時、AWS ユーザ認証中は認証処理を優先するため、マトリクスボタンの操作に対する反応が遅れます。
3. AWS Web アプリケーションと接続中に DALI マスタコントローラ GUI との接続・通信を行う場合、双方から同時に Frame 発行を行わないようにしてください。Frame の応答を正しく受け取れない場合があります。
4. AWS Web アプリケーションのログ表示は、MQTT 通信を使用しています。MQTT 通信では順序保証されないため、DALI コマンド表示が前後入れ替わることがあります。
5. AWS 接続時、AWS との接続が切れることがあります。自動で接続が復帰しますが、その間、AWS Web アプリケーションは、応答しません。
6. AWS Web アプリケーションのログ表示は、コマンド名が表示されたログは自身が送信したログです。他者が送信したデータはコマンド名が表示されません。
7. OTA は開始後、すべての機能が一時停止し、完了後に再起動します

5.8　OTA

1. OTA の手順

無線による（OTA）更新を試す際は以下の URL の”手順まとめ”を参照ください。
https://github.com/renesas/amazon-freertos/wiki/OTA%E3%81%AE%E6%B4%BB%E7%94%A8
但し、”手順まとめ”の2番は変更不要です。

2. AWS コンソールからの OTA 手順

OTA 更新ファイルを AWS にアップロードし、AWS コンソールからの OTA 実施手順は、
以下の URL を参照ください。
https://docs.aws.amazon.com/ja_jp/freertos/latest/userguide/ota-console-workflow.html

3. OTA の基礎部分を知りたい場合、以下の URL を参照ください。
https://docs.aws.amazon.com/ja_jp/freertos/latest/userguide/freertos-ota-dev.html

6.　エラー時の対応

6.1　ビルドできない

6.1.1　プロジェクトのフォルダ階層が深い場合

配置したプロジェクトファイルのフォルダ階層が深すぎますとプロジェクトをビルド出来ない時があります。
その際は、プロジェクトファイルのフォルダ階層を下げてください。

```
例)
この例では、AAAA、XXXX は、プロジェクト用に作成された任意のフォルダ名です。
AAAA を削除します。
C:/AAAA/XXXX/プロジェクトファイル
↓
C:/XXXX/プロジェクトファイル
```

6.1.2　FIT モジュールを変更した場合

インポートしたデモプロジェクトに対して、ソフトウェアコンポーネントによる FIT モジュールの変更などを行いビルドした際、以下のようなエラーが発生する場合があります。

```
fatal error: ファイル名: No such file or directory
```

使用する FIT モジュールが削除されている可能性があるため、削除されている場合は再度登録を行ってください。

6.2　DALI 機器と接続できない

6.2.1　DALI マスタコントローラ GUI の場合

使用する COM ポート、ボーレートなどの通信設定が異なっている可能性があります。

1.COM ポートの確認

デバイスマネージャーを開き、表示される COM ポートを確認してください。

2.通信設定

DALI マスタコントローラ GUI の通信設定は以下となります。

Table 69 Communication settings

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

注：Control Device 接続確認に使用しないでください。
DALI マスタコントローラ GUI は Control Gear に対する制御を行うため、Control Device と

の接続確認は行えません。Control Device との接続確認は AWS Web アプリケーションを使用してください。

6.2.2　AWS Web アプリケーションの場合

- PC が AWS にログインできない

再度 AWS Web アプリケーション章の内容を確認し、ユーザ登録が行えているか確認してください。

- RX65N ボードが Wi-Fi 接続できない

使用する証明書、登録している SSID とパスワードが異なっている可能性があります。
再度 AWS Web アプリケーション章の内容を確認し、設定内容が合っているか確認してください。また、RX Cloud Option Board の User SW を押下しながら再起動すると SSID の表示、MQTT との通信ログが表示されますので、MQTT との接続確認にご使用いただけます。

Figure 134 Screen displayed by pressing the user switch

Figure 135 Log display screen when connected to MQTT

※別途 Tera Tarm などの通信 Log 表示ソフトが必要です。Log 表示ソフトの通信設定は Table 64 と同様です。

※使用時、DALI マスタコントローラ GUI は接続できません。接続する場合は Reset SW 押下で再起動してください。

7. 参考ドキュメント

- 統合開発環境 e2studio ユーザーズマニュアル　入門ガイド (R20UT4204)
- RX ファミリ　フラッシュモジュール　Firmware Integration Technology(r01an2184)
- RX ファミリ　GPIO モジュール　Firmware Integration Technology (r01an1721)
- RX ファミリ　SCI モジュール　Firmware Integration Technology (r01an1815)
- RX ファミリ　ADC モジュール　Firmware Integration Technology (r01an1666)
- RX65N グループ、RX651 グループ　ユーザーマニュアル　ハードウェア編(R01UH0590)
- RX Family Cloud Option Board User's Manual (r12um0039)
- RX65N グループ　Target Board for RX65N ユーザーズマニュアル(r12um0038)
- DALI マスタコントローラ GUI ユーザーズマニュアル(r20ut0715)
- Applilet EZ for HCD Controller　ユーザーズマニュアル(r20ut0435)
- EZ-0012　RL78/I1A DC/DC LED 制御評価ボード　ユーザーズマニュアル(r01uh0363)
- RL78/I1A による照明通信(受信編)　アプリケーションノート(r01an1115)
- RL78/I1A による照明通信(送信編)　アプリケーションノート(r01an3193)
- RL78/I1A DALI-2 Control Gear 基本(102) 調光(207) 調色(209Tc) Application Notes(r01an4654)
- DALI 規格書
  IEC62386-103ed1.1
  IEC62386-101ed2.1
- Amazon FreeRTOS ユーザーガイド
  https://docs.aws.amazon.com/ja_jp/freertos/latest/userguide/what-is-amazon-freertos.html.

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	June.19.20	All	新規作成
1.01	Nov.21.22	All	AWS のポリシー設定を最小構成に修正

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 静電気対策
   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。
2. 電源投入時の処置
   電源投入時は、製品の状態は不定です。電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. 電源オフ時における入力信号
   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。
4. 未使用端子の処理
   未使用端子は、「未使用端子の処理」に従って処理してください。CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。
5. クロックについて
   リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
6. 入力端子の印加波形
   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V_{IL}（Max.）から V_{IH}（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}（Max.）から V_{IH}（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。
7. リザーブアドレス（予約領域）のアクセス禁止
   リザーブアドレス（予約領域）のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
8. 製品間の相違について
   型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違うと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ幅射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。回路、ソフトウェアおよびこれらに関連する情報を使用する場合、お客様の責任において、お客様の機器・システムを設計ください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品または本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
6. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。

   標準水準：　コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット等

   高品質水準：輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等

   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（「脆弱性問題」といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因しまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
10. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
12. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
13. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
14. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注 1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

(Rev.5.0-1　2020.10)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

商標について

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/