User's Manual

22

RL78 ファミリ

DALI-2 Input Device ライブラリ

ユーザーズマニュアル Push Button(301)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.1.00　2023.10

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 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. DALI301 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提供している DALI103i ライブラリ専用の拡張ライブラリです。

DALI103i ライブラリの仕様は DALI103i ライブラリ ユーザーズマニュアルを参照してください。

本ライブラリでは、IEC62386-301ed1.0 (以降、DALI301)にて規定された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。Instance Type 1 (Push Button)の Instance を持つ Input Device を実装したい場合に使用してください。

表 1-1 処理範囲

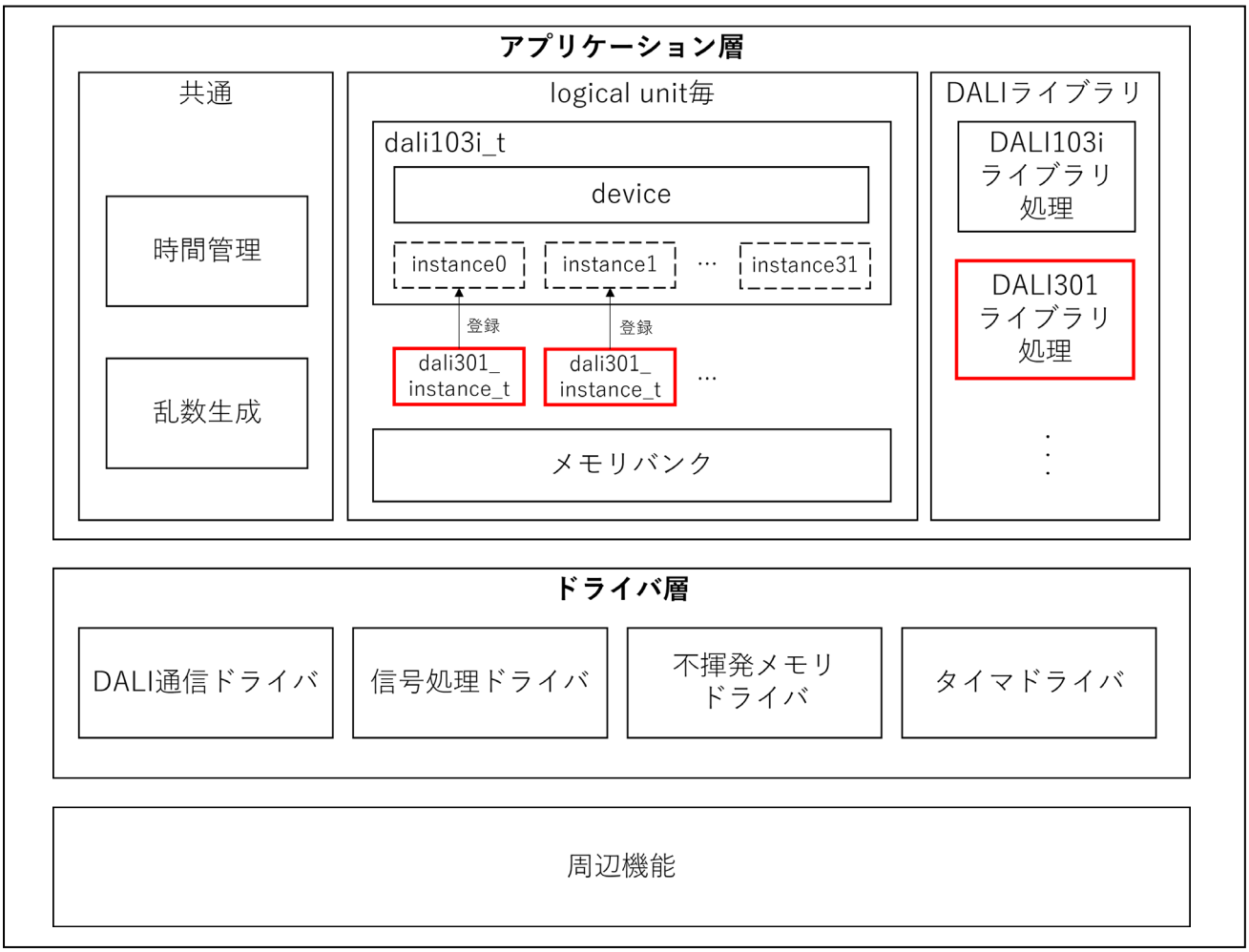
ユーザ作成処理	ライブラリ処理
・Push Button 入力制御 ・Push Button 異常検出	・受信 24bit Forward Frame 処理 ・送信 Backward Frame 発行 ・送信 Event Message Frame 発行 ・タイミング制御 ・DALI 変数操作

本ライブラリでは、DALI103i ライブラリを使って定義した logical unit に登録可能な Instance Type 1 の instance を提供します。

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。
赤線で囲んだ部分が本ライブラリとなります。本ライブラリは DALI103i ライブラリに拡張することを前提としています。

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 1-2 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
IEC62386-301 Edition 1.0	Renesas CC-RL V1.11.00	r_dali_301_cc_gen2_v1_00.lib
	IAR C/C++ Compiler for Renesas RL78 V4.21.4	r_dali_301_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 1-3 ファイル一覧

ファイル名	説明
r_dali_301_cc_gen2_v1_00.lib r_dali_301_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_dali301_api.h	ライブラリヘッダファイル
r_dali301_ivar.h	instance変数モジュールの定義ヘッダファイル
r_dali301_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 1-4 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 1-5 ライブラリリソース(可変)に記載します。

表 1-4 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	2,423 [byte]
		RAMサイズ	0 [byte]
	最大スタックサイズ		42 [byte] (R_DALI301_SetInputSignal関数)
IAR	ライブラリリソース	ROMサイズ	2,696 [byte]
		RAMサイズ	0 [byte]
	最大スタックサイズ		56 [byte] (R_DALI301_GetInputNotification関数)

表 1-5 ライブラリリソース(可変)

コンパイラ	項目	RAMサイズ
CC-RL	dali301_instance_t	122 [byte / instance]
IAR	dali301_instance_t	122 [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 1-6 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. dali301_instance_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。

なお、DALI103i ライブラリでの要件に加えて必要となる要件のみ記載します。

2.1 ハードウェア要件

2.1.1 Push Button

instance type 1 の instance は信号処理装置として Push Button を適用する必要があります。

2.1.2 異常検出機構

instance type 1 の instance 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。なお、異常内容はメーカ依存になっており、最大 4 種類の異常検出まで認められています。

この機能はオプションです。

2.2 ソフトウェア要件

2.2.1 DALI301 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。本ライブラリでは instance type 1 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(dali301_instance_t)を提供します。dali301_instance_t 型変数のことを DALI301 instance モジュールと呼びます。

必要な数の DALI301 instance モジュールを定義し、DALI103i モジュールに登録してください。

2.2.2 Push Button ドライバ

信号処理装置となる Push Button の押下状態を取得するドライバを実装してください。

なお、Push Button の ON/OFF 動作時に bouncing(接点の状態が変化する際、一時的に ON/OFF を繰り返す不安定な現象)が発生することがあります。ハードウェアで bouncing を抑制する回路になっていない場合、ソフトウェアで bouncing 対策処理を実装してください。

2.2.3 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。

この機能はオプションです。

- instance type 1 の instance に関わる異常発生
  R_DALI301_AddInstanceByteError 関数
- instance type 1 の instance に関わる異常解消
  R_DALI301_RemoveInstanceByteError 関数

3. DALI301 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
dali301_instance_t	DALI301 instance モジュール型

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 event information一覧

マクロ名	マクロ値	説明
DALI301_EVENT_BUTTON_RELEASED	0x00000000	button released イベント
DALI301_EVENT_BUTTON_PRESSED	0x00000001	button pressed イベント
DALI301_EVENT_SHORT_PRESS	0x00000002	short press イベント
DALI301_EVENT_DOUBLE_PRESS	0x00000005	double press イベント
DALI301_EVENT_LONG_PRESS_START	0x00000009	long press start イベント
DALI301_EVENT_LONG_PRESS_REPEAT	0x0000000B	long press repeat イベント
DALI301_EVENT_LONG_PRESS_STOP	0x0000000C	long press stop イベント
DALI301_EVENT_BUTTON_FREE	0x0000000E	button free イベント
DALI301_EVENT_BUTTON_STUCK	0x0000000F	button stuck イベント

表 3-3 instance error一覧

マクロ名	マクロ値	説明
DALI301_ERRBYTE _MANUFACTURER_SPECIFIC_ERROR_1	0x10	manufacturer specific error 1 ビット
DALI301_ERRBYTE _MANUFACTURER_SPECIFIC_ERROR_2	0x20	manufacturer specific error 2 ビット
DALI301_ERRBYTE _MANUFACTURER_SPECIFIC_ERROR_3	0x40	manufacturer specific error 3 ビット
DALI301_ERRBYTE _MANUFACTURER_SPECIFIC_ERROR_4	0x80	manufacturer specific error 4 ビット

表 3-4 input signal (dali301_input_signal_t)一覧

マクロ名	マクロ値	説明
DALI301_INPUT_SIGNAL_PRESSED	0	Push button 押下状態の input signal 値
DALI301_INPUT_SIGNAL_RELEASED	1	Push button 非押下状態の input signal 値

本ライブラリで提供する戻り値を以下に記載します。

表 3-5 戻り値 (dali301_return_t) 一覧

定義	戻り値	説明
DALI301_RETURN_OK	0	正常終了
DALI301_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。

instance NVM 型構造体 (dali301_instance_nvm_t) の定義

```
typedef struct
{
    dali103i_instance_nvm_t base;
    dali301_ivar_nvm_t add;
} dali301_instance_nvm_t;
```

instance default 型構造体 (dali301_instance_default_t) の定義

```
typedef struct
{
    dali301_input_signal_t signal;
    uint8_t t_short_min;
    uint8_t t_double_min;
} dali301_instance_default_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。

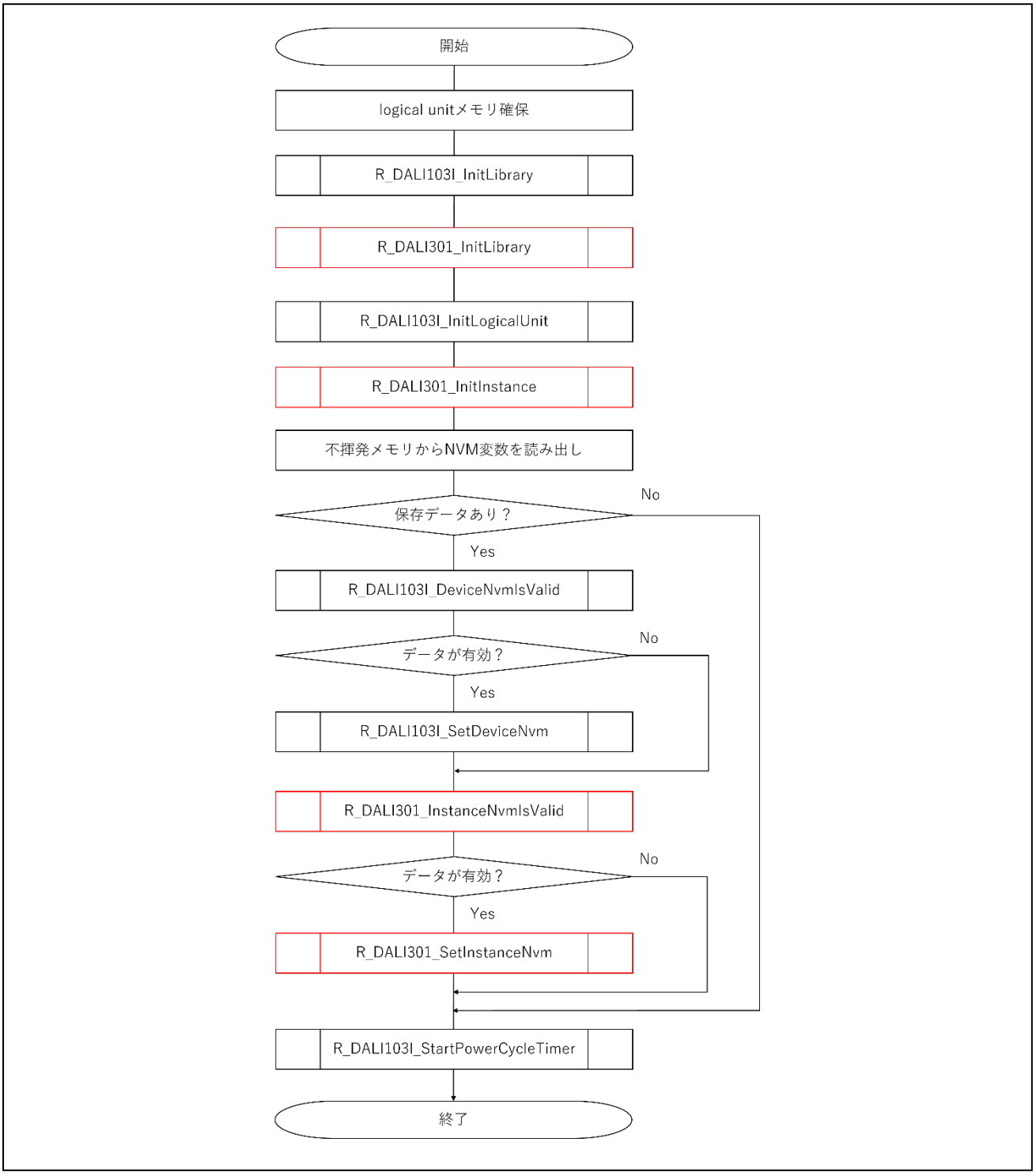
表 3-6 API関数一覧

関数名	説明
R_DALI301_InitLibrary	DALI301 ライブラリの初期化
R_DALI301_InitInstance	instance の初期化
R_DALI301_InstanceNvmIsValid	instance NVM 変数値の有効範囲内チェック
R_DALI301_SetInstanceNvm	instance NVM 変数値の設定
R_DALI301_GetInstanceNvm	instance NVM 変数値の取得
R_DALI301_InstanceNvmIsChanged	instance NVM 変数値変更チェック
R_DALI301_InstanceIsActive	instanceActive の状態確認
R_DALI301_SetInputSignal	input signal の設定
R_DALI301_GetInputNotification	input notification イベントの取得
R_DALI301_AddInstanceErrorByte	instance error の追加
R_DALI301_RemoveInstanceErrorByte	instance error の除去
R_DALI301_GetInstanceErrorByte	instanceErrorByte 変数値の取得
R_DALI301_GetLibraryVersion	ライブラリバージョン取得

3.4 概略フローチャート

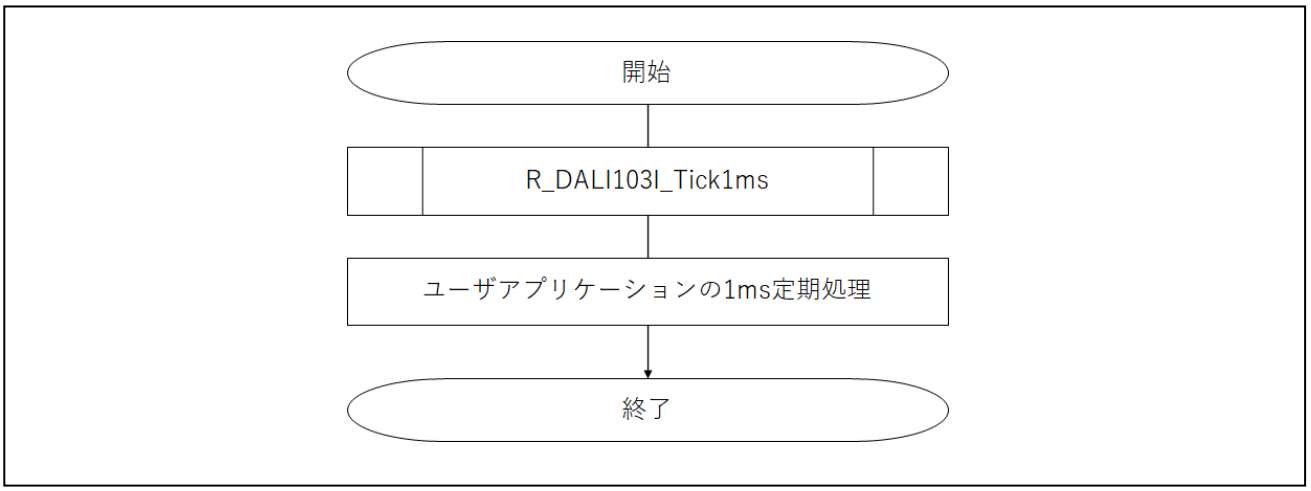
3.4.1 初期化時

初期化時のフローを記載します。赤い枠で囲んだ関数が本ライブラリ提供関数です。



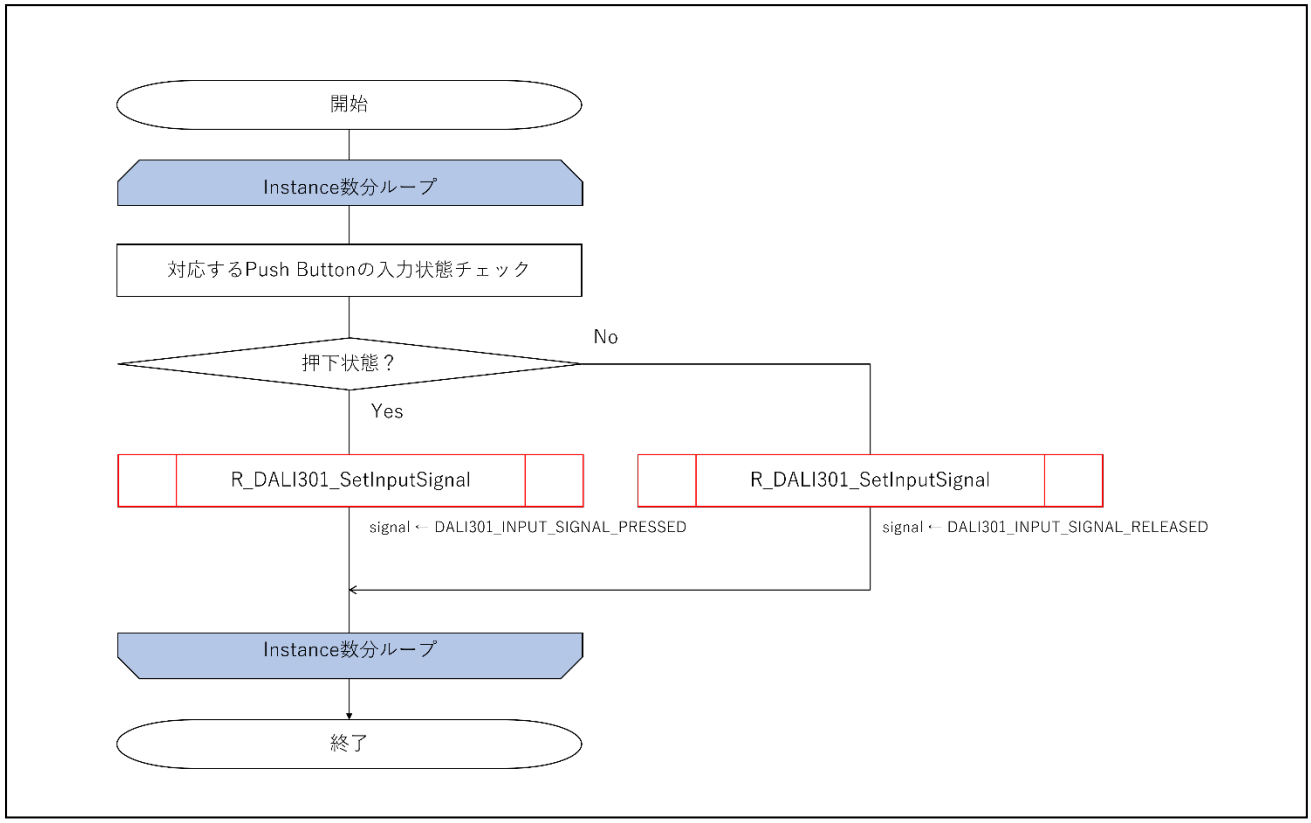
3.4.2 1ms 定期処理

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。



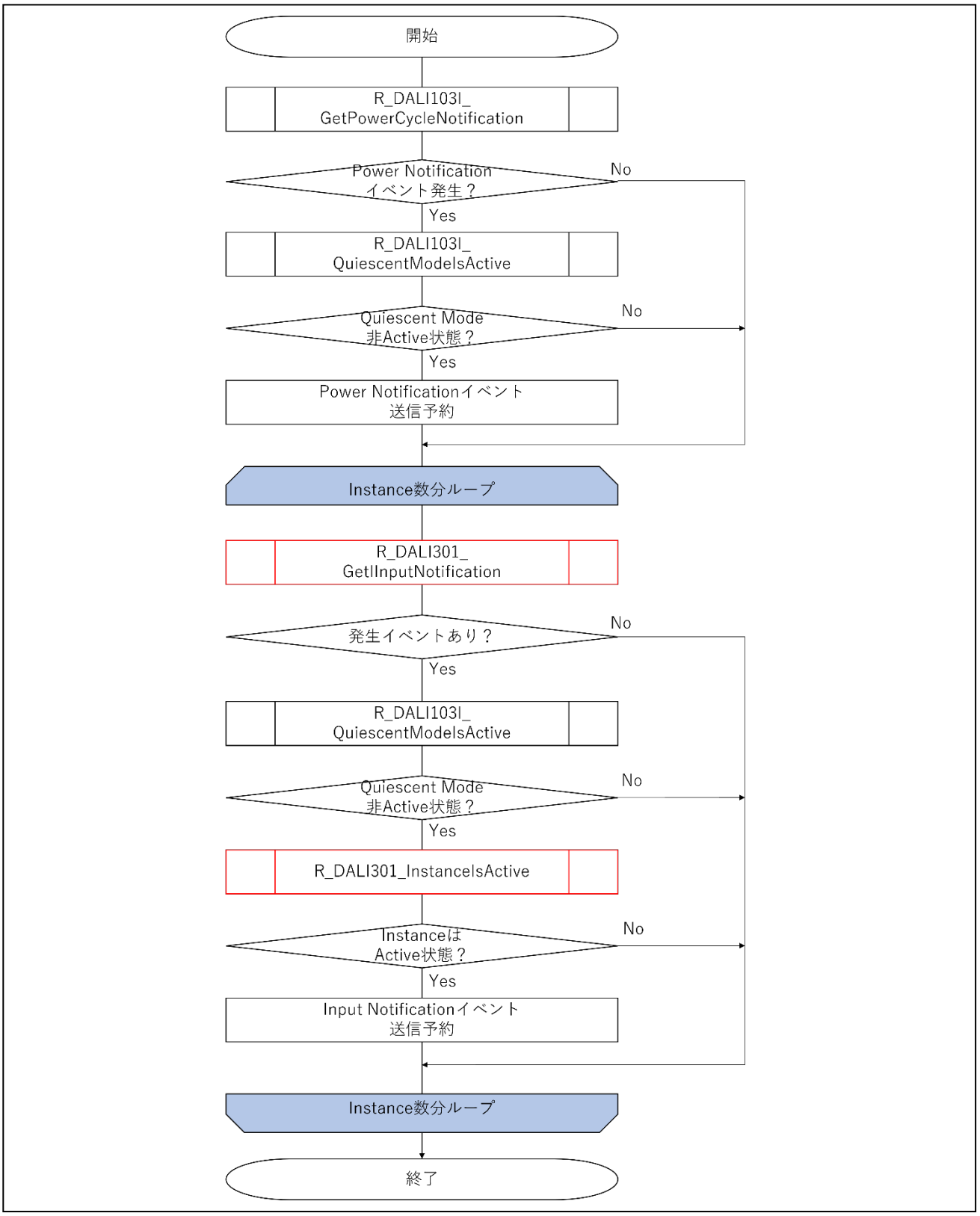
3.4.3 Input Signal 更新処理

Input Signal 更新のフローを記載します。赤い枠で囲んだ関数が本ライブラリ提供関数です。



3.4.4 Event Message 処理

Event Message 処理のフローを記載します。赤い枠で囲んだ関数が本ライブラリ提供関数です。

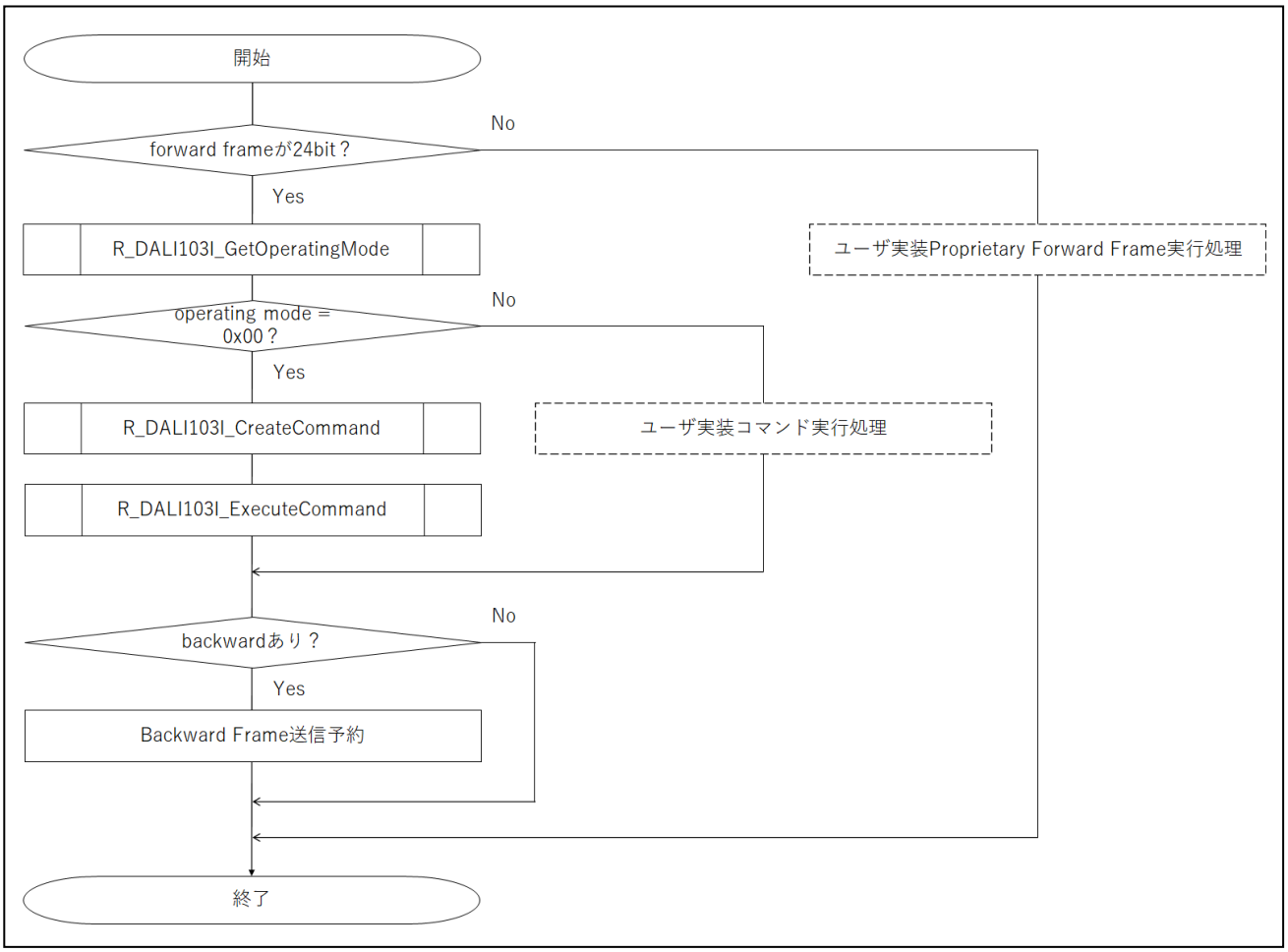


3.4.5 Forward Frame 受信時

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。

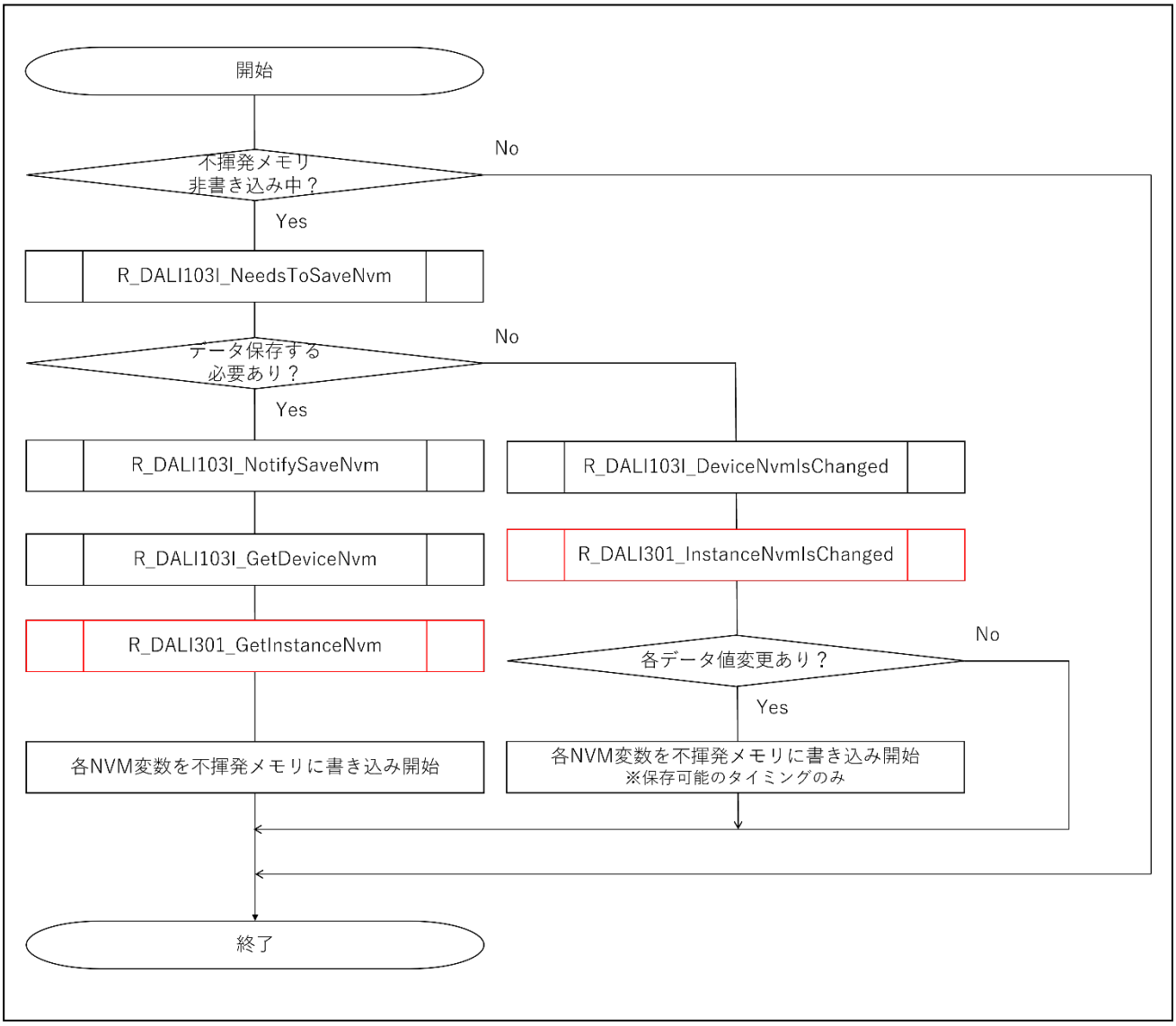
また、0 以外の operating mode はオプション機能です。独自モードが必要な場合実装の上、R_DALI103I_InitLogicalUnit 関数にてモード番号を登録してください。



3.4.6 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。

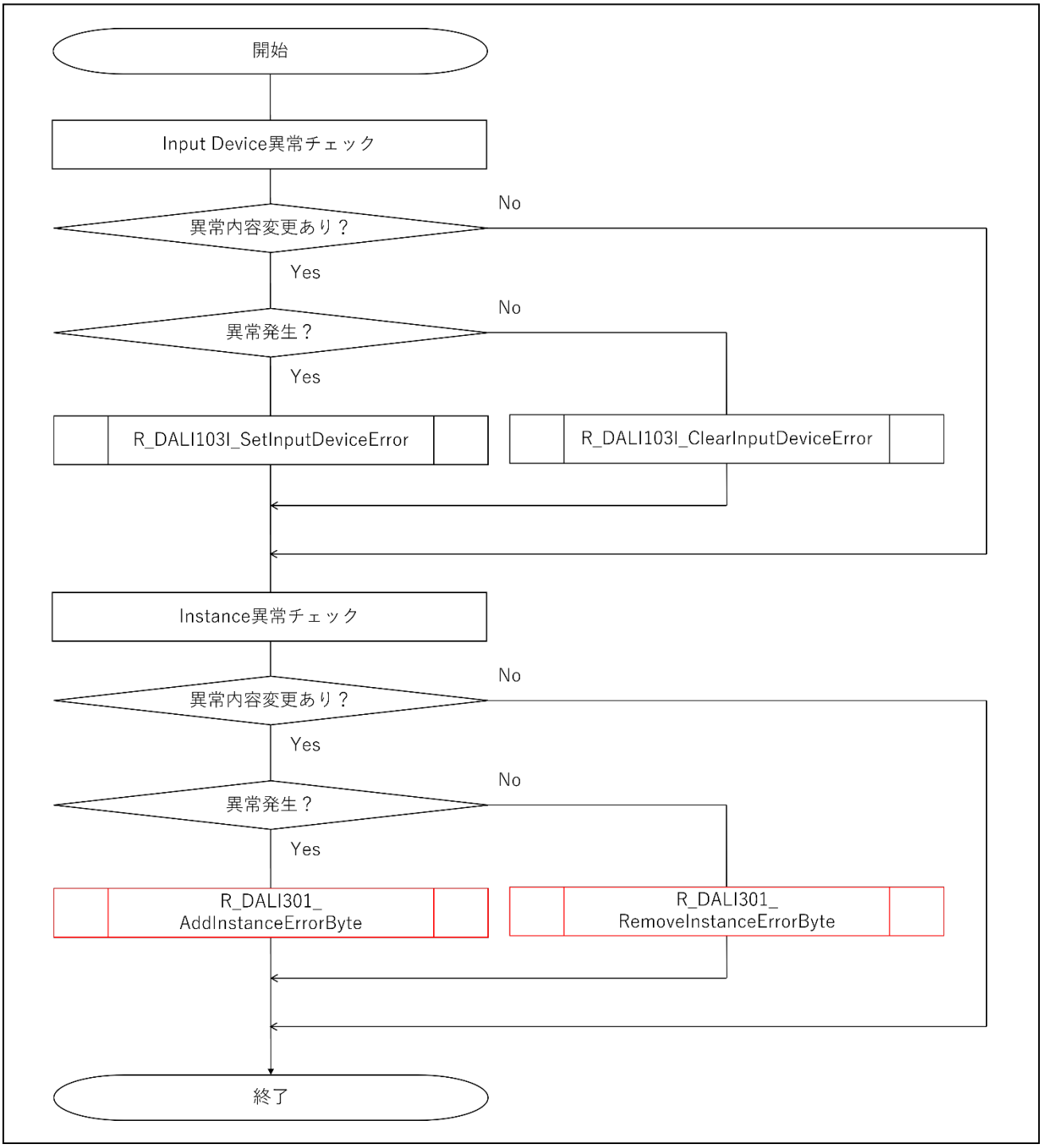
SAVE PERSISTENT VARIABLES コマンド受信から 300ms 以内に不揮発メモリに保存完了することが規定されています。また、SAVE PERSISTANT VARIABLES コマンドを受信していなくとも NVM 変数値に変更があった場合は、30 秒以内に保存することが規定されています。それぞれ規定時間内に保存完了するように定期的にチェックを行い、処理を行ってください。赤い枠で囲んだ関数が本ライブラリ提供関数です。



3.4.7 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。赤い枠で囲んだ関数が本ライブラリ提供関数です。



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_DALI301_InitLibrary

【概　要】
DALI301 ライブラリの初期化を行います。

【書　式】

dali301_return_t R_DALI301_InitLibrary(void)

【前提条件】
1. R_DALI103I_InitLibrary 関数が正常終了していること。

【引　数】
なし

【戻り値】

値	説　明
DALI301_RETURN_OK	正常終了
DALI301_RETURN_ERR	パラメータエラー

3.5.2 R_DALI301_InitInstance

【概　要】

DALI301 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。dali301_instance_t 型は InstanceType 1 の instance を提供します。

【書　式】

```
dali301_return_t R_DALI301_InitInstance(dali103i_t * p_this,
                                        dali301_instance_t * p_instance,
                                        const dali301_instance_default_t * p_default_value)
```

【前提条件】

1. R_DALI103I_InitLibrary 関数が正常終了していること。
2. R_DALI301_InitLibrary 関数が正常終了していること。
3. 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
dali301_instance_t * p_instance	DALI301 instance モジュールへのポインタ
const dali301_instance_default_t * p_default_value	fuctory burn-in の default 値 有効範囲 - signal： DALI301_INPUT_SIGNAL_PRESSED, DALI301_INPUT_SIGNAL_RELEASED - t_short_min：10～255 - t_double_min：10～100

【戻り値】

値	説　明
DALI301_RETURN_OK	正常終了
DALI301_RETURN_ERR	パラメータエラー - 引数設定を見直してください。

3.5.3 R_DALI301_InstanceNvmIsValid

【概　要】

dali301_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_DALI301_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_DALI301_InstanceNvmIsValid(const dali301_instance_t * p_this,
                                  const dali301_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_DALI301_InitLibrary 関数が正常終了していること。
3. R_DALI103I_InitLogicalUnit 関数が正常終了していること。
4. R_DALI301_InitInstance 関数が正常終了していること。

【引　数】

引　数	説　明
const dali301_instance_t * p_this	DALI301 instance モジュールへのポインタ
const dali301_instance_nvm_t * p_nvm	DALI301 instance NVM 変数へのポインタ 有効範囲： - base.instance_group0：0x00～0x1F, 0xFF - base.instance_group1：0x00～0x1F, 0xFF - base.instance_group2：0x00～0x1F, 0xFF - base.instance_active：true, false - base.event_filter：0x00000000～0x000000FF - base.event_scheme：0x00～0x04 - base.event_priority：0x02～0x05 - add.t_short：t_short_min～0xFF - add.t_double：0x00, t_double_min～0x64 - add.t_repeat：0x05～0x64 - add.t_stuck：0x05～0xFF

【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

3.5.4 R_DALI301_SetInstanceNvm

【概　要】

DALI301 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_DALI301_SetInstanceNvm(dali301_instance_t * p_this,
                              const dali301_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_DALI301_InitLibrary 関数が正常終了していること。
3. R_DALI103I_InitLogicalUnit 関数が正常終了していること。
4. R_DALI301_InitInstance 関数が正常終了していること。
5. R_DALI301_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
dali301_instance_t * p_this	DALI301 instance モジュールへのポインタ
const dali301_instance_nvm_t * p_nvm	DALI301 instance NVM 変数へのポインタ

【戻り値】

なし

3.5.5 R_DALI301_GetInstanceNvm

【概　要】

DALI301 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_DALI301_GetInstanceNvm(const dali301_instance_t * p_this,
                              dali301_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_DALI301_InitLibrary 関数が正常終了していること。
3. R_DALI103I_InitLogicalUnit 関数が正常終了していること。
4. R_DALI301_InitInstance 関数が正常終了していること。
5. 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 dali301_instance_t * p_this	DALI301 instance モジュールへのポインタ
dali301_instance_nvm_t * p_nvm	DALI301 instance NVM 変数へのポインタ

【戻り値】

なし

3.5.6 R_DALI301_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_DALI301_InstanceNvmIsChanged(dali301_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_DALI301_InitLibrary 関数が正常終了していること。
3. R_DALI103I_InitLogicalUnit 関数が正常終了していること。
4. R_DALI301_InitInstance 関数が正常終了していること。
5. 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
dali301_instance_t * p_this	DALI301 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

3.5.7 R_DALI301_InstanceIsActive

【概　要】

指定した DALI301 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_DALI301_InstanceIsActive(const dali301_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_DALI301_InitLibrary 関数が正常終了していること。
3. R_DALI103I_InitLogicalUnit 関数が正常終了していること。
4. R_DALI301_InitInstance 関数が正常終了していること。
5. 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 dali301_instance_t	DALI301 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

3.5.8 R_DALI301_SetInputSignal

【概　要】

指定した DALI301 instance モジュールに input signal を設定します。
instance に対応した Push Button の押下状態を随時設定してください。

【書　式】

```
void R_DALI301_SetInputSignal(dali301_instance_t * p_this,
                              dali301_input_signal_t signal)
```

【前提条件】

1. R_DALI103I_InitLibrary 関数が正常終了していること。
2. R_DALI301_InitLibrary 関数が正常終了していること。
3. R_DALI103I_InitLogicalUnit 関数が正常終了していること。
4. R_DALI301_InitInstance 関数が正常終了していること。
5. 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
dali301_instance_t * p_this	DALI301 モジュールへのポインタ
dali301_input_signal_t signal	input signal 設定値

【戻り値】

なし

3.5.9 R_DALI301_GetInputNotification

【概　要】

指定した DALI301 instance モジュールの input notification イベントを取得します。
本関数にて取得した input notification イベントを priority 設定に従った時間で送信してください。

【書　式】

```
dali103i_event_t R_DALI301_GetInputNotification(dali103i_t * p_this,
                                                dali301_instance_t * p_instance)
```

【前提条件】

1. R_DALI103I_InitLibrary 関数が正常終了していること。
2. R_DALI301_InitLibrary 関数が正常終了していること。
3. R_DALI103I_InitLogicalUnit 関数が正常終了していること。
4. R_DALI301_InitInstance 関数が正常終了していること。
5. 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
dali301_instance_t * p_instance	DALI301 instance モジュールへのポインタ

【戻り値】

メンバ	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

3.5.10 R_DALI301_AddInstanceErrorByte

【概　要】

指定した DALI301 instance モジュールに対し指定したエラーを instanceErrorByte に追加設定します。
特定のエラーが発生した場合に、対応したマクロを使用して呼び出してください。

【書　式】

```
void R_DALI301_AddInstanceErrorByte(dali301_instance_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_DALI301_InitLibrary 関数が正常終了していること。
3. R_DALI103I_InitLogicalUnit 関数が正常終了していること。
4. R_DALI301_InitInstance 関数が正常終了していること。
5. 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
dali301_instance_t * p_this	DALI301 instance モジュールへのポインタ
uint8_t error	instance error の追加設定値 有効範囲 - DALI301_ERRBYTE_MANUFACTURER_SPECIFIC_ERROR_1 - DALI301_ERRBYTE_MANUFACTURER_SPECIFIC_ERROR_2 - DALI301_ERRBYTE_MANUFACTURER_SPECIFIC_ERROR_3 - DALI301_ERRBYTE_MANUFACTURER_SPECIFIC_ERROR_4 ※上記マクロを OR 指定することで複数指定も可能

【戻り値】

なし

3.5.11 R_DALI301_RemoveInstanceErrorByte

【概　要】

指定した DALI301 instance モジュールに対し指定したエラーを instanceErrorByte から除去設定します。
特定のエラーが解消した場合に、対応したマクロを使用して呼び出してください。

【書　式】

```
void R_DALI301_RemoveInstanceErrorByte(dali301_instance_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_DALI301_InitLibrary 関数が正常終了していること。
3. R_DALI103I_InitLogicalUnit 関数が正常終了していること。
4. R_DALI301_InitInstance 関数が正常終了していること。
5. 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
dali301_instance_t * p_this	DALI301 instance モジュールへのポインタ
uint8_t error	instance error の除去設定値 有効範囲 - DALI301_ERRBYTE_MANUFACTURER_SPECIFIC_ERROR_1 - DALI301_ERRBYTE_MANUFACTURER_SPECIFIC_ERROR_2 - DALI301_ERRBYTE_MANUFACTURER_SPECIFIC_ERROR_3 - DALI301_ERRBYTE_MANUFACTURER_SPECIFIC_ERROR_4 ※上記マクロを OR 指定することで複数指定も可能

【戻り値】

なし

3.5.12 R_DALI301_GetInstanceErrorByte

【概　要】

instanceErrorByte 設定値を取得します。

【書　式】

uint8_t R_DALI301_GetInstanceErrorByte(const dali301_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_DALI301_InitLibrary 関数が正常終了していること。
3. R_DALI103I_InitLogicalUnit 関数が正常終了していること。
4. R_DALI301_InitInstance 関数が正常終了していること。
5. 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 dali301_instance_t * p_this	DALI301 モジュールへのポインタ

【戻り値】

値	説　明
uint8_t	instanceErrorByte 設定値

3.5.13 R_DALI301_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_DALI301_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 Push Button(301)編

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	Oct.04.23	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 Push Button(301)編

発行年月日　2023年10月04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

R01US0649JJ0100