User's Manual

RL78 ファミリ

DALI-2 Input Device ライブラリ

ユーザーズマニュアル 基本(103)編

16 ビット・シングルチップ・マイクロコンピュータ

本資料に記載の全ての情報は本資料発行時点のものであり、ルネサス エレクトロニクスは、
予告なしに、本資料に記載した製品または仕様を変更することがあります。
ルネサス エレクトロニクスのホームページなどにより公開される最新情報をご確認ください。

ルネサス エレクトロニクス
www.renesas.com

Rev.1.00 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。

© 2022 Renesas Electronics Corporation. All rights reserved.

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 Input Device 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/G23 ユーザーズマニュアル ハードウェア編	R01UH0896JJ0100
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. DALI103i ライブラリ概要 1

1.1 ライブラリ機能概要 1
1.2 ソフトウェア構成 2
1.3 対応規格 3
1.4 ファイル一覧 3
1.5 リソース 4
1.6 開発環境 4
1.7 注意事項 5

2. プログラミング環境 6

2.1 ハードウェア要件 6
2.1.1 DALI 通信回路 6
2.1.2 不揮発領域 6
2.1.3 異常検出機構 6
2.1.4 信号処理装置 6
2.2 ソフトウェア要件 7
2.2.1 DALI103i モジュール定義 7
2.2.2 DALI103i Instance モジュール定義 7
2.2.3 DALI 通信ドライバ 7
2.2.4 時間管理 8
2.2.5 乱数生成 8
2.2.6 不揮発領域アクセス 8
2.2.7 異常通知 8
2.2.8 メモリバンク実体定義 9
2.2.9 メモリバンクアクセス関数実装 12

3. DALI103i ライブラリ機能 15

3.1 データ型、戻り値の定義 15
3.2 構造体一覧 16
3.3 API関数一覧 18
3.4 概略フローチャート 19
3.4.1 初期化時 19
3.4.2 1ms 定期処理 20
3.4.3 Event Message 処理 21

3.4.4 Forward Frame 受信時 22
3.4.5 不揮発データ処理 23
3.4.6 異常処理 24
3.5 API関数仕様 25
3.5.1 R_DALI103I_InitLibrary 25
3.5.2 R_DALI103I_InitLogicalUnit 27
3.5.3 R_DALI103I_InitInstance 29
3.5.4 R_DALI103I_DeviceNvmIsValid 30
3.5.5 R_DALI103I_InstanceNvmIsValid 31
3.5.6 R_DALI103I_SetDeviceNvm 32
3.5.7 R_DALI103I_SetInstanceNvm 33
3.5.8 R_DALI103I_GetDeviceNvm 34
3.5.9 R_DALI103I_GetInstanceNvm 35
3.5.10 R_DALI103I_DeviceNvmIsChanged 36
3.5.11 R_DALI103I_InstanceNvmIsChanged 37
3.5.12 R_DALI103I_NeedsToSaveNvm 38
3.5.13 R_DALI103I_NotifySaveNvm 39
3.5.14 R_DALI103I_StartPowerCycleTimer 40
3.5.15 R_DALI103I_GetOperatingMode 41
3.5.16 R_DALI103I_Tick1ms 42
3.5.17 R_DALI103I_SetInputDeviceError 43
3.5.18 R_DALI103I_ClearInputDeviceError 44
3.5.19 R_DALI103I_SetInstanceError 45
3.5.20 R_DALI103I_ClearInstanceError 46
3.5.21 R_DALI103I_InstanceIsActive 47
3.5.22 R_DALI103I_GetPowerCycleNotification 48
3.5.23 R_DALI103I_SetInputSignal 49
3.5.24 R_DALI103I_GetInstanceEventFilter 51
3.5.25 R_DALI103I_GenerateInputNotification 52
3.5.26 R_DALI103I_GetTestFrame 53
3.5.27 R_DALI103I_IdentificationIsActive 54
3.5.28 R_DALI103I_QuiescentModeIsActive 55
3.5.29 R_DALI103I_CreateCommand 56
3.5.30 R_DALI103I_ExecuteCommand 57
3.5.31 R_DALI103I_GetLibraryVersion 58

1. DALI103i 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 Input Device 用ライブラリとして、IEC62386-103 規格(以降、DALI103)のうち Input Device に対する仕様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、主に送受信時の処理、タイミング処理等を担っています。ハードウェアに関する規格部分 (ハードウェア依存部) 及び電源制御についてはアプリケーションノートを参考に、DALI 規格に準拠するようユーザで作りこみを行ってください。

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 ・不揮発データアクセス ・異常検出 ・Instance の状態更新	・受信 24bit 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 ・送信 Event Message Frame 処理

本ライブラリでは、DALI 通信によって受信した 24bit 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、ユーザアプリケーションによって受信した 24bit Forward Frame を、API 関数を呼び出すことでライブラリに通知、処理を行います。

DALI 変数設定コマンドや DALI 変数設定値取得コマンドなど、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

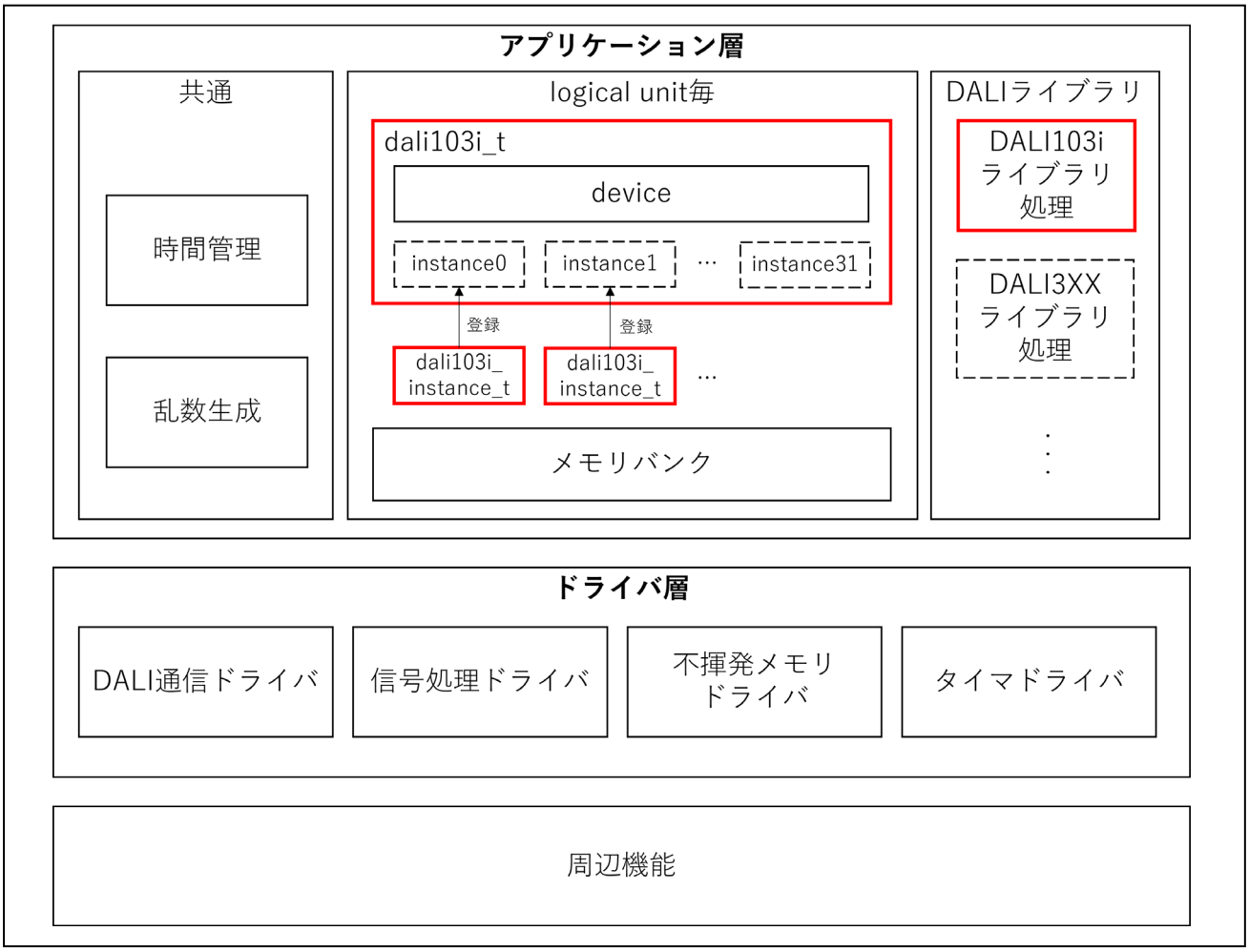
注：DALI システムとして、一つの DALI ネットワークに対して Input Device の最大接続数は 64 です。DALI ネットワークの構成とハードウェアの制限を加味し、合計で 64 を超えない数で設定してください。

1.2 ソフトウェア構成

本ライブラリを使用した場合における、Input Device のソフトウェア構成を以下に示します。
赤線で囲んだ部分を本ライブラリが提供します。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 IEC62386-3XX 規格の仕様をライブラリ化した DALI3XX ライブラリを使用して拡張することが可能です。

dali103i_t 型(詳細は後述)変数は Input Device の logical unit のパラメータから構成され、Input Device における device の実体を 1 つと instance の実体を登録できる受け口を 32 個保有しています。
dali103i_instance_t 型(詳細は後述)変数は dali103i_t 型に登録可能な instance のパラメータを保有しています。
ユーザは、必要な数の logical unit 及び instance を実体化することで所望の Input Device を実現できます。

図 1-1 Input Device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およびコンパイラ環境は以下となります。

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

1.4 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_103i_cc_gen2_v1_00.lib r_dali_103i_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_dali103i_api.h	ライブラリヘッダファイル
r_dali103i_dvar.h	device変数モジュールの定義ヘッダファイル
r_dali103i_ivar.h	instance変数モジュールの定義ヘッダファイル
r_dali103i_timer.h	タイマモジュールの定義ヘッダファイル
r_dali103i_list.h	リストモジュールの定義ヘッダファイル
r_dali103i_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_dali103i_itx_if.h	汎用InstanceType I/Fモジュールの定義ヘッダファイル
r_dali103i_ftx_if.h	汎用FeatureType I/Fモジュールの定義ヘッダファイル
r_dali103i_device.h	deviceモジュールの定義ヘッダファイル
r_dali103i_instance.h	instanceモジュールの定義ヘッダファイル
r_dali103i_common.h	複数モジュールにて使用する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を以下に示します。

Input Device 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)、Input Device 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	13,432 [byte]
		RAMサイズ	4 [byte]
	最大スタックサイズ		74 [byte] (R_DALI103I_ExecuteCommand関数)
IAR	ライブラリリソース	ROMサイズ	16,088 [byte]
		RAMサイズ	4 [byte]
	最大スタックサイズ		68 [byte] (R_DALI103I_ExecuteCommand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	dali103i_t	152 [byte / logical unit]
	dali103i_instance_t	92 [byte / instance]
IAR	dali103i_t	152 [byte / logical unit]
	dali103i_instance_t	92 [byte / instance]

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 V2022-04
	C コンパイラ	Renesas CC-RL V1.11.00
	CPU コア	RL78-S2/S3 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 V4.21.4
	C コンパイラ	IAR C/C++ Compiler For Renesas RL78 V4.21.4
	CPU コア	RL78-S3 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中の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. dali103i_t 型構造体や dali103i_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 Input Device 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

Input Device は不揮発領域に 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 異常検出機構

Input Device 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が必要になります。ユーザアプリケーションにて検出部分を実装するようにしてください。

2.1.4 信号処理装置

Input Device を構成する一部として instance というものがあります。instance はスイッチ、センサなどの入力信号を処理することで動作します。

用途に合わせた信号処理装置を実装してください。

2.2 ソフトウェア要件

2.2.1 DALI103i 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 Input Device に相当する)の単位を logical unit といいます。本ライブラリは Input Device 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(dali103i_t)を提供します。dali103i_t 型の変数のことを DALI103i モジュールと呼びます。

必要な logical unit 数分の DALI103i モジュールを定義してください。

2.2.2 DALI103i Instance モジュール定義

DALI 規格では 1 つの Input Device につき、必要個数に応じて 1～32 個の instance(信号処理装置)を実装することができます。また、instance は扱う入力信号に応じて 32 種類の instance type に分類されています。

各 instance type と定義される規格を以下に記載します。

表 2-1 instance typeと定義規格

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

DALI103 規格をライブラリ化した本ライブラリでは、instance type 0 の instance 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(dali103i_instance_t)を提供します。dali103i_instance_t 型変数のことを DALI103i instance モジュールと呼びます。instance type 0 の instance を必要な場合、必要な数分の DALI103i instance モジュールを定義してください。

また、instance type 1 以降の instance を実装したい場合は、独自に当該部分を実装するか別途 DALI3XX ライブラリを使用してください。

2.2.3 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以下アプリケーションノートを参照してください。

「RL78/G23 照明通信マスタボード 初期ファームウェア」 (R01AN6460JJ0100)

2.2.4 時間管理

本ライブラリは時間管理を行う 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_DALI103I_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。

2.2.5 乱数生成

Input Device 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 dali103i_general_callback_t 型構造体変数のメンバ Get Random Value 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

2.2.6 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
IEC62396-103 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。

2.2.7 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。

- Input Device に関わる異常発生
  R_DALI103I_SetInputDeviceError 関数
- Input Device に関わる異常解消
  R_DALI103I_ClearInputDeviceError 関数
- instance type 0 の instance に関わる異常発生
  R_DALI103I_SetInstanceError 関数
- instance type 0 の instance に関わる異常解消
  R_DALI103I_ClearInstanceError 関数

2.2.8 メモリバンク実体定義

Input Device 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。IEC62386-103 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。Input Device 及び logical unit に関する情報が格納されます。

表 2-2 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 2-3 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 2-4 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 2-5 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 2-6 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 3.5.1 R_DALI103I_InitLibrary 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 2-7 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 　DALI103I_MB_IS_NOT_IMPLEMENTED：指定したバンクが未実装 　DALI103I_MB_LOCATION_IS_NOT_IMPLEMENTED：指定したロケーションが未実装 　DALI103I_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 　DALI103I_MB_IS_NOT_IMPLEMENTED：指定したバンクが未実装 　DALI103I_MB_LOCATION_IS_NOT_IMPLEMENTED：指定したロケーションが未実装 　DALI103I_MB_EXECUTE_ERROR：実行エラー

2.2.9.4 UNLATCH READ 関数

IEC62386-103 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

IEC62386-103 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、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. DALI103i 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
dali103_t	DALI103i モジュール型
dali103i_instance_t	DALI103i instance モジュール型
dali103i_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
DALI103I_TEST_FRAME_MAX	(4)	Test frame の最大フレーム数
DALI103I_INSTANCE_NUM_MAX	(32)	Instance 最大数
DALI103I_NO_ANSWER	((int16_t)-1)	backward frame なし
DALI103I_ANSWER_IS_CORRUPTED	((int16_t)-2)	corrupted backward frame

本ライブラリで提供する戻り値を以下に記載します。

表 3-3 戻り値 (dali103i_return_t) 一覧

定義	戻り値	説明
DALI103I_RETURN_OK	0	正常終了
DALI103I_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 (dali103i_general_callback_t) の定義

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

メモリバンク操作コールバック関数型構造体 (dali103i_mb_if_callback_t) の定義

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

イベントメッセージ型構造体 (dali103i_event_t) の定義

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

Test Frame 型構造体 (dali103i_test_frame_t) の定義

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

device NVM 型構造体 (dali103i_device_nvm_t) の定義

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

instance NVM 型構造体 (dali103i_instance_nvm_t) の定義

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

3.3 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。

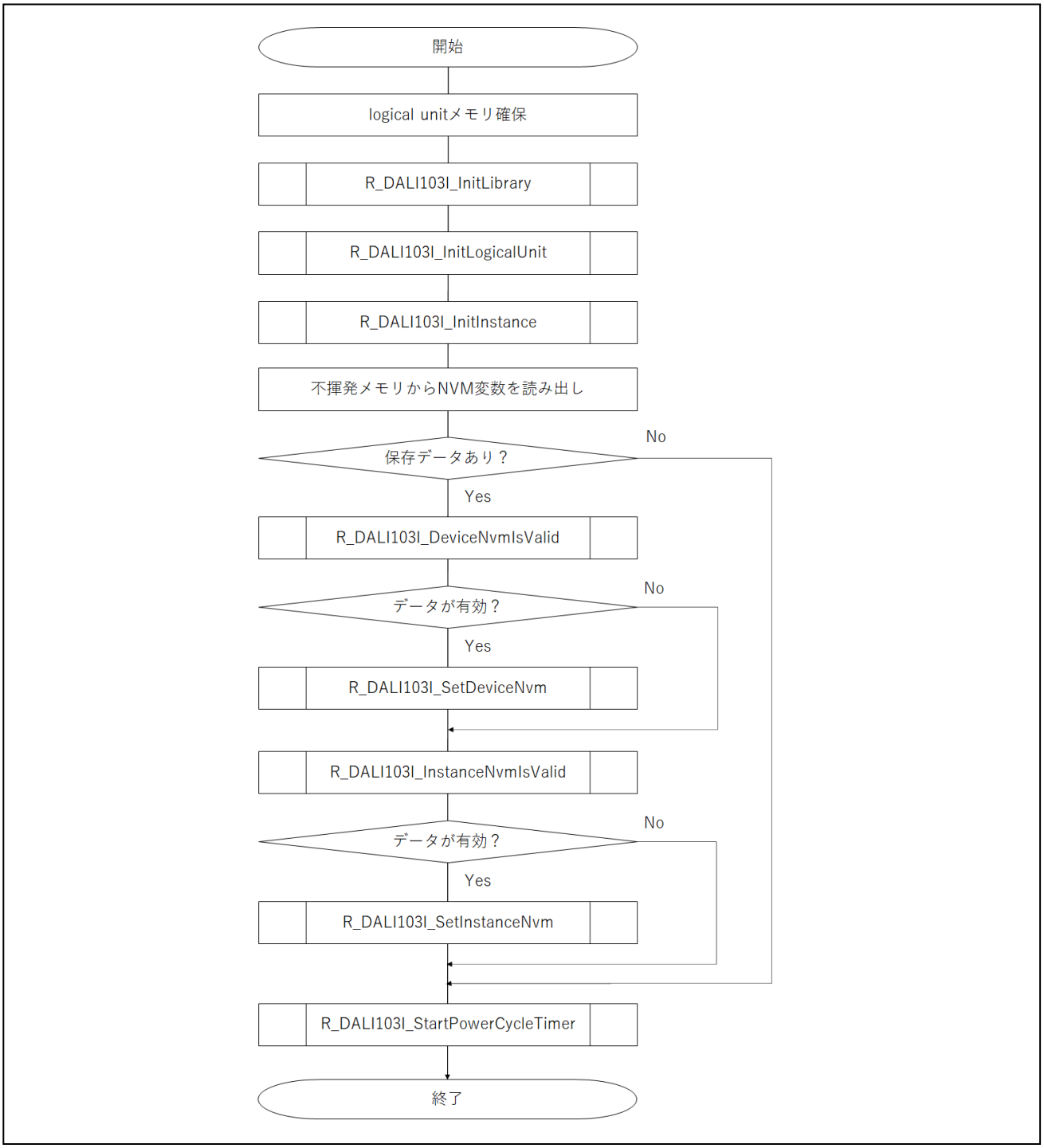
表 3-4 API関数一覧

関数名	説明
R_DALI103I_InitLibrary	DALI103i ライブラリの初期化
R_DALI103I_InitLogicalUnit	logical unit の初期化
R_DALI103I_InitInstance	instance の初期化
R_DALI103I_DeviceNvmIsValid	device NVM 変数値の有効範囲内チェック
R_DALI103I_InstanceNvmIsValid	instance NVM 変数値の有効範囲内チェック
R_DALI103I_SetDeviceNvm	device NVM 変数値の設定
R_DALI103I_SetInstanceNvm	instance NVM 変数値の設定
R_DALI103I_GetDeviceNvm	device NVM 変数値の取得
R_DALI103I_GetInstanceNvm	instance NVM 変数値の取得
R_DALI103I_DeviceNvmIsChanged	device NVM 変数値変更チェック
R_DALI103I_InstanceNvmIsChanged	instance NVM 変数値変更チェック
R_DALI103I_NeedsToSaveNvm	NVM 変数保存必要チェック
R_DALI103I_NotifySaveNvm	NVM 変数保存完了通知
R_DALI103I_StartPowerCycleTimer	power cycle notification タイマ開始
R_DALI103I_GetOperatingMode	operating mode 値取得
R_DALI103I_Tick1ms	内部動作を 1ms 進行
R_DALI103I_SetInputDeviceError	InputDeviceError 詳細情報の設定
R_DALI103I_ClearInputDeviceError	InputDeviceError 情報のクリア
R_DALI103I_SetInstanceError	InstanceError の設定
R_DALI103I_ClearInstanceError	InstanceError のクリア
R_DALI103I_InstanceIsActive	InstanceActive の状態確認
R_DALI103I_GetPowerCycleNotification	power cycle notification イベントの取得
R_DALI103I_SetInputSignal	input signal の設定
R_DALI103I_GetInstanceEventFilter	instance event filter の取得
R_DALI103I_GenerateInputNotification	input notification event の生成
R_DALI103I_GetTestFrame	test frame の取得
R_DALI103I_IdentificationIsActive	identification の active チェック
R_DALI103I_QuiescentModeIsActive	quiescent mode の active チェック
R_DALI103I_CreateCommand	コマンド情報作成
R_DALI103I_ExecuteCommand	受信コマンド実行
R_DALI103I_GetLibraryVersion	ライブラリバージョン取得

3.4 概略フローチャート

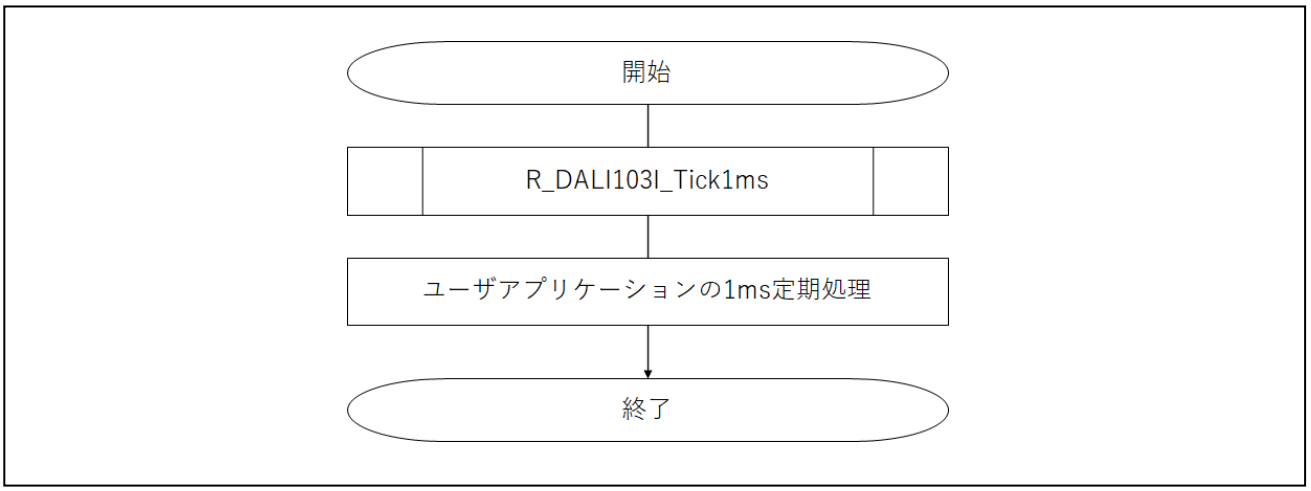
3.4.1 初期化時

初期化時のフローを記載します。



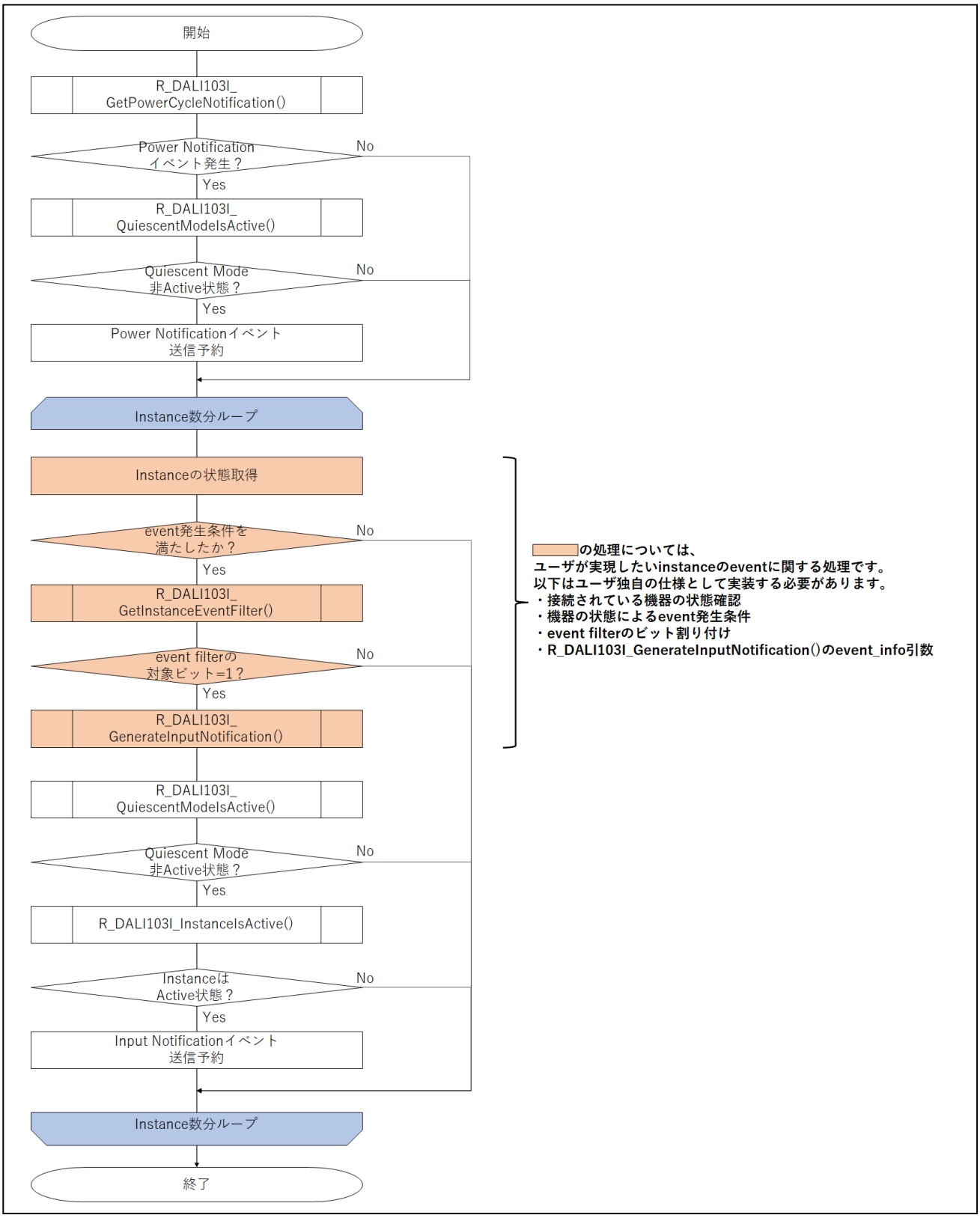
3.4.2 1ms 定期処理

1ms 定期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 Event Message 処理

Event Message 処理のフローを記載します。

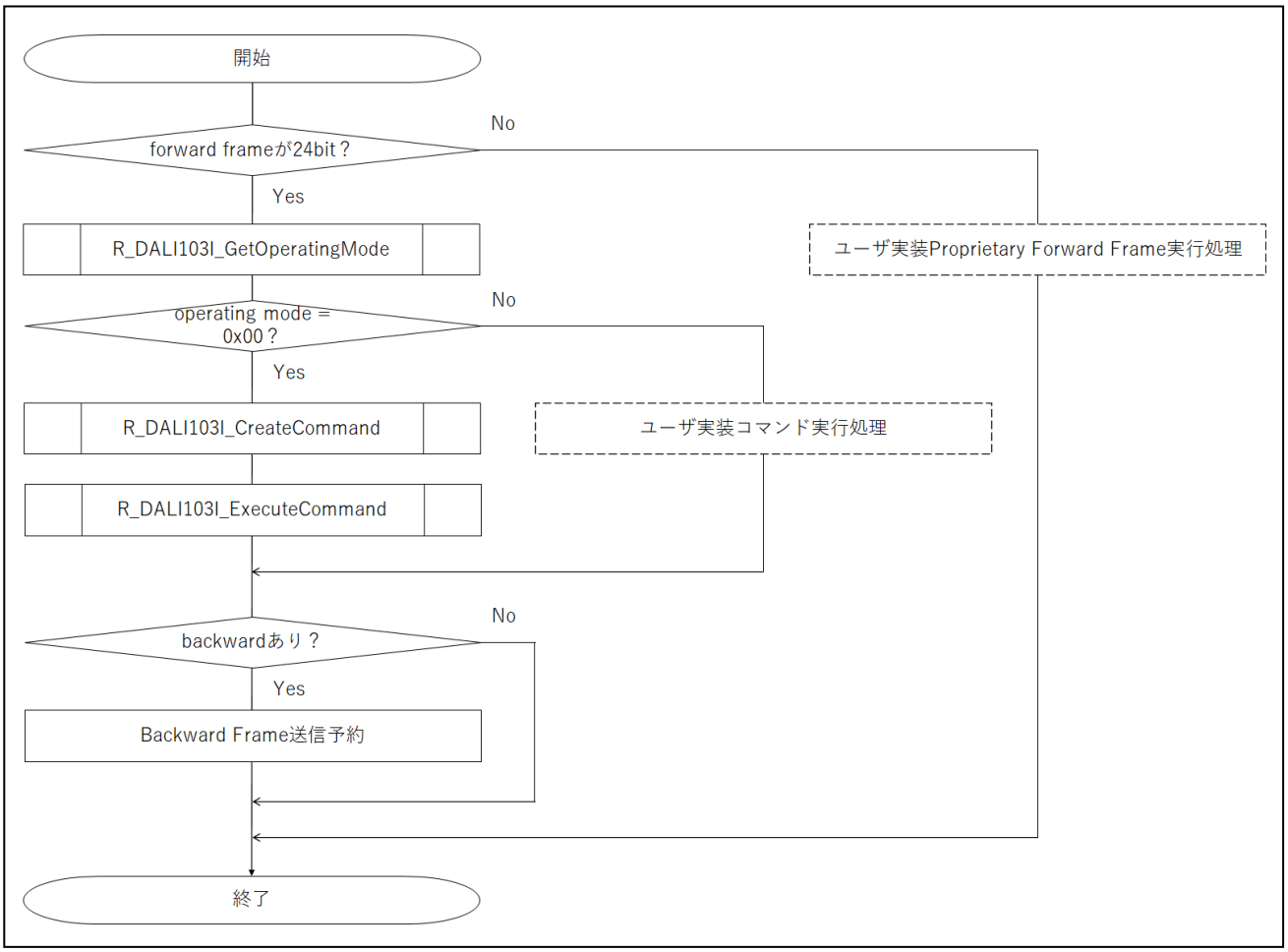


3.4.4 Forward Frame 受信時

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, 32bit 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。

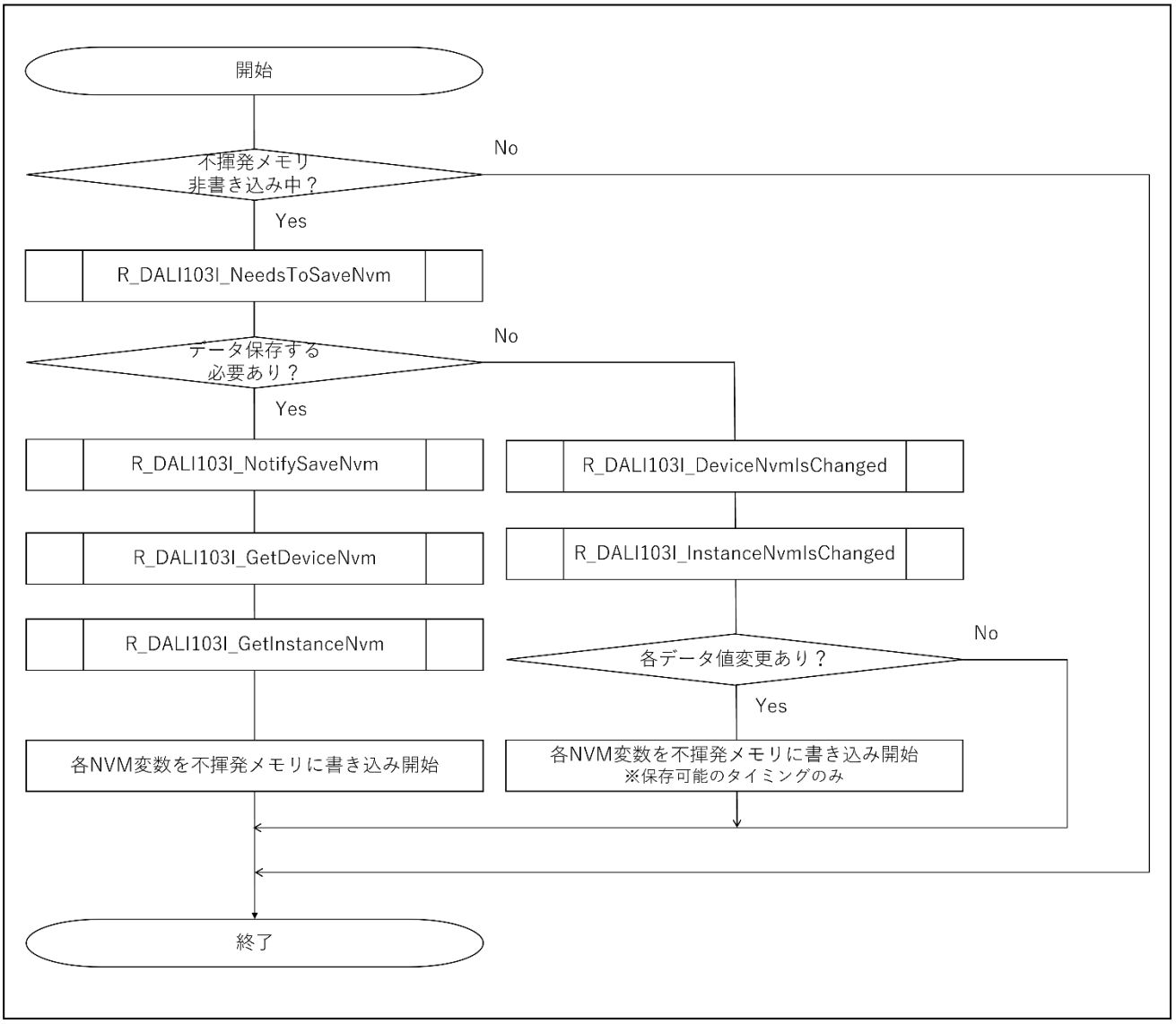
また、0 以外の operating mode はオプション機能です。独自モードが必要な場合実装の上、R_DALI103I_InitLogicalUnit 関数にてモード番号を登録してください。



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。

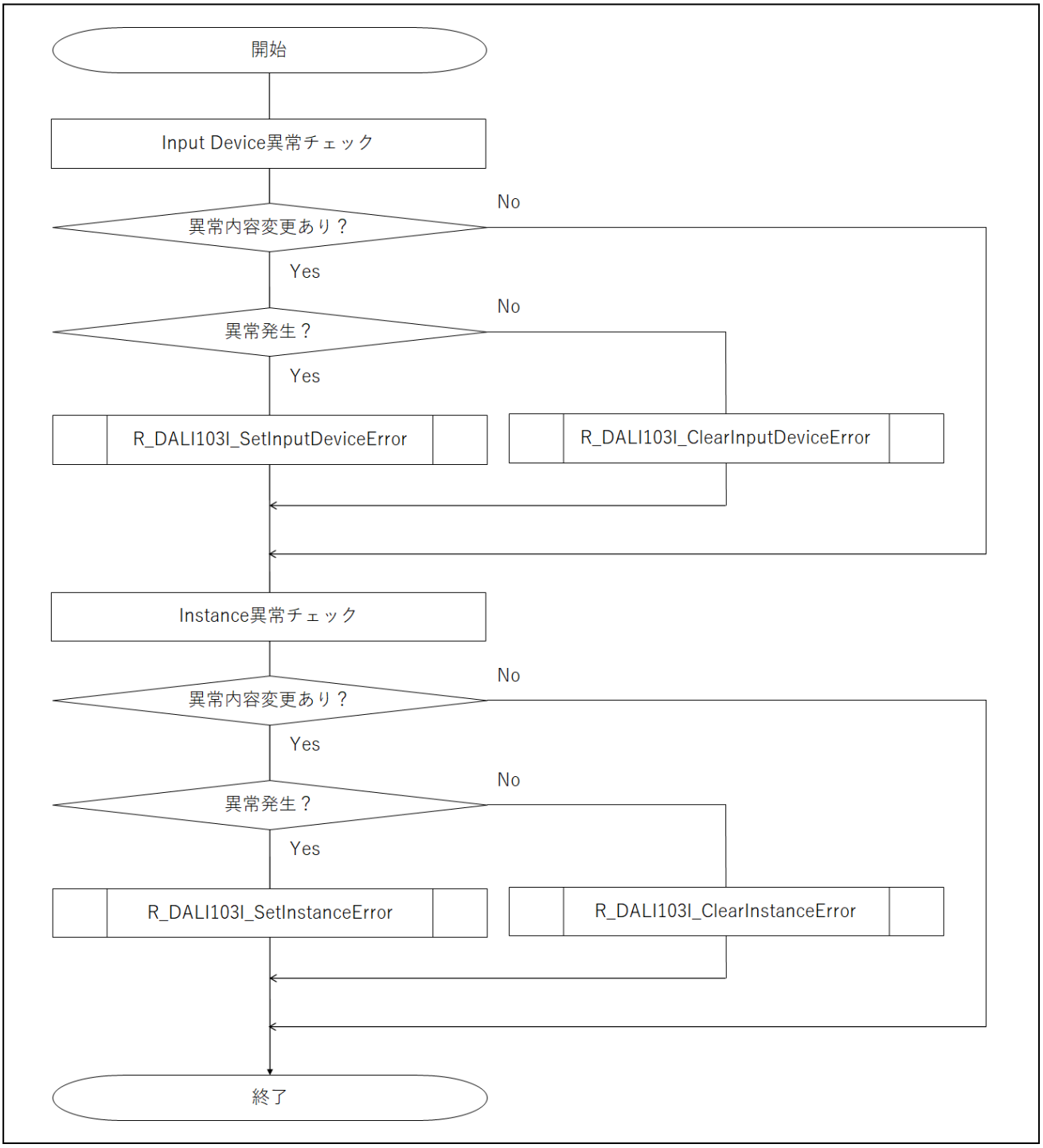
SAVE PERSISTENT VARIABLES コマンド受信から 300ms 以内に不揮発メモリに保存完了することが規定されています。また、SAVE PERSISTANT VARIABLES コマンドを受信していなくとも NVM 変数値に変更があった場合は、30 秒以内に保存することが規定されています。それぞれ規定時間内に保存完了するように定期的にチェックを行い、処理を行ってください。



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。

Input Device 異常、Instance 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 R_DALI103I_InitLibrary

【概　要】

DALI103i ライブラリの初期化を行います。

【書　式】

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

【前提条件】

特になし。

【引　数】

引　数	説　明
const dali103i_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 2.2.5 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。 NULL での設定は許容されません。
const dali103i_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
DALI103I_RETURN_OK	正常終了
DALI103I_RETURN_ERR	パラメータエラー - 引数設定を見直してください。

3.5.2 R_DALI103I_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】

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

【前提条件】

1. R_DALI103I_InitLibrary 関数が正常終了していること。

【引　数】

引　数	説　明
dali103i_t * p_this	DALI103i 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。
uint8_t default_operating_mode	operating mode の default 値 有効範囲： - operating_mode：0x00, 0x80～0xFF
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
DALI103I_RETURN_OK	正常終了
DALI103I_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_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 };
```

3.5.3 R_DALI103I_InitInstance

【概　要】

DALI103i instance モジュールを初期化し、DALI103i モジュール (dali103i_t 型) に instance を登録します。

dali103i_instance_t 型は InstanceType 0 の instance を提供します。InstanceType 0 以外の instance を使用する場合は InstanceType にあった DALI3XX ライブラリを使用してください。

【書　式】

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

【前提条件】

1. R_DALI103I_InitLibrary 関数が正常終了していること。
2. R_DALI103I_InitLogicalUnit 関数が正常終了していること。

【引　数】

引　数	説　明
dali103i_t * p_this	DALI103i モジュールへのポインタ
dali103i_instance_t * p_instance	DALI103i instance モジュールへのポインタ
uint8_t resolution	input signal の解像度 有効範囲：1～255

【戻り値】

値	説　明
DALI103I_RETURN_OK	正常終了
DALI103I_RETURN_ERR	パラメータエラー - 引数設定を見直してください。

3.5.4 R_DALI103I_DeviceNvmIsValid

【概　要】

dali103i_device_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_DALI103I_SetDeviceNvm 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_DALI103I_DeviceNvmIsValid(const dali103i_t * p_this,
                                 const dali103i_device_nvm_t * p_nvm)
```

【前提条件】

1. R_DALI103I_InitLibrary 関数が正常終了していること。
2. R_DALI103I_InitLogicalUnit 関数が正常終了していること。

【引　数】

引　数	説　明
const dali103i_t * p_this	DALI103i モジュールへのポインタ
const dali103i_device_nvm_t * p_nvm	DALI103i device NVM 変数へのポインタ 有効範囲： - short_address：0x00～0x3F, 0xFF - device_groups：0x00000000～0xFFFFFFFF - random_address：0x000000～0xFFFFFF - operating_mode：0x00, 0x80～0xFF - application_active：true, false - power_cycle_notification：enable, disable - event_priority：0x2～0x5

【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

3.5.5 R_DALI103I_InstanceNvmIsValid

【概　要】

dali103i_instance_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_DALI103I_SetInstanceNvm 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_DALI103I_InstanceNvmIsValid(const dali103i_instance_t * p_this,
                                   const dali103i_instance_nvm_t * p_nvm)
```

【前提条件】

1. R_DALI103I_InitLibrary 関数が正常終了していること。
2. R_DALI103I_InitLogicalUnit 関数が正常終了していること。
3. R_DALI103I_InitInstance 関数が正常終了していること。

【引　数】

引　数	説　明
const dali103i_instance_t * p_this	DALI103i instance モジュールへのポインタ
const dali103i_instance_nvm_t * p_nvm	DALI103i instance NVM 変数へのポインタ 有効範囲： - instance_group0：0x00～0x1F, 0xFF - instance_group1：0x00～0x1F, 0xFF - instance_group2：0x00～0x1F, 0xFF - instance_active：true, false - event_filter：0x000000～0xFFFFFF - event_scheme：0x0～0x4 - event_priority：0x2～0x5

【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

3.5.6 R_DALI103I_SetDeviceNvm

【概　要】

DALI103i モジュールに device NVM 変数値を設定します。

電源投入時に不揮発メモリに device 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_DALI103I_SetDeviceNvm(dali103i_t * p_this,
                             const dali103i_device_nvm_t * p_nvm)
```

【前提条件】

1. R_DALI103I_InitLibrary 関数が正常終了していること。
2. R_DALI103I_InitLogicalUnit 関数が正常終了していること。
3. R_DALI103I_DeviceNvmIsValid 関数で device 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
dali103i_t * p_this	DALI103i モジュールへのポインタ
const dali103i_device_nvm_t * p_nvm	DALI103i device NVM 変数へのポインタ

【戻り値】

なし

3.5.7 R_DALI103I_SetInstanceNvm

【概　要】

DALI103i instance モジュールに instance NVM 変数値を設定します。

電源投入時に不揮発メモリに instance 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_DALI103I_SetInstanceNvm(dali103i_instance_t * p_this,
                               const dali103i_instance_nvm_t * p_nvm)
```

【前提条件】

1. R_DALI103I_InitLibrary 関数が正常終了していること。
2. R_DALI103I_InitLogicalUnit 関数が正常終了していること。
3. R_DALI103I_InitInstance 関数が正常終了していること。
4. R_DALI103I_InstanceNvmIsValid 関数で instance 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
dali103i_instance_t * p_this	DALI103i instance モジュールへのポインタ
const dali103i_instance_nvm_t * p_nvm	DALI103i instance NVM 変数へのポインタ

【戻り値】

なし

3.5.8 R_DALI103I_GetDeviceNvm

【概　要】

DALI103i モジュールから device NVM 変数設定値を取得します。
不揮発メモリに最新の device NVM 変数値を保存する際に使用してください。

【書　式】

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

【前提条件】

1. R_DALI103I_InitLibrary 関数が正常終了していること。
2. R_DALI103I_InitLogicalUnit 関数が正常終了していること。
3. R_DALI103I_StartPowerCycleTimer 関数で power cycle notification タイマが開始されていること。

【引　数】

引　数	説　明
const dali103i_t * p_this	DALI103i モジュールへのポインタ
dali103i_device_nvm_t * p_nvm	DALI103i device NVM 変数へのポインタ

【戻り値】

なし

3.5.9 R_DALI103I_GetInstanceNvm

【概　要】

DALI103i instance モジュールから instance NVM 変数設定値を取得します。
不揮発メモリに最新の instance NVM 変数値を保存する際に使用してください。

【書　式】

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

【前提条件】

1. R_DALI103I_InitLibrary 関数が正常終了していること。
2. R_DALI103I_InitLogicalUnit 関数が正常終了していること。
3. R_DALI103I_InitInstance 関数が正常終了していること。
4. R_DALI103I_StartPowerCycleTimer 関数で power cycle notification タイマが開始されていること。

【引　数】

引　数	説　明
const dali103i_instance_t * p_this	DALI103i instance モジュールへのポインタ
dali103i_instance_nvm_t * p_nvm	DALI103i instance NVM 変数へのポインタ

【戻り値】

なし

3.5.10 R_DALI103I_DeviceNvmIsChanged

【概　要】

少なくとも一つの device 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 device 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_DALI103I_DeviceNvmIsChanged(dali103i_t * p_this)
```

【前提条件】

1. R_DALI103I_InitLibrary 関数が正常終了していること。
2. R_DALI103I_InitLogicalUnit 関数が正常終了していること。
3. R_DALI103I_StartPowerCycleTimer 関数で power cycle notification タイマが開始されていること。

【引　数】

引　数	説　明
dali103i_t * p_this	DALI103i 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

3.5.11 R_DALI103I_InstanceNvmIsChanged

【概　要】

少なくとも一つの instance 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 instance 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_DALI103I_InstanceNvmIsChanged(dali103i_instance_t * p_this)
```

【前提条件】

1. R_DALI103I_InitLibrary 関数が正常終了していること。
2. R_DALI103I_InitLogicalUnit 関数が正常終了していること。
3. R_DALI103I_InitInstance 関数が正常終了していること。
4. R_DALI103I_StartPowerCycleTimer 関数で power cycle notification タイマが開始されていること。

【引　数】

引　数	説　明
dali103i_instance_t * p_this	DALI103i instance 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

3.5.12 R_DALI103I_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_DALI103I_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_DALI103I_NeedsToSaveNvm(const dali103i_t * p_this)
```

【前提条件】

1. R_DALI103I_InitLibrary 関数が正常終了していること。
2. R_DALI103I_InitLogicalUnit 関数が正常終了していること。
3. R_DALI103I_InitInstance 関数が正常終了していること。
4. R_DALI103I_StartPowerCycleTimer 関数で power cycle notification タイマが開始されていること。

【引　数】

引　数	説　明
const dali103i_t * p_this	DALI103i 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

3.5.13 R_DALI103I_NotifySaveNvm

【概　要】

R_DALI103I_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_DALI103I_NotifySaveNvm(dali103i_t * p_this)
```

【前提条件】

1. R_DALI103I_InitLibrary 関数が正常終了していること。
2. R_DALI103I_InitLogicalUnit 関数が正常終了していること。
3. R_DALI103I_InitInstance 関数が正常終了していること。
4. R_DALI103I_StartPowerCycleTimer 関数で power cycle notification タイマが開始されていること。
5. R_DALI103I_NeedsToSaveNvm 関数の戻り値が true 状態であること。

【引　数】

引　数	説　明
dali103i_t * p_this	DALI103i モジュールへのポインタ

【戻り値】

なし

3.5.14 R_DALI103I_StartPowerCycleTimer

【概　要】

各変数を power on value に設定し、power cycle notification タイマをスタートします。

本関数呼び出し時、Power on value が存在する変数群は Power on value に設定されます。

ハードウェアの電源投入から本関数の呼び出しまでの時間を考慮して、総合して 1.3～5.0s になる値を設定してください。

【書　式】

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

【前提条件】

1. R_DALI103I_InitLibrary 関数が正常終了していること。
2. R_DALI103I_InitLogicalUnit 関数が正常終了していること。
3. R_DALI103I_InitInstance 関数が正常終了していること。

【引　数】

引　数	説　明
dali103i_t * p_this	DALI103i モジュールへのポインタ
uint16_t msec	power cycle notification 発生までの時間

【戻り値】

なし

3.5.15 R_DALI103I_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_DALI103I_ExecuteCommand 関数は operating mode が 0 の場合の処理関数です。

【書　式】

uint8_t R_DALI103I_GetOperatingMode(const dali103i_t * p_this)

【前提条件】

1. R_DALI103I_InitLibrary 関数が正常終了していること。
2. R_DALI103I_InitLogicalUnit 関数が正常終了していること。
3. R_DALI103I_InitInstance 関数が正常終了していること。
4. R_DALI103I_StartPowerCycleTimer 関数で power cycle notification タイマが開始されていること。

【引　数】

引　数	説　明
const dali103i_t * p_this	DALI103i モジュールへのポインタ

【戻り値】

値	説　明
0x00	IEC62386-103 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_DALI103I_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。

3.5.16 R_DALI103I_Tick1ms

【概　要】

DALI103i 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_DALI103I_Tick1ms(dali103i_t * p_this)
```

【前提条件】

1. R_DALI103I_InitLibrary 関数が正常終了していること。
2. R_DALI103I_InitLogicalUnit 関数が正常終了していること。
3. R_DALI103I_InitInstance 関数が正常終了していること。
4. R_DALI103I_StartPowerCycleTimer 関数で power cycle notification タイマが開始されていること。

【引　数】

引　数	説　明
dali103i_t * p_this	DALI103i モジュールへのポインタ

【戻り値】

なし

3.5.17 R_DALI103I_SetInputDeviceError

【概　要】

Input Device Error の詳細情報を設定します。

【書　式】

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

【前提条件】

1. R_DALI103I_InitLibrary 関数が正常終了していること。
2. R_DALI103I_InitLogicalUnit 関数が正常終了していること。
3. R_DALI103I_InitInstance 関数が正常終了していること。
4. R_DALI103I_StartPowerCycleTimer 関数で power cycle notification タイマが開始されていること。

【引　数】

引　数	説　明
dali103i_t * p_this	DALI103i モジュールへのポインタ
uint8_t error	エラー詳細情報 0x00～0xFE：詳細なエラー番号 ※エラー情報はユーザが定義 0xFF：詳細不明なエラー

【戻り値】

なし

3.5.18 R_DALI103I_ClearInputDeviceError

【概　要】
Input Device Error をクリアします。

【書　式】

void R_DALI103I_ClearInputDeviceError(dali103i_t * p_this)

【前提条件】
1. R_DALI103I_InitLibrary 関数が正常終了していること。
2. R_DALI103I_InitLogicalUnit 関数が正常終了していること。
3. R_DALI103I_InitInstance 関数が正常終了していること。
4. R_DALI103I_StartPowerCycleTimer 関数で power cycle notification タイマが開始されていること。

【引　数】

引　数	説　明
dali103i_t * p_this	DALI103i モジュールへのポインタ

【戻り値】
なし

3.5.19 R_DALI103I_SetInstanceError

【概　要】
指定した DALI103i instance モジュールに対して instance error を設定します。

【書　式】

void R_DALI103I_SetInstanceError(dali103i_instance_t * p_this)

【前提条件】
1. R_DALI103I_InitLibrary 関数が正常終了していること。
2. R_DALI103I_InitLogicalUnit 関数が正常終了していること。
3. R_DALI103I_InitInstance 関数が正常終了していること。
4. R_DALI103I_StartPowerCycleTimer 関数で power cycle notification タイマが開始されていること。

【引　数】

引　数	説　明
dali103i_instance_t * p_this	DALI103i instance モジュールへのポインタ

【戻り値】
なし

3.5.20 R_DALI103I_ClearInstanceError

【概　要】
指定した DALI103i instance モジュールの instance error をクリアします。

【書　式】

void R_DALI103I_ClearInstanceError(dali103i_instance_t * p_this)

【前提条件】
1. R_DALI103I_InitLibrary 関数が正常終了していること。
2. R_DALI103I_InitLogicalUnit 関数が正常終了していること。
3. R_DALI103I_InitInstance 関数が正常終了していること。
4. R_DALI103I_StartPowerCycleTimer 関数で power cycle notification タイマが開始されていること。

【引　数】

引　数	説　明
dali103i_instance_t * p_this	DALI103i instance モジュールへのポインタ

【戻り値】
なし

3.5.21 R_DALI103I_InstanceIsActive

【概　要】

指定した DALI103i instance モジュールが Active かどうかを取得します。
本関数の戻り値が false のとき、input notification イベントを送信することはできません。

【書　式】

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

【前提条件】

1. R_DALI103I_InitLibrary 関数が正常終了していること。
2. R_DALI103I_InitLogicalUnit 関数が正常終了していること。
3. R_DALI103I_InitInstance 関数が正常終了していること。
4. R_DALI103I_StartPowerCycleTimer 関数で power cycle notification タイマが開始されていること。

【引　数】

引　数	説　明
const dali103i_instance_t	DALI103i instance モジュールへのポインタ

【戻り値】

値	説　明
true	instance が Active
false	instance が非 Active

3.5.22 R_DALI103I_GetPowerCycleNotification

【概　要】

Power Cycle Notification イベントを取得します。

【書　式】

dali103i_event_t R_DALI103I_GetPowerCycleNotification(dali103i_t * p_this)

【前提条件】

1. R_DALI103I_InitLibrary 関数が正常終了していること。
2. R_DALI103I_InitLogicalUnit 関数が正常終了していること。
3. R_DALI103I_InitInstance 関数が正常終了していること。
4. R_DALI103I_StartPowerCycleTimer 関数で power cycle notification タイマが開始されていること。

【引　数】

引　数	説　明
dali103i_t * p_this	DALI103i モジュールへのポインタ

【戻り値】

値	説　明
bool is_exist	イベントの有無
dali103i_forward_frame_t frame	is_exist=true のとき、power cycle notification イベントを格納

3.5.23 R_DALI103I_SetInputSignal

【概　要】

指定した DALI103i instance モジュールに対して Input Signal を設定します。

【書　式】

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

【前提条件】

1. R_DALI103I_InitLibrary 関数が正常終了していること。
2. R_DALI103I_InitLogicalUnit 関数が正常終了していること。
3. R_DALI103I_InitInstance 関数が正常終了していること。
4. R_DALI103I_StartPowerCycleTimer 関数で power cycle notification タイマが開始されていること。

【引　数】

引　数	説　明
dali103i_instance_t * p_this	DALI103i instance モジュールへのポインタ
const uint8_t * p_signal	input signal 配列へのポインタ

【戻り値】

なし

(1) p_signal 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。
要件として以下を満たしてください。
・配列の要素数は instance の resolution/8 を最も近い整数に切り上げた数にしてください。
・配列に対する input signal 値の格納方法はリトルエンディアン、かつ、LSB 詰めにしてください。

配列の設定例は以下です。

例１） resolution = 3 で input signal 値に 0x05 を設定する場合

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

例２） resolution = 10 で input signal 値に 0x145 を設定する場合

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

例３） resolution = 100 で input signal 値に 0x123456789ABCDEF を設定する場合

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

3.5.24 R_DALI103I_GetInstanceEventFilter

【概　要】

指定した DALI103i instance モジュールの event filter を通知します。

【書　式】

uint32_t R_DALI103I_GetInstanceEventFilter(const dali103i_instance_t * p_this)

【前提条件】

1. R_DALI103I_InitLibrary 関数が正常終了していること。
2. R_DALI103I_InitLogicalUnit 関数が正常終了していること。
3. R_DALI103I_InitInstance 関数が正常終了していること。
4. R_DALI103I_StartPowerCycleTimer 関数で power cycle notification タイマが開始されていること。

【引　数】

引　数	説　明
const dali103i_instance_t * p_this	DALI103i instance モジュールへのポインタ

【戻り値】

値	説　明
uint32_t	instance event filter 設定値

3.5.25 R_DALI103I_GenerateInputNotification

【概　要】

指定した DALI103i instance モジュールの input notification イベントを発生させます。

Instance Type 0 の Instance である dali103i_instance_t 型 instance では標準で規定された input notification イベントは存在していません。そのためユーザ定義のイベントを発生させるために使用します。発生させたイベントを送信するか否かは event filter 設定に影響しますので、R_DALI103I_GetInstanceEventFilter 関数と組み合わせて使用してください。

【書　式】

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

【前提条件】

1. R_DALI103I_InitLibrary 関数が正常終了していること。
2. R_DALI103I_InitLogicalUnit 関数が正常終了していること。
3. R_DALI103I_InitInstance 関数が正常終了していること。
4. R_DALI103I_StartPowerCycleTimer 関数で power cycle notification タイマが開始されていること。

【引　数】

引　数	説　明
const dali103i_t * p_this	DALI103i モジュールへのポインタ
const dali103i_instance_t * p_instance	DALI103i instance モジュールへのポインタ
uint16_t event_info	event information 情報 (下位 10bit)

【戻り値】

値	説　明
bool is_exist	イベントの有無
dali103i_forward_frame_t frame	is_exist=true のとき、input notification イベントを格納

3.5.26 R_DALI103I_GetTestFrame

【概　要】

Test Frame を取得します。

【書　式】

dali103i_test_frame_t R_DALI103I_GetTestFrame(dali103i_t * p_this)

【前提条件】

1. R_DALI103I_InitLibrary 関数が正常終了していること。
2. R_DALI103I_InitLogicalUnit 関数が正常終了していること。
3. R_DALI103I_InitInstance 関数が正常終了していること。
4. R_DALI103I_StartPowerCycleTimer 関数で power cycle notification タイマが開始されていること。

【引　数】

引　数	説　明
dali103i_t * p_this	DALI103i モジュールへのポインタ

【戻り値】

値	説　明
uint8_t num	test frame の数
dali103i_forward_frame_t frame	test frame の配列

3.5.27 R_DALI103I_IdentificationIsActive

【概　要】

Identification が active 状態か否かを取得します。Identification とは、Input Device 設置者が特定の Input Device 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、ユーザアプリケーションは「LED を点滅させる」「音を出す」など視覚的または聴覚的手段、またはスマートデバイスやツールへのワイヤレス送信などにより識別可能な操作で照明装置を制御する必要があります。

なお、Identification の処理はオプション機能となります。

【書　式】

bool R_DALI103I_IdentificationIsActive(const dali103i_t * p_this)

【前提条件】

1. R_DALI103I_InitLibrary 関数が正常終了していること。
2. R_DALI103I_InitLogicalUnit 関数が正常終了していること。
3. R_DALI103I_InitInstance 関数が正常終了していること。
4. R_DALI103I_StartPowerCycleTimer 関数で power cycle notification タイマが開始されていること。

【引　数】

引　数	説　明
const dali103i_t * p_this	DALI103i 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

3.5.28 R_DALI103I_QuiescentModeIsActive

【概 要】

QuiescentMode が active 状態か否かを取得します。
本関数の戻り値が true である間、ユーザアプリケーションは Forward Frame を送信しないでください。

【書 式】

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

【前提条件】

1. R_DALI103I_InitLibrary 関数が正常終了していること。
2. R_DALI103I_InitLogicalUnit 関数が正常終了していること。
3. R_DALI103I_InitInstance 関数が正常終了していること。
4. R_DALI103I_StartPowerCycleTimer 関数で power cycle notification タイマが開始されていること。

【引 数】

引 数	説 明
const dali103i_t * p_this	DALI103i モジュールへのポインタ

【戻り値】

値	説 明
true	QuiescentMode が active
false	QuiescentMode が非 active

3.5.29 R_DALI103I_CreateCommand

【概　要】

DALI 通信ドライバを通して受信した 24bit Forward Frame を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】

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

【前提条件】

1. R_DALI103I_InitLibrary 関数が正常終了していること。
2. R_DALI103I_InitLogicalUnit 関数が正常終了していること。
3. R_DALI103I_InitInstance 関数が正常終了していること。
4. R_DALI103I_StartPowerCycleTimer 関数で power cycle notification タイマが開始されていること。

【引　数】

引　数	説　明
const dali103i_t * p_this	DALI103i モジュールへのポインタ
uint32_t forward	受信した 24bit 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
uint8_t address_byte	address byte データ
uint8_t instance_byte	instans byte データ
uint8_t opcode_byte	opcode byte データ
bool is_received_twice	twice 状態
dali103i_daddr_t device_addressing	device addressing
dali103i_iaddr_t instance_addressing	instance addressing
dali103i_cmd_type_t type	コマンドタイプ
dali103i_target_t target	コマンド実行ターゲット

3.5.30 R_DALI103I_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_DALI103I_ExecuteCommand(dali103i_t * p_this,
                                  const dali103i_cmd_t * p_cmd)
```

【前提条件】

1. R_DALI103I_InitLibrary 関数が正常終了していること。
2. R_DALI103I_InitLogicalUnit 関数が正常終了していること。
3. R_DALI103I_InitInstance 関数が正常終了していること。
4. R_DALI103I_StartPowerCycleTimer 関数で power cycle notification タイマが開始されていること。
5. R_DALI103I_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
dali103i_t * p_this	DALI103i モジュールへのポインタ
const dali103i_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
DALI103I_NO_ANSWER	backward frame なし
DALI103I_BACKWARD_IS_CORRUPTED	corrupted backward frame

3.5.31 R_DALI103I_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】

```
uint16_t R_DALI103I_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 Input Device ライブラリ ユーザーズマニュアル 基本(103)編

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	Nov.,8.22	ー	初版発行

RL78ファミリ DALI-2 Input Deviceライブラリ
ユーザーズマニュアル 基本(103)編

発行年月日　2022年11月8日　　Rev.1.00

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

R01US0593JJ0100