User's Manual

RL78 ファミリ

DALI-2 Contror Gear ライブラリ

ユーザーズマニュアル 基本(102)編

16 ビット・シングルチップ・マイクロコンピュータ

ルネサス エレクトロニクス
www.renesas.com

Rev.1.01 2022.11

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

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/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。

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。

このマニュアルの使い方

1. 目的と対象者

このマニュアルは、RL78 マイクロコントローラで DALI システムの Control Gear を開発するユーザを対象としています。

このマニュアルを使用するには、電気回路、論理回路、マイクロコンピュータに関する基本的な知識が必要です。

このマニュアルは、大きく分類すると、製品の概要、仕様、使用上の注意で構成されています。

本ライブラリは、注意事項を十分確認の上、使用してください。注意事項は、各章の本文中、各章の最後、注意事項の章に記載しています。

改訂記録は旧版の記載内容に対して訂正または追加した主な箇所をまとめたものです。改訂内容すべてを記録したものではありません。詳細は、このマニュアルの本文でご確認ください。

DALI ライブラリでは次のドキュメントを用意しています。ドキュメントは最新版を使用してください。最新版はルネサス エレクトロニクスのホームページに掲載されています。

ドキュメントの種類	記載内容	資料名	資料番号
ユーザーズマニュアル ハードウェア編	ハードウェアの仕様（ピン配置、メモリマップ、周辺機能の仕様、電気的特性、タイミング）と動作説明 ※周辺機能の使用方法はアプリケーションノートを参照してください。	RL78/I1A ユーザーズマニュアル ハードウェア編	R01UH0169JJ0320
ユーザーズマニュアル ソフトウェア編	CPU 命令セットの説明	RL78 ファミリ ユーザーズマニュアル ソフトウェア編	R01US0015JJ0220
アプリケーションノート	周辺機能の使用方法、応用例 参考プログラム C 言語によるプログラムの作成方法	ルネサス エレクトロニクスホームページに掲載されています。	
Renesas Technical Update	製品の仕様、ドキュメント等に関する速報		

2. 略語および略称の説明

略語／略称	英語名	日本語名
DALI	Digital Addressable Lighing Interface	照明制御の国際標準規格
NVM	Non-Volatile Memory	不揮発メモリ

目次

1. DALI102 ライブラリ概要

1.1 ライブラリ機能概要

本ライブラリは、DALI 通信におけるスレーブ(Control Gear)用ライブラリとして、DALI102 規格のハードウェア非依存部分の処理を実現しています。

DALI 規格には、通信タイミング、変数保存等のハードウェアに関する規格と、Forward Frame 受信時の処理等のソフトウェアに関する規格の両方が定義されています。本ライブラリは、汎用性を持たせるために主にハードウェアに関する規格部分(ハードウェア依存部)及び電源制御を省いたものとなっています。

本ライブラリを使用するには、DALI 規格について理解する必要があります。

表 1.1 処理範囲

ユーザ作成処理	ライブラリ処理
・H/W 設定 ・DALI 通信ドライバ ・タイマ制御 ・メモリバンク実体／制御 ・調光制御 ・不揮発データアクセス ・異常検出	・受信 16bit Forward Frame 処理 ・送信 Backward Frame 発行 ・タイミング制御 ・DALI 変数操作 ・メモリバンク操作

本ライブラリでは、DALI 通信によって受信した 16bit Forward Frame に基づいて処理を行います。ライブラリはハードウェア依存部分を持たないためにユーザアプリケーションによって受信した 16bit Forward Frame を指定 API 関数でライブラリに指定することにより処理を行います。

16bit Forward Frame にて指定されるコマンドは DALI 変数設定コマンドや DALI 変数設定値取得コマンドなど様々存在します。アプリケーション側の設定変更が必要な場合などは、必要に応じてアプリケーションに通知を行います。

本ライブラリは、1 デバイスで複数の logical unit を実現することができます。また、複数のメモリバンク実装に対応できます。対応できる最大 logical unit 数、及び、実装可能なメモリバンクのバンク番号は以下の通りです。

表 1.2 logical unit及びメモリバンク仕様

項目	値
最大 logical unit 数	64 注
実装可能メモリバンク番号	0～199

注：DALI システムとして、一つの DALI ネットワークに対して Control Gear の最大接続数は 64 です。DALI ネットワークの構成とハードウェアの制限を加味し、合計で 64 を超えない数で設定してください。

1.2 ソフトウェア構成

本ライブラリを使用した場合のControl Gearソフトウェア構成を以下に示します。

赤線で囲んだ部分が本ライブラリとなります。本ライブラリはハードウェア非依存部の処理を実現します。本ライブラリはアプリケーション層の一部となり、DALI通信、不揮発データアクセス等を行います。また、本ライブラリは別途IEC62386 Part 2XX系の仕様をライブラリ化したDALI2XXライブラリを使用して拡張することが可能です。

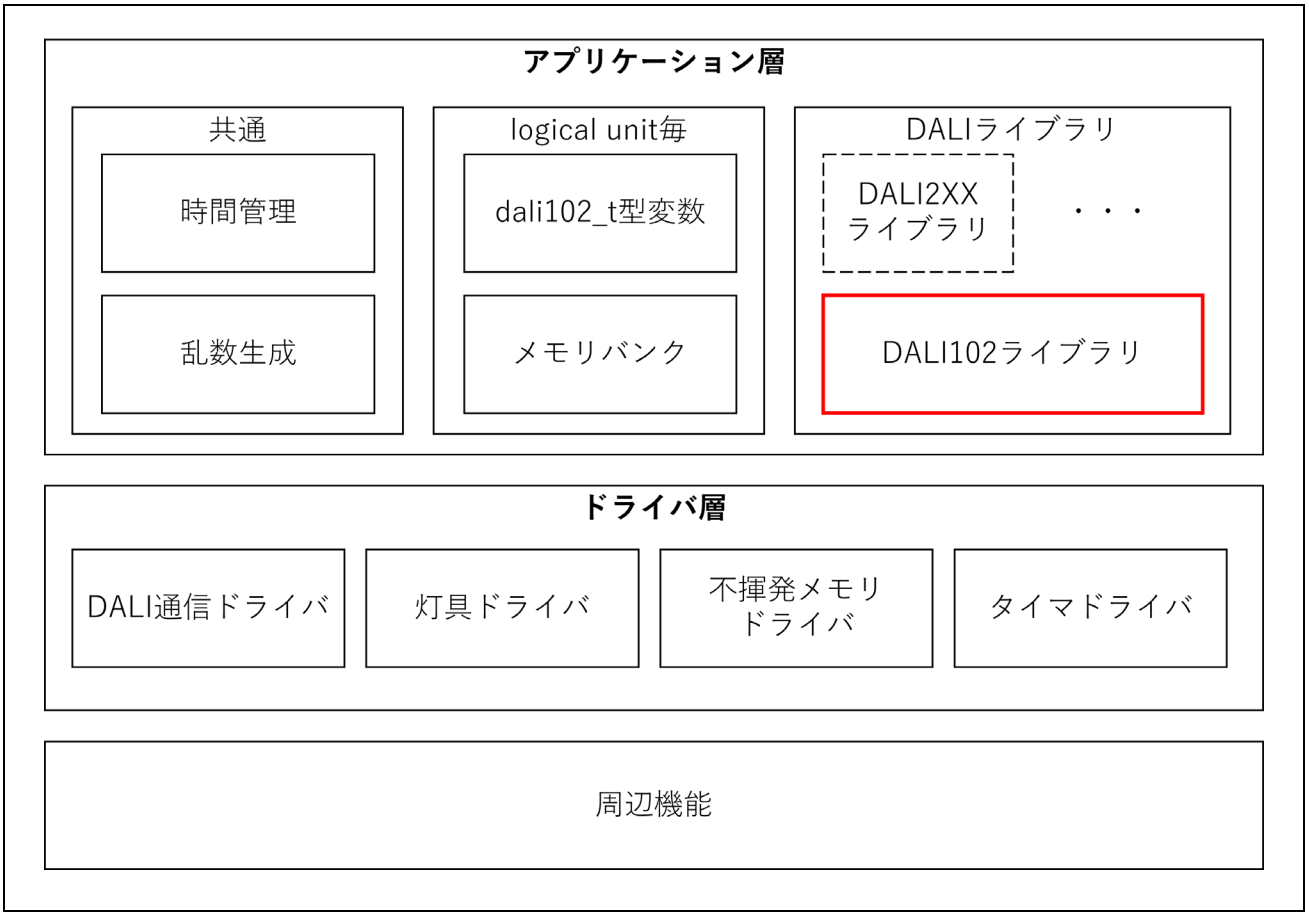


図 1.1 Control Gearソフトウェア構成図

1.3 対応規格

本ライブラリで対応している規格は以下となります。

表 1.3 対応規格とライブラリ名

対応規格	コンパイラ	ライブラリ名
IEC62386-102 Edition 2.0	Renesas CC-RL V1.10.00	r_dali_102_cc_gen2_v1_00.lib
	IAR C/C++ Compiler for Renesas RL78 V4.21.2.2420	r_dali_102_iar_gen2_v1_00.a

1.4 ファイル一覧

本ライブラリが提供するファイル一覧を記載します。

表 1.4 ファイル一覧

ファイル名	説明
r_dali_102_cc_gen2_v1_00.lib r_dali_102_iar_gen2_v1_00.a	CC-RL版ライブラリファイル IAR版ライブラリファイル
r_dali102_api.h	ライブラリヘッダファイル
r_dali102_common.h	複数モジュールにて使用する定義ヘッダファイル
r_dali102_var.h	変数モジュールの定義ヘッダファイル
r_dali102_timer.h	タイマモジュールの定義ヘッダファイル
r_dali102_fade.h	フェードモジュールの定義ヘッダファイル
r_dali102_mb_if.h	メモリバンクI/Fモジュールの定義ヘッダファイル
r_dali102_list.h	リストモジュールの定義ヘッダファイル
r_dali102_dtx_if.h	汎用Device Type I/Fモジュールの定義ヘッダファイル
r_dali102_dt6_if.h	Device Type 6 I/Fモジュールの定義ヘッダファイル
r_dali102_dt8_if.h	Device Type 8 I/Fモジュールの定義ヘッダファイル

1.5 リソース

本ライブラリが必要とするライブラリのリソース(ROM/RAM サイズ、最大スタックサイズ)を以下に示します。

Control Gear の実装内容に依存しないリソースを表 1.5 ライブラリリソース(固定)、Control Gear の実装内容に依存するリソースを表 1.6 ライブラリリソース(可変)に記載します。

表 1.5 ライブラリリソース(固定)

コンパイラ	項目		サイズ
CC-RL	ライブラリリソース	ROMサイズ	11,561 [byte]
		RAMサイズ	4 [byte]
	最大スタックサイズ		98 [byte] (R_DALI102_Tick1ms関数)
IAR	ライブラリリソース	ROMサイズ	13,245 [byte]
		RAMサイズ	6 [byte]
	最大スタックサイズ		116 [byte] (R_DALI102_Tick1ms関数)

表 1.6 ライブラリリソース(可変)

コンパイラ	項目	RAMサイズ
CC-RL	dali102_t	108 [byte / logical unit]
IAR	dali102_t	108 [byte / logical unit]

1.6 開発環境

本ライブラリ開発時の環境を以下に記載します。

表 1.7 ライブラリ開発環境

コンパイラ	項目	内容
CC-RL	統合開発環境	e2studio V2021-04
	C コンパイラ	Renesas CC-RL V1.10.00
	CPU コア	RL78-S2 コア
	最適化レベル	サイズ優先
	言語規格	GNU ISO C99
IAR	統合開発環境	IAR Embedded Workbench for Renesas RL78 V8.5.2.7561
	C コンパイラ	IAR C/C++ Compiler For Renesas RL78 V4.21.3.2447
	CPU コア	RL78-S2 コア
	最適化レベル	サイズ優先
	言語規格	GNU ISO C99

1.7 注意事項

1. 本ライブラリの API 関数はユーザアプリケーションにて割り込みハンドラからの呼び出しを禁止します。
2. 本ライブラリを含んだプログラムのループ処理を最大 1ms 未満で実行できるようにしてください。ループ処理が 1ms 以上で動作する環境下では DALI 規格仕様を満たしません。
3. dali102_t 型構造体や dali102_cmd_t 型構造体は参照専用の構造体です。

2. プログラミング環境

この章では、ユーザが本ライブラリを使用してControl Gear動作を行う上で、必要なハードウェア環境とソフトウェア環境について説明します。

2.1 ハードウェア要件

2.1.1 DALI 通信回路

DALI 通信を実現するには IEC62386-101 規格に規定された動作を行う通信回路が必要となります。

2.1.2 不揮発領域

Control Gear は NVM データと呼ばれるデータを退避し、電源再投入後もデータ値を保持する仕様があるため、専用の不揮発領域を確保してください。

2.1.3 調光制御回路

Control Gear は Application Controller からのコマンドにより灯具の明るさを指定されます。本ライブラリ経由で取得した明るさに合わせて灯具を調光制御する回路が必要となります。

2.1.4 異常検出機構

Control Gear は、動作上の異常を検出し、内部保持の変数にその状態を保持したうえで Application Controller の問い合わせに対して回答する必要があります。そのため、ハードウェア的に異常を検出する機構(例えば灯具異常など)が必要になります。

2.2 ソフトウェア要件

2.2.1 DALI102 モジュール定義

1 つのハードウェアの中で定義される論理的な Bus unit(本書では Control Gear に相当する)の単位を logical unit といいます。本ライブラリは Control Gear の logical unit を構成するために必要なパラメータをまとめた構造体型(dali102_t)を提供します。dali102_t 型変数のことを DALI102 モジュールと呼びます。

必要な logical unit 数分の DALI102 モジュールを定義してください 。

2.2.2 DALI 通信ドライバ

ハードウェア要件に記載している DALI 通信回路を制御し、IEC62386-101 規格を満たすドライバを実装してください。DALI 通信ドライバの詳細は RL78/I1A DALI Control Gear 基本(102) 調光(207) 調色(209Tc) サンプルアプリケーション(R01AN6177)を参照してください。

2.2.3 時間管理

本ライブラリは 1ms 定期に呼び出されることで時間管理を行う API 関数が存在します。ユーザアプリケーションで 1ms インターバルタイマを実装し、R_DALI102_Tick1ms 関数を呼んでください。

なお、タイマの精度が悪い場合は規格に反することがありますので、呼び出し間隔の誤差は±10%未満に収まるようにしてください。

2.2.4 乱数生成

Control Gear では再現性のない 24bit の乱数を生成することを求められます。ユーザアプリケーションにて、0x000000 から 0xFFFFFE の範囲で再現性のない乱数を生成する関数を実装してください。引数と戻り値のフォーマットは以下となります。この関数は本ライブラリを使用する上で必須です。

実装した関数は dali102_general_callback_t 型構造体変数のメンバ GetRandomValue にコールバック関数として登録してください。

関数フォーマット	
引　数	なし
戻り値	uint32_t 0x000000から0xFFFFFEの範囲の値

2.2.5 不揮発領域アクセス

ハードウェア要件に記載している不揮発領域に対してアクセスを行う処理を実装してください

IEC62396-102 規格を満たすため、書き込み処理が 300ms 以内に完了するようにしてください。

2.2.6 異常通知

ハードウェア要件に記載している異常検出機構にて異常状態が発生及び解消したときは、以下の API 関数を呼んでください。

●灯具の異常発生
R_DALI102_SetLampFailure 関数
●灯具の異常解消
R_DALI102_ClearLampFailure 関数
●Control Gear 全体に関わる異常発生
R_DALI102_SetControlGearFailure 関数
●Control Gear 全体に関わる異常解消
R_DALI102_ClearControlGearFailure 関数

2.2.7 調光制御

ハードウェア要件に記載している DALI 通信回路を制御し、R_DALI102_GetActualLevel 関数 又は R_DALI102_GetActualLevelHighRes 関数 を定期的に呼び出して取得した actual level に合わせて調光を行ってください。

基本的には取得した actual level が示す調光率に即時調光させる必要があります。しかし、調光率が 0%から 0%以外に変更されたタイミングに限っては適切な startup 時間を設ける必要があります。

startup 時間とは、消灯状態の灯具に点灯指示を出してから点灯状態になるまでの時間のことです。使用する灯具の特性に合わせて時間を設定してください。

startup の開始時に R_DALI102_NotifyBeginStartup 関数を、startup 終了時に R_DALI102_NotifyEndStartup 関数をそれぞれ呼び出してください。

また、灯具が点灯状態になったとき R_DALI102_SetLampOn 関数を、消灯状態になったとき R_DALI102_ClearLampOn 関数をそれぞれ呼び出してください。

2.2.8 メモリバンク実体定義

Control Gear が持つメモリバンクの構造やその内容は一部を除いてユーザ依存となるため本ライブラリには含んでいません。IEC62386-102 規格書を参考にメモリバンクの実体を実装してください。

規格書で規定されている仕様を2.2.8.1～2.2.8.3に示します。

2.2.8.1 メモリバンク 0

メモリバンク 0 は logical unit 毎に必須となるメモリバンクです。Control Gear 及び logical unit に関する情報が格納されます。

表 2.1 メモリバンク0のメモリマップ(1/2)

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

表 2.2 メモリバンク0のメモリマップ(2/2)

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

2.2.8.2 メモリバンク 1

メモリバンク 1 は logical unit 毎に任意で追加可能なメモリバンクです。OEM 仕様に関する追加情報を設定するために予約されています。

表 2.3 メモリバンク1のメモリマップ(1/2)

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

表 2.4 メモリバンク1のメモリマップ(2/2)

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

2.2.8.3 メモリバンク 2～199

メモリバンク 2～199 は logical unit 毎に任意で追加可能なメモリバンクです。各バンクの内容は基本仕様を満たす範囲内であれば自由に定義をすることが可能です。

表 2.5 メモリバンク2～199のメモリマップ

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

2.2.9 メモリバンクアクセス関数実装

本ライブラリ単体では、ユーザが実体定義したメモリバンクにアクセスすることができません。そのため、提供するコールバック関数のプロトタイプに合わせて各アクセス関数を実装してください。

詳細は R_DALI102_InitLibrary 関数 の関数仕様を参照してください。

各アクセス関数は本ライブラリを使用する上で必須の関数と任意実装（オプション）の関数があります。各アクセス関数の実装が必須か否かを示す表を以下に示します。

表 2.6 メモリバンクアクセス関数要否一覧

アクセス関数	必須／オプション
RESET 関数	必須
READ 関数	必須
WRITE 関数	必須
UNLATCH READ 関数	オプション
CANCEL WRITE 関数	オプション

2.2.9.1 RESET 関数

引数にて指定したバンクの全 location を RESET 値に設定する関数を実装してください。引数と戻り値のフォーマットは以下となります。この関数は本ライブラリを使用する上で必須です。

関数フォーマット	
引　数	uint8_t unit 　logical unitの番号 uint8_t bank 　リセット対象とするバンク番号
戻り値	void

2.2.9.2 READ 関数

引数にて指定した location の値を読み出す関数を実装してください。引数と戻り値のフォーマットは以下となります。この関数は本ライブラリを使用する上で必須です。

関数フォーマット	
引　数	uint8_t unit 　logical unitの番号 uint8_t bank 　読み出し対象とするバンク番号 uint8_t location 　読み出すバンクの指定ロケーション
戻り値	int16_t 　0x00 - 0xFF：読み出し値 　DALI102_MB_IS_NOT_IMPLEMENTED：指定したバンクが未実装 　DALI102_MB_LOCATION_IS_NOT_IMPLEMENTED：指定したロケーションが未実装 　DALI102_MB_EXECUTE_ERROR：実行エラー

2.2.9.3 WRITE 関数

引数にて指定した location に値を書き込む関数を実装してください。引数と戻り値のフォーマットは以下となります。この関数は本ライブラリを使用する上で必須です。

関数フォーマット	
引　数	uint8_t unit 　logical unitの番号 uint8_t bank 　書き込み対象とするバンク番号 uint8_t location 　書き込むバンクの指定ロケーション 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 　DALI102_MB_IS_NOT_IMPLEMENTED：指定したバンクが未実装 　DALI102_MB_LOCATION_IS_NOT_IMPLEMENTED：指定したロケーションが未実装 　DALI102_MB_EXECUTE_ERROR：実行エラー

2.2.9.4 UNLATCH READ 関数

IEC62386-102 規格のメモリバンク規定としてマルチバイト（連続する複数 location のメモリデータを組み合わせて一つのデータとして意味を成す）データが許されています。マルチバイトデータを 1byte ずつ読み出す途中でデータ値の一部が変更されてしまうと一つのマルチバイトデータとして有効性がなくなってしまいます。そのため、ユーザはメモリバンク内にマルチバイトデータを配置する際に、対象マルチバイトデータ読み出し開始から読み出し終了までの間、値が変更されないようデータラッチ（保持）する機能を実装することが推奨されています。マルチバイトデータに対するデータラッチ機能を実装した場合のみ、引数にて指定した logical unit の読み出し中におけるデータラッチを解除する関数を実装してください。この関数は本ライブラリを使用する上でオプションとなります。

関数フォーマット	
引　数	uint8_t unit 　logical unitの番号
戻り値	void

2.2.9.5 CANCEL WRITE 関数

IEC62386-102 規格のメモリバンク規定としてマルチバイト（連続する複数 location のメモリデータを組み合わせて一つのデータとして意味を成す）データが許されています。マルチバイトデータを 1byte ずつ更新する途中で書き込みをやめてしまうと一つのマルチバイトデータとして有効性がなくなってしまいます。そのため、ユーザはメモリバンク内にマルチバイトデータを配置する際に、そのため、マルチバイトデータの書き込み時に「最初から最後までのデータが書き込まれた場合のみ一括してデータを書き込む」または「最後のデータまで書き込みが行われなかった場合に更新前のデータに書き戻す」処理を行う機能が必要になります。最後までのデータ書き込みが行われなかったことを明確に通知するための関数になります。

メモリバンクの実体定義内に上記の仕様を実装している場合のみ、引数にて指定した logical unit のマルチバイトデータに対する書き込みをキャンセルする関数を実装してください。この関数は本ライブラリを使用する上でオプションとなります。

関数フォーマット	
引　数	uint8_t unit 　logical unitの番号
戻り値	void

3. DALI102 ライブラリ機能

本ライブラリの機能について説明します。

3.1 データ型、戻り値の定義

本ライブラリで提供するデータ型を以下に記載します。

表 3.1 データ型一覧

型名	説明
dali102_t	DALI102 モジュール型
dali102_cmd_t	コマンド情報型

本ライブラリで提供する定義マクロを以下に記載します。

表 3.2 マクロ一覧

マクロ名	マクロ値	説明
DALI102_NO_ANSWER	(int16_t)-1	Backward データなし
DALI102_MB_IS_NOT_IMPLEMENTED	(-1)	メモリバンク未実装
DALI102_MB_LOCATION_IS_NOT_IMPLEMENTED	(-2)	ロケーション未実装
DALI102_MB_EXECUTE_ERROR	(-3)	実行エラー

表 3.3 Light source type(dali102_light_source_type_t)一覧

マクロ名	マクロ値	説明
LIGHT_SOURCE_TYPE_LOW_PRESSURE_FLUORESCENT	0	低圧蛍光灯
LIGHT_SOURCE_TYPE_HID	2	HID
LIGHT_SOURCE_TYPE_LOW_VOLTAGE_HALOGEN	3	定電圧ハロゲン灯
LIGHT_SOURCE_TYPE_INCANDESCENT	4	白熱灯
LIGHT_SOURCE_TYPE_LED	6	LED
LIGHT_SOURCE_TYPE_OLED	7	有機 EL
LIGHT_SOURCE_TYPE_OTHER	252	その他
LIGHT_SOURCE_TYPE_UNKNOWN	253	不明
LIGHT_SOURCE_TYPE_NO_LIGHT	254	灯具なし

本ライブラリで提供する戻り値を以下に記載します。

表 3.4 戻り値(dali102_return_t)一覧

定義	戻り値	説明
DALI102_RETURN_OK	0	正常終了
DALI102_RETURN_ERR	-1	異常終了

3.2 構造体一覧

本ライブラリで提供する構造体を以下に記載します。

汎用コールバック関数型構造体(dali102_general_callback_t)の定義

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

メモリバンク操作コールバック関数型構造体(dali102_mb_if_callback_t)の定義

```
typdef struct
{
    void (*Reset)(uint8_t unit, uint8_t bank);
    int16_t (*Read)(uint8_t unit, uint8_t bank, uint8_t location);
    int16_t (*Write)(uint8_t unit, uint8_t bank, uint8_t location, uint8_t data);
    void (*UnlatchRead)(uint8_t unit);
    void (*CancelWrite)(uint8_t unit);
} dali102_mb_if_callback_t;
```

default 値型構造体(dali102_default_t)の定義

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

NVM 変数型構造体(dali102_nvm_t)の定義

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

3.3 API 関数一覧

本ライブラリの API 関数一覧を以下に記載します。

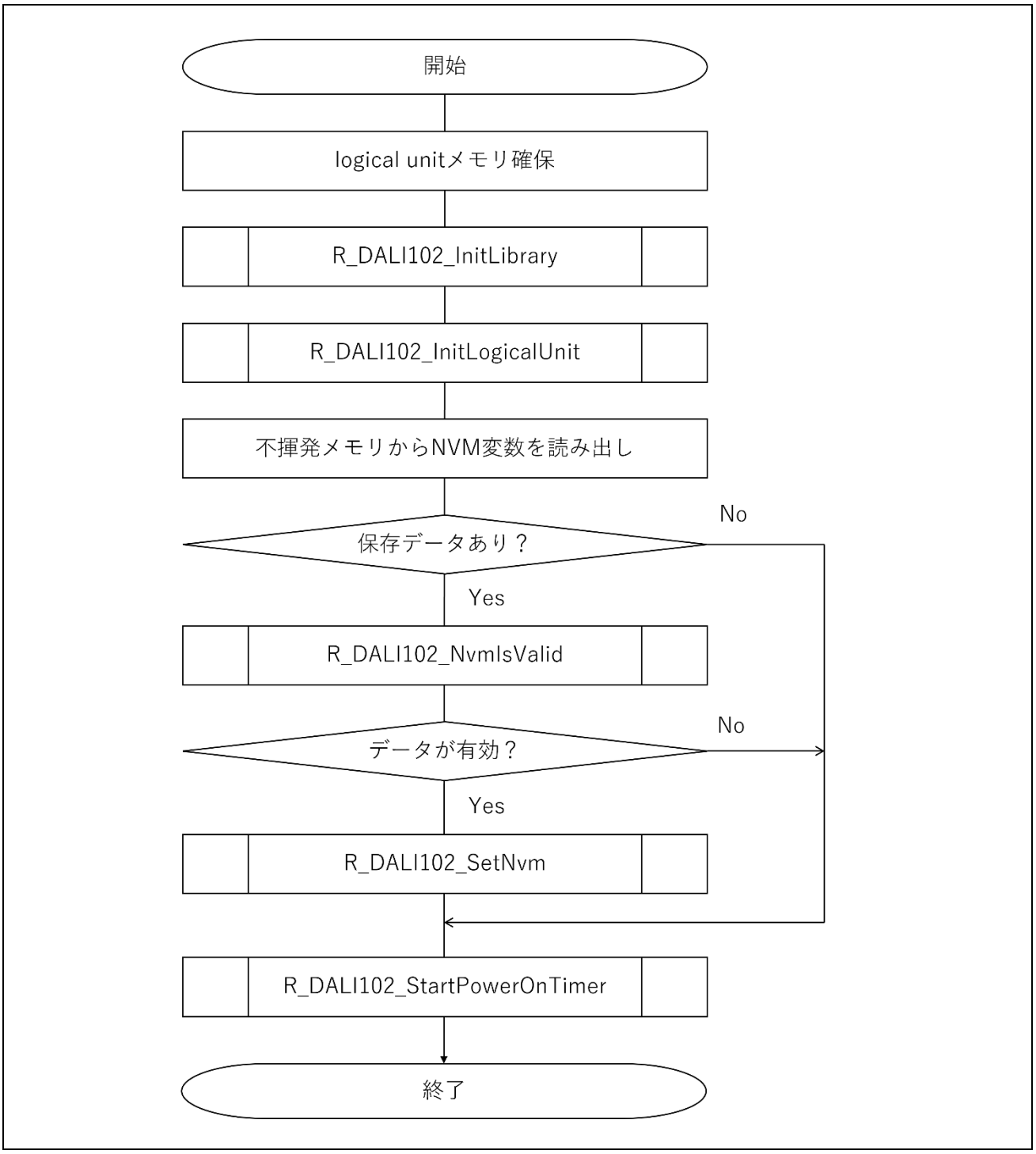
表 3.5 API関数一覧

関数名	説明
R_DALI102_InitLibrary	DALI102 ライブラリの初期化
R_DALI102_InitLogicalUnit	logical unit の初期化
R_DALI102_NvmIsValid	NVM 変数値の有効範囲内チェック
R_DALI102_SetNvm	NVM 変数値の設定
R_DALI102_GetNvm	NVM 変数値の取得
R_DALI102_NvmIsChanged	NVM 変数値変更チェック
R_DALI102_NeedsToSaveNvm	NVM 変数保存必要チェック
R_DALI102_NotifySaveNvm	NVM 変数保存通知
R_DALI102_StartPowerOnTimer	power on timer 開始
R_DALI102_GetOperatingMode	operating mode 値取得
R_DALI102_Tick1ms	内部動作を 1ms 進行
R_DALI102_NotifyBeginStartup	startup の開始通知
R_DALI102_NotifyEndStartup	startup の終了通知
R_DALI102_SetLampOn	lamp on 状態のセット
R_DALI102_ClearLampOn	lamp on 状態のクリア
R_DALI102_SetLampFailure	lamp failure 状態のセット
R_DALI102_ClearLampFailure	lamp failure 状態のクリア
R_DALI102_SetControlGearFailure	control gear failure 状態のセット
R_DALI102_ClearControlGearFailure	control gear failure 状態のクリア
R_DALI102_NotifySystemFailure	system failure 通知
R_DALI102_GetActualLevel	actual level 取得
R_DALI102_GetActualLevelHighRes	高分解能 actual level 取得
R_DALI102_IdentificationIsActive	identification の active チェック
R_DALI102_CreateCommand	コマンド情報作成
R_DALI102_ExecuteCommand	受信コマンド実行
R_DALI102_GetLibraryVersion	ライブラリバージョン取得

3.4 概略フローチャート

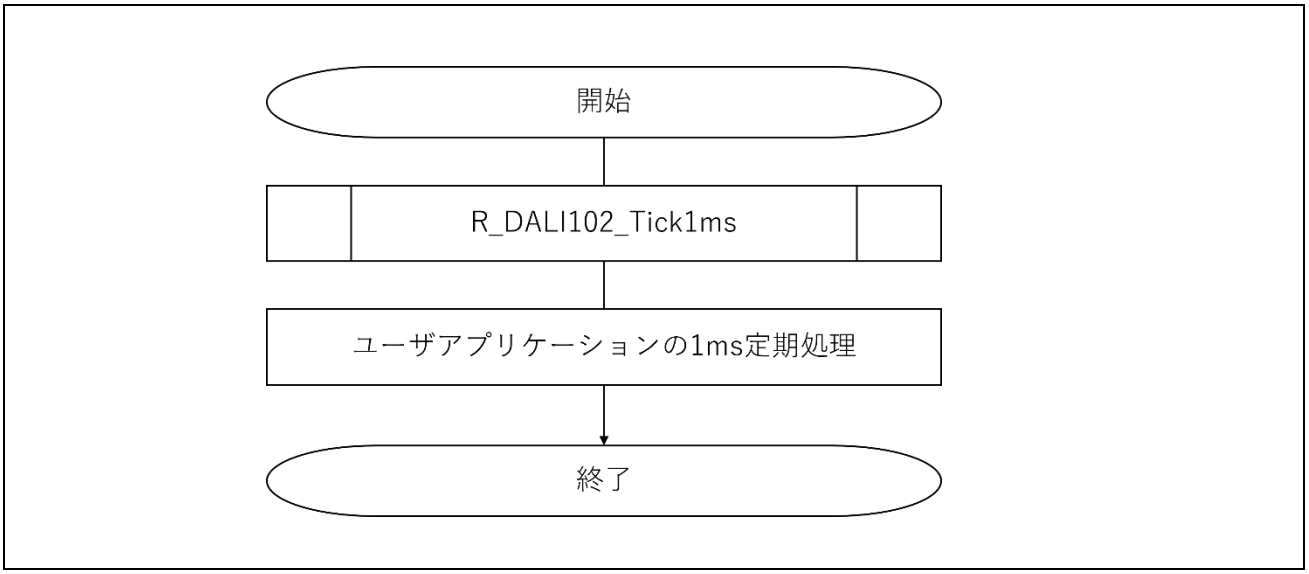
3.4.1 初期化時

初期化時のフローを記載します。



3.4.2 1ms 定期処理

1ms 定義処理のフローを記載します。この処理は 1ms 間隔で処理を行ってください。

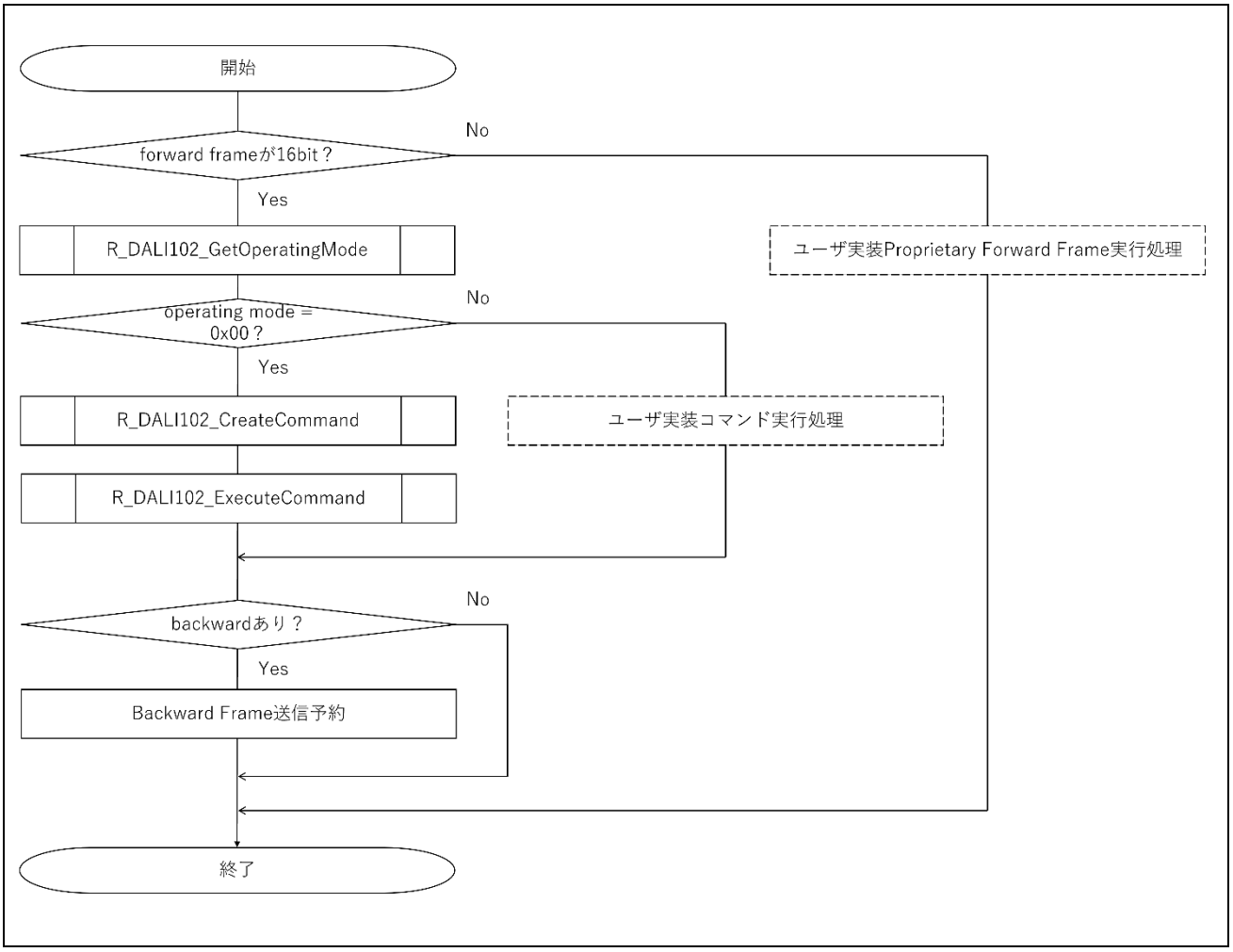


3.4.3 Forward Frame 受信時

Forward Frame 受信時処理のフローを記載します。DALI 通信バスが Forward Frame を受信した際に処理を行ってください。

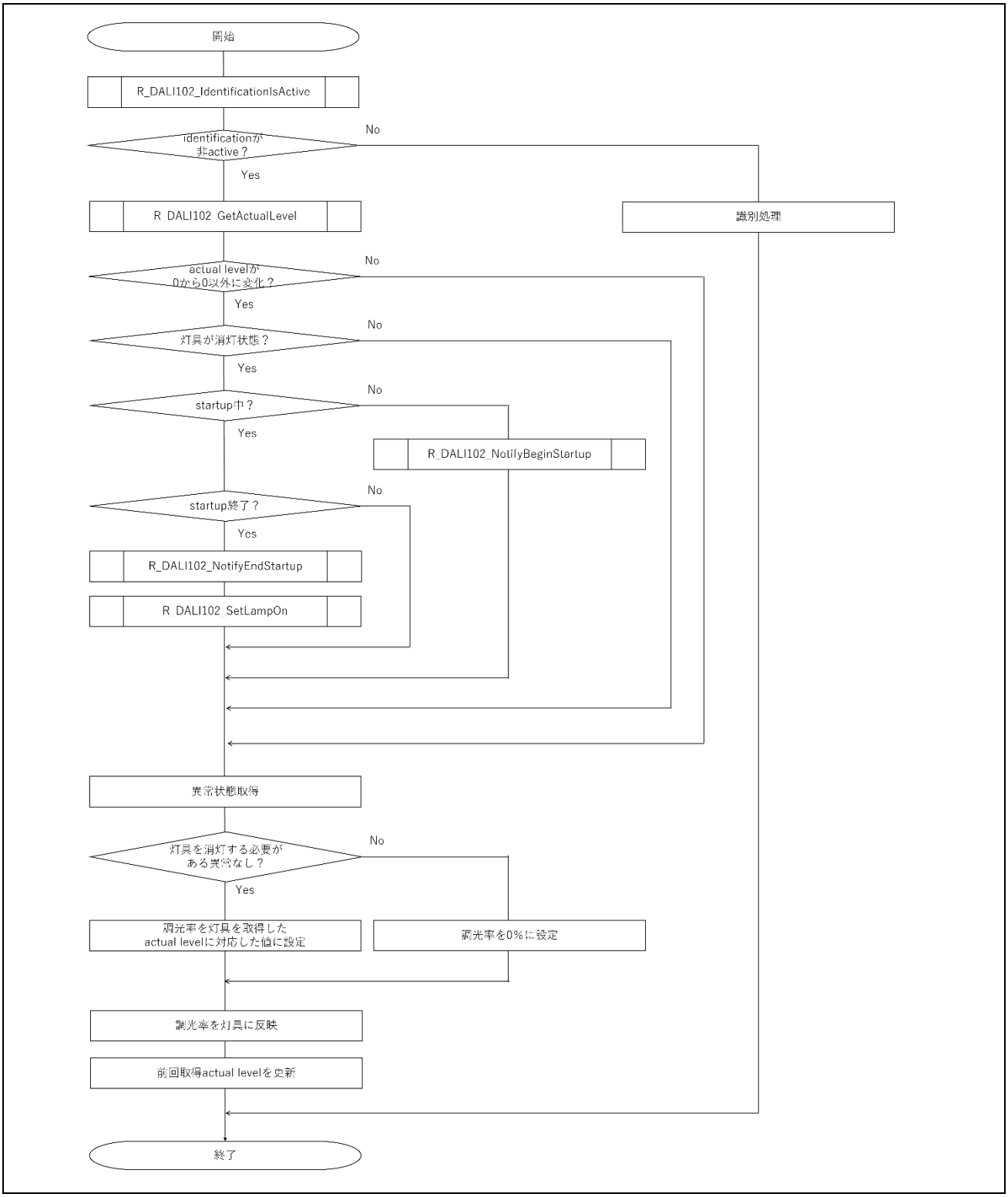
Proprietary Forward Frame(16bit 超、かつ, 20bit, 24bit 以外の Forward Frame)に対する処理はオプション機能となります。DALI 通信ドライバ及びアプリケーションが対応している場合に実装してください。

また、0 以外の operating mode はオプション機能です。独自モードが必要な場合実装の上、R_DALI102_InitLogicalUnit 関数にてモード番号を登録してください。



3.4.4 調光処理

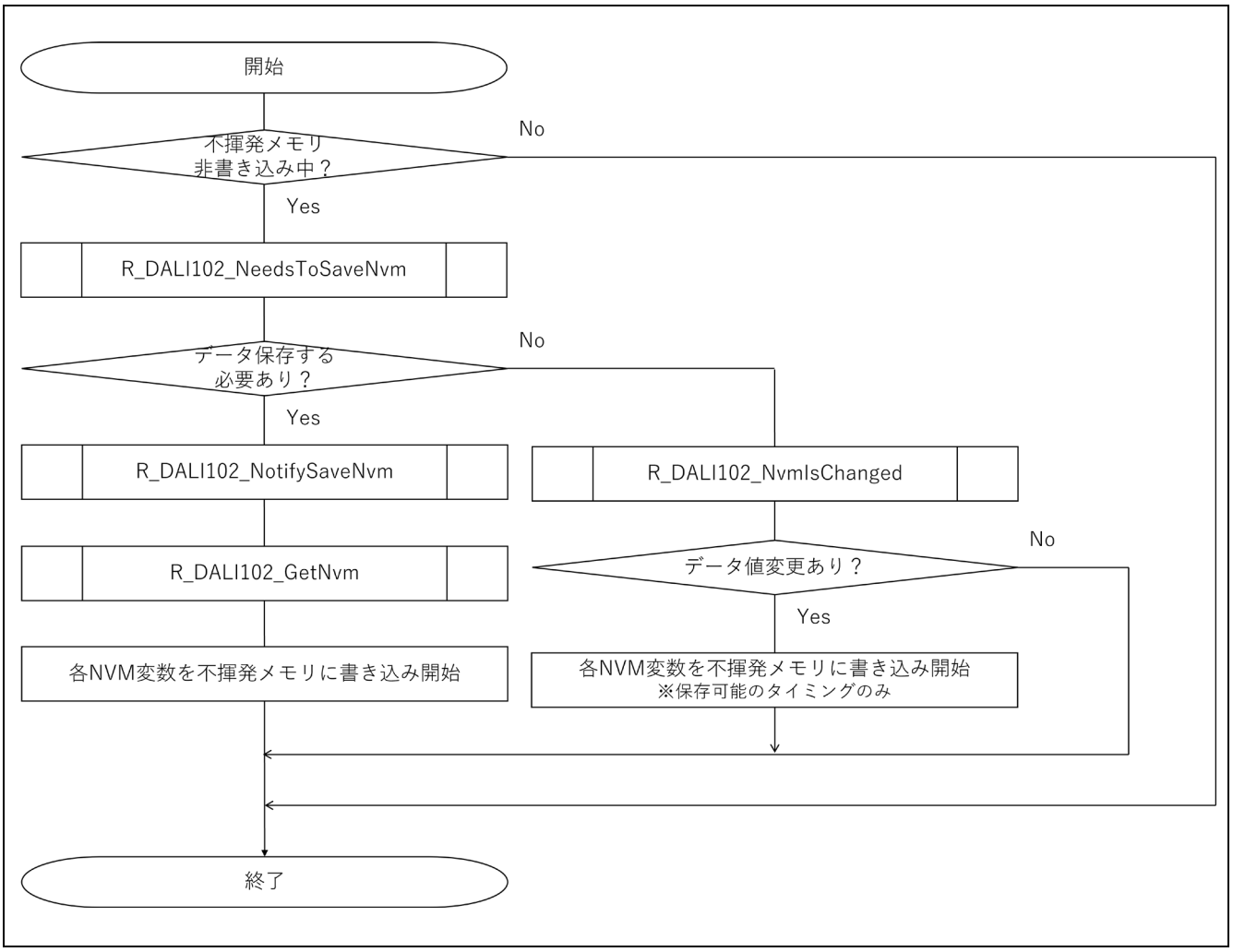
調光処理のフローを記載します。定期的に処理を行ってください。
また、識別処理はユーザ依存処理になります。ユーザで仕様定義と実装をしてください。



3.4.5 不揮発データ処理

不揮発データ処理のフローを記載します。

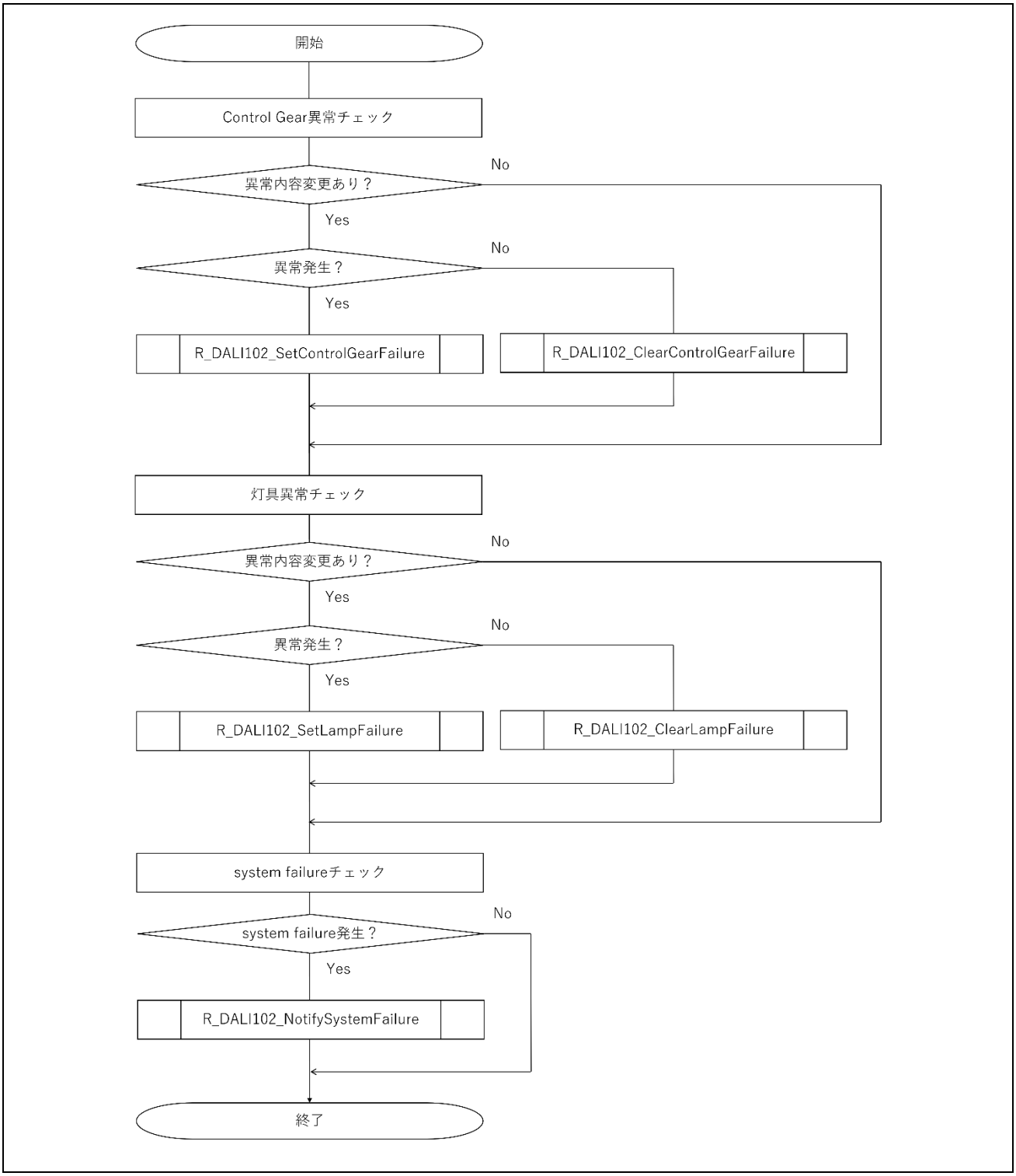
SAVE PERSISTENT VARIABLES コマンド受信から 300ms 以内に不揮発メモリに保存完了することが規定されています。また、SAVE PERSISTANT VARIABLES VARIABLES コマンドを受信していなくとも NVM 変数値に変更があった場合は、30 秒以内に保存することが規定されています。それぞれ規定時間内に保存完了するように定期的にチェックを行い、処理を行ってください。



3.4.6 異常処理

異常処理のフローを記載します。異常状態が更新されたタイミングで呼び出してください。

Control Gear 異常、灯具異常それぞれの詳細仕様はハードウェア及びソフトウェアに依存します。環境に合わせて仕様を定義、実装を検討してください。



3.5 API 関数仕様

本ライブラリの API 関数仕様を以下に記載します。

3.5.1.1 R_DALI102_InitLibrary

【概　要】
DALI102 ライブラリの初期化を行います。

【書　式】

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

【前提条件】
特になし。

【引　数】

引　数	説　明
const dali102_general_callback_t * p_gen_callback	汎用コールバック関数へのポインタ [メンバ] .GetRandomValue 2.2.4章に記載している関数フォーマットに沿った関数のポインタを設定してください。 NULL での設定は許容されません。
const dali102_mb_if_callback_t * p_mb_callback	メモリバンクコールバック関数へのポインタ [メンバ] .Reset 2.2.9.1章に記載している関数フォーマットに沿った関数のポインタを設定してください。 NULL での設定は許容されません。 .Read 2.2.9.2章に記載している関数フォーマットに沿った関数のポインタを設定してください。 NULL での設定は許容されません。 .Write 2.2.9.3章に記載している関数フォーマットに沿った関数のポインタを設定してください。 NULL での設定は許容されません。 .UnlatchRead 2.2.9.4章に記載している関数フォーマットに沿った関数のポインタを設定してください。 オプション機能ですので、NULL での設定が許容されます。 .CancelWrite 2.2.9.5章に記載している関数フォーマットに沿った関数のポインタを設定してください。 オプション機能ですので、NULL での設定が許容されます。

【戻り値】

値	説　明
DALI102_RETURN_OK	正常終了
DALI102_RETURN_ERR	パラメータエラー - 引数設定を見直してください。

3.5.1.2 R_DALI102_InitLogicalUnit

【概　要】

指定した logical unit の初期化を行います。
必要な logical unit の数だけ呼び出してください。

【書　式】

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

【前提条件】

1. R_DALI102_InitLibrary 関数が正常終了していること。

【引　数】

引　数	説　明
dali102_t * p_this	DALI102 モジュールへのポインタ
uint8_t unit_index	logical unit の index 番号 有効範囲：0x00～0x3F ※使用するメモリバンク 0 の location 0x1A に指定する index 番号と同じ値を設定してください。
uint16_t phm_fade_time_ms	物理的にフェード可能な最小時間 [単位：ms] 有効範囲：1～700
const dali102_default_t * p_default_value	factory burn-in の default 値 有効範囲： - phm：1～254 - operating_mode：0x00, 0x80～0xFF
const uint8_t * p_light_source_list	light source type 配列へのポインタ 設定方法は次頁を参照してください。
const uint8_t * p_mode_list	operating mode 配列へのポインタ 設定方法は次頁を参照してください。

【戻り値】

値	説　明
DALI102_RETURN_OK	正常終了
DALI102_RETURN_ERR	パラメータエラー - 引数設定を見直してください。

(1) p_light_source_list パラメータの設定

uint8_t 型配列の先頭ポインタを設定してください。
配列の先頭要素に登録する light source の数を、2 つ目の要素以降に dali102_light_source_type_t 列挙型にて定義された light source type を代入します。配列の設定例は以下です。

例 1）LED のみ登録する場合

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

例 2）HID と OLED を登録する場合

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

(2) p_mode_list パラメータの設定

uint8_t 型配列の先頭ポインタを設定してください。
配列の先頭要素に登録する operating mode の数を、2 つ目の要素以降に operating mode 値を代入します。
使用要件として、以下を満たしてください。

- 必ず 0x00（DALI 規定モード）を含めてください。
- manufacturer specific mode(0x80～0xFF)の範囲で追加登録してください。
- 登録した operating mode のいずれかを p_default_value.operating_mode に設定してください。

配列の設定例は以下です。

例 1）DALI 規定モードのみ登録する場合

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

例 2）DALI 規定モードと manufacturer specific mode として 0x80 と 0x90 を登録する場合

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

3.5.1.3 R_DALI102_NvmIsValid

【概　要】

dali102_nvm_t 型変数のメンバに設定している値が全て有効範囲内かどうかを返します。
後述の R_DALI102_SetNvm 関数に値を設定する前に必ず呼び出してチェックしてください。

【書　式】

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

【前提条件】

1. R_DALI102_InitLibrary 関数が正常終了していること。
2. R_DALI102_InitLogicalUnit 関数が正常終了していること。

【引　数】

引　数	説　明
const dali102_t * p_this	DALI102 モジュールへのポインタ
const dali102_nvm_t * p_nvm	DALI102 モジュール用 NVM 変数へのポインタ 有効範囲： - last_light_level：0x00, min_level～max_level - power_on_level：0x00～0xFF - system_failure_level：0x00～0xFF - min_level：phm～max_level - max_level：min_level～0xFE - fade_rate：0x01～0x0F - fade_time：0x00～0x0F - extended_fade_time.base：0x00～0x0F - extended_fade_time.multiplier：0x00～0x04 - short_address：0x00～0x3F, 0xFF - random_address：0x000000～0xFFFFFF - operating_mode：0x00、0x80～0xFF - gear_groups：0x0000～0xFFFF - scene：0x00～0xFF

【戻り値】

値	説　明
true	全ての変数が有効範囲内
false	少なくとも一つの変数が有効範囲外

3.5.1.4 R_DALI102_SetNvm

【概　要】

DALI102 モジュールに NVM 変数値を設定します。

電源投入時に不揮発メモリに NVM 変数のデータが保存されているときに、読み出したデータを設定するために使用してください。

【書　式】

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

【前提条件】

1. R_DALI102_InitLibrary 関数が正常終了していること。
2. R_DALI102_InitLogicalUnit 関数が正常終了していること。
3. R_DALI102_NvmIsValid 関数で NVM 変数が有効範囲内であることを確認していること。

【引　数】

引　数	説　明
dali102_t * p_this	DALI102 モジュールへのポインタ
const dali102_nvm_t * p_nvm	DALI102 モジュール用 NVM 変数へのポインタ

【戻り値】

なし

3.5.1.5 R_DALI102_GetNvm

【概　要】

DALI102 モジュールから NVM 変数設定値を取得します。
不揮発メモリに最新の NVM 変数値を保存する際に使用してください。

【書　式】

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

【前提条件】

1. R_DALI102_InitLibrary 関数が正常終了していること。
2. R_DALI102_InitLogicalUnit 関数が正常終了していること。

【引　数】

引　数	説　明
const dali102_t * p_this	DALI102 モジュールへのポインタ
dali102_nvm_t * p_nvm	DALI102 モジュール用 NVM 変数へのポインタ

【戻り値】

なし

3.5.1.6 R_DALI102_NvmIsChanged

【概　要】

少なくとも一つの NVM 変数値に変更があったかどうかを取得します。

本関数の戻り値が true だった場合、ハードウェアの状態に応じて NVM 変数を不揮発メモリに保存してください。

本関数にて取得できる状態は、前回本関数を呼ばれたとき（初回呼び出し時は起動時）からが対象となります。連続して呼び出すと戻り値が false になりますのでご注意ください。

【書　式】

bool R_DALI102_NvmIsChanged (dali102_t * p_this)

【前提条件】

1. R_DALI102_InitLibrary 関数が正常終了していること。
2. R_DALI102_InitLogicalUnit 関数が正常終了していること。

【引　数】

引　数	説　明
dali102_t * p_this	DALI102 モジュールへのポインタ

【戻り値】

値	説　明
true	値変更あり
false	値変更なし

3.5.1.7 R_DALI102_NeedsToSaveNvm

【概　要】

NVM 変数の保存要求(SAVE PERSISTENT VARIABLES コマンドを受信)があったかどうかを取得します。

本関数の戻り値が true だった場合、ハードウェアの状態に応じて NVM 変数値を不揮発メモリに保存してください。本関数での保存の要求に対して保存処理ができる状態になったら、後述の R_DALI102_NotifySaveNvm 関数を呼び出すことで通知してください。通知を行うまで本関数で取得できる状態はクリアされません。

【書　式】

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

【前提条件】

1. R_DALI102_InitLibrary 関数が正常終了していること。
2. R_DALI102_InitLogicalUnit 関数が正常終了していること。

【引　数】

引　数	説　明
const dali102_t * p_this	DALI102 モジュールへのポインタ

【戻り値】

値	説　明
true	保存する必要あり
false	保存する必要なし

3.5.1.8 R_DALI102_NotifySaveNvm

【概　要】

R_DALI102_NeedsToSaveNvm 関数による保存要求に対して NVM 変数値を不揮発メモリに保存する処理ができる状態になった際に呼び出してください。

なるべく早い段階で本関数を呼び出すことで、次の保存要求を受け付けられる時間が長くなります。

【書　式】

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

【前提条件】

1. R_DALI102_InitLibrary 関数が正常終了していること。
2. R_DALI102_InitLogicalUnit 関数が正常終了していること。
3. R_DALI102_NeedsToSaveNvm 関数の戻り値が true 状態であること。

【引　数】

引　数	説　明
dali102_t * p_this	DALI102 モジュールへのポインタ

【戻り値】

なし

3.5.1.9 R_DALI102_StartPowerOnTimer

【概　要】

電源投入から Power On Level が反映されるまでのタイマを開始します。

IEC62386-102 規格上、電源投入から Power on Level が反映されるまでの時間は 600ms±10%以内である必要があります。

ハードウェアの電源投入から本関数を呼ぶまでの時間を加味した上で規格を満たす時間を設定してください。

【書　式】

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

【前提条件】

1. R_DALI102_InitLibrary 関数が正常終了していること。
2. R_DALI102_InitLogicalUnit 関数が正常終了していること。

【引　数】

引　数	説　明
dali102_t * p_this	DALI102 モジュールへのポインタ
uint16_t msec	Power on level を適用するまでの時間

【戻り値】

なし

3.5.1.10 R_DALI102_GetOperatingMode

【概　要】

動作中の operating mode 値を取得します。

取得した operating mode 設定値に合わせて、受信した Forward Frame に対する処理を切り替えてください。本ライブラリが提供する R_DALI102_ExecuteCommand 関数は operating mode が 0 の場合の処理関数です。

【書　式】

```
uint8_t R_DALI102_GetOperatingMode ( const dali102_t * p_this )
```

【前提条件】

1. R_DALI102_InitLibrary 関数が正常終了していること。
2. R_DALI102_InitLogicalUnit 関数が正常終了していること。

【引　数】

引　数	説　明
const dali102_t * p_this	DALI102 モジュールへのポインタ

【戻り値】

値	説　明
0x00	IEC62386-102 規格にて規定されている受信 Forward Frame に対する処理を行うモードです。 後述の R_DALI102_ExecuteCommand 関数を実行することで処理を行ってください。
0x80～0xFF	ユーザ独自実装のモードです。 受信した Forward Frame は取得した operating mode 値に対応したユーザ実装処理にて処理を行ってください。

3.5.1.11 R_DALI102_Tick1ms

【概　要】
DALI102 モジュールの内部動作を 1ms 進めます。
1ms 毎に定期的に呼び出してください。

【書　式】

void R_DALI102_Tick1ms (dali102_t * p_this)

【前提条件】
1. R_DALI102_InitLibrary 関数が正常終了していること。
2. R_DALI102_InitLogicalUnit 関数が正常終了していること。

【引　数】

引　数	説　明
dali102_t * p_this	DALI102 モジュールへのポインタ

【戻り値】
なし

3.5.1.12 R_DALI102_NotifyBeginStartup

【概　要】

灯具の startup 開始を通知します。

灯具が消灯している状態から点灯状態に移行するにあたり、点灯までにかかる準備段階の開始時に呼び出してください。

【書　式】

```
void R_DALI102_NotifyBeginStartup ( dali102_t * p_ths )
```

【前提条件】

1. R_DALI102_InitLibrary 関数が正常終了していること。
2. R_DALI102_InitLogicalUnit 関数が正常終了していること。

【引　数】

引　数	説　明
dali102_t * p_this	DALI102 モジュールへのポインタ

【戻り値】

なし

3.5.1.13 R_DALI102_NotifyEndStartup

【概　要】

灯具の startup 終了を通知します。
R_DALI102_NotifyBeginStartup 関数呼び出し後、灯具が実際に点灯した直後に呼び出してください。

【書　式】

void R_DALI102_NotifyEndStartup (dali102_t * p_this)

【前提条件】

1. R_DALI102_InitLibrary 関数が正常終了していること。
2. R_DALI102_InitLogicalUnit 関数が正常終了していること。
3. R_DALI102_NotifyBeginStartup 関数を事前に呼び出していること。

【引　数】

引　数	説　明
dali102_t * p_this	DALI102 モジュールへのポインタ

【戻り値】

なし

3.5.1.14 R_DALI102_SetLampOn

【概　要】
灯具の点灯状態を通知します。
灯具が点灯状態になった直後に呼び出してください。

【書　式】

void R_DALI102_SetLampOn (dali102_t * p_this)

【前提条件】
1. R_DALI102_InitLibrary 関数が正常終了していること。
2. R_DALI102_InitLogicalUnit 関数が正常終了していること。

【引　数】

引　数	説　明
dali102_t * p_this	DALI102 モジュールへのポインタ

【戻り値】
なし

3.5.1.15 R_DALI102_ClearLampOn

【概　要】
灯具の消灯を通知します。
灯具が消灯になった直後に呼び出してください。

【書　式】

void R_DALI102_ClearLampOn (dali102_t * p_this)

【前提条件】
1. R_DALI102_InitLibrary 関数が正常終了していること。
2. R_DALI102_InitLogicalUnit 関数が正常終了していること。

【引　数】

引　数	説　明
dali102_t * p_this	DALI102 モジュールへのポインタ

【戻り値】
なし

3.5.1.16 R_DALI102_SetLampFailure

【概　要】

灯具に関する異常発生を通知します。

灯具に関する異常が発生した場合に呼び出してください。

なお、灯具に関する具体的な異常内容はハードウェア及びソフトウェアに依存します。ユーザで定義した異常内容と関連付けてご使用ください。（例：灯具との接続回路のショート／オープン）

【書　式】

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

【前提条件】

1. R_DALI102_InitLibrary 関数が正常終了していること。
2. R_DALI102_InitLogicalUnit 関数が正常終了していること。

【引　数】

引　数	説　明
dali102_t * p_this	DALI102 モジュールへのポインタ

【戻り値】

なし

3.5.1.17 R_DALI102_ClearLampFailure

【概　要】
灯具に関する異常解消を通知します。
灯具に関する異常がすべて解消した場合に呼び出してください。

【書　式】

void R_DALI102_ClearLampFailure (dali102_t * p_this)

【前提条件】
1. R_DALI102_InitLibrary 関数が正常終了していること。
2. R_DALI102_InitLogicalUnit 関数が正常終了していること。
3. R_DALI102_SetLampFailure 関数を事前に呼び出していること。

【引　数】

引　数	説　明
dali102_t * p_this	DALI102 モジュールへのポインタ

【戻り値】
なし

3.5.1.18 R_DALI102_SetControlGearFailure

【概　要】

Control Gear 全体に関する異常発生を通知します。

Control Gear 全体に関する異常が発生した場合に呼び出してください。

なお、Control Gear 全体に関する具体的な異常内容はハードウェア及びソフトウェアに依存します。ユーザで定義した異常内容と関連付けてご使用ください。（例：動作保証以上の温度上昇）

【書　式】

void R_DALI102_SetControlGearFailure (dali102_t * p_this)

【前提条件】

1. R_DALI102_InitLibrary 関数が正常終了していること。
2. R_DALI102_InitLogicalUnit 関数が正常終了していること。

【引　数】

引　数	説　明
dali102_t * p_this	DALI102 モジュールへのポインタ

【戻り値】

なし

3.5.1.19 R_DALI102_ClearControlGearFailure

【概　要】

Control Gear 全体に関する異常解消を通知します。
Control Gear 全体に関する異常がすべて解消した場合に呼び出してください。

【書　式】

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

【前提条件】

1. R_DALI102_InitLibrary 関数が正常終了していること。
2. R_DALI102_InitLogicalUnit 関数が正常終了していること。
3. R_DALI102_SetControlGearFailure 関数を事前に呼び出していること。

【引　数】

引　数	説　明
dali102_t * p_this	DALI102 モジュールへのポインタ

【戻り値】

なし

3.5.1.20 R_DALI102_NotifySystemFailure

【概　要】

system failure が発生したことを通知します。

DALI 通信バスが ACTIVE 状態(LOW レベル)を 500ms±10%継続したタイミングで呼び出してください。

本関数呼び出し後も system failure 状態が継続していても定期的に呼び出しする必要はなく、また、DALI 通信バスが IDLE 状態(HIGH レベル)に変化したとしてもライブラリとしては行うべき処理はありません。

【書　式】

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

【前提条件】

1. R_DALI102_InitLibrary 関数が正常終了していること。
2. R_DALI102_InitLogicalUnit 関数が正常終了していること。

【引　数】

引　数	説　明
dali102_t * p_this	DALI102 モジュールへのポインタ

【戻り値】

なし

3.5.1.21 R_DALI102_GetActualLevel

【概　要】

actual level を取得します。
定期的に本関数を呼び出して、取得した actual level に対応した調光率で灯具を調光させてください。
actual level と調光率の対応は IEC62386-102 規格書を参照してください。

【書　式】

void R_DALI102_GetActualLevel (const dali102_t * p_this)

【前提条件】

1. R_DALI102_InitLibrary 関数が正常終了していること。
2. R_DALI102_InitLogicalUnit 関数が正常終了していること。

【引　数】

引　数	説　明
const dali102_t * p_this	DALI102 モジュールへのポインタ

【戻り値】

値	説　明
0x00～0xFE	actual level 値

3.5.1.22 R_DALI102_GetActualLevelHighRes

【概　要】

高分解能 actual level を取得します。本関数は actual level の範囲 0x00～0xFE を 256 倍した 0x0000～0xFE00 の範囲に拡張した値が取得できます。より詳細な調光制御を行いたい場合に使用します。

定期的に本関数を呼び出して、取得した actual level に対応した調光率で灯具を調光させてください。

actual level と調光率の対応は IEC62386-102 規格書を参照してください。

【書　式】

void R_DALI102_GetActualLevelHighRes (const dali102_t * p_this)

【前提条件】

1. R_DALI102_InitLibrary 関数が正常終了していること。
2. R_DALI102_InitLogicalUnit 関数が正常終了していること。

【引　数】

引　数	説　明
const dali102_t * p_this	DALI102 モジュールへのポインタ

【戻り値】

値	説　明
0x0000～0xFE00	actual level 値 (単位：actual level / 256)

3.5.1.23 R_DALI102_IdentificationIsActive

【概　要】

Identification が active 状態か否かを取得します。Identification とは、Control Gear 設置者が特定の Control Gear を識別できるようにする、試運転中に使用される一時的な状態のことを指します。

本関数の戻り値が true である間、指定 logical unit を識別できる処理を実行してください。

Identification 中の識別処理内容としては、actual level を一時的に無視し、灯具を 0～100%の任意の調光率で調光することができます。具体的な処理内容はユーザ依存であるため、セット仕様を加味した内容をユーザで実装してください。（例：1 秒間隔で点滅）

【書　式】

bool R_DALI102_IdentificationIsActive (const dali102_t * p_this)

【前提条件】

1. R_DALI102_InitLibrary 関数が正常終了していること。
2. R_DALI102_InitLogicalUnit 関数が正常終了していること。

【引　数】

引　数	説　明
const dali102_t * p_this	DALI102 モジュールへのポインタ

【戻り値】

値	説　明
true	Identification が active
false	Identification が非 active

3.5.1.24 R_DALI102_CreateCommand

【概　要】

DALI 通信ドライバを通して受信した 16bit Forward Frame から本ライブラリで処理できるコマンド情報を作成します。

【書　式】

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

【前提条件】

1. R_DALI102_InitLibrary 関数が正常終了していること。
2. R_DALI102_InitLogicalUnit 関数が正常終了していること。

【引　数】

引　数	説　明
uint16_t forward	受信した 16bit Forward Frame
bool twice	twice 状態 直近 100ms 以内に受信した frame と同じ frame：true それ以外：false

【戻り値】

メンバ変数	説　明
dali102_cmd_num_t num	コマンド番号
uint8_t address_byte	address byte データ
uint8_t opcode_byte	opcode byte データ
bool is_received_twice	twice 状態
uint16_t execute_condition	コマンド実行条件

3.5.1.25 R_DALI102_ExecuteCommand

【概　要】

受信した DALI コマンドを実行します。
本関数の戻り値に応じて DALI 通信バスに Backward Frame を送信してください。

【書　式】

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

【前提条件】

1. R_DALI102_InitLibrary 関数が正常終了していること。
2. R_DALI102_InitLogicalUnit 関数が正常終了していること。
3. R_DALI102_CreateCommand 関数にてコマンド情報を取得していること。

【引　数】

引　数	説　明
dali102_t * p_this	DALI102 モジュールへのポインタ
const dali102_cmd_t * p_cmd	コマンド情報へのポインタ

【戻り値】

値	説　明
0x00～0xFF	backward frame
DALI102_NO_ANSWER	backward frame なし

3.5.1.26 R_DALI102_GetLibraryVersion

【概　要】
本ライブラリのバージョン番号を取得します。

【書　式】

uint8_t R_DALI102_GetLibraryVersion (void)

【前提条件】
なし

【引　数】
なし

【戻り値】

値	説　明
uint16_t	バージョン番号（形式：0xXXYY） XX：メジャーバージョン YY：マイナーバージョン

改訂記録	RL78 ファミリ DALI-2 Control Gear ライブラリ ユーザーズマニュアル 基本(102)編

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	15th, Jun., 2022	ー	初版発行
1.01	1st, Nov., 2022	ー	一部表現を変更

RL78ファミリ DALI-2 Control Gearライブラリ
ユーザーズマニュアル 基本(102)編

発行年月日　2022年11月1日　　Rev.1.01

発行　　ルネサス エレクトロニクス株式会社
　　〒135-0061　東京都江東区豊洲3-2-24（豊洲フォレシア）

RL78 ファミリ

ルネサス エレクトロニクス株式会社

R01US0535JJ0101