

# e<sup>2</sup> studio コード生成ツール

統合開発環境

ユーザーズマニュアル RL78 APIリファレンス編

対象デバイス

RL78 ファミリ

本資料に記載の全ての情報は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。

## 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。お客様の機器・システムの設計において、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には、お客様の責任において行ってください。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まれます。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、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、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。
5. 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。

6. 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。
7. 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。
8. 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、当社は、一切その責任を負いません。
9. 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。
10. 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いたします。
11. 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。
12. 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。

# このマニュアルの使い方

|     |  |
|-----|--|
| 対象者 | このマニュアルは、コード生成ツールの機能を理解し、それをを用いたアプリケーション・システムを開発するユーザを対象としています。  |
| 目的  | このマニュアルは、コード生成ツールの持つソフトウェア機能をユーザに理解していただき、これを使用するシステムのハードウェア、ソフトウェア開発の参照用資料として役立つことを目的としています。  |
| 構成  | このマニュアルは、大きく分けて次の内容で構成しています。<br><br><a href="#">1. 概 説</a><br><a href="#">2. 出力ファイル</a><br><a href="#">3.API 関数</a>  |
| 読み方 | このマニュアルを読むにあたっては、電気、論理回路、マイクロコンピュータに関する一般知識が必要となります。<br><br>凡例    データ表記の重み    :    左が上位桁、右が下位桁<br><br>アクティブ・ロウの表記: <u>XXX</u> (端子、信号名称に上線)<br><br>注                                :    本文中につけた注の説明<br><br>注意                              :    気をつけて読んでいただきたい内容<br><br>備考                                :    本文中の補足説明<br><br>数の表記                         :    10 進数 ... XXXX<br><br>:    16 進数 ... 0XXXXX |

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。

# 目次

|        |   |     |
|--------|---|-----|
| 1.     | 概 説   | 6   |
| 1.1    | 概 要   | 6   |
| 1.2    | 特 長   | 6   |
| 2.     | 出力ファイル  | 7   |
| 2.1    | 説 明   | 7   |
| 3.     | API 関数  | 22  |
| 3.1    | 概 要   | 22  |
| 3.2    | 関数リファレンス  | 22  |
| 3.2.1  | 共 通   | 24  |
| 3.2.2  | クロック発生回路  | 29  |
| 3.2.3  | ポート機能   | 43  |
| 3.2.4  | 高速オンチップ・オシレータ・クロック周波数補正機能                         | 46  |
| 3.2.5  | タイマ・アレイ・ユニット                                      | 52  |
| 3.2.6  | タイマ RJ  | 67  |
| 3.2.7  | タイマ RD  | 82  |
| 3.2.8  | タイマ RG  | 102 |
| 3.2.9  | タイマ RX  | 110 |
| 3.2.10 | 16 ビット・タイマ KB                                     | 118 |
| 3.2.11 | 16 ビット・タイマ KC0                                    | 131 |
| 3.2.12 | 16 ビット・タイマ KB2                                    | 138 |
| 3.2.13 | リアルタイム・クロック                                       | 162 |
| 3.2.14 | サブシステム・クロック周波数測定回路                                | 200 |
| 3.2.15 | 12 ビット・インターバル・タイマ                                 | 207 |
| 3.2.16 | 8 ビット・インターバル・タイマ                                  | 215 |
| 3.2.17 | 16 ビット・ウエイクアップ・タイマ                                | 223 |
| 3.2.18 | クロック出力／ブザー出力制御回路                                  | 230 |
| 3.2.19 | ウォッチドッグ・タイマ                                       | 236 |
| 3.2.20 | プログラマブル・ゲイン計装アンプ付き 24 ビット $\Delta\Sigma$ /D コンバータ | 241 |
| 3.2.21 | A/D コンバータ   | 254 |
| 3.2.22 | コンフィギュラブル・アンプ                                     | 270 |
| 3.2.23 | 温度センサ   | 276 |
| 3.2.24 | 24 ビット $\Delta\Sigma$ /D コンバータ                    | 283 |
| 3.2.25 | D/A コンバータ   | 296 |
| 3.2.26 | プログラマブル・ゲイン・アンプ                                   | 308 |
| 3.2.27 | コンパレータ  | 315 |
| 3.2.28 | コンパレータ / プログラマブル・ゲイン・アンプ                          | 323 |

|        |                               |     |
|--------|-------------------------------|-----|
| 3.2.29 | シリアル・アレイ・ユニット                 | 334 |
| 3.2.30 | シリアル・アレイ・ユニット4                | 374 |
| 3.2.31 | アシンクロナス・シリアル・インタフェース LIN-UART | 387 |
| 3.2.32 | シリアル・インタフェース IICA             | 406 |
| 3.2.33 | LCDコントローラ／ドライバ                | 429 |
| 3.2.34 | サウンド・ジェネレータ                   | 440 |
| 3.2.35 | DMAコントローラ                     | 446 |
| 3.2.36 | データ・トランスファ・コントローラ             | 455 |
| 3.2.37 | イベントリンクコントローラ                 | 463 |
| 3.2.38 | 割り込み機能                        | 467 |
| 3.2.39 | キー割り込み機能                      | 483 |
| 3.2.40 | 電圧検出回路                        | 489 |
| 3.2.41 | バッテリー・バックアップ機能                | 510 |
| 3.2.42 | 発振停止検出回路                      | 516 |
| 3.2.43 | SPIインタフェース                    | 524 |
| 3.2.44 | オペアンプ                         | 533 |
| 3.2.45 | データ演算回路                       | 542 |
| 3.2.46 | 32ビット積和演算回路                   | 552 |
| 3.2.47 | 12ビットA/Dコンバータ                 | 562 |
| 3.2.48 | 12ビットD/Aコンバータ                 | 575 |
| 3.2.49 | オペアンプ&アナログスイッチ                | 582 |
| 3.2.50 | ボルテージ・リファレンス                  | 591 |
| 3.2.51 | サンプリング出カタイマ／ディテクタ             | 596 |
| 3.2.52 | 外部サンプリング                      | 605 |
| 3.2.53 | シリアル・インタフェース UARTMG           | 612 |
| 3.2.54 | アンプ・ユニット                      | 627 |
| 3.2.55 | データ・フラッシュ・ライブラリ               | 636 |

## 1. 概 説

コード生成ツールは、デバイス・ドライバを自動生成するソフトウェア・ツールです。このドキュメントでは、コード生成ツールが出力するファイルおよび API 関数について説明します。

### 1.1 概 要

コード生成ツールは、GUI ベースで各種情報を設定することにより、マイクロコントローラの端子配置状況（端子配置表、端子配置図）／マイクロコントローラが提供している周辺機能（クロック発生回路、ポート機能など）を制御するうえで必要なソース・コード（デバイス・ドライバ・プログラム：C ソース・ファイル、ヘッダ・ファイル）を出力することができます。

### 1.2 特 長

以下に、コード生成ツールの特長を示します。

- コード生成機能  
コード生成では、GUI ベースで設定した情報に応じたデバイス・ドライバ・プログラムを出力するだけでなく、main 関数を含んだサンプル・プログラム、リンク・ディレクティブ・ファイルなどといったビルド環境一式を出力することもできます。
- レポート機能  
端子配置／コード生成を用いて設定した情報を各種形式のファイルで出力し、設計資料として利用することができます。
- リネーム機能  
コード生成が出力するファイル名、およびソース・コードに含まれている API 関数の関数名については、デフォルトの名前が付与されますが、ユーザ独自の名前に変更することもできます。
- ユーザ・コード保護機能  
各 API 関数には、ユーザが独自にコードを追加できるように、ユーザ・コード記述用のコメントが設けられています。

[ユーザ・コード記述用のコメント]

```
/* Start user code. Do not edit comment generated here */
```

```
/* End user code. Do not edit comment generated here */
```

このコメント内にコードを記述すると、再度コード生成した場合でもユーザが記述したコードは保護されます。

## 2. 出力ファイル

本章では、コード生成ツールが出力するファイルについて説明します。

### 2.1 説明

以下に、コード生成ツールが出力するファイルの一覧を示します。

表 2.1 出力ファイル

| 周辺機能                                  | ファイル名                                   | API 関数名  | 出力 (*1)                              |
|---------------------------------------|---|--|--------------------------------------|
| 共通                                    | r_main.c または<br>r_cg_main.c             | main<br>R_MAIN_UserInit  | ○<br>○                               |
|                                       | r_systeminit.c または<br>r_cg_systeminit.c | hdwinit<br>R_Systeminit  | ○<br>○                               |
|                                       | r_cg_macrodriver.h                      | —  | —                                    |
|                                       | r_cg_userdefine.h                       | —  | —                                    |
| クロック発生回路                              | r_cg_cgc.c                              | R_CGC_Create<br>R_CGC_Set_ClockMode<br>R_CGC_RAMECC_Start<br>R_CGC_RAMECC_Stop<br>R_CGC_StackPointer_Start<br>R_CGC_StackPointer_Stop<br>R_CGC_ClockMonitor_Start<br>R_CGC_ClockMonitor_Stop | ○<br>×<br>○<br>○<br>○<br>○<br>○<br>○ |
|                                       | r_cg_cgc_user.c                         | R_CGC_Create_UserInit<br>r_cg_ram_ecc_interrupt<br>r_cg_stackpointer_interrupt<br>r_cg_clockmonitor_interrupt<br>R_CGC_Get_ResetSource   | ×<br>○<br>○<br>○<br>○                |
|                                       | r_cg_cgc.h                              | —  | —                                    |
| ポート機能                                 | r_cg_port.c                             | R_PORT_Create  | ○                                    |
|                                       | r_cg_port_user.c                        | R_PORT_Create_UserInit   | ×                                    |
|                                       | r_cg_port.h                             | —  | —                                    |
| 高速オンチップ・<br>オシレータ・ク<br>ロック周波数補正<br>機能 | r_cg_hofc.c                             | R_HOFC_Create<br>R_HOFC_Start<br>R_HOFC_Stop   | ○<br>○<br>○                          |
|                                       | r_cg_hofc_user.c                        | R_HOFC_Create_UserInit<br>r_hofc_interrupt   | ×<br>○                               |
|                                       | r_cg_hofc.h                             | —  | —                                    |

| 周辺機能         | ファイル名                                  | API 関数名  | 出力 (*1)  |
|--------------|--|--|--|
| タイマ・アレイ・ユニット | r_cg_timer.c または r_cg_tau.c            | R_TAUm_Create<br>R_TAUm_Channeln_Start<br>R_TAUm_Channeln_Higher8bits_Start<br>R_TAUm_Channeln_Lower8bits_Start<br>R_TAUm_Channeln_Stop<br>R_TAUm_Channeln_Higher8bits_Stop<br>R_TAUm_Channeln_Lower8bits_Stop<br>R_TAUm_Reset<br>R_TAUm_Set_PowerOff<br>R_TAUm_Channeln_Get_PulseWidth<br>R_TAUm_Channeln_Set_SoftwareTriggerOn | ○<br>○<br>○<br>○<br>○<br>○<br>○<br>○<br>×<br>○<br>○<br>○ |
|              | r_cg_timer_user.c または r_cg_tau_user.c  | R_TAUm_Create_UserInit<br>r_taum_channeln_interrupt<br>r_taum_channeln_higher8bits_interrupt   | ×<br>○<br>○  |
|              | r_cg_timer.h または r_cg_tau.h            | —  | —  |
| タイマ RJ       | r_cg_timer.c または r_cg_tmrj.c           | R_TMR_RJn_Create<br>R_TMR_RJn_Start<br>R_TMR_RJn_Stop<br>R_TMR_RJn_Set_PowerOff<br>R_TMR_RJn_Get_PulseWidth<br>R_TMRJn_Create<br>R_TMRJn_Start<br>R_TMRJn_Stop<br>R_TMRJn_Set_PowerOff<br>R_TMRJn_Get_PulseWidth   | ○<br>○<br>○<br>○<br>○<br>○<br>○<br>○<br>○<br>×<br>×      |
|              | r_cg_timer_user.c または r_cg_tmrj_user.c | R_TMR_RJn_Create_UserInit<br>r_tmr_rjn_interrupt<br>R_TMRJn_Create_UserInit<br>r_tmrjn_interrupt   | ×<br>○<br>×<br>○   |
|              | r_cg_timer.h または r_cg_tmrj.h           | —  | —  |



| 周辺機能   | ファイル名                                    | API 関数名   | 出力 (*1)   |
|--------|--|---|---|
| タイマ RD | r_cg_timer.c または<br>r_cg_tmr.c           | R_TMR_RDn_Create<br>R_TMR_RDn_Start<br>R_TMR_RDn_Stop<br>R_TMR_RDn_Set_PowerOff<br>R_TMR_RDn_ForcedOutput_Start<br>R_TMR_RDn_ForcedOutput_Stop<br>R_TMR_RDn_Get_PulseWidth<br>R_TMRDn_Create<br>R_TMRDn_Start<br>R_TMRDn_Stop<br>R_TMRDn_Set_PowerOff<br>R_TMRDn_ForcedOutput_Start<br>R_TMRDn_ForcedOutput_Stop<br>R_TMRDn_Get_PulseWidth<br>R_TMRD_Set_PowerOff | ○<br>○<br>○<br>×<br>×<br>×<br>○<br>○<br>○<br>○<br>×<br>×<br>×<br>○<br>× |
|        | r_cg_timer_user.c または<br>r_cg_tmr_user.c | R_TMR_RDn_Create_UserInit<br>r_tmr_rdn_interrupt<br>R_TMRDn_Create_UserInit<br>r_tmr_rdn_interrupt  | ×<br>○<br>×<br>○  |
|        | r_cg_timer.h<br>または<br>r_cg_tmr.h        | —   | —   |
| タイマ RG | r_cg_timer.c                             | R_TMR_RG0_Create<br>R_TMR_RG0_Start<br>R_TMR_RG0_Stop<br>R_TMR_RG0_Set_PowerOff<br>R_TMR_RG0_Get_PulseWidth   | ○<br>○<br>○<br>×<br>○   |
|        | r_cg_timer_user.c                        | R_TMR_RG0_Create_UserInit<br>r_tmr_rg0_interrupt  | ×<br>○  |
|        | r_cg_timer.h                             | —   | —   |
| タイマ RX | r_cg_tmr.c                               | R_TMRX_Create<br>R_TMRX_Start<br>R_TMRX_Stop<br>R_TMRX_Get_BufferValue<br>R_TMRX_Set_PowerOff   | ○<br>○<br>○<br>○<br>×   |
|        | r_cg_tmr_user.c                          | R_TMRX_Create_UserInit<br>r_tmr_interrupt   | ×<br>○  |
|        | r_cg_tmr.h                               | —   | —   |

| 周辺機能              | ファイル名                                      | API 関数名   | 出力 (*1)  |
|-------------------|--|---|--|
| 16 ビット・タイマ<br>KB  | r_cg_timer.c または<br>r_cg_tmkb.c            | R_TMR_KB_Create<br>R_TMR_KBm_Start<br>R_TMR_KBm_Stop<br>R_TMR_KBm_Set_PowerOff<br>R_TMR_KBmn_ForcedOutput_Start<br>R_TMR_KBmn_ForcedOutput_Stop<br>R_TMR_KBm_BatchOverwriteRequestOn<br>R_TMR_KBm_ForcedOutput_mn_Start<br>R_TMR_KBm_ForcedOutput_mn_Stop<br>R_TMR_KBm_Reset  | ○<br>○<br>○<br>×<br>○<br>○<br>○<br>○<br>○<br>×   |
|                   | r_cg_timer_user.c また<br>は r_cg_tmkb_user.c | R_TMR_KBm_Create_UserInit<br>r_tmr_kbm_interrupt  | ×<br>○   |
|                   | r_cg_timer.h または<br>r_cg_tmkb.h            | —   | —  |
| 16 ビット・タイマ<br>KC0 | r_cg_timer.c                               | R_TMR_KC0_Create<br>R_TMR_KC0_Start<br>R_TMR_KC0_Stop<br>R_TMR_KC0_Set_PowerOff   | ○<br>○<br>○<br>×   |
|                   | r_cg_timer_user.c                          | R_TMR_KC0_Create_UserInit<br>r_tmr_kc0_interrupt  | ×<br>○   |
|                   | r_cg_timer.h                               | —   | —  |
| 16 ビット・タイマ<br>KB2 | r_cg_kb2.c                                 | R_KB2m_Create<br>R_KB2m_Start<br>R_KB2m_Stop<br>R_KB2m_Set_PowerOff<br>R_KB2m_Simultaneous_Start<br>R_KB2m_Simultaneous_Stop<br>R_KB2m_Synchronous_Start<br>R_KB2m_Synchronous_Stop<br>R_KB2m_TKBO0_Forced_Output_Stop_Function1_Start<br>R_KB2m_TKBO0_Forced_Output_Stop_Function1_Stop<br>R_KB2m_TKBO1_Forced_Output_Stop_Function1_Start<br>R_KB2m_TKBO1_Forced_Output_Stop_Function1_Stop<br>R_KB2m_TKBO0_DitheringFunction_Start<br>R_KB2m_TKBO0_DitheringFunction_Stop<br>R_KB2m_TKBO1_DitheringFunction_Start<br>R_KB2m_TKBO1_DitheringFunction_Stop<br>R_KB2m_TKBO0_SmoothStartFunction_Start<br>R_KB2m_TKBO0_SmoothStartFunction_Stop<br>R_KB2m_TKBO1_SmoothStartFunction_Start<br>R_KB2m_TKBO1_SmoothStartFunction_Stop<br>R_KB2m_Set_BatchOverwriteRequestOn | ○<br>○<br>○<br>×<br>○<br>○<br>○<br>○<br>○<br>○<br>○<br>○<br>○<br>○<br>○<br>○<br>○<br>○<br>○<br>○<br>○<br>○<br>○<br>○ |
|                   | r_cg_kb2_user.c                            | R_KB2m_Create_UserInit<br>r_kb2m_interrupt  | ×<br>○   |
|                   | r_cg_kb2.h                                 | —   | —  |

| 周辺機能                         | ファイル名                      | API 関数名                           | 出力 (*1) |
|------------------------------|----------------------------|-----------------------------------|---------|
| リアルタイム・クロック                  | r_cg_rtc.c                 | R_RTC_Create                      | ○       |
|                              |                            | R_RTC_Start                       | ○       |
|                              |                            | R_RTC_Stop                        | ○       |
|                              |                            | R_RTC_Set_PowerOff                | ○       |
|                              |                            | R_RTC_Set_HourSystem              | ×       |
|                              |                            | R_RTC_Set_CounterValue            | ○       |
|                              |                            | R_RTC_Set_CalendarCounterValue    | ○       |
|                              |                            | R_RTC_Set_BinaryCounterValue      | ○       |
|                              |                            | R_RTC_Get_CounterValue            | ○       |
|                              |                            | R_RTC_Get_CalendarCounterValue    | ○       |
|                              |                            | R_RTC_Get_BinaryCounterValue      | ○       |
|                              |                            | R_RTC_Set_ConstPeriodInterruptOn  | ○       |
|                              |                            | R_RTC_Set_ConstPeriodInterruptOff | ○       |
|                              |                            | R_RTC_Set_AlarmOn                 | ○       |
|                              |                            | R_RTC_Set_CalendarAlarmOn         | ○       |
|                              |                            | R_RTC_Set_BinaryAlarmOn           | ○       |
|                              |                            | R_RTC_Set_AlarmOff                | ○       |
|                              |                            | R_RTC_Set_AlarmValue              | ○       |
|                              |                            | R_RTC_Set_CalendarAlarmValue      | ○       |
|                              | R_RTC_Set_BinaryAlarmValue | ○                                 |         |
|                              | R_RTC_Get_AlarmValue       | ○                                 |         |
| R_RTC_Get_CalendarAlarmValue | ○                          |                                   |         |
| R_RTC_Get_BinaryAlarmValue   | ○                          |                                   |         |
| R_RTC_Set_RTC1HZOn           | ○                          |                                   |         |
| R_RTC_Set_RTC1HZOff          | ○                          |                                   |         |
| R_RTC_Set_RTCOUTOn           | ○                          |                                   |         |
| R_RTC_Set_RTCOUTOff          | ○                          |                                   |         |
|                              | r_cg_rtc_user.c            | R_RTC_Create_UserInit             | ×       |
|                              |                            | r_rtc_interrupt                   | ○       |
|                              |                            | r_rtc_callback_constperiod        | ○       |
|                              |                            | r_rtc_callback_alarm              | ○       |
|                              |                            | r_rtc_alarminterrupt              | ○       |
|                              | r_rtc_periodicinterrupt    | ○                                 |         |
| r_rtc_callback_periodic      | ○                          |                                   |         |
|                              | r_cg_rtc.h                 | —                                 | —       |
| サブシステム・クロック周波数測定回路           | r_cg_fmc.c                 | R_FMC_Create                      | ○       |
|                              |                            | R_FMC_Start                       | ○       |
|                              | R_FMC_Stop                 | ○                                 |         |
| R_FMC_Set_PowerOff           | ×                          |                                   |         |
|                              | r_cg_fmc_user.c            | R_FMC_Create_UserInit             | ×       |
|                              |                            | r_fmc_interrupt                   | ○       |
|                              | r_cg_fmc.h                 | —                                 | —       |
| 12ビット・インターバル・タイマ             | r_cg_it.c                  | R_IT_Create                       | ○       |
|                              |                            | R_IT_Start                        | ○       |
|                              |                            | R_IT_Stop                         | ○       |
|                              |                            | R_IT_Reset                        | ×       |
|                              | R_IT_Set_PowerOff          | ×                                 |         |
|                              | r_cg_it_user.c             | R_IT_Create_UserInit              | ×       |
|                              |                            | r_it_interrupt                    | ○       |
|                              | r_cg_it.h                  | —                                 | —       |

| 周辺機能                                 | ファイル名                | API 関数名  | 出力 (*1)                         |
|--------------------------------------|----------------------|--|---------------------------------|
| 8ビット・インターバル・タイマ                      | r_cg_it8bit.c        | R_IT8bitm_Channeln_Create<br>R_IT8bitm_Channeln_Start<br>R_IT8bitm_Channeln_Stop<br>R_IT8bitm_Channeln_Set_PowerOff<br>R_IT8bitm_Set_PowerOff                                  | ○<br>○<br>○<br>×<br>×           |
|                                      | r_cg_it8bit_user.c   | R_IT8bitm_Channeln_Create_UserInit<br>r_it8bitm_channeln_interrupt   | ×<br>○                          |
|                                      | r_cg_it8bit.h        | —  | —                               |
| 16ビット・ウエイクアップ・タイマ                    | r_cg_timer.c         | R_WUTM_Create<br>R_WUTM_Start<br>R_WUTM_Stop<br>R_WUTM_Set_PowerOff  | ○<br>○<br>○<br>×                |
|                                      | r_cg_timer_user.c    | R_WUTM_Create_UserInit<br>r_wutm_interrupt   | ×<br>○                          |
|                                      | r_cg_timer.h         | —  | —                               |
| クロック出力／ブザー出力制御回路                     | r_cg_pclbuz.c        | R_PCLBUZn_Create<br>R_PCLBUZn_Start<br>R_PCLBUZn_Stop<br>R_PCLBUZ_Set_PowerOff   | ○<br>○<br>○<br>×                |
|                                      | r_cg_pclbuz_user.c   | R_PCLBUZn_Create_UserInit  | ×                               |
|                                      | r_cg_pclbuz.h        | —  | —                               |
| ウォッチドッグ・タイマ                          | r_cg_wdt.c           | R_WDT_Create<br>R_WDT_Restart  | ○<br>○                          |
|                                      | r_cg_wdt_user.c      | R_WDT_Create_UserInit<br>r_wdt_interrupt   | ×<br>○                          |
|                                      | r_cg_wdt.h           | —  | —                               |
| プログラマブル・ゲイン計装アンプ付き 24ビット ΔΣA/D コンバータ | r_cg_pga_dsad.c      | R_PGA_DSAD_Create<br>R_PGA_DSAD_Start<br>R_PGA_DSAD_Stop<br>R_PGA_DSAD_Set_PowerOff<br>R_PGA_DSAD_Get_AverageResult<br>R_PGA_DSAD_Get_Result<br>R_PGA_DSAD_CAMP_OffsetTrimming | ○<br>○<br>○<br>×<br>○<br>○<br>○ |
|                                      | r_cg_pga_dsad_user.c | R_PGA_DSAD_Create_UserInit<br>r_pga_dsad_interrupt_conversion<br>r_pga_dsad_interrupt_scan<br>r_pga_dsad_conversion_interrupt<br>r_pga_dsad_scan_interrupt                     | ×<br>○<br>○<br>○<br>○           |
|                                      | r_cg_pga_dsad.h      | —  | —                               |

| 周辺機能                           | ファイル名             | API 関数名   | 出力 (*1)  |
|--------------------------------|-------------------|---|--|
| A/D コンバータ                      | r_cg_adc.c        | R_ADC_Create<br>R_ADC_Set_OperationOn<br>R_ADC_Set_OperationOff<br>R_ADC_Start<br>R_ADC_Stop<br>R_ADC_Reset<br>R_ADC_Set_PowerOff<br>R_ADC_Set_ADChannel<br>R_ADC_Set_SnoozeOn<br>R_ADC_Set_SnoozeOff<br>R_ADC_Set_TestChannel<br>R_ADC_Get_Result<br>R_ADC_Get_Result_8bit | ○<br>○<br>○<br>○<br>○<br>○<br>×<br>×<br>×<br>×<br>×<br>×<br>○<br>○ |
|                                | r_cg_adc_user.c   | R_ADC_Create_UserInit<br>r_adc_interrupt  | ×<br>○   |
|                                | r_cg_adc.h        | —   | —  |
| コンフィギュラブル・アンプ                  | r_cg_camp.c       | R_CAMP_Create<br>R_CAMPn_Start<br>R_CAMPn_Stop<br>R_CAMP_Set_PowerOff   | ○<br>○<br>○<br>×   |
|                                | r_cg_camp_user.c  | R_CAMP_Create_UserInit  | ×  |
|                                | r_cg_camp.h       | —   | —  |
| 温度センサ                          | r_cg_tmtps.c      | R_TMPS_Create<br>R_TMPS_Start<br>R_TMPS_Stop<br>R_TMPS_Reset<br>R_TMPS_Set_PowerOff   | ○<br>○<br>○<br>×<br>×  |
|                                | r_cg_tmtps_user.c | R_TMPS_Create_UserInit  | ×  |
|                                | r_cg_tmtps.h      | —   | —  |
| 24 ビット $\Delta\Sigma$ /D コンバータ | r_cg_dsadc.c      | R_DSADC_Create<br>R_DSADC_Set_OperationOn<br>R_DSADC_Set_OperationOff<br>R_DSADC_Start<br>R_DSADC_Stop<br>R_DSADC_Reset<br>R_DSADC_Set_PowerOff<br>R_DSADC_Channeln_Get_Result<br>R_DSADC_Channeln_Get_Result_16bit   | ○<br>○<br>○<br>○<br>○<br>○<br>×<br>×<br>○<br>○                     |
|                                | r_cg_dsadc_user.c | R_DSADC_Create_UserInit<br>r_dsadc_interrupt<br>r_dsadzcn_interrupt   | ×<br>○<br>○  |
|                                | r_cg_dsadc.h      | —   | —  |

| 周辺機能                     | ファイル名               | API 関数名  | 出力 (*1)  |
|--------------------------|---------------------|--|--|
| D/A コンバータ                | r_cg_dac.c          | R_DAC_Create<br>R_DACn_Start<br>R_DACn_Stop<br>R_DAC_Set_PowerOff<br>R_DACn_Set_ConversionValue<br>R_DAC_Change_OutputVoltage_8bit<br>R_DAC_Change_OutputVoltage<br>R_DACn_Create<br>R_DAC_Reset | ○<br>○<br>○<br>×<br>○<br>○<br>○<br>○<br>○<br>× |
|                          | r_cg_dac_user.c     | R_DAC_Create_UserInit<br>R_DACn_Create_UserInit  | ×<br>×   |
|                          | r_cg_dac.h          | —  | —  |
| プログラマブル・ゲイン・アンプ          | r_cg_pga.c          | R_PGA_Create<br>R_PGA_Start<br>R_PGA_Stop<br>R_PGA_Reset<br>R_PGA_Set_PowerOff   | ○<br>○<br>○<br>×<br>×                          |
|                          | r_cg_pga_user.c     | R_PGA_Create_UserInit  | ×  |
|                          | r_cg_pga.h          | —  | —  |
| コンパレータ                   | r_cg_comp.c         | R_COMP_Create<br>R_COMPn_Start<br>R_COMPn_Stop<br>R_COMP_Reset<br>R_COMP_Set_PowerOff  | ○<br>○<br>○<br>×<br>×                          |
|                          | r_cg_comp_user.c    | R_COMP_Create_UserInit<br>r_compn_interrupt  | ×<br>○   |
|                          | r_cg_comp.h         | —  | —  |
| コンパレータ / プログラマブル・ゲイン・アンプ | r_cg_comppga.c      | R_COMPPGA_Create<br>R_COMPPGA_Set_PowerOff<br>R_COMPn_Start<br>R_COMPn_Stop<br>R_PGA_Start<br>R_PGA_Stop<br>R_PWMOPT_Start<br>R_PWMOPT_Stop  | ○<br>×<br>○<br>○<br>○<br>○<br>○<br>○           |
|                          | r_cg_comppga_user.c | R_COMPPGA_Create_UserInit<br>r_compn_interrupt   | ×<br>○   |
|                          | r_cg_comppga.h      | —  | —  |

| 周辺機能                      | ファイル名                                  | API 関数名                          | 出力 (*1) |
|---------------------------|--|----------------------------------|---------|
| シリアル・アレイ・ユニット             | r_cg_serial.c または r_cg_sau.c           | R_SAUm_Create                    | ○       |
|                           |  | R_SAUm_Reset                     | ×       |
|                           |  | R_SAUm_Set_PowerOff              | ×       |
|                           |  | R_SAUm_Set_SnoozeOn              | ×       |
|                           |  | R_SAUm_Set_SnoozeOff             | ×       |
|                           |  | R_UARTn_Create                   | ○       |
|                           |  | R_UARTn_Start                    | ○       |
|                           |  | R_UARTn_Stop                     | ○       |
|                           |  | R_UARTn_Send                     | ○       |
|                           |  | R_UARTn_Receive                  | ○       |
|                           | r_cg_serial_user.c または r_cg_sau_user.c | R_SAUm_Create_UserInit           | ×       |
|                           |  | r_uartn_interrupt_send           | ○       |
|                           |  | r_uartn_interrupt_receive        | ○       |
|                           |  | r_uartn_interrupt_error          | ○       |
|                           |  | r_uartn_callback_sendend         | ○       |
|                           |  | r_uartn_callback_receiveend      | ○       |
|                           |  | r_uartn_callback_error           | ○       |
|                           |  | r_uartn_callback_softwareoverrun | ○       |
|                           |  | r_csirn_interrupt                | ○       |
|                           |  | r_csirn_callback_sendend         | ○       |
|                           | r_cg_serial.h または r_cg_sau.h           | —                                | —       |
|                           |  |                                  |         |
|                           |  |                                  |         |
|                           |  |                                  |         |
|                           |  |                                  |         |
|                           |  |                                  |         |
|                           |  |                                  |         |
|                           |  |                                  |         |
|                           |  |                                  |         |
|                           |  |                                  |         |
| シリアル・アレイ・ユニット4            | r_cg_serial.c                          | R_DALIn_Create                   | ○       |
|                           |  | R_DALIn_Start                    | ○       |
|                           |  | R_DALIn_Stop                     | ○       |
|                           |  | R_DALIn_Send                     | ○       |
|                           | R_DALIn_Receive                        | ○                                |         |
|                           | r_cg_serial_user.c                     | r_dalin_interrupt_send           | ○       |
| r_dalin_interrupt_receive |  | ○                                |         |
|                           | r_cg_serial.h                          | r_dalin_interrupt_error          | ○       |
|                           |  | —                                | —       |

| 周辺機能                          | ファイル名                                   | API 関数名   | 出力 (*1)  |
|-------------------------------|---|---|--|
| アシンクロナス・シリアル・インタフェース LIN-UART | r_cg_serial.c                           | R_UARTFn_Create<br>R_UARTFn_Start<br>R_UARTFn_Stop<br>R_UARTFn_Set_PowerOff<br>R_UARTFn_Send<br>R_UARTFn_Receive<br>R_UARTFn_Set_DataComparisonOn<br>R_UARTFn_Set_DataComparisonOff   | ○<br>○<br>○<br>×<br>○<br>○<br>○<br>○                               |
|                               | r_cg_serial_user.c                      | R_UARTFn_Create_UserInit<br>r_uartfn_interrupt_send<br>r_uartfn_interrupt_receive<br>r_uartfn_interrupt_error<br>r_uartfn_callback_sendend<br>r_uartfn_callback_receiveend<br>r_uartfn_callback_error<br>r_uartfn_callback_softwareoverrun<br>r_uartfn_callback_expbitdetect<br>r_uartfn_callback_idmatch | ×<br>○<br>○<br>○<br>○<br>○<br>○<br>○<br>○<br>○<br>○                |
|                               | r_cg_serial.h                           | —   | —  |
| シリアル・インタフェース IICA             | r_cg_serial.c または r_cg_iica.c           | R_IICAn_Create<br>R_IICAn_StopCondition<br>R_IICAn_Stop<br>R_IICAn_Reset<br>R_IICAn_Set_PowerOff<br>R_IICAn_Master_Send<br>R_IICAn_Master_Receive<br>R_IICAn_Slave_Send<br>R_IICAn_Slave_Receive<br>R_IICAn_Set_SnoozeOn<br>R_IICAn_Set_SnoozeOff<br>R_IICAn_Set_WakeupOn<br>R_IICAn_Set_WakeupOff        | ○<br>×<br>○<br>×<br>×<br>○<br>○<br>○<br>○<br>○<br>○<br>○<br>○<br>○ |
|                               | r_cg_serial_user.c または r_cg_iica_user.c | R_IICAn_Create_UserInit<br>r_iican_interrupt<br>r_iican_callback_master_sendend<br>r_iican_callback_master_receiveend<br>r_iican_callback_master_error<br>r_iican_callback_slave_sendend<br>r_iican_callback_slave_receiveend<br>r_iican_callback_slave_error<br>r_iican_callback_getstopcondition        | ×<br>○<br>○<br>○<br>○<br>○<br>○<br>○<br>○<br>×                     |
|                               | r_cg_serial.h または r_cg_iica.h           | —   | —  |
| LCD コントローラ / ドライバ             | r_cg_lcd.c                              | R_LCD_Create<br>R_LCD_Start<br>R_LCD_Stop<br>R_LCD_Set_VoltageOn<br>R_LCD_Set_VoltageOff<br>R_LCD_Set_PowerOff<br>R_LCD_VoltageOn<br>R_LCD_VoltageOff   | ○<br>○<br>○<br>○<br>○<br>○<br>○<br>○                               |
|                               | r_cg_lcd_user.c                         | R_LCD_Create_UserInit<br>r_lcd_interrupt  | ×<br>○   |
|                               | r_cg_lcd.h                              | —   | —  |



| 周辺機能              | ファイル名            | API 関数名  | 出力 (*1)  |
|-------------------|------------------|--|--|
| サウンド・ジェネレータ       | r_cg_sg.c        | R_SG_Create<br>R_SG_Start<br>R_SG_Stop   | ○<br>○<br>○                                    |
|                   | r_cg_sg_user.c   | R_SG_Create_UserInit<br>r_sg_interrupt   | ×<br>○   |
|                   | r_cg_sg.h        | —  | —  |
| DMA コントローラ        | r_cg_dmac.c      | R_DMACHn_Create<br>R_DMACHn_Start<br>R_DMACHn_Stop<br>R_DMACHn_Set_SoftwareTriggerOn   | ○<br>○<br>○<br>○<br>○                          |
|                   | r_cg_dmac_user.c | R_DMACHn_Create_UserInit<br>R_DMACHn_Set_SoftwareTriggerOn<br>r_dmacn_interrupt  | ×<br>×<br>○                                    |
|                   | r_cg_dmac.h      | —  | —  |
| データ・トランスファ・コントローラ | r_cg_dtc.c       | R_DTC_Create<br>R_DTCn_Start<br>R_DTCn_Stop<br>R_DTC_Set_PowerOff<br>R_DTCDn_Start<br>R_DTCDn_Stop   | ○<br>○<br>○<br>×<br>○<br>○                     |
|                   | r_cg_dtc_user.c  | R_DTC_Create_UserInit  | ×  |
|                   | r_cg_dtc.h       | —  | —  |
| イベントリンクコントローラ     | r_cg_elc.c       | R_ELC_Create<br>R_ELC_Stop   | ○<br>○   |
|                   | r_cg_elc_user.c  | R_ELC_Create_UserInit  | ×  |
|                   | r_cg_elc.h       | —  | —  |
| 割り込み機能            | r_cg_intc.c      | R_INTC_Create<br>R_INTCn_Start<br>R_INTCn_Stop<br>R_INTCLRn_Start<br>R_INTCLRn_Stop<br>R_INTRTCICn_Start<br>R_INTRTCICn_Stop<br>R_INTFO_Start<br>R_INTFO_Stop<br>R_INTFO_ClearFlag | ○<br>○<br>○<br>○<br>○<br>○<br>○<br>○<br>○<br>○ |
|                   | r_cg_intc_user.c | R_INTC_Create_UserInit<br>r_intcn_interrupt<br>r_intclrn_interrupt<br>r_intrtcicn_interrupt<br>r_intfo_interrupt   | ×<br>○<br>○<br>○<br>○                          |
|                   | r_cg_intc.h      | —  | —  |

| 周辺機能           | ファイル名                                   | API 関数名  | 出力 (*1)   |
|----------------|---|--|---|
| キー割り込み機能       | r_cg_intc.c または<br>r_cg_key.c           | R_KEY_Create<br>R_KEY_Start<br>R_KEY_Stop  | ○<br>○<br>○   |
|                | r_cg_intc_user.c または<br>r_cg_key_user.c | R_KEY_Create_UserInit<br>r_key_interrupt   | ×<br>○  |
|                | r_cg_intc.h または<br>r_cg_key.h           | —  | —   |
| 電圧検出回路         | r_cg_lvd.c                              | R_LVD_Create<br>R_LVD_InterruptMode_Start<br>R_LVD_Start_VDD<br>R_LVD_Start_VBAT<br>R_LVD_Start_VRTC<br>R_LVD_Start_EXLVD<br>R_LVD_Stop_VDD<br>R_LVD_Stop_VBAT<br>R_LVD_Stop_VRTC<br>R_LVD_Stop_EXLVD<br>R_LVI_Create<br>R_LVI_InterruptMode_Start | ○<br>○<br>○<br>○<br>○<br>○<br>○<br>○<br>○<br>○<br>○<br>○<br>○ |
|                | r_cg_lvd_user.c                         | R_LVD_Create_UserInit<br>r_lvd_interrupt<br>r_lvd_vddinterrupt<br>r_lvd_vbatinterrupt<br>r_lvd_vrtcinterrupt<br>r_lvd_exlvdinterrupt<br>R_LVI_Create_UserInit<br>r_lvi_interrupt   | ×<br>○<br>○<br>○<br>○<br>○<br>○<br>×<br>○                     |
|                | r_cg_lvd.h                              | —  | —   |
| バッテリー・バックアップ機能 | r_cg_bup.c                              | R_BUP_Create<br>R_BUP_Start<br>R_BUP_Stop  | ○<br>○<br>○   |
|                | r_cg_bup_user.c                         | R_BUP_Create_UserInit<br>r_bup_interrupt   | ×<br>○  |
|                | r_cg_bup.h                              | —  | —   |
| 発振停止検出回路       | r_cg_osdc.c                             | R_OSDC_Create<br>R_OSDC_Start<br>R_OSDC_Stop<br>R_OSDC_Set_PowerOff<br>R_OSDC_Reset  | ○<br>○<br>○<br>×<br>×   |
|                | r_cg_osdc_user.c                        | R_OSDC_Create_UserInit<br>r_osdc_interrupt   | ×<br>○  |
|                | r_cg_osdc.h                             | —  | —   |

| 周辺機能         | ファイル名                                   | API 関数名   | 出力 (*1)                         |
|--------------|---|---|---------------------------------|
| SPI インタフェース  | r_cg_saic.c または<br>r_cg_spi.c           | R_SAIC_Create<br>R_SAIC_Write<br>R_SAIC_Read<br>R_SPI_Create<br>R_SPI_Write<br>R_SPI_Read   | ○<br>○<br>×<br>○<br>○<br>○      |
|              | r_cg_saic_user.c<br>または r_cg_spi_user.c | R_SAIC_Create_UserInit<br>R_SPI_Create_UserInit   | ×<br>×                          |
|              | r_cg_saic.h または<br>r_cg_spi.h           | —   | —                               |
| オペアンプ        | r_cg_opamp.c                            | R_OPAMP_Create<br>R_OPAMP_Set_ReferenceCircuitOn<br>R_OPAMP_Set_ReferenceCircuitOff<br>R_OPAMPn_Start<br>R_OPAMPn_Stop<br>R_OPAMPn_Set_PrechargeOn<br>R_OPAMPn_Set_PrechargeOff | ○<br>○<br>○<br>○<br>○<br>×<br>× |
|              | r_cg_opamp_user.c                       | R_OPAMP_Create_UserInit   | ×                               |
|              | r_cg_opamp.h                            | —   | —                               |
| データ演算回路      | r_cg_doc.c                              | R_DOC_Create<br>R_DOC_SetMode<br>R_DOC_WriteData<br>R_DOC_GetResult<br>R_DOC_ClearFlag<br>R_DOC_Set_PowerOff<br>R_DOC_Reset   | ○<br>○<br>○<br>○<br>○<br>×<br>× |
|              | r_cg_doc_user.c                         | R_DOC_Create_UserInit<br>r_doc_interrupt  | ×<br>○                          |
|              | r_cg_doc.h                              | —   | —                               |
| 32 ビット積和演算回路 | r_cg_mac32bit.c                         | R_MAC32Bit_Create<br>R_MAC32Bit_Reset<br>R_MAC32Bit_Set_PowerOff<br>R_MAC32Bit_MULUnsigned<br>R_MAC32Bit_MULSigned<br>R_MAC32Bit_MACUnsigned<br>R_MAC32Bit_MACSigned            | ○<br>×<br>×<br>×<br>×<br>×<br>× |
|              | r_cg_mac32bit_user.c                    | R_MAC32Bit_Create_UserInit<br>r_mac32bit_interrupt_flow   | ×<br>○                          |
|              | r_cg_mac32bit.h                         | —   | —                               |

| 周辺機能              | ファイル名               | API 関数名   | 出力 (*1)  |
|-------------------|---------------------|---|--|
| 12ビット A/D コンバータ   | r_cg_12adc.c        | R_12ADC_Create<br>R_12ADC_Start<br>R_12ADC_Stop<br>R_12ADC_Get_ValueResult<br>R_12ADC_Set_ADChannel<br>R_12ADC_TemperatureSensorOutput_On<br>R_12ADC_TemperatureSensorOutput_Off<br>R_12ADC_InternalReferenceVoltage_On<br>R_12ADC_InternalReferenceVoltage_Off<br>R_12ADC_Set_PowerOff | ○<br>○<br>○<br>○<br>×<br>×<br>×<br>×<br>×<br>× |
|                   | r_cg_12adc_user.c   | R_12ADC_Create_UserInit<br>r_12adc_interrupt  | ×<br>○   |
|                   | r_cg_12adc.h        | —   | —  |
| 12ビット D/A コンバータ   | r_cg_12da.c         | R_12DA_Create<br>R_12DAn_Start<br>R_12DAn_Stop<br>R_12DA_Set_PowerOff<br>R_12DAn_Set_ConversionValue  | ○<br>○<br>○<br>×<br>○                          |
|                   | r_cg_12da_user.c    | R_12DA_Create_UserInit  | ×  |
|                   | r_cg_12da.h         | —   | —  |
| オペアンプ&アナログスイッチ    | r_cg_ampansw.c      | R_AMPANSW_Create<br>R_OPAMPm_Set_ReferenceCircuitOn<br>R_OPAMPm_Set_ReferenceCircuitOff<br>R_OPAMPm_Start<br>R_OPAMPm_Stop<br>R_ANSW_ChargePumpm_On<br>R_ANSW_ChargePumpm_Off   | ○<br>○<br>○<br>○<br>○<br>○<br>○                |
|                   | r_cg_ampansw_user.c | R_AMPANSW_Create_UserInit   | ×  |
|                   | r_cg_ampansw.h      | —   | —  |
| ボルテージ・リファレンス      | r_cg_vr.c           | R_VR_Create<br>R_VR_Start<br>R_VR_Stop  | ○<br>○<br>○                                    |
|                   | r_cg_vr_user.c      | R_VR_Create_UserInit  | ×  |
|                   | r_cg_vr.h           | —   | —  |
| サンプリング出力タイマ/ディテクタ | r_cg_smotd.c        | R_SMOTD_Create<br>R_SMOTD_Start<br>R_SMOTD_Stop<br>R_SMOTD_Set_PowerOFF   | ○<br>○<br>○<br>×                               |
|                   | r_cg_smotd_user.c   | R_SMOTD_Create_UserInit<br>r_smotd_counterA_interrupt<br>r_smotd_counterB_interrupt<br>r_smotd_smpn_interrupt   | ×<br>○<br>○<br>○                               |
|                   | r_cg_smotd.h        | —   | —  |

| 周辺機能                | ファイル名              | API 関数名   | 出力 (*1)                              |
|---------------------|--------------------|---|--------------------------------------|
| 外部サンプリング            | r_cg_exsd.c        | R_EXSD_Create<br>R_EXSD_Start<br>R_EXSD_Stop<br>R_EXSD_Set_PowerOff   | ○<br>○<br>○<br>×                     |
|                     | r_cg_exsd_user.c   | R_EXSD_Create_UserInit<br>r_exsd_interrupt  | ×<br>○                               |
|                     | r_cg_exsd.h        | —   | —                                    |
| シリアル・インタフェース UARTMG | r_cg_uartmg.c      | R_UARTMGn_Create<br>R_UARTMGn_Start<br>R_UARTMGn_Stop<br>R_UARTMGn_Send<br>R_UARTMGn_Receive  | ○<br>○<br>○<br>○<br>○                |
|                     | r_cg_uartmg_user.c | R_UARTMGn_Create_UserInit<br>r_uartmgn_interrupt_send<br>r_uartmgn_interrupt_receive<br>r_uartmgn_interrupt_error<br>r_uartmgn_callback_sendend<br>r_uartmgn_callback_receiveend<br>r_uartmgn_interrupt_error<br>r_uartmgn_callback_softwareoverrun | ×<br>○<br>○<br>○<br>○<br>○<br>○<br>○ |
|                     | r_cg_usetmg.h      | —   | —                                    |
| アンプ・ユニット            | r_cg_amp.c         | R_AMP_Create<br>R_AMP_Set_PowerOn<br>R_AMP_Set_PowerOff<br>R_PGA1_Start<br>R_PGA1_Stop<br>R_AMPn_Start<br>R_AMPn_Stop   | ○<br>○<br>○<br>○<br>○<br>○<br>○      |
|                     | r_cg_amp_user.c    | R_AMP_Create_UserInit   | ×                                    |
|                     | r_cg_amp.h         | —   | —                                    |
| データ・フラッシュ・ライブラリ     | r_cg_pfdl.c        | R_FDL_Create<br>R_FDL_Open<br>R_FDL_Close<br>R_FDL_Write<br>R_FDL_Read<br>R_FDL_Erase   | ○<br>○<br>○<br>○<br>○<br>○           |
|                     | r_cg_pfdl.h        | —   | —                                    |

- \*1 [コード生成 . プロパティ ,API 関数の出力設定] がデフォルト (設定にあわせてすべて出力する) の場合  
○ : 周辺機能パネルの設定により自動で出力されます。  
× : "コード・プレビュー" から、API のプロパティを開き、"関数を使用する" の設定により、出力されます。

## 3. API関数

本章では、コード生成が出力するAPI関数について説明します。

### 3.1 概要

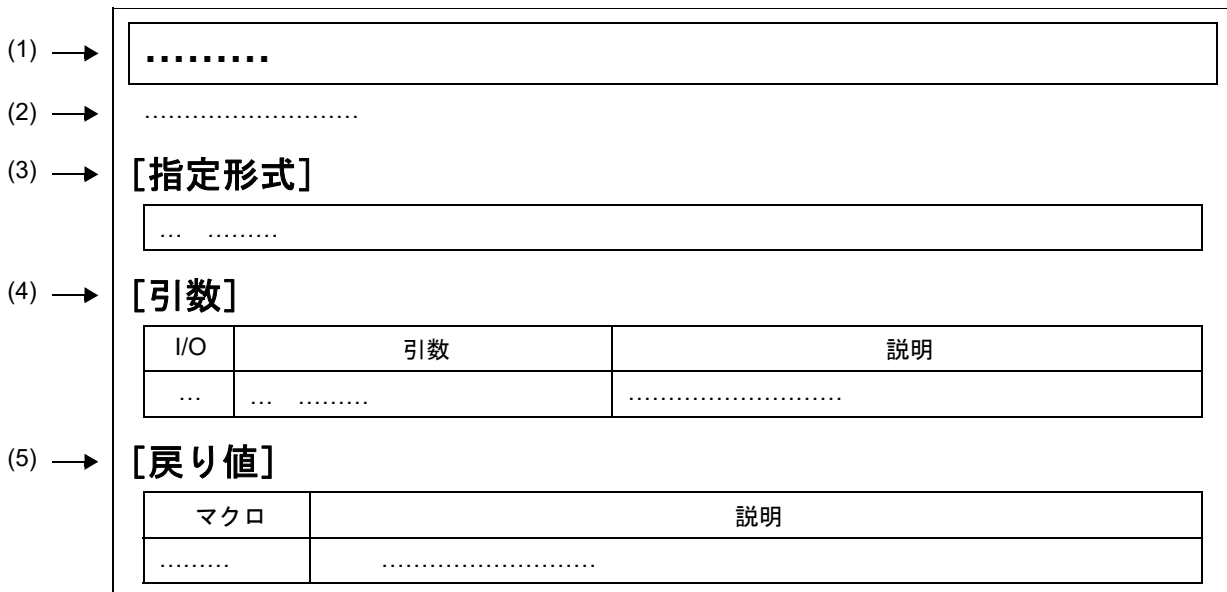
以下に、コード生成がAPI関数を出力する際の命名規則を示します。

- マクロ名  
すべて大文字。  
なお、先頭に“数字”が付与されている場合、該当数字（16進数値）とマクロ値は同値。
- ローカル変数名  
すべて小文字。
- グローバル変数名  
先頭に“g”を付与し、構成単語の先頭のみ大文字。
- グローバル変数へのポインタ名  
先頭に“gp”を付与し、構成単語の先頭のみ大文字。
- 列挙指定子 enum の要素名  
すべて大文字。

### 3.2 関数リファレンス

本節では、コード生成が出力するAPI関数について、次の記述フォーマットに従って説明します。

図 3.1 API関数の記述フォーマット



- (1) 名称  
API関数の名称を示しています。
- (2) 機能  
API関数の機能概要を示しています。
- (3) [指定形式]  
API関数をC言語で呼び出す際の記述形式を示しています。
- (4) [引数]  
API関数の引数を次の形式で示しています。

| I/O | 引数  | 説明  |
|-----|-----|-----|
| (a) | (b) | (c) |

- (a) I/O
  - 引数の種類
  - I ... 入力引数
  - O ... 出力引数
- (b) 引数
  - 引数のデータ・タイプ
- (c) 説明
  - 引数の説明

- (5) [戻り値]  
API関数からの戻り値を次の形式で示しています。

| マクロ | 説明  |
|-----|-----|
| (a) | (b) |

- (a) マクロ
  - 戻り値のマクロ
- (b) 説明
  - 戻り値の説明

### 3.2.1 共 通

以下に、コード生成が共通用として出力する API 関数の一覧を示します。

表 3.1 共通用 API 関数

| API 関数名         | 機能概要   |
|-----------------|--|
| hdwinit         | 各種ハードウェアを制御するうえで必要となる初期化処理を行います。<br>Renesas 製コンパイラ提供のスタートアップ・ルーチンから自動で呼ばれます。 |
| R_Systeminit    | 各種周辺機能を制御するうえで必要となる初期化処理を行います。   |
| main            | main 関数です。   |
| R_MAIN_UserInit | ユーザ独自の初期化処理を行います。  |



## hdwinit

各種ハードウェアを制御するうえで必要となる初期化処理を行います。

備考 本 API 関数の呼び出しは、スタートアップ・ルーチンから行ってください。

### [指定形式]

```
void hdwinit ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_Systeminit

各種周辺機能を制御するうえで必要となる初期化処理を行います。

備考 本 API 関数は、[hdwinit](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_Systeminit ( void );
```

### [引数]

なし

### [戻り値]

なし

**main**

main 関数です。

備考 本 API 関数の呼び出しは、スタートアップ・ルーチンから行ってください。

**[指定形式]**

```
void main ( void );
```

**[引数]**

なし

**[戻り値]**

なし

## R\_MAIN\_UserInit

ユーザ独自の初期化処理を行います。

備考           本 API 関数は、`main` のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void   R_MAIN_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

### 3.2.2 クロック発生回路

以下に、コード生成がクロック発生回路（リセット機能、オンチップ・デバッグ機能などを含む）用として出力するAPI関数の一覧を示します。

表 3.2 クロック発生回路用 API 関数

| API 関数名                                      | 機能概要   |
|--|--|
| <a href="#">R_CGC_Create</a>                 | クロック発生回路（リセット機能、オンチップ・デバッグ機能などを含む）を制御するうえで必要となる初期化処理を行います。 |
| <a href="#">R_CGC_Create_UserInit</a>        | クロック発生回路（リセット機能、オンチップ・デバッグ機能などを含む）に関するユーザ独自の初期化処理を行います。    |
| <a href="#">r_cgc_ram_ecc_interrupt</a>      | RAM 1 bit 訂正 / 2 bit エラー検出割り込み INTRAM の発生に伴う処理を行います。       |
| <a href="#">r_cgc_stackpointer_interrupt</a> | スタック・ポインタ・オーバフロー / アンダフロー割り込み INTSPM の発生に伴う処理を行います。        |
| <a href="#">r_cgc_clockmonitor_interrupt</a> | クロック・モニタ割り込み INTCLM の発生に伴う処理を行います。                         |
| <a href="#">R_CGC_Get_ResetSource</a>        | 内部リセットの発生に伴う処理を行います。                                       |
| <a href="#">R_CGC_Set_ClockMode</a>          | CPU クロック / 周辺ハードウェア・クロックを変更します。                            |
| <a href="#">R_CGC_RAMECC_Start</a>           | RAM-ECC 機能を開始します。  |
| <a href="#">R_CGC_RAMECC_Stop</a>            | RAM-ECC 機能を終了します。  |
| <a href="#">R_CGC_StackPointer_Start</a>     | CPU スタック・ポインタ・モニタ機能を開始します。                                 |
| <a href="#">R_CGC_StackPointer_Stop</a>      | CPU スタック・ポインタ・モニタ機能を終了します。                                 |
| <a href="#">R_CGC_ClockMonitor_Start</a>     | クロック・モニタを開始します。  |
| <a href="#">R_CGC_ClockMonitor_Stop</a>      | クロック・モニタを終了します。  |

## R\_CGC\_Create

クロック発生回路（リセット機能, オンチップ・デバッグ機能などを含む）を制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_CGC_Create ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_CGC\_Create\_UserInit

クロック発生回路（リセット機能, オンチップ・デバッグ機能などを含む）に関するユーザ独自の初期化処理を行います。

備考           本 API 関数は、[R\\_CGC\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_CGC_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

## r\_cgc\_ram\_ecc\_interrupt

RAM 1 bit 訂正／2 bit エラー検出割り込み INTRAM の発生に伴う処理を行います。

備考 本 API 関数は、RAM 1 bit 訂正／2 bit エラー検出割り込み INTRAM に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_cgc_ram_ecc_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_cgc_ram_ecc_interrupt ( void );
```

### [引数]

なし

### [戻り値]

なし



## r\_cgc\_stackpointer\_interrupt

スタック・ポインタ・オーバーフロー／アンダフロー割り込み INTSPM の発生に伴う処理を行います。

備考 本 API 関数は、スタック・ポインタ・オーバーフロー／アンダフロー割り込み INTSPM に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_cgc_stackpointer_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_cgc_stackpointer_interrupt ( void );
```

### [引数]

なし

### [戻り値]

なし

## r\_cgc\_clockmonitor\_interrupt

クロック・モニタ割り込み INTCLM の発生に伴う処理を行います。

備考 本 API 関数は、クロック・モニタ割り込み INTCLM に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_cgc_clockmonitor_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_cgc_clockmonitor_interrupt ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_CGC\_Get\_ResetSource

内部リセットの発生に伴う処理を行います。

### [指定形式]

```
void R_CGC_Get_ResetSource ( void );
```

### [引数]

なし

### [戻り値]

なし

**R\_CGC\_Set\_ClockMode**

CPU クロック／周辺ハードウェア・クロックを変更します。

**[指定形式]**

```
#include "r_cg_macrodriver.h"
#include "r_cg_cgc.h"
MD_STATUS R_CGC_Set_ClockMode ( clock_mode_t mode );
```

**[引数]**

| I/O | 引数                 | 説明  |
|-----|--------------------|---|
| I   | clock_mode_t mode; | CPU クロック／周辺ハードウェア・クロックの種類<br>HIOCLK : 高速オンチップ・オシレータ<br>SYSX1CLK : X1 クロック<br>SYSEXTCLK : 外部メイン・システム・クロック<br>SUBXT1CLK : XT1 クロック<br>SUBEXTCLK : 外部サブシステム・クロック |

**[戻り値]**

| マクロ         | 説明       |
|-------------|----------|
| MD_OK       | 正常終了     |
| MD_ERROR1   | 異常終了     |
| MD_ERROR2   | 異常終了     |
| MD_ERROR3   | 異常終了     |
| MD_ERROR4   | 異常終了     |
| MD_ARGERROR | 引数の指定が不正 |

## R\_CGC\_RAMECC\_Start

RAM-ECC 機能を開始します。

### [指定形式]

```
void R_CGC_RAMECC_Start ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_CGC\_RAMECC\_Stop

RAM-ECC 機能を終了します。

### [指定形式]

```
void R_CGC_RAMECC_Stop ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_CGC\_StackPointer\_Start

CPU スタック・ポインタ・モニタ機能を開始します。

### [指定形式]

```
void R_CGC_StackPointer_Start ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_CGC\_StackPointer\_Stop

CPU スタック・ポインタ・モニタ機能を終了します。

### [指定形式]

```
void R_CGC_StackPointer_Stop ( void );
```

### [引数]

なし

### [戻り値]

なし



**R\_CGC\_ClockMonitor\_Start**

クロック・モニタを開始します。

**[指定形式]**

```
void R_CGC_ClockMonitor_Start ( void );
```

**[引数]**

なし

**[戻り値]**

なし

**R\_CGC\_ClockMonitor\_Stop**

クロック・モニタを終了します。

**[指定形式]**

```
void R_CGC_ClockMonitor_Stop ( void );
```

**[引数]**

なし

**[戻り値]**

なし

### 3.2.3 ポート機能

以下に、コード生成がポート機能用として出力する API 関数の一覧を示します。

表 3.3 ポート機能用 API 関数

| API 関数名                                | 機能概要                          |
|--|-------------------------------|
| <a href="#">R_PORT_Create</a>          | ポート機能を制御するうえで必要となる初期化処理を行います。 |
| <a href="#">R_PORT_Create_UserInit</a> | ポート機能に関するユーザ独自の初期化処理を行います。    |

## R\_PORT\_Create

ポート機能を制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_PORT_Create ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_PORT\_Create\_UserInit

ポート機能に関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_PORT\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_PORT_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

### 3.2.4 高速オンチップ・オシレータ・クロック周波数補正機能

以下に、コード生成が高速オンチップ・オシレータ・クロック周波数補正機能用として出力する API 関数の一覧を示します。

表 3.4 高速オンチップ・オシレータ・クロック周波数補正機能用 API 関数

| API 関数名                                | 機能概要  |
|--|---|
| <a href="#">R_HOFC_Create</a>          | 高速オンチップ・オシレータ・クロック周波数補正機能を制御するうえで必要となる初期化処理を行います。 |
| <a href="#">R_HOFC_Create_UserInit</a> | 高速オンチップ・オシレータ・クロック周波数補正機能に関するユーザ独自の初期化処理を行います。    |
| <a href="#">r_hofc_interrupt</a>       | 高速オンチップ・オシレータ・クロック周波数補正機能完了割り込みの発生に伴う処理を行います。     |
| <a href="#">R_HOFC_Start</a>           | 高速オンチップ・オシレータ・クロック周波数補正機能を開始します。                  |
| <a href="#">R_HOFC_Stop</a>            | 高速オンチップ・オシレータ・クロック周波数補正機能を終了します。                  |

## R\_HOFC\_Create

高速オンチップ・オシレータ・クロック周波数補正機能を制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_HOFC_Create ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_HOFC\_Create\_UserInit

高速オンチップ・オシレータ・クロック周波数補正機能に関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_HOFC\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_HOFC_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし



## r\_hofc\_interrupt

高速オンチップ・オシレータ・クロック周波数補正機能完了割り込みの発生に伴う処理を行います。

備考 本 API 関数は、高速オンチップ・オシレータ・クロック周波数補正機能完了割り込み INTCR に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_hofc_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_hofc_interrupt ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_HOFC\_Start

高速オンチップ・オシレータ・クロック周波数補正機能を開始します。

### [指定形式]

```
void R_HOFC_Start ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_HOFC\_Stop

高速オンチップ・オシレータ・クロック周波数補正機能を終了します。

### [指定形式]

```
void R_HOFC_Stop ( void );
```

### [引数]

なし

### [戻り値]

なし

### 3.2.5 タイマ・アレイ・ユニット

以下に、コード生成がタイマ・アレイ・ユニット用として出力する API 関数の一覧を示します。

表 3.5 タイマ・アレイ・ユニット用 API 関数

| API 関数名   | 機能概要  |
|---|---|
| <a href="#">R_TAUm_Create</a>                         | タイマ・アレイ・ユニットを制御するうえで必要となる初期化処理を行います。                    |
| <a href="#">R_TAUm_Create_UserInit</a>                | タイマ・アレイ・ユニットに関するユーザ独自の初期化処理を行います。                       |
| <a href="#">r_taum_channeln_interrupt</a>             | タイマ割り込み INTT $Mmn$ の発生に伴う処理を行います。                       |
| <a href="#">r_taum_channeln_higher8bits_interrupt</a> | タイマ割り込み INTT $MmnH$ の発生に伴う処理を行います。                      |
| <a href="#">R_TAUm_Channeln_Start</a>                 | チャンネル $n$ のカウントを開始します。                                  |
| <a href="#">R_TAUm_Channeln_Higher8bits_Start</a>     | チャンネル $n$ のカウント（上位 8 ビット）を開始します。                        |
| <a href="#">R_TAUm_Channeln_Lower8bits_Start</a>      | チャンネル $n$ のカウント（下位 8 ビット）を開始します。                        |
| <a href="#">R_TAUm_Channeln_Stop</a>                  | チャンネル $n$ のカウントを終了します。                                  |
| <a href="#">R_TAUm_Channeln_Higher8bits_Stop</a>      | チャンネル $n$ のカウント（上位 8 ビット）を終了します。                        |
| <a href="#">R_TAUm_Channeln_Lower8bits_Stop</a>       | チャンネル $n$ のカウント（下位 8 ビット）を終了します。                        |
| <a href="#">R_TAUm_Reset</a>                          | タイマ・アレイ・ユニットをリセットします。                                   |
| <a href="#">R_TAUm_Set_PowerOff</a>                   | タイマ・アレイ・ユニットに対するクロック供給を停止します。                           |
| <a href="#">R_TAUm_Channeln_Get_PulseWidth</a>        | Tl $mn$ 端子に対する入力信号（入力パルス）のパルス間隔、またはハイ/ロウ・レベルの測定幅を獲得します。 |
| <a href="#">R_TAUm_Channeln_Set_SoftwareTriggerOn</a> | ワンショット・パルス出力のためのトリガ（ソフトウェア・トリガ）を発生させます。                 |

## R\_TAUm\_Create

タイマ・アレイ・ユニットを制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_TAUm_Create ( void );
```

備考 *m*は、ユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_TAUm\_Create\_UserInit

タイマ・アレイ・ユニットに関するユーザ独自の初期化処理を行います。

備考 本API関数は、[R\\_TAUm\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_TAUm_Create_UserInit ( void );
```

備考 *m*は、ユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし

## r\_taum\_channeln\_interrupt

タイマ割り込み INTTM $m$ n の発生に伴う処理を行います。

備考 本 API 関数は、タイマ割り込み INTTM $m$ n に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_taum_channeln_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_taum_channeln_interrupt ( void );
```

備考  $m$  はユニット番号を、 $n$  はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## r\_taum\_channeln\_higher8bits\_interrupt

タイマ割り込み INTT $Mn$ H の発生に伴う処理を行います。

備考 本 API 関数は、タイマ割り込み INTT $Mn$ H に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_taum_channeln_higher8bits_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_taum_channeln_higher8bits_interrupt ( void );
```

備考  $m$  はユニット番号を、 $n$  はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし



## R\_TAUm\_Channeln\_Start

チャンネル  $n$  のカウントを開始します。

備考 本 API 関数を呼び出してからカウント処理を開始するまでの時間は、該当機能の種類（インターバル・タイマ、方形波出力、外部イベント・カウンタなど）により異なります。

### [指定形式]

```
void R_TAUm_Channeln_Start ( void );
```

備考  $m$  はユニット番号を、 $n$  はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_TAUm\_Channeln\_Higher8bits\_Start

チャンネル  $n$  のカウント（上位 8 ビット）を開始します。

備考 本 API 関数の呼び出しは、タイマ・アレイ・ユニットを 8 ビット・タイマとして使用している場合に限られます。

### [指定形式]

```
void R_TAUm_Channeln_Higher8bits_Start ( void );
```

備考  $m$  はユニット番号を、 $n$  はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_TAUm\_Channeln\_Lower8bits\_Start

チャンネル  $n$  のカウント（下位 8 ビット）を開始します。

- 備考 1. 本 API 関数の呼び出しは、タイマ・アレイ・ユニットを 8 ビット・タイマとして使用している場合に  
限られます。
- 備考 2. 本 API 関数を呼び出してからカウント処理を開始するまでの時間は、該当機能の種類（インターバル・  
タイマ、外部イベント・カウンタ、ディレイ・カウンタなど）により異なります。

### [指定形式]

```
void R_TAUm_Channeln_Lower8bits_Start ( void );
```

備考  $m$  はユニット番号を、 $n$  はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_TAUm\_Channeln\_Stop

チャンネル  $n$  のカウントを終了します。

### [指定形式]

```
void R_TAUm_Channeln_Stop ( void );
```

備考  $m$  はユニット番号を,  $n$  はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_TAUm\_Channeln\_Higher8bits\_Stop

チャンネル  $n$  のカウント（上位 8 ビット）を終了します。

備考 本 API 関数の呼び出しは、タイマ・アレイ・ユニットを 8 ビット・タイマとして使用している場合に  
限られます。

### [指定形式]

```
void R_TAUm_Channeln_Higher8bits_Stop ( void );
```

備考  $m$  はユニット番号を、 $n$  はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_TAUm\_Channeln\_Lower8bits\_Stop

チャンネル  $n$  のカウント（下位 8 ビット）を終了します。

備考 本 API 関数の呼び出しは、タイマ・アレイ・ユニットを 8 ビット・タイマとして使用している場合に  
限られます。

### [指定形式]

```
void R_TAUm_Channeln_Lower8bits_Stop ( void );
```

備考  $m$  はユニット番号を、 $n$  はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_TAUm\_Reset

タイマ・アレイ・ユニットをリセットします。

### [指定形式]

```
void R_TAUm_Reset ( void );
```

備考 *m*は、ユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_TAUm\_Set\_PowerOff

タイマ・アレイ・ユニットに対するクロック供給を停止します。

備考 本API関数の呼び出しにより、タイマ・アレイ・ユニットはリセット状態へと移行します。  
このため、本API関数の呼び出し後、制御レジスタへの書き込みは無視されます。

### [指定形式]

```
void R_TAUm_Set_PowerOff ( void );
```

備考 *m*は、ユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし



## R\_TAUm\_Channeln\_Get\_PulseWidth

Tl $m$ n 端子に対する入力信号（入力パルス）のパルス間隔、またはハイ/ロウ・レベルの測定幅を獲得します。

### [指定形式]

```
#include "r_cg_macrodriver.h"
void R_TAUm_Channeln_Get_PulseWidth ( uint32_t * const width );
```

備考  $m$ はユニット番号を、 $n$ はチャンネル番号を意味します。

### [引数]

| I/O | 引数                      | 説明                                |
|-----|-------------------------|-----------------------------------|
| ○   | uint32_t * const width; | 測定幅 (0x0 ~ 0x1FFFF) を格納する領域へのポインタ |

### [戻り値]

なし

## R\_TAUm\_Channeln\_Set\_SoftwareTriggerOn

ワンショット・パルス出力のためのトリガ（ソフトウェア・トリガ）を発生させます。

### [指定形式]

```
void R_TAUm_Channeln_Set_SoftwareTriggerOn ( void );
```

備考  $m$ はユニット番号を,  $n$ はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

### 3.2.6 タイマRJ

以下に、コード生成がタイマRJ用として出力するAPI関数の一覧を示します。

表 3.6 タイマRJ用API関数

| API関数名                                    | 機能概要                                 |
|---|--------------------------------------|
| <a href="#">R_TMR_RJn_Create</a>          | 16ビット・タイマRJnを制御するうえで必要となる初期化処理を行います。 |
| <a href="#">R_TMR_RJn_Create_UserInit</a> | 16ビット・タイマRJnに関するユーザ独自の初期化処理を行います。    |
| <a href="#">r_tmr_rjn_interrupt</a>       | タイマ割り込みの発生に伴う処理を行います。                |
| <a href="#">R_TMR_RJn_Start</a>           | 16ビット・タイマRJnのカウンタ処理を開始します。           |
| <a href="#">R_TMR_RJn_Stop</a>            | 16ビット・タイマRJnのカウンタ処理を終了します。           |
| <a href="#">R_TMR_RJn_Set_PowerOff</a>    | 16ビット・タイマRJnに対するクロック供給を停止します。        |
| <a href="#">R_TMR_RJn_Get_PulseWidth</a>  | 16ビット・タイマRJnのパルス幅を読み出します。            |
| <a href="#">R_TMRJn_Create</a>            | 16ビット・タイマRJnを制御するうえで必要となる初期化処理を行います。 |
| <a href="#">R_TMRJn_Create_UserInit</a>   | 16ビット・タイマRJnに関するユーザ独自の初期化処理を行います。    |
| <a href="#">r_tmrjn_interrupt</a>         | タイマ割り込みの発生に伴う処理を行います。                |
| <a href="#">R_TMRJn_Start</a>             | 16ビット・タイマRJnのカウンタ処理を開始します。           |
| <a href="#">R_TMRJn_Stop</a>              | 16ビット・タイマRJnのカウンタ処理を終了します。           |
| <a href="#">R_TMRJn_Set_PowerOff</a>      | 16ビット・タイマRJnに対するクロック供給を停止します。        |
| <a href="#">R_TMRJn_Get_PulseWidth</a>    | 16ビット・タイマRJnのパルス幅を読み出します。            |

## R\_TMR\_RJn\_Create

16ビット・タイマRJnを制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_TMR_RJn_Create ( void );
```

備考 *n*は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_TMR\_RJn\_Create\_UserInit

16ビット・タイマRJnに関するユーザ独自の初期化処理を行います。

備考 本API関数は、[R\\_TMR\\_RJn\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_TMR_RJn_Create_UserInit ( void );
```

備考 *n*は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## r\_tmr\_rjn\_interrupt

タイマ割り込みの発生に伴う処理を行います。

備考 本 API 関数は、タイマ割り込みに対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_tmr_rjn_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_tmr_rjn_interrupt ( void );
```

備考  $n$  は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

**R\_TMR\_RJn\_Start**

16ビット・タイマRJnのカウント処理を開始します。

**[指定形式]**

```
void R_TMR_RJn_Start ( void );
```

備考 *n*は、チャンネル番号を意味します。

**[引数]**

なし

**[戻り値]**

なし

## R\_TMR\_RJn\_Stop

16 ビット・タイマ RJn のカウント処理を終了します。

### [指定形式]

```
void R_TMR_RJn_Stop ( void );
```

備考  $n$  は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし



## R\_TMR\_RJn\_Set\_PowerOff

16 ビット・タイマ RJn に対するクロック供給を停止します。

備考 本 API 関数の呼び出しにより、16 ビット・タイマ RJn はリセット状態へと移行します。  
このため、本 API 関数の呼び出し後、制御レジスタへの書き込みは無視されます。

### [指定形式]

```
void R_TMR_RJn_Set_PowerOff ( void );
```

備考 *n* は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_TMR\_RJn\_Get\_PulseWidth

16 ビット・タイマ RJn のパルス幅を読み出します。

- 備考 1. 本 API 関数の呼び出しは、16 ビット・タイマ RJn をパルス幅測定モード／パルス周期測定モードで使用している場合に限られます。
- 備考 2. パルス幅の計測中にオーバフロー（2 回以上）が発生した場合、正常なパルス幅を読み出すことはできません。

### [指定形式]

```
#include "r_cg_macrodriver.h"
void R_TMR_RJn_Get_PulseWidth ( uint32_t * const active_width );
```

備考 *n* は、チャンネル番号を意味します。

### [引数]

| I/O | 引数                             | 説明                                      |
|-----|--------------------------------|---|
| O   | uint32_t * const active_width; | TRJ0IO 端子から読み出したアクティブ・レベル幅を格納する領域へのポインタ |

### [戻り値]

なし

## R\_TMRJn\_Create

16ビット・タイマ R $n$ を制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_TMRJn_Create ( void );
```

備考  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_TMRJn\_Create\_UserInit

16ビット・タイマ R $n$ に関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_TMRJn\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_TMRJn_Create_UserInit ( void );
```

備考  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## r\_tmrjn\_interrupt

タイマ割り込みの発生に伴う処理を行います。

備考 本API関数は、タイマ割り込みに対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_tmrjn_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_tmrjn_interrupt ( void );
```

備考 *n*は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_TMRJn\_Start

16 ビット・タイマ R $Jn$  のカウント処理を開始します。

### [指定形式]

```
void R_TMRJn_Start ( void );
```

備考  $n$  は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_TMRJn\_Stop

16ビット・タイマ R $J_n$ のカウント処理を終了します。

### [指定形式]

```
void R_TMRJn_Stop ( void );
```

備考  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_TMRJn\_Set\_PowerOff

16 ビット・タイマ R $Jn$ に対するクロック供給を停止します。

備考 本 API 関数の呼び出しにより、16 ビット・タイマ R $Jn$  はリセット状態へと移行します。  
このため、本 API 関数の呼び出し後、制御レジスタへの書き込みは無視されます。

### [指定形式]

```
void R_TMRJn_Set_PowerOff ( void );
```

備考  $n$  は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし



## R\_TMRJn\_Get\_PulseWidth

16 ビット・タイマ R $J_n$  のパルス幅を読み出します。

備考 1. 本 API 関数の呼び出しは、16 ビット・タイマ R $J_n$  をパルス幅測定モード／パルス周期測定モードで使用している場合に限られます。

備考 2. パルス幅の計測中にオーバフロー（2 回以上）が発生した場合、正常なパルス幅を読み出すことはできません。

### [指定形式]

```
#include "r_cg_macrodriver.h"
void R_TMRJn_Get_PulseWidth ( uint32_t * const active_width );
```

備考  $n$  は、チャンネル番号を意味します。

### [引数]

| I/O | 引数                             | 説明                                      |
|-----|--------------------------------|---|
| O   | uint32_t * const active_width; | TRJ0IO 端子から読み出したアクティブ・レベル幅を格納する領域へのポインタ |

### [戻り値]

なし

### 3.2.7 タイマ RD

以下に、コード生成がタイマ RD 用として出力する API 関数の一覧を示します。

表 3.7 タイマ RD 用 API 関数

| API 関数名                      | 機能概要                                       |
|------------------------------|--|
| R_TMR_RDn_Create             | 16 ビット・タイマ RD $n$ を制御するうえで必要となる初期化処理を行います。 |
| R_TMR_RDn_Create_UserInit    | 16 ビット・タイマ RD $n$ に関するユーザ独自の初期化処理を行います。    |
| r_tmr_rdn_interrupt          | タイマ割り込みの発生に伴う処理を行います。                      |
| R_TMR_RDn_Start              | 16 ビット・タイマ RD $n$ のカウント処理を開始します。           |
| R_TMR_RDn_Stop               | 16 ビット・タイマ RD $n$ のカウント処理を終了します。           |
| R_TMR_RDn_Set_PowerOff       | 16 ビット・タイマ RD $n$ に対するクロック供給を停止します。        |
| R_TMR_RDn_ForcedOutput_Start | 16 ビット・タイマ RD $n$ のパルス出力強制遮断処理を開始します。      |
| R_TMR_RDn_ForcedOutput_Stop  | 16 ビット・タイマ RD $n$ のパルス出力強制遮断処理を終了します。      |
| R_TMR_RDn_Get_PulseWidth     | 16 ビット・タイマ RD $n$ のパルス幅を読み出します。            |
| R_TMRDn_Create               | 16 ビット・タイマ RD $n$ を制御するうえで必要となる初期化処理を行います。 |
| R_TMRDn_Create_UserInit      | 16 ビット・タイマ RD $n$ に関するユーザ独自の初期化処理を行います。    |
| r_tmrdn_interrupt            | タイマ割り込みの発生に伴う処理を行います。                      |
| R_TMRDn_Start                | 16 ビット・タイマ RD $n$ のカウント処理を開始します。           |
| R_TMRDn_Stop                 | 16 ビット・タイマ RD $n$ のカウント処理を終了します。           |
| R_TMRDn_Set_PowerOff         | 16 ビット・タイマ RD $n$ に対するクロック供給を停止します。        |
| R_TMRDn_ForcedOutput_Start   | 16 ビット・タイマ RD $n$ のパルス出力強制遮断処理を開始します。      |
| R_TMRDn_ForcedOutput_Stop    | 16 ビット・タイマ RD $n$ のパルス出力強制遮断処理を終了します。      |
| R_TMRDn_Get_PulseWidth       | 16 ビット・タイマ RD $n$ のパルス幅を読み出します。            |
| R_TMRD_Set_PowerOff          | 16 ビット・タイマ RD に対するクロック供給を停止します。            |

## R\_TMR\_RDn\_Create

16ビット・タイマ RD $n$ を制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_TMR_RDn_Create ( void );
```

備考  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_TMR\_RDn\_Create\_UserInit

16ビット・タイマ RD $n$ に関するユーザ独自の初期化処理を行います。

備考 本API関数は、[R\\_TMR\\_RDn\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_TMR_RDn_Create_UserInit ( void );
```

備考  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## r\_tmr\_rdn\_interrupt

タイマ割り込みの発生に伴う処理を行います。

備考 本 API 関数は、タイマ割り込みに対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_tmr_rdn_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_tmr_rdn_interrupt ( void );
```

備考 *n* はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_TMR\_RDn\_Start

16ビット・タイマ RDn のカウント処理を開始します。

### [指定形式]

```
void R_TMR_RDn_Start ( void );
```

備考 *n* は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_TMR\_RDn\_Stop

16ビット・タイマ RD $n$ のカウンタ処理を終了します。

### [指定形式]

```
void R_TMR_RDn_Stop ( void );
```

備考  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_TMR\_RDn\_Set\_PowerOff

16 ビット・タイマ RD $n$ に対するクロック供給を停止します。

備考 本 API 関数の呼び出しにより、16 ビット・タイマ RD $n$  はリセット状態へと移行します。  
このため、本 API 関数の呼び出し後、制御レジスタへの書き込みは無視されます。

### [指定形式]

```
void R_TMR_RDn_Set_PowerOff ( void );
```

備考  $n$  は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし



## R\_TMR\_RDn\_ForcedOutput\_Start

16ビット・タイマ RD $n$ のパルス出力強制遮断処理を開始します。

### [指定形式]

```
void R_TMR_RDn_ForcedOutput_Start ( void );
```

備考  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_TMR\_RDn\_ForcedOutput\_Stop

16 ビット・タイマ RD $n$  のパルス出力強制遮断処理を終了します。

備考 本 API 関数の呼び出しは、16 ビット・タイマ RD $n$  がカウント停止状態（タイマ RD スタート・レジスタ（TRDSTR）の TSTART ビットが 0）の場合に限られます。

### [指定形式]

```
void R_TMR_RDn_ForcedOutput_Stop ( void );
```

備考  $n$  は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_TMR\_RDn\_Get\_PulseWidth

16 ビット・タイマ RD $n$ のパルス幅を読み出します。

- 備考 1. 本 API 関数の呼び出しは、16 ビット・タイマ RD $n$ をインプット・キャプチャ機能で使用している場合に限られます。
- 備考 2. パルス幅の計測中にオーバフロー（2 回以上）が発生した場合、正常なパルス幅を読み出すことはできません。

### [指定形式]

```
#include "r_cg_macrodriver.h"
#include "r_cg_timer.h"
MD_STATUS R_TMR_RDn_Get_PulseWidth ( uint32_t * const active_width, uint32_t * const
inactive_width, timer_channel_t channel );
```

備考  $n$ は、チャンネル番号を意味します。

### [引数]

| I/O | 引数                                  | 説明   |
|-----|-------------------------------------|--|
| O   | uint32_t<br>* const active_width;   | 読み出したアクティブ・レベル幅を格納する領域へのポインタ   |
| O   | uint32_t<br>* const inactive_width; | 読み出したインアクティブ・レベル幅を格納する領域へのポインタ   |
| I   | timer_channel_t channel;            | 読み出し対象端子<br>TMCHANNELA : TRDIOA $n$ 端子<br>TMCHANNELB : TRDIOB $n$ 端子<br>TMCHANNELC : TRDIOC $n$ 端子<br>TMCHANNELD : TRDIOD $n$ 端子 |

### [戻り値]

| マクロ   | 説明   |
|-------|------|
| MD_OK | 正常終了 |

## R\_TMRDn\_Create

16 ビット・タイマ RD $n$ を制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_TMRDn_Create ( void );
```

備考  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_TMRDn\_Create\_UserInit

16ビット・タイマ RD $n$ に関するユーザ独自の初期化処理を行います。

備考 本API関数は、[R\\_TMRDn\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_TMRDn_Create_UserInit ( void );
```

備考  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

**r\_tmr $d$ n\_interrupt**

タイマ割り込みの発生に伴う処理を行います。

備考 本 API 関数は、タイマ割り込みに対応した割り込み処理として呼び出されます。

**[指定形式]**

CA78K0R コンパイラの場合

```
__interrupt static void r_tmr $d$ n_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_tmr $d$ n_interrupt ( void );
```

備考  $n$  はチャンネル番号を意味します。

**[引数]**

なし

**[戻り値]**

なし

**R\_TMRDn\_Start**

16ビット・タイマ RD $n$ のカウンタ処理を開始します。

**[指定形式]**

```
void R_TMRDn_Start ( void );
```

備考  $n$ は、チャネル番号を意味します。

**[引数]**

なし

**[戻り値]**

なし

## R\_TMRDn\_Stop

16ビット・タイマ RD $n$ のカウンタ処理を終了します。

### [指定形式]

```
void R_TMRDn_Stop ( void );
```

備考  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし



## R\_TMRDn\_Set\_PowerOff

16 ビット・タイマ RD $n$ に対するクロック供給を停止します。

備考 本 API 関数の呼び出しにより、16 ビット・タイマ RD $n$  はリセット状態へと移行します。  
このため、本 API 関数の呼び出し後、制御レジスタへの書き込みは無視されます。

### [指定形式]

```
void R_TMRDn_Set_PowerOff ( void );
```

備考  $n$  は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_TMRDn\_ForcedOutput\_Start

16ビット・タイマ RD $n$ のパルス出力強制遮断処理を開始します。

### [指定形式]

```
void R_TMRDn_ForcedOutput_Start ( void );
```

備考  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_TMRDn\_ForcedOutput\_Stop

16 ビット・タイマ RD $n$ のパルス出力強制遮断処理を終了します。

備考 本 API 関数の呼び出しは、16 ビット・タイマ RD $n$ がカウント停止状態（タイマ RD スタート・レジスタ（TRDSTR）の TSTART ビットが 0）の場合に限られます。

### [指定形式]

```
void R_TMRDn_ForcedOutput_Stop ( void );
```

備考  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_TMRDn\_Get\_PulseWidth

16 ビット・タイマ RD $n$  のパルス幅を読み出します。

- 備考 1. 本 API 関数の呼び出しは、16 ビット・タイマ RD $n$  をインプット・キャプチャ機能で使用している場合に限られます。
- 備考 2. パルス幅の計測中にオーバフロー（2 回以上）が発生した場合、正常なパルス幅を読み出すことはできません。

### [指定形式]

```
#include "r_cg_macrodriver.h"
#include "r_cg_timer.h"
MD_STATUS R_TMRDn_Get_PulseWidth ( uint32_t * const active_width, uint32_t * const
inactive_width, timer_channel_t channel );
```

備考  $n$  は、チャンネル番号を意味します。

### [引数]

| I/O | 引数                                  | 説明   |
|-----|-------------------------------------|--|
| O   | uint32_t<br>* const active_width;   | 読み出したアクティブ・レベル幅を格納する領域へのポインタ   |
| O   | uint32_t<br>* const inactive_width; | 読み出したインアクティブ・レベル幅を格納する領域へのポインタ   |
| I   | timer_channel_t channel;            | 読み出し対象端子<br>TMCHANNELA : TRDIOAn 端子<br>TMCHANNELB : TRDIOBn 端子<br>TMCHANNELC : TRDIOcn 端子<br>TMCHANNELD : TRDIODn 端子 |

### [戻り値]

| マクロ   | 説明   |
|-------|------|
| MD_OK | 正常終了 |

## R\_TMRD\_Set\_PowerOff

16ビット・タイマ RD に対するクロック供給を停止します。

備考 本 API 関数の呼び出しにより、16ビット・タイマ RD はリセット状態へと移行します。  
このため、本 API 関数の呼び出し後、制御レジスタへの書き込みは無視されます。

### [指定形式]

```
void R_TMRD_Set_PowerOff ( void );
```

### [引数]

なし

### [戻り値]

なし

### 3.2.8 タイマ RG

以下に、コード生成がタイマ RG 用として出力する API 関数の一覧を示します。

表 3.8 タイマ RG 用 API 関数

| API 関数名                                   | 機能概要                                    |
|---|---|
| <a href="#">R_TMR_RG0_Create</a>          | 16 ビット・タイマ RG0 を制御するうえで必要となる初期化処理を行います。 |
| <a href="#">R_TMR_RG0_Create_UserInit</a> | 16 ビット・タイマ RG0 に関するユーザ独自の初期化処理を行います。    |
| <a href="#">r_tmr_rg0_interrupt</a>       | タイマ割り込みの発生に伴う処理を行います。                   |
| <a href="#">R_TMR_RG0_Start</a>           | 16 ビット・タイマ RG0 のカウント処理を開始します。           |
| <a href="#">R_TMR_RG0_Stop</a>            | 16 ビット・タイマ RG0 のカウント処理を終了します。           |
| <a href="#">R_TMR_RG0_Set_PowerOff</a>    | 16 ビット・タイマ RG0 に対するクロック供給を停止します。        |
| <a href="#">R_TMR_RG0_Get_PulseWidth</a>  | 16 ビット・タイマ RG0 のパルス幅を読み出します。            |

## R\_TMR\_RG0\_Create

16ビット・タイマ RG0 を制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_TMR_RG0_Create ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_TMR\_RG0\_Create\_UserInit

16ビット・タイマ RG0 に関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_TMR\\_RG0\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_TMR_RG0_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし



## r\_tmr\_rg0\_interrupt

タイマ割り込みの発生に伴う処理を行います。

備考 本 API 関数は、タイマ割り込みに対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_tmr_rg0_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_tmr_rg0_interrupt ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_TMR\_RG0\_Start

16ビット・タイマRG0のカウンタ処理を開始します。

### [指定形式]

```
void R_TMR_RG0_Start ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_TMR\_RG0\_Stop

16ビット・タイマ RG0 のカウント処理を終了します。

### [指定形式]

```
void R_TMR_RG0_Stop ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_TMR\_RG0\_Set\_PowerOff

16 ビット・タイマ RG0 に対するクロック供給を停止します。

備考           本 API 関数の呼び出しにより、16 ビット・タイマ RG0 はリセット状態へと移行します。  
このため、本 API 関数の呼び出し後、制御レジスタへの書き込みは無視されます。

### [指定形式]

```
void   R_TMR_RG0_Set_PowerOff ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_TMR\_RG0\_Get\_PulseWidth

16ビット・タイマ RG0 のパルス幅を読み出します。

備考 1. 本 API 関数の呼び出しは、16ビット・タイマ RG0 をインプット・キャプチャ機能で使用している場合に限られます。

備考 2. パルス幅の計測中にオーバフロー（2回以上）が発生した場合、正常なパルス幅を読み出すことはできません。

### [指定形式]

```
#include "r_cg_macrodriver.h"
#include "r_cg_timer.h"
MD_STATUS R_TMR_RG0_Get_PulseWidth ( uint32_t * const active_width, uint32_t * const
inactive_width, timer_channel_t channel );
```

### [引数]

| I/O | 引数  | 説明   |
|-----|---|--|
| ○   | uint32_t<br>* const <i>active_width</i> ;   | TRGIOA 端子から読み出したアクティブ・レベル幅を格納する領域へのポインタ                        |
| ○   | uint32_t<br>* const <i>inactive_width</i> ; | TRGIOA 端子から読み出したインアクティブ・レベル幅を格納する領域へのポインタ                      |
| l   | timer_channel_t <i>channel</i> ;            | 読み出し対象端子<br>TMCHANNELA : TRGIOA0 端子<br>TMCHANNELB : TRGIOB0 端子 |

### [戻り値]

| マクロ   | 説明   |
|-------|------|
| MD_OK | 正常終了 |

### 3.2.9 タイマ RX

以下に、コード生成がタイマ RX 用として出力する API 関数の一覧を示します。

表 3.9 タイマ RX 用 API 関数

| API 関数名                                | 機能概要                                       |
|--|--|
| <a href="#">R_TMRX_Create</a>          | 16 ビット・タイマ RX を制御するうえで必要となる初期化処理を行います。     |
| <a href="#">R_TMRX_Create_UserInit</a> | 16 ビット・タイマ RX に関するユーザ独自の初期化処理を行います。        |
| <a href="#">r_tmrx_interrupt</a>       | タイマ割り込みの発生に伴う処理を行います。                      |
| <a href="#">R_TMRX_Start</a>           | 16 ビット・タイマ RX のカウント処理を開始します。               |
| <a href="#">R_TMRX_Stop</a>            | 16 ビット・タイマ RX のカウント処理を終了します。               |
| <a href="#">R_TMRX_Set_PowerOff</a>    | 16 ビット・タイマ RX に対するクロック供給を停止します。            |
| <a href="#">R_TMRX_Get_BufferValue</a> | 16 ビット・タイマ RX の TRX レジスタのバッファ・レジスタを読み出します。 |

## R\_TMRX\_Create

16ビット・タイマRXを制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_TMRX_Create ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_TMRX\_Create\_UserInit

16 ビット・タイマ RX に関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_TMRX\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_TMRX_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし



## r\_tmrx\_interrupt

タイマ割り込みの発生に伴う処理を行います。

備考 本 API 関数は、タイマ割り込みに対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_tmrx_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_tmrx_interrupt ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_TMRX\_Start

16ビット・タイマRXのカウンタ処理を開始します。

### [指定形式]

```
void R_TMRX_Start ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_TMRX\_Stop

16ビット・タイマRXのカウンタ処理を終了します。

### [指定形式]

```
void R_TMRX_Stop ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_TMRX\_Set\_PowerOff

16 ビット・タイマ RX に対するクロック供給を停止します。

備考           本 API 関数の呼び出しにより、16 ビット・タイマ RX はリセット状態へと移行します。  
このため、本 API 関数の呼び出し後、制御レジスタへの書き込みは無視されます。

### [指定形式]

```
void   R_TMRX_Set_PowerOff ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_TMRX\_Get\_BufferValue

16ビット・タイマRXのTRXレジスタのバッファ・レジスタを読み出します。

### [指定形式]

```
#include "r_cg_macrodriver.h"
void R_TMRX_Get_BufferValue(uint32_t * const value)
```

### [引数]

| I/O | 引数                         | 説明                               |
|-----|----------------------------|----------------------------------|
| O   | uint32_t<br>* const value; | TRXレジスタのバッファ・レジスタの値を格納する領域へのポインタ |

### [戻り値]

なし

## 3.2.10 16 ビット・タイマ KB

以下に、コード生成が 16 ビット・タイマ KB 用として出力する API 関数の一覧を示します。

表 3.10 16 ビット・タイマ KB 用 API 関数

| API 関数名   | 機能概要                                   |
|---|--|
| <a href="#">R_TMR_KB_Create</a>                   | 16 ビット・タイマ KB を制御するうえで必要となる初期化処理を行います。 |
| <a href="#">R_TMR_KBm_Create_UserInit</a>         | 16 ビット・タイマ KB に関するユーザ独自の初期化処理を行います。    |
| <a href="#">r_tmr_kbm_interrupt</a>               | タイマ割り込みの発生に伴う処理を行います。                  |
| <a href="#">R_TMR_KBm_Start</a>                   | 16 ビット・タイマ KB のカウント処理を開始します。           |
| <a href="#">R_TMR_KBm_Stop</a>                    | 16 ビット・タイマ KB のカウント処理を終了します。           |
| <a href="#">R_TMR_KBm_Set_PowerOff</a>            | 16 ビット・タイマ KB に対するクロック供給を停止します。        |
| <a href="#">R_TMR_KBmn_ForcedOutput_Start</a>     | 強制出力停止機能に使用するトリガ信号の入力を許可します。           |
| <a href="#">R_TMR_KBmn_ForcedOutput_Stop</a>      | 強制出力停止機能に使用するトリガ信号の入力を禁止します。           |
| <a href="#">R_TMR_KBm_BatchOverwriteRequestOn</a> | コンペア・レジスタの一斉書き換えを許可します。                |
| <a href="#">R_TMR_KBm_ForcedOutput_mn_Start</a>   | 強制出力停止機能に使用するトリガ信号の入力を許可します。           |
| <a href="#">R_TMR_KBm_ForcedOutput_mn_Stop</a>    | 強制出力停止機能に使用するトリガ信号の入力を禁止します。           |
| <a href="#">R_TMR_KBm_Reset</a>                   | 16 ビット・タイマ KB をリセットします。                |

## R\_TMR\_KB\_Create

16ビット・タイマ KB を制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_TMR_KB_Create ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_TMR\_KBm\_Create\_UserInit

16 ビット・タイマ KB に関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_TMR\\_KB\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_TMR_KBm_Create_UserInit ( void );
```

備考 *m* は、ユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし



## r\_tmr\_kbm\_interrupt

タイマ割り込みの発生に伴う処理を行います。

備考 本 API 関数は、タイマ割り込みに対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_tmr_kbm_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_tmr_kbm_interrupt ( void );
```

備考 *m* は、ユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_TMR\_KBm\_Start

16 ビット・タイマ KB のカウント処理を開始します。

### [指定形式]

```
void R_TMR_KBm_Start ( void );
```

備考 *m* は、ユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_TMR\_KBm\_Stop

16ビット・タイマ KB のカウント処理を終了します。

### [指定形式]

```
void R_TMR_KBm_Stop ( void );
```

備考 *m* は、ユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_TMR\_KBm\_Set\_PowerOff

16 ビット・タイマ KB に対するクロック供給を停止します。

備考 本 API 関数の呼び出しにより、16 ビット・タイマ KB はリセット状態へと移行します。  
このため、本 API 関数の呼び出し後、制御レジスタへの書き込みは無視されます。

### [指定形式]

```
void R_TMR_KBm_Set_PowerOff ( void );
```

備考 *m* は、ユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_TMR\_KBmn\_ForcedOutput\_Start

強制出力停止機能に使用するトリガ信号の入力を許可します。

### [指定形式]

```
void R_TMR_KBmn_ForcedOutput_Start ( void );
```

備考  $m$ はユニット番号を,  $n$ はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_TMR\_KBmn\_ForcedOutput\_Stop

強制出力停止機能に使用するトリガ信号の入力を禁止します。

### [指定形式]

```
void R_TMR_KBmn_ForcedOutput_Stop ( void );
```

備考  $m$ はユニット番号を,  $n$ はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_TMR\_KBm\_BatchOverwriteRequestOn

コンペア・レジスタの一斉書き換えを許可します。

備考           コンペア・レジスタの内容を一斉に書き換えるタイミングは、本API関数を呼び出したのち、カウント値とコンペア・レジスタに設定された値が一致した際、または外部トリガが発生した際となります。

### [指定形式]

```
void   R_TMR_KBm_BatchOverwriteRequestOn ( void );
```

備考           *m* はユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_TMR\_KBm\_ForcedOutput\_mn\_Start

強制出力停止機能に使用するトリガ信号の入力を許可します。

### [指定形式]

```
void R_TMR_KBm_ForcedOutput_mn_Start ( void );
```

備考  $m$ はユニット番号を,  $n$ はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし



## R\_TMR\_KBm\_ForcedOutput\_mn\_Stop

強制出力停止機能に使用するトリガ信号の入力を禁止します。

### [指定形式]

```
void R_TMR_KBm_ForcedOutput_mn_Stop ( void );
```

備考  $m$ はユニット番号を,  $n$ はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_TMR\_KBm\_Reset

タイマ・アレイ・ユニットをリセットします。

### [指定形式]

```
void R_TMR_KBm_Reset ( void );
```

備考 *m*は、ユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし

### 3.2.11 16 ビット・タイマ KC0

以下に、コード生成が 16 ビット・タイマ KC0 用として出力する API 関数の一覧を示します。

表 3.11 16 ビット・タイマ KC0 用 API 関数

| API 関数名                                   | 機能概要                                    |
|---|---|
| <a href="#">R_TMR_KC0_Create</a>          | 16 ビット・タイマ KC0 を制御するうえで必要となる初期化処理を行います。 |
| <a href="#">R_TMR_KC0_Create_UserInit</a> | 16 ビット・タイマ KC0 に関するユーザ独自の初期化処理を行います。    |
| <a href="#">r_tmr_kc0_interrupt</a>       | タイマ割り込みの発生に伴う処理を行います。                   |
| <a href="#">R_TMR_KC0_Start</a>           | 16 ビット・タイマ KC0 のカウント処理を開始します。           |
| <a href="#">R_TMR_KC0_Stop</a>            | 16 ビット・タイマ KC0 のカウント処理を終了します。           |
| <a href="#">R_TMR_KC0_Set_PowerOff</a>    | 16 ビット・タイマ KC0 に対するクロック供給を停止します。        |

## R\_TMR\_KC0\_Create

16ビット・タイマ KC0 を制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_TMR_KC0_Create ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_TMR\_KC0\_Create\_UserInit

16ビット・タイマ KC0 に関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_TMR\\_KC0\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_TMR_KC0_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

## r\_tmr\_kc0\_interrupt

タイマ割り込みの発生に伴う処理を行います。

備考 本 API 関数は、タイマ割り込みに対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_tmr_kc0_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_tmr_kc0_interrupt ( void );
```

### [引数]

なし

### [戻り値]

なし

**R\_TMR\_KC0\_Start**

16ビット・タイマ KC0 のカウント処理を開始します。

**[指定形式]**

```
void R_TMR_KC0_Start ( void );
```

**[引数]**

なし

**[戻り値]**

なし

## R\_TMR\_KC0\_Stop

16ビット・タイマ KC0 のカウント処理を終了します。

### [指定形式]

```
void R_TMR_KC0_Stop ( void );
```

### [引数]

なし

### [戻り値]

なし



## R\_TMR\_KC0\_Set\_PowerOff

16 ビット・タイマ KC0 に対するクロック供給を停止します。

備考           本 API 関数の呼び出しにより、16 ビット・タイマ KC0 はリセット状態へと移行します。  
このため、本 API 関数の呼び出し後、制御レジスタへの書き込みは無視されます。

### [指定形式]

```
void R_TMR_KC0_Set_PowerOff ( void );
```

### [引数]

なし

### [戻り値]

なし

## 3.2.12 16ビット・タイマ KB2

以下に、コード生成が16ビット・タイマKB2用として出力するAPI関数の一覧を示します。

表 3.12 16ビット・タイマKB2用API関数

| API関数名  | 機能概要  |
|---|---|
| <a href="#">R_KB2m_Create</a>   | 16ビット・タイマKB2を制御するうえで必要となる初期化処理を行います。          |
| <a href="#">R_KB2m_Create_UserInit</a>                                      | 16ビット・タイマKB2に関するユーザ独自の初期化処理を行います。             |
| <a href="#">r_kb2m_interrupt</a>  | タイマ割り込みINTTKB2mの発生に伴う処理を行います。                 |
| <a href="#">R_KB2m_Start</a>  | 16ビット・タイマKB2のカウント処理を開始します。                    |
| <a href="#">R_KB2m_Stop</a>   | 16ビット・タイマKB2のカウント処理を終了します。                    |
| <a href="#">R_KB2m_Set_PowerOff</a>   | 16ビット・タイマKB2に対するクロック供給を停止します。                 |
| <a href="#">R_KB2m_Simultaneous_Start</a>                                   | 同時スタート&ストップ・モードを開始します。                        |
| <a href="#">R_KB2m_Simultaneous_Stop</a>                                    | 同時スタート&ストップ・モードを終了します。                        |
| <a href="#">R_KB2m_Synchronous_Start</a>                                    | タイマ・スタート&クリア・モードを開始します。                       |
| <a href="#">R_KB2m_Synchronous_Stop</a>                                     | タイマ・スタート&クリア・モードを終了します。                       |
| <a href="#">R_KB2m_TKBO<sub>n</sub>0_Forced_Output_Stop_Function1_Start</a> | タイマ出力TKBO <sub>n</sub> 0に対する強制出力停止機能1を開始します。  |
| <a href="#">R_KB2m_TKBO<sub>n</sub>0_Forced_Output_Stop_Function1_Stop</a>  | タイマ出力TKBO <sub>n</sub> 0に対する強制出力停止機能1を終了します。  |
| <a href="#">R_KB2m_TKBO<sub>n</sub>1_Forced_Output_Stop_Function1_Start</a> | タイマ出力TKBO <sub>n</sub> 1に対する強制出力停止機能2を開始します。  |
| <a href="#">R_KB2m_TKBO<sub>n</sub>1_Forced_Output_Stop_Function1_Stop</a>  | タイマ出力TKBO <sub>n</sub> 1に対する強制出力停止機能2を終了します。  |
| <a href="#">R_KB2m_TKBO<sub>n</sub>0_DitheringFunction_Start</a>            | タイマ出力TKBO <sub>n</sub> 0に対するディザリング機能を開始します。   |
| <a href="#">R_KB2m_TKBO<sub>n</sub>0_DitheringFunction_Stop</a>             | タイマ出力TKBO <sub>n</sub> 0に対するディザリング機能を終了します。   |
| <a href="#">R_KB2m_TKBO<sub>n</sub>1_DitheringFunction_Start</a>            | タイマ出力TKBO <sub>n</sub> 1に対するディザリング機能を開始します。   |
| <a href="#">R_KB2m_TKBO<sub>n</sub>1_DitheringFunction_Stop</a>             | タイマ出力TKBO <sub>n</sub> 1に対するディザリング機能を終了します。   |
| <a href="#">R_KB2m_TKBO<sub>n</sub>0_SmoothStartFunction_Start</a>          | タイマ出力TKBO <sub>n</sub> 0に対するソフト・スタート機能を開始します。 |
| <a href="#">R_KB2m_TKBO<sub>n</sub>0_SmoothStartFunction_Stop</a>           | タイマ出力TKBO <sub>n</sub> 0に対するソフト・スタート機能を終了します。 |
| <a href="#">R_KB2m_TKBO<sub>n</sub>1_SmoothStartFunction_Start</a>          | タイマ出力TKBO <sub>n</sub> 1に対するソフト・スタート機能を開始します。 |
| <a href="#">R_KB2m_TKBO<sub>n</sub>1_SmoothStartFunction_Stop</a>           | タイマ出力TKBO <sub>n</sub> 1に対するソフト・スタート機能を終了します。 |
| <a href="#">R_KB2m_Set_BatchOverwriteRequestOn</a>                          | コンペア・レジスタの一斉書き換えを許可します。                       |

## R\_KB2*m*\_Create

16ビット・タイマ KB2 を制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_KB2m_Create ( void );
```

備考 *m*は、ユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_KB2m\_Create\_UserInit

16ビット・タイマ KB2 に関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_KB2m\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_KB2m_Create_UserInit ( void );
```

備考 *m* は、ユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし

## r\_kb2m\_interrupt

タイマ割り込み INTTKB2*m*の発生に伴う処理を行います。

備考 本API関数は、タイマ割り込み INTTKB2*m*に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_kb2m_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_kb2m_interrupt ( void );
```

備考 *m*は、ユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_KB2*m*\_Start

16 ビット・タイマ KB2 のカウント処理を開始します。

### [指定形式]

```
void R_KB2m_Start ( void );
```

備考 *m*は、ユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_KB2*m*\_Stop

16ビット・タイマ KB2 のカウント処理を終了します。

### [指定形式]

```
void R_KB2m_Stop ( void );
```

備考 *m*は、ユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_KB2*m*\_Set\_PowerOff

16 ビット・タイマ KB2 に対するクロック供給を停止します。

### [指定形式]

```
void R_KB2m_Set_PowerOff ( void );
```

備考 *m*は、ユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし



## R\_KB2*m*\_Simultaneous\_Start

同時スタート & ストップ・モードを開始します。

### [指定形式]

```
void R_KB2m_Simultaneous_Start ( void );
```

備考 *m*は、ユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_KB2*m*\_Simultaneous\_Stop

同時スタート & ストップ・モードを終了します。

### [指定形式]

```
void R_KB2m_Simultaneous_Stop ( void );
```

備考 *m*は、ユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_KB2m\_Synchronous\_Start

タイマ・スタート & クリア・モードを開始します。

### [指定形式]

```
void R_KB2m_Synchronous_Start ( void );
```

備考 *m* は、ユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_KB2*m*\_Synchronous\_Stop

タイマ・スタート & クリア・モードを終了します。

### [指定形式]

```
void R_KB2m_Synchronous_Stop ( void );
```

備考 *m*は、ユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_KB2m\_TKBO*n*0\_Forced\_Output\_Stop\_Function1\_Start

タイマ出力 TKBO*n*0 に対する強制出力停止機能 1 を開始します。

### [指定形式]

```
void R_KB2m_TKBOn0_Forced_Output_Stop_Function1_Start ( void );
```

備考 *m* はユニット番号を, *n* はチャネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

**R\_KB2*m*\_TKB0*n*0\_Forced\_Output\_Stop\_Function1\_Stop**

タイマ出力 TKB0*n*0 に対する強制出力停止機能 1 を終了します。

**[指定形式]**

```
void R_KB2m_TKB0n0_Forced_Output_Stop_Function1_Stop ( void );
```

備考 *m*はユニット番号を, *n*はチャンネル番号を意味します。

**[引数]**

なし

**[戻り値]**

なし

**R\_KB2m\_TKBO $n$ 1\_Forced\_Output\_Stop\_Function1\_Start**

タイマ出力 TKBO $n$ 1 に対する強制出力停止機能 2 を開始します。

**[指定形式]**

```
void R_KB2m_TKBO $n$ 1_Forced_Output_Stop_Function1_Start ( void );
```

備考  $m$  はユニット番号を,  $n$  はチャネル番号を意味します。

**[引数]**

なし

**[戻り値]**

なし

**R\_KB2m\_TKBOn1\_Forced\_Output\_Stop\_Function1\_Stop**

タイマ出力 TKBOn1 に対する強制出力停止機能 2 を終了します。

**[指定形式]**

```
void R_KB2m_TKBOn1_Forced_Output_Stop_Function1_Stop ( void );
```

備考 *m* はユニット番号を, *n* はチャンネル番号を意味します。

**[引数]**

なし

**[戻り値]**

なし



## R\_KB2m\_TKBO*n*0\_DitheringFunction\_Start

タイマ出力 TKBO*n*0 に対するディザリング機能を開始します。

### [指定形式]

```
void R_KB2m_TKBOn0_DitheringFunction_Start ( void );
```

備考 *m*はユニット番号を, *n*はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_KB2*m*\_TKB0*n*0\_DitheringFunction\_Stop

タイマ出力 TKB0*n*0 に対するディザリング機能を終了します。

### [指定形式]

```
void R_KB2m_TKB0n0_DitheringFunction_Stop ( void );
```

備考 *m*はユニット番号を, *n*はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_KB2m\_TKBO*n*\_DitheringFunction\_Start

タイマ出力 TKBO*n* に対するディザリング機能を開始します。

### [指定形式]

```
void R_KB2m_TKBOn_DitheringFunction_Start ( void );
```

備考 *m*はユニット番号を, *n*はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_KB2*m*\_TKB0*n*1\_DitheringFunction\_Stop

タイマ出力 TKB0*n*1 に対するディザリング機能を終了します。

### [指定形式]

```
void R_KB2m_TKB0n1_DitheringFunction_Stop ( void );
```

備考 *m*はユニット番号を, *n*はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

### R\_KB2*m*\_TKB0*n*0\_SmoothStartFunction\_Start

タイマ出力 TKB0*n*0 に対するソフト・スタート機能を開始します。

#### [指定形式]

```
void R_KB2m_TKB0n0_SmoothStartFunction_Start ( void );
```

備考 *m*はユニット番号を, *n*はチャンネル番号を意味します。

#### [引数]

なし

#### [戻り値]

なし

### R\_KB2*m*\_TKB0*n*0\_SmoothStartFunction\_Stop

タイマ出力 TKB0*n*0 に対するソフト・スタート機能を終了します。

#### [指定形式]

```
void R_KB2m_TKB0n0_SmoothStartFunction_Stop ( void );
```

備考 *m*はユニット番号を, *n*はチャネル番号を意味します。

#### [引数]

なし

#### [戻り値]

なし

## R\_KB2*m*\_TKB0*n*1\_SmoothStartFunction\_Start

タイマ出力 TKB0*n*1 に対するソフト・スタート機能を開始します。

### [指定形式]

```
void R_KB2m_TKB0n1_SmoothStartFunction_Start ( void );
```

備考 *m*はユニット番号を, *n*はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

**R\_KB2*m*\_TKB0*n*1\_SmoothStartFunction\_Stop**

タイマ出力 TKB0*n*1 に対するソフト・スタート機能を終了します。

**[指定形式]**

```
void R_KB2m_TKB0n1_SmoothStartFunction_Stop ( void );
```

備考 *m*はユニット番号を, *n*はチャネル番号を意味します。

**[引数]**

なし

**[戻り値]**

なし



## R\_KB2m\_Set\_BatchOverwriteRequestOn

コンペア・レジスタの一斉書き換えを許可します。

備考           コンペア・レジスタの内容を一斉に書き換えるタイミングは、本API関数を呼び出したのち、カウント値とコンペア・レジスタに設定された値が一致した際、または外部トリガが発生した際となります。

### [指定形式]

```
void   R_KB2m_Set_BatchOverwriteRequestOn ( void );
```

備考           *m* はユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし

## 3.2.13 リアルタイム・クロック

以下に、コード生成がリアルタイム・クロック用として出力する API 関数の一覧を示します。

表 3.13 リアルタイム・クロック用 API 関数

| API 関数名                           | 機能概要   |
|-----------------------------------|--|
| R_RTC_Create                      | リアルタイム・クロックを制御するうえで必要となる初期化処理を行います。            |
| R_RTC_Create_UserInit             | リアルタイム・クロックに関するユーザ独自の初期化処理を行います。               |
| r_rtc_interrupt                   | リアルタイム・クロック割り込み INTRTC の発生に伴う処理を行います。          |
| R_RTC_Start                       | リアルタイム・クロック（年、月、曜日、日、時、分、秒）のカウントを開始します。        |
| R_RTC_Stop                        | リアルタイム・クロック（年、月、曜日、日、時、分、秒）のカウントを終了します。        |
| R_RTC_Set_PowerOff                | リアルタイム・クロックに対するクロック供給を停止します。                   |
| R_RTC_Set_HourSystem              | リアルタイム・クロックの時間制（12 時間制、24 時間制）を設定します。          |
| R_RTC_Set_CounterValue            | リアルタイム・クロックにカウント値を設定します。                       |
| R_RTC_Set_CalendarCounterValue    | リアルタイム・クロックにカウント値を設定します。（カレンダーモード設定時）          |
| R_RTC_Set_BinaryCounterValue      | リアルタイム・クロックにカウント値を設定します。（バイナリモード設定時）           |
| R_RTC_Get_CounterValue            | リアルタイム・クロックのカウント値を読み出します。                      |
| R_RTC_Get_CalendarCounterValue    | リアルタイム・クロックにカウント値を読み出します。（カレンダーモード設定時）         |
| R_RTC_Get_BinaryCounterValue      | リアルタイム・クロックにカウント値を読み出します。（バイナリモード設定時）          |
| R_RTC_Set_ConstPeriodInterruptOn  | 割り込み INTRTC の発生周期を設定したのち、定周期割り込み機能を開始します。      |
| R_RTC_Set_ConstPeriodInterruptOff | 定周期割り込み機能を終了します。                               |
| r_rtc_callback_constperiod        | 定周期割り込み INTRTC の発生に伴う処理を行います。                  |
| R_RTC_Set_AlarmOn                 | アラーム割り込み機能を開始します。                              |
| R_RTC_Set_CalendarAlarmOn         | アラーム割り込み機能を開始します。（カレンダーモード設定時）                 |
| R_RTC_Set_BinaryAlarmOn           | アラーム割り込み機能を開始します。（バイナリモード設定時）                  |
| R_RTC_Set_AlarmOff                | アラーム割り込み機能を終了します。                              |
| R_RTC_Set_AlarmValue              | アラームの発生条件（曜日、時、分）を設定します。                       |
| R_RTC_Set_CalenderAlarmValue      | アラームの発生条件（年、月、曜日、日、時、分、秒）を設定します。（カレンダーモード設定時）  |
| R_RTC_Set_BinaryAlarmValue        | アラームの発生条件を設定します。（バイナリモード設定時）                   |
| R_RTC_Get_AlarmValue              | アラームの発生条件（曜日、時、分）を読み出します。                      |
| R_RTC_Get_CalendarAlarmValue      | アラームの発生条件（年、月、曜日、日、時、分、秒）を読み出します。（カレンダーモード設定時） |

| API 関数名                                    | 機能概要                                  |
|--|---------------------------------------|
| <a href="#">R_RTC_Get_BinaryAlarmValue</a> | アラームの発生条件を読み出します。(バイナリモード設定時)         |
| <a href="#">r_rtc_callback_alarm</a>       | アラーム割り込み INTRTC の発生に伴う処理を行います。        |
| <a href="#">R_RTC_Set_RTC1HZOn</a>         | RTC1HZ 端子に対する補正クロック (1 Hz) の出力を許可します。 |
| <a href="#">R_RTC_Set_RTC1HZOff</a>        | RTC1HZ 端子に対する補正クロック (1 Hz) の出力を禁止します。 |
| <a href="#">R_RTC_Set_RTCOUTOn</a>         | RTCOUT の出力を許可します。                     |
| <a href="#">R_RTC_Set_RTCOUTOff</a>        | RTCOUT の出力を禁止します。                     |
| <a href="#">r_rtc_alarminerrupt</a>        | アラーム割り込み INTRTCALM の発生に伴う処理を行います。     |
| <a href="#">r_rtc_periodicinterrupt</a>    | 周期割り込み INTRTCPRD の発生に伴う処理を行います。       |
| <a href="#">r_rtc_callback_periodic</a>    | 定周期割り込み INTRTC の発生に伴う処理を行います。         |

## R\_RTC\_Create

リアルタイム・クロックを制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_RTC_Create ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_RTC\_Create\_UserInit

リアルタイム・クロックに関するユーザ独自の初期化処理を行います。

備考 本API関数は、[R\\_RTC\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_RTC_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

## r\_rtc\_interrupt

リアルタイム・クロック割り込み INTRTC の発生に伴う処理を行います。

備考 本 API 関数は、リアルタイム・クロック割り込み INTRTC に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_rtc_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_rtc_interrupt ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_RTC\_Start

リアルタイム・クロック（年，月，曜日，日，時，分，秒）のカウントを開始します。

### [指定形式]

```
void R_RTC_Start ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_RTC\_Stop

リアルタイム・クロック（年，月，曜日，日，時，分，秒）のカウントを終了します。

### [指定形式]

```
void R_RTC_Stop ( void );
```

### [引数]

なし

### [戻り値]

なし



## R\_RTC\_Set\_PowerOff

リアルタイム・クロックに対するクロック供給を停止します。

- 備考 1. 本 API 関数の呼び出しにより、リアルタイム・クロックはリセット状態へと移行します。このため、本 API 関数の呼び出し後、制御レジスタへの書き込みは無視されます。
- 備考 2. 本 API 関数では、周辺イネーブル・レジスタ *n* の RTCEN ビットを操作することにより、リアルタイム・クロックに対するクロック供給の停止を実現しています。このため、本 API 関数の呼び出しを行った際には、RTCEN ビットを共用している他の周辺装置（インターバル・タイマなど）に対するクロック供給も停止することになります。

### [指定形式]

```
void R_RTC_Set_PowerOff ( void );
```

### [引数]

なし

### [戻り値]

なし

**R\_RTC\_Set\_HourSystem**

リアルタイム・クロックの時間制（12時間制，24時間制）を設定します。

**[指定形式]**

```
#include "r_cg_macrodriver.h"
#include "r_cg_rtc.h"
MD_STATUS R_RTC_Set_HourSystem ( rtc_hour_system_t hour_system );
```

**[引数]**

| I/O | 引数  | 説明   |
|-----|---|--|
| I   | <code>rtc_hour_system_t hour_system;</code> | 時間制の種類<br>HOUR12 : 12時間制<br>HOUR24 : 24時間制 |

**[戻り値]**

| マクロ         | 説明                |
|-------------|-------------------|
| MD_OK       | 正常終了              |
| MD_BUSY1    | カウント処理を実行中（設定変更前） |
| MD_BUSY2    | カウント処理を停止中（設定変更後） |
| MD_ARGERROR | 引数の指定が不正          |

備考 MD\_BUSY1, または MD\_BUSY2 が返却される場合は, カウンタの動作が停止している, またはカウンタの動作開始待ち時間が短いことに起因している可能性があるため, ヘッダ・ファイル `r_cg_rtc.h` で定義されているマクロ `RTC_WAITTIME` の値を大きくしてください。

## R\_RTC\_Set\_CounterValue

リアルタイム・クロックにカウント値を設定します。

### [指定形式]

```
#include "r_cg_macrodriver.h"
#include "r_cg_rtc.h"
MD_STATUS R_RTC_Set_CounterValue ( rtc_counter_value_t counter_write_val );
```

### [引数]

| I/O | 引数  | 説明    |
|-----|---|-------|
| I   | rtc_counter_value_t<br>counter_write_val; | カウント値 |

備考 以下に、リアルタイム・クロックのカウント値 rtc\_counter\_value\_t の構成を示します。

```
typedef struct {
    uint8_t sec; /* 秒 */
    uint8_t min; /* 分 */
    uint8_t hour; /* 時 */
    uint8_t day; /* 日 */
    uint8_t week; /* 曜日 (0 : 日曜日, 6 : 土曜日) */
    uint8_t month; /* 月 */
    uint16_t year; /* 年 */
} rtc_counter_value_t;
```

### [戻り値]

| マクロ      | 説明                 |
|----------|--------------------|
| MD_OK    | 正常終了               |
| MD_BUSY1 | カウント処理を実行中 (設定変更前) |
| MD_BUSY2 | カウント処理を停止中 (設定変更後) |

備考 MD\_BUSY1、または MD\_BUSY2 が返却される場合は、カウンタの動作が停止している、またはカウンタの動作開始待ち時間が短いことに起因している可能性があるため、ヘッダ・ファイル r\_cg\_rtc.h で定義されているマクロ RTC\_WAITTIME の値を大きくしてください。

## R\_RTC\_Set\_CalendarCounterValue

リアルタイム・クロックにカウント値を設定します。(カレンダーモード設定時)

### [指定形式]

```
#include "r_cg_macrodriver.h"
#include "r_cg_rtc.h"
MD_STATUS R_RTC_Set_CalendarCounterValue ( rtc_counter_value_t counter_write_val );
```

### [引数]

| I/O | 引数  | 説明    |
|-----|---|-------|
| I   | rtc_counter_value_t<br>counter_write_val; | カウント値 |

備考 カウント値 rtc\_counter\_value\_t についての詳細は、[R\\_RTC\\_Set\\_CounterValue](#) を参照してください。

### [戻り値]

| マクロ      | 説明                 |
|----------|--------------------|
| MD_OK    | 正常終了               |
| MD_BUSY1 | カウント処理を実行中 (設定変更前) |

備考 MD\_BUSY1 が返却される場合は、カウンタの動作開始待ち時間が短いことに起因している可能性があるため、ヘッダ・ファイル r\_cg\_rtc.h で定義されているマクロ RTC\_WAITTIME の値を大きくしてください。

## R\_RTC\_Set\_BinaryCounterValue

リアルタイム・クロックにカウント値を設定します。(バイナリモード設定時)

### [指定形式]

```
#include "r_cg_macrodriver.h"
#include "r_cg_rtc.h"
MD_STATUS R_RTC_Set_BinaryCounterValue ( uint32_t counter_write_val );
```

### [引数]

| I/O | 引数                             | 説明    |
|-----|--------------------------------|-------|
| I   | uint32_t<br>counter_write_val; | カウント値 |

### [戻り値]

| マクロ      | 説明                 |
|----------|--------------------|
| MD_OK    | 正常終了               |
| MD_BUSY1 | カウント処理を実行中 (設定変更前) |

**備考** MD\_BUSY1 が返却される場合は、カウンタの動作開始待ち時間が短いことに起因している可能性があります。そのため、ヘッダ・ファイル r\_cg\_rtc.h で定義されているマクロ RTC\_WAITTIME の値を大きくしてください。

## R\_RTC\_Get\_CounterValue

リアルタイム・クロックのカウント値を読み出します。

### [指定形式]

```
#include "r_cg_macrodriver.h"
#include "r_cg_rtc.h"
MD_STATUS R_RTC_Get_CounterValue ( rtc_counter_value_t * const counter_read_val );
```

### [引数]

| I/O | 引数   | 説明                       |
|-----|--|--------------------------|
| ○   | rtc_counter_value_t<br>* const counter_read_val; | 読み出したカウント値を格納する構造体へのポインタ |

備考 カウント値 rtc\_counter\_value\_t についての詳細は、[R\\_RTC\\_Set\\_CounterValue](#) を参照してください。

### [戻り値]

| マクロ      | 説明                |
|----------|-------------------|
| MD_OK    | 正常終了              |
| MD_BUSY1 | カウント処理を実行中（読み出し前） |
| MD_BUSY2 | カウント処理を停止中（読み出し後） |

備考 MD\_BUSY1, または MD\_BUSY2 が返却される場合は、カウンタの動作が停止している、またはカウンタの動作開始待ち時間が短いことに起因している可能性があるため、ヘッダ・ファイル r\_cg\_rtc.h で定義されているマクロ RTC\_WAITTIME の値を大きくしてください。

## R\_RTC\_Get\_CalendarCounterValue

リアルタイム・クロックのカウンタ値を読み出します。(カレンダーモード設定時)

### [指定形式]

```
#include "r_cg_macrodriver.h"
#include "r_cg_rtc.h"
MD_STATUS R_RTC_Get_CalendarCounterValue ( rtc_counter_value_t * const
counter_read_val );
```

### [引数]

| I/O | 引数   | 説明                       |
|-----|--|--------------------------|
| ○   | rtc_counter_value_t<br>* const counter_read_val; | 読み出したカウンタ値を格納する構造体へのポインタ |

備考 カウンタ値 rtc\_counter\_value\_t についての詳細は、[R\\_RTC\\_Set\\_CounterValue](#) を参照してください。

### [戻り値]

| マクロ      | 説明     |
|----------|--------|
| MD_OK    | 正常終了   |
| MD_ERROR | 読み出し失敗 |

## R\_RTC\_Get\_BinaryCounterValue

リアルタイム・クロックのカウント値を読み出します。(バイナリモード設定時)

### [指定形式]

```
#include "r_cg_macrodriver.h"
#include "r_cg_rtc.h"
MD_STATUS R_RTC_Get_BinaryCounterValue ( uint32_t * const counter_read_val );
```

### [引数]

| I/O | 引数                                    | 説明                       |
|-----|---------------------------------------|--------------------------|
| ○   | uint32_t<br>* const counter_read_val; | 読み出したカウント値を格納する構造体へのポインタ |

### [戻り値]

| マクロ      | 説明     |
|----------|--------|
| MD_OK    | 正常終了   |
| MD_ERROR | 読み出し失敗 |



## R\_RTC\_Set\_ConstPeriodInterruptOn

割り込み INTRTC の発生周期を設定したのち、定周期割り込み機能を開始します。

### [指定形式]

```
#include "r_cg_macrodriver.h"
#include "r_cg_rtc.h"
MD_STATUS R_RTC_Set_ConstPeriodInterruptOn ( rtc_int_period_t period );
```

### [引数]

| I/O | 引数                               | 説明  |
|-----|----------------------------------|---|
| I   | rtc_int_period_t <i>period</i> ; | 割り込み INTRTC の発生周期<br>HALFSEC : 0.5 秒<br>ONESEC : 1 秒<br>ONEMIN : 1 分<br>ONEHOUR : 1 時間<br>ONEDAY : 1 日<br>ONEMONTH : 1 カ月 |

### [戻り値]

| マクロ         | 説明       |
|-------------|----------|
| MD_OK       | 正常終了     |
| MD_ARGERROR | 引数の指定が不正 |

## R\_RTC\_Set\_ConstPeriodInterruptOff

定周期割り込み機能を終了します。

### [指定形式]

```
void R_RTC_Set_ConstPeriodInterruptOff ( void );
```

### [引数]

なし

### [戻り値]

なし

## r\_rtc\_callback\_constperiod

定周期割り込み INTRTC の発生に伴う処理を行います。

備考 本 API 関数は、定周期割り込み INTRTC に対応した割り込み処理 [r\\_rtc\\_interrupt](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
static void r_rtc_callback_constperiod ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_RTC\_Set\_AlarmOn

アラーム割り込み機能を開始します。

### [指定形式]

```
void R_RTC_Set_AlarmOn ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_RTC\_Set\_CalendarAlarmOn

アラーム割り込み機能を開始します。(カレンダーモード設定時)

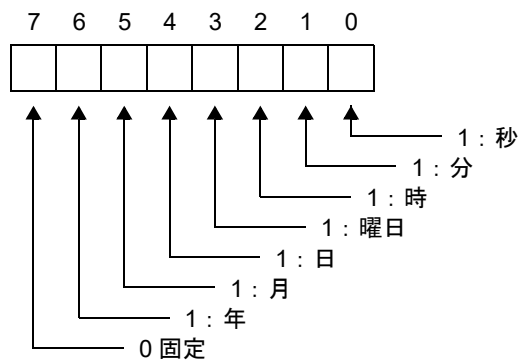
### [指定形式]

```
#include "r_cg_macrodriver.h"
#include "r_cg_rtc.h"
void R_RTC_Set_CalendarAlarmOn ( uint8_t enb_set );
```

### [引数]

| I/O | 引数               | 説明        |
|-----|------------------|-----------|
| I   | uint8_t enb_set; | アラームイネーブル |

備考 以下にアラームイネーブル set\_enb の各ビットに対する意味を示します。



### [戻り値]

なし

## R\_RTC\_Set\_BinaryAlarmOn

アラーム割り込み機能を開始します。(バイナリモード設定時)

### [指定形式]

```
void R_RTC_Set_BinaryAlarmOn ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_RTC\_Set\_AlarmOff

アラーム割り込み機能を終了します。

### [指定形式]

```
void R_RTC_Set_AlarmOff ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_RTC\_Set\_AlarmValue

アラームの発生条件（曜日，時，分）を設定します。

### [指定形式]

```
#include "r_cg_macrodriver.h"
#include "r_cg_rtc.h"
void R_RTC_Set_AlarmValue ( rtc_alarm_value_t alarm_val );
```

### [引数]

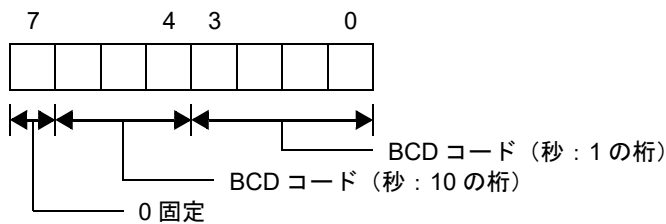
| I/O | 引数                           | 説明                |
|-----|------------------------------|-------------------|
| I   | rtc_alarm_value_t alarm_val; | アラームの発生条件（曜日，時，分） |

備考 以下に，アラームの発生条件 rtc\_alarm\_value\_t の構成を示します。（構成内容はデバイスによって異なります）

```
typedef struct {
    uint8_t alarmws; /* 秒 */
    uint8_t alarmwm; /* 分 */
    uint8_t alarmwh; /* 時 */
    uint8_t alarmww; /* 曜日 (0:日曜日, 6:土曜日) */
    uint8_t alarmwd; /* 日 */
    uint8_t alarmwmt; /* 月 */
    uint16_t alarmwyy; /* 年 */
} rtc_alarm_value_t;
```

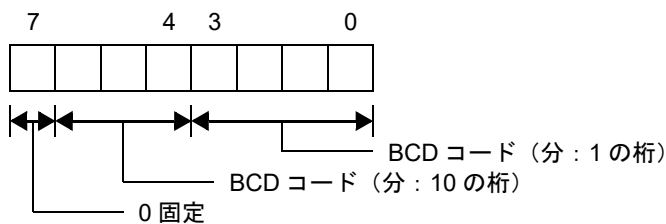
#### - alarmwm (秒)

以下に，構成メンバ alarmws の各ビットに対する意味を示します。



#### - alarmwm (分)

以下に，構成メンバ alarmwm の各ビットに対する意味を示します。



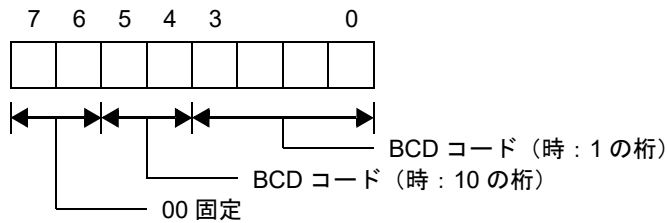


## - alarmwh (時)

以下に、構成メンバ alarmwh の各ビットに対する意味を示します。

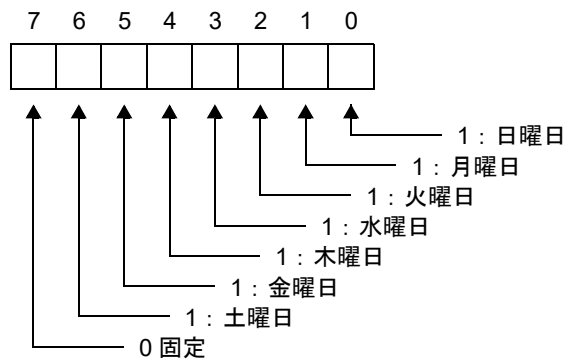
なお、ビット5は、リアルタイム・クロックが12時間制の場合、以下の意味となります。

0: 午前  
1: 午後



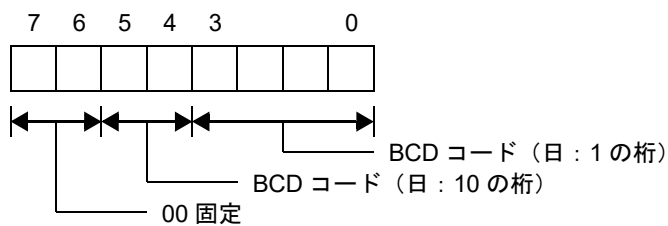
## - alarmww (曜日)

以下に、構成メンバ alarmww の各ビットに対する意味を示します。



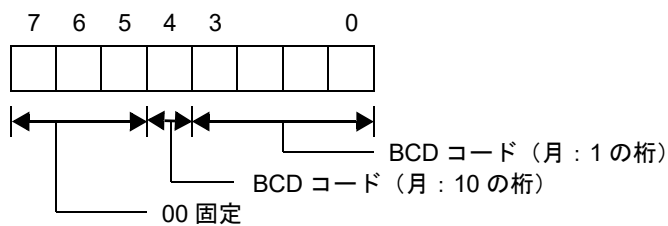
## - alarmwd (日)

以下に、構成メンバ alarmwd の各ビットに対する意味を示します。



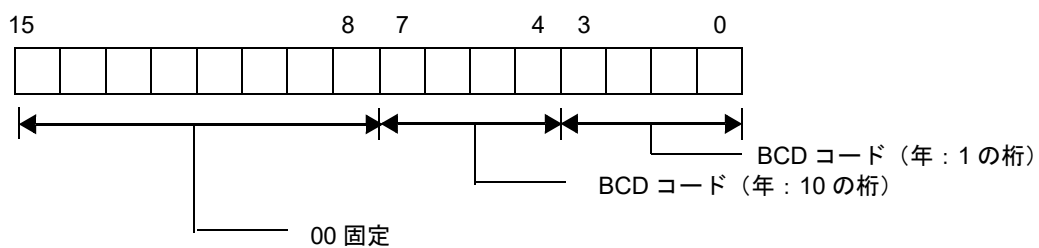
## - alarmwmn (月)

以下に、構成メンバ alarmwmn の各ビットに対する意味を示します。



- alarmwy (年)

以下に、構成メンバ alarmwy の各ビットに対する意味を示します。



[戻り値]

なし

## R\_RTC\_Set\_CalenderAlarmValue

アラームの発生条件（年，月，曜日，日，時，分，秒）を設定します。（カレンダーモード設定時）

### [指定形式]

```
#include "r_cg_macrodriver.h"
#include "r_cg_rtc.h"
void R_RTC_Set_CalenderAlarmValue ( rtc_alarm_value_t alarm_val );
```

### [引数]

| I/O | 引数                           | 説明                        |
|-----|------------------------------|---------------------------|
| I   | rtc_alarm_value_t alarm_val; | アラームの発生条件（年，月，曜日，日，時，分，秒） |

備考 アラームの発生条件 rtc\_alarm\_value\_t についての詳細は、[R\\_RTC\\_Set\\_AlarmValue](#) を参照してください。

### [戻り値]

なし

## R\_RTC\_Set\_BinaryAlarmValue

アラームの発生条件を設定します。(バイナリモード設定時)

### [指定形式]

```
#include "r_cg_macrodriver.h"
#include "r_cg_rtc.h"
void R_RTC_Set_BinaryAlarmValue ( uint32_t alarm_enable, uint32_t alarm_val );
```

### [引数]

| I/O | 引数                           | 説明  |
|-----|------------------------------|---|
| I   | uint32_t <i>alarm_enable</i> | アラームイネーブル<br>(バイナリカウンタアラーム許可レジスタへ値を設定します)       |
| I   | uint32_t <i>alarm_val</i>    | アラームの発生条件 (カウント値)<br>(バイナリカウンタアラームレジスタへ値を設定します) |

### [戻り値]

なし

## R\_RTC\_Get\_AlarmValue

アラームの発生条件（曜日，時，分）を読み出します。

### [指定形式]

```
#include "r_cg_macrodriver.h"
#include "r_cg_rtc.h"
void R_RTC_Get_AlarmValue ( rtc_alarm_value_t * const alarm_val );
```

備考 アラームの発生条件 `rtc_alarm_value_t` についての詳細は、[R\\_RTC\\_Set\\_AlarmValue](#) を参照してください。

### [引数]

| I/O | 引数  | 説明                      |
|-----|---|-------------------------|
| O   | <code>rtc_alarm_value_t</code><br><code>* const alarm_val;</code> | 読み出した発生条件を格納する構造体へのポインタ |

### [戻り値]

なし

## R\_RTC\_Get\_CalendarAlarmValue

アラームの発生条件（年，月，曜日，日，時，分，秒）を読み出します。（カレンダーモード設定時）

### [指定形式]

```
#include "r_cg_macrodriver.h"
#include "r_cg_rtc.h"
void R_RTC_Get_CalendarAlarmValue ( rtc_alarm_value_t * const alarm_val );
```

備考 アラームの発生条件 `rtc_alarm_value_t` についての詳細は、[R\\_RTC\\_Set\\_AlarmValue](#) を参照してください。

### [引数]

| I/O | 引数  | 説明                      |
|-----|---|-------------------------|
| O   | <code>rtc_alarm_value_t</code><br><code>* const alarm_val;</code> | 読み出した発生条件を格納する構造体へのポインタ |

### [戻り値]

なし

## R\_RTC\_Get\_BinaryAlarmValue

アラームの発生条件を読み出します。(バイナリモード設定時)

### [指定形式]

```
#include "r_cg_macrodriver.h"
#include "r_cg_rtc.h"
void R_RTC_Get_BinaryAlarmValue ( uint32_t * const alarm_enable, uint32_t * const
alarm_val );
```

### [引数]

| I/O | 引数                            | 説明                           |
|-----|-------------------------------|------------------------------|
| ○   | uint32_t * const alarm_enable | 読み出したアラームイネーブル値を格納する変数へのポインタ |
| ○   | uint32_t * const alarm_val    | 読み出した発生条件を格納する変数へのポインタ       |

### [戻り値]

なし

## r\_rtc\_callback\_alarm

アラーム割り込み INTRTC の発生に伴う処理を行います。

備考 本 API 関数は、アラーム割り込み INTRTC に対応した割り込み処理 [r\\_rtc\\_interrupt](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
static void r_rtc_callback_alarm ( void );
```

### [引数]

なし

### [戻り値]

なし



## R\_RTC\_Set\_RTC1HZOn

RTC1HZ 端子に対する補正クロック (1 Hz) の出力を許可します。

### [指定形式]

```
void R_RTC_Set_RTC1HZOn ( void );
```

### [引数]

なし

### [戻り値]

なし

**R\_RTC\_Set\_RTC1HZOff**

RTC1HZ 端子に対する補正クロック (1 Hz) の出力を禁止します。

**[指定形式]**

```
void R_RTC_Set_RTC1HZOff ( void );
```

**[引数]**

なし

**[戻り値]**

なし

## R\_RTC\_Set\_RTCOUTOn

RTCOUT の出力を許可します。

### [指定形式]

```
void R_RTC_Set_RTCOUTOn ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_RTC\_Set\_RTCOUTOff

RTCOUT の出力を禁止します。

### [指定形式]

```
void R_RTC_Set_RTCOUTOff ( void );
```

### [引数]

なし

### [戻り値]

なし

## r\_rtc\_alarminerrupt

アラーム割り込み INTRTCALM の発生に伴う処理を行います。

備考 本 API 関数は、アラーム割り込み INTRTCALM に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_rtc_alarminerrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_rtc_alarminerrupt ( void );
```

### [引数]

なし

### [戻り値]

なし

## r\_rtc\_periodicinterrupt

周期割り込み INTRTCPRD の発生に伴う処理を行います。

備考 本 API 関数は、周期割り込み INTRTCPRD に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_rtc_periodicinterrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_rtc_periodicinterrupt ( void );
```

### [引数]

なし

### [戻り値]

なし

## r\_rtc\_callback\_periodic

定周期割り込み INTRTC の発生に伴う処理を行います。

備考 本 API 関数は、定周期割り込み INTRTC に対応した割り込み処理 [r\\_rtc\\_interrupt](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
static void r_rtc_callback_periodic ( void );
```

### [引数]

なし

### [戻り値]

なし

### 3.2.14 サブシステム・クロック周波数測定回路

以下に、コード生成がサブシステム・クロック周波数測定回路用として出力する API 関数の一覧を示します。

表 3.14 サブシステム・クロック周波数測定回路用 API 関数

| API 関数名                               | 機能概要                                       |
|---------------------------------------|--|
| <a href="#">R_FMC_Create</a>          | サブシステム・クロック周波数測定回路を制御するうえで必要となる初期化処理を行います。 |
| <a href="#">R_FMC_Create_UserInit</a> | サブシステム・クロック周波数測定回路に関するユーザ独自の初期化処理を行います。    |
| <a href="#">r_fmc_interrupt</a>       | 周波数測定完了割り込み INTFM の発生に伴う処理を行います。           |
| <a href="#">R_FMC_Start</a>           | サブシステム・クロック周波数測定回路を利用した周波数の測定を開始します。       |
| <a href="#">R_FMC_Stop</a>            | サブシステム・クロック周波数測定回路を利用した周波数の測定を終了します。       |
| <a href="#">R_FMC_Set_PowerOff</a>    | サブシステム・クロック周波数測定回路に対するクロック供給を停止します。        |



## R\_FMC\_Create

サブシステム・クロック周波数測定回路を制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_FMC_Create ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_FMC\_Create\_UserInit

サブシステム・クロック周波数測定回路に関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_FMC\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_FMC_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

## r\_fmc\_interrupt

周波数測定完了割り込み INTFM の発生に伴う処理を行います。

備考 本 API 関数は、周波数測定完了割り込み INTFM に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_fmc_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_fmc_interrupt ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_FMC\_Start

サブシステム・クロック周波数測定回路を利用した周波数の測定を開始します。

### [指定形式]

```
void R_FMC_Start ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_FMC\_Stop

サブシステム・クロック周波数測定回路を利用した周波数の測定を終了します。

### [指定形式]

```
void R_FMC_Stop ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_FMC\_Set\_PowerOff

サブシステム・クロック周波数測定回路に対するクロック供給を停止します。

備考           本 API 関数の呼び出しにより、サブシステム・クロック周波数測定回路はリセット状態へと移行します。  
このため、本 API 関数の呼び出し後、制御レジスタへの書き込みは無視されます。

### [指定形式]

```
void   R_FMC_Set_PowerOff ( void );
```

### [引数]

なし

### [戻り値]

なし

### 3.2.15 12 ビット・インターバル・タイマ

以下に、コード生成が 12 ビット・インターバル・タイマ用として出力する API 関数の一覧を示します。

表 3.15 12 ビット・インターバル・タイマ用 API 関数

| API 関数名                              | 機能概要                                       |
|--------------------------------------|--|
| <a href="#">R_IT_Create</a>          | 12 ビット・インターバル・タイマを制御するうえで必要となる初期化処理を行います。  |
| <a href="#">R_IT_Create_UserInit</a> | 12 ビット・インターバル・タイマに関するユーザ独自の初期化処理を行います。     |
| <a href="#">r_it_interrupt</a>       | 12 ビット・インターバル・タイマ割り込み INTIT の発生に伴う処理を行います。 |
| <a href="#">R_IT_Start</a>           | 12 ビット・インターバル・タイマのカウントを開始します。              |
| <a href="#">R_IT_Stop</a>            | 12 ビット・インターバル・タイマのカウントを終了します。              |
| <a href="#">R_IT_Reset</a>           | 12 ビット・インターバル・タイマをリセットします。                 |
| <a href="#">R_IT_Set_PowerOff</a>    | 12 ビット・インターバル・タイマに対するクロック供給を停止します。         |

## R\_IT\_Create

12ビット・インターバル・タイマを制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_IT_Create ( void );
```

### [引数]

なし

### [戻り値]

なし



## R\_IT\_Create\_UserInit

12ビット・インターバル・タイマに関するユーザ独自の初期化処理を行います。

備考 本API関数は、[R\\_IT\\_Create](#)のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_IT_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

## r\_it\_interrupt

12ビット・インターバル・タイマ割り込み INTIT の発生に伴う処理を行います。

備考 本 API 関数は、12ビット・インターバル・タイマ割り込み INTIT に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_it_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_it_interrupt ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_IT\_Start

12ビット・インターバル・タイマのカウントを開始します。

### [指定形式]

```
void R_IT_Start ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_IT\_Stop

12ビット・インターバル・タイマのカウントを終了します。

### [指定形式]

```
void R_IT_Stop ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_IT\_Reset

12ビット・インターバル・タイマをリセットします。

### [指定形式]

```
void R_IT_Reset ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_IT\_Set\_PowerOff

12 ビット・インターバル・タイマに対するクロック供給を停止します。

- 備考 1. 本 API 関数の呼び出しにより、12 ビット・インターバル・タイマはリセット状態へと移行します。このため、本 API 関数の呼び出し後、制御レジスタへの書き込みは無視されます。
- 備考 2. 本 API 関数では、周辺イネーブル・レジスタ *n* の RTCEN ビットを操作することにより、12 ビット・インターバル・タイマに対するクロック供給の停止を実現しています。このため、本 API 関数の呼び出しを行った際には、RTCEN ビットを共用している他の周辺装置（リアルタイム・クロックなど）に対するクロック供給も停止することになります。

### [指定形式]

```
void R_IT_Set_PowerOff ( void );
```

### [引数]

なし

### [戻り値]

なし

### 3.2.16 8ビット・インターバル・タイマ

以下に、コード生成が8ビット・インターバル・タイマ用として出力するAPI関数の一覧を示します。

表 3.16 8ビット・インターバル・タイマ用API関数

| API関数名   | 機能概要  |
|--|---|
| <a href="#">R_IT8bitm_Channeln_Create</a>          | 8ビット・インターバル・タイマを制御するうえで必要となる初期化処理を行います。                         |
| <a href="#">R_IT8bitm_Channeln_Create_UserInit</a> | 8ビット・インターバル・タイマに関するユーザ独自の初期化処理を行います。                            |
| <a href="#">r_it8bitm_channeln_interrupt</a>       | 8ビット・インターバル・タイマ割り込み INTIT $n$ 0, または INTIT $n$ 1 の発生に伴う処理を行います。 |
| <a href="#">R_IT8bitm_Channeln_Start</a>           | 8ビット・インターバル・タイマのカウントを開始します。                                     |
| <a href="#">R_IT8bitm_Channeln_Stop</a>            | 8ビット・インターバル・タイマのカウントを終了します。                                     |
| <a href="#">R_IT8bitm_Channeln_Set_PowerOff</a>    | 8ビット・インターバル・タイマに対するクロック供給を停止します。                                |
| <a href="#">R_IT8bitm_Set_PowerOff</a>             | 8ビット・インターバル・タイマに対するクロック供給を停止します。                                |

## R\_IT8bitm\_Channeln\_Create

8ビット・インターバル・タイマを制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_IT8bitm_Channeln_Create ( void );
```

備考  $m$ はユニット番号を,  $n$ はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし



## R\_IT8bitm\_Channeln\_Create\_UserInit

8ビット・インターバル・タイマに関するユーザ独自の初期化処理を行います。

備考 本API関数は、[R\\_IT8bitm\\_Channeln\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_IT8bitm_Channeln_Create_UserInit ( void );
```

備考  $m$ はユニット番号を、 $n$ はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## r\_it8bitm\_channeln\_interrupt

8ビット・インターバル・タイマ割り込み INTIT $n$ 0, または INTIT $n$ 1 の発生に伴う処理を行います。

備考 本API関数は、8ビット・インターバル・タイマ割り込み INTIT $n$ 0, または INTIT $n$ 1 に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_it8bitm_channeln_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_it8bitm_channeln_interrupt ( void );
```

備考  $m$ はユニット番号を,  $n$ はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_IT8bit $m$ \_Channel $n$ \_Start

8ビット・インターバル・タイマのカウントを開始します。

### [指定形式]

```
void R_IT8bit $m$ _Channel $n$ _Start ( void );
```

備考  $m$ はユニット番号を,  $n$ はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_IT8bit $m$ \_Channel $n$ \_Stop

8ビット・インターバル・タイマのカウントを終了します。

### [指定形式]

```
void R_IT8bit $m$ _Channel $n$ _Stop ( void );
```

備考  $m$ はユニット番号を,  $n$ はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_IT8bitm\_Channeln\_Set\_PowerOff

8ビット・インターバル・タイマに対するクロック供給を停止します。

備考 本API関数の呼び出しにより、8ビット・インターバル・タイマはリセット状態へと移行します。  
このため、本API関数の呼び出し後、制御レジスタへの書き込みは無視されます。

### [指定形式]

```
void R_IT8bitm_Channeln_Set_PowerOff ( void );
```

備考  $m$ はユニット番号を、 $n$ はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_IT8bitm\_Set\_PowerOff

8ビット・インターバル・タイマに対するクロック供給を停止します。

備考 本API関数の呼び出しにより、8ビット・インターバル・タイマはリセット状態へと移行します。  
このため、本API関数の呼び出し後、制御レジスタへの書き込みは無視されます。

### [指定形式]

```
void R_IT8bitm_Set_PowerOff ( void );
```

備考 *m*はユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし

### 3.2.17 16 ビット・ウエイクアップ・タイマ

以下に、コード生成が 16 ビット・ウエイクアップ・タイマ用として出力する API 関数の一覧を示します。

表 3.17 16 ビット・ウエイクアップ・タイマ用 API 関数

| API 関数名                                | 機能概要                                       |
|--|--|
| <a href="#">R_WUTM_Create</a>          | 16 ビット・ウエイクアップ・タイマを制御するうえで必要となる初期化処理を行います。 |
| <a href="#">R_WUTM_Create_UserInit</a> | 16 ビット・ウエイクアップ・タイマに関するユーザ独自の初期化処理を行います。    |
| <a href="#">r_wutm_interrupt</a>       | タイマ割り込みの発生に伴う処理を行います。                      |
| <a href="#">R_WUTM_Start</a>           | 16 ビット・ウエイクアップ・タイマのカウント処理を開始します。           |
| <a href="#">R_WUTM_Stop</a>            | 16 ビット・ウエイクアップ・タイマのカウント処理を終了します。           |
| <a href="#">R_WUTM_Set_PowerOff</a>    | 16 ビット・ウエイクアップ・タイマに対するクロック供給を停止します。        |

## R\_WUTM\_Create

16ビット・ウエイクアップ・タイマを制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_WUTM_Create ( void );
```

### [引数]

なし

### [戻り値]

なし



## R\_WUTM\_Create\_UserInit

16ビット・ウェイクアップ・タイマに関するユーザ独自の初期化処理を行います。

備考 本API関数は、[R\\_WUTM\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_WUTM_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

## r\_wutm\_interrupt

タイマ割り込みの発生に伴う処理を行います。

備考 本 API 関数は、タイマ割り込みに対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_wutm_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_wutm_interrupt ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_WUTM\_Start

16ビット・ウエイクアップ・タイマのカウンタ処理を開始します。

### [指定形式]

```
void R_WUTM_Start ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_WUTM\_Stop

16ビット・ウエイクアップ・タイマのカウンタ処理を終了します。

### [指定形式]

```
void R_WUTM_Stop ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_WUTM\_Set\_PowerOff

16 ビット・ウエイクアップ・タイマに対するクロック供給を停止します。

備考 本 API 関数の呼び出しにより、16 ビット・ウエイクアップ・タイマはリセット状態へと移行します。  
このため、本 API 関数の呼び出し後、制御レジスタへの書き込みは無視されます。

### [指定形式]

```
void R_WUTM_Set_PowerOff ( void );
```

### [引数]

なし

### [戻り値]

なし

### 3.2.18 クロック出力／ブザー出力制御回路

以下に、コード生成がクロック出力／ブザー出力制御回路用として出力する API 関数の一覧を示します。

表 3.18 クロック出力／ブザー出力制御回路用 API 関数

| API 関数名                                   | 機能概要                                     |
|---|--|
| <a href="#">R_PCLBUZn_Create</a>          | クロック出力／ブザー出力制御回路を制御するうえで必要となる初期化処理を行います。 |
| <a href="#">R_PCLBUZn_Create_UserInit</a> | クロック出力／ブザー出力制御回路に関するユーザ独自の初期化処理を行います。    |
| <a href="#">R_PCLBUZn_Start</a>           | クロック出力／ブザー出力を開始します。                      |
| <a href="#">R_PCLBUZn_Stop</a>            | クロック出力／ブザー出力を停止します。                      |
| <a href="#">R_PCLBUZ_Set_PowerOff</a>     | クロック出力／ブザー出力制御回路に対するクロック供給を停止します。        |

## R\_PCLBUZn\_Create

クロック出力／ブザー出力制御回路を制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_PCLBUZn_Create ( void );
```

備考  $n$ は、出力端子を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_PCLBUZn\_Create\_UserInit

クロック出力／ブザー出力制御回路に関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_PCLBUZn\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_PCLBUZn_Create_UserInit ( void );
```

備考 *n*は、出力端子を意味します。

### [引数]

なし

### [戻り値]

なし



## R\_PCLBUZn\_Start

クロック出力／ブザー出力を開始します。

### [指定形式]

```
void R_PCLBUZn_Start ( void );
```

備考  $n$ は、出力端子を意味します。

### [引数]

なし

### [戻り値]

なし

**R\_PCLBUZn\_Stop**

クロック出力／ブザー出力を停止します。

**[指定形式]**

```
void R_PCLBUZn_Stop ( void );
```

備考  $n$ は、出力端子を意味します。

**[引数]**

なし

**[戻り値]**

なし

## R\_PCLBUZ\_Set\_PowerOff

クロック出力／ブザー出力制御回路に対するクロック供給を停止します。

備考 1. 本 API 関数の呼び出しにより、クロック出力／ブザー出力制御回路はリセット状態へと移行します。このため、本 API 関数の呼び出し後、制御レジスタへの書き込みは無視されます。

備考 2. 本 API 関数では、周辺イネーブル・レジスタ *n* の RTCEN ビットを操作することにより、クロック出力／ブザー出力制御回路に対するクロック供給の停止を実現しています。このため、本 API 関数の呼び出しを行った際には、RTCEN ビットを共用している他の周辺装置（リアルタイム・クロックなど）に対するクロック供給も停止することになります。

### [指定形式]

```
void R_PCLBUZ_Set_PowerOff ( void );
```

### [引数]

なし

### [戻り値]

なし

### 3.2.19 ウォッチドッグ・タイマ

以下に、コード生成がウォッチドッグ・タイマ用として出力する API 関数の一覧を示します。

表 3.19 ウォッチドッグ・タイマ用 API 関数

| API 関数名                               | 機能概要                                   |
|---------------------------------------|--|
| <a href="#">R_WDT_Create</a>          | ウォッチドッグ・タイマを制御するうえで必要となる初期化処理を行います。    |
| <a href="#">R_WDT_Create_UserInit</a> | ウォッチドッグ・タイマに関するユーザ独自の初期化処理を行います。       |
| <a href="#">r_wdt_interrupt</a>       | インターバル割り込み INTWDTI の発生に伴う処理を行います。      |
| <a href="#">R_WDT_Restart</a>         | ウォッチドッグ・タイマのカウンタをクリアしたのち、カウント処理を再開します。 |

## R\_WDT\_Create

ウォッチドッグ・タイマを制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_WDT_Create ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_WDT\_Create\_UserInit

ウォッチドッグ・タイマに関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_WDT\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_WDT_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

## r\_wdt\_interrupt

インターバル割り込み INTWDTI の発生に伴う処理を行います。

備考 本 API 関数は、インターバル割り込み INTWDTI に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_wdt_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_wdt_interrupt ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_WDT\_Restart

ウォッチドッグ・タイマのカウンタをクリアしたのち、カウント処理を再開します。

### [指定形式]

```
void R_WDT_Restart ( void );
```

### [引数]

なし

### [戻り値]

なし



### 3.2.20 プログラマブル・ゲイン計装アンプ付き 24 ビット $\Delta\Sigma$ /D コンバータ

以下に、コード生成がプログラマブル・ゲイン計装アンプ付き 24 ビット  $\Delta\Sigma$ /D コンバータ用として出力する API 関数の一覧を示します。

表 3.20 プログラマブル・ゲイン計装アンプ付き 24 ビット  $\Delta\Sigma$ /D コンバータ用 API 関数

| API 関数名   | 機能概要  |
|---|---|
| <a href="#">R_PGA_DSAD_Create</a>               | プログラマブル・ゲイン計装アンプ付き 24 ビット $\Delta\Sigma$ /D コンバータを制御するうえで必要となる初期化処理を行います。             |
| <a href="#">R_PGA_DSAD_Create_UserInit</a>      | プログラマブル・ゲイン計装アンプ付き 24 ビット $\Delta\Sigma$ /D コンバータに関するユーザ独自の初期化処理を行います。                |
| <a href="#">r_pga_dsad_interrupt_conversion</a> | 24 ビット $\Delta\Sigma$ /D コンバータ変換終了割り込み INTDSAD の発生に伴う処理を行います。                         |
| <a href="#">r_pga_dsad_interrupt_scan</a>       | 24 ビット $\Delta\Sigma$ /D コンバータスキャン完了割り込み INTDSADS の発生に伴う処理を行います。                      |
| <a href="#">R_PGA_DSAD_Start</a>                | A/D 変換を開始します。   |
| <a href="#">R_PGA_DSAD_Stop</a>                 | A/D 変換を終了します。   |
| <a href="#">R_PGA_DSAD_Set_PowerOff</a>         | プログラマブル・ゲイン計装アンプ付き 24 ビット $\Delta\Sigma$ /D コンバータに対するクロック供給を停止します。                    |
| <a href="#">R_PGA_DSAD_Get_AverageResult</a>    | A/D 変換結果の平均値を読み出します。  |
| <a href="#">R_PGA_DSAD_Get_Result</a>           | A/D 変換結果を読み出します。  |
| <a href="#">R_PGA_DSAD_CAMP_OffsetTrimming</a>  | コンフィギュラブル・アンプをプログラマブル・ゲイン計装アンプ付き 24 ビット $\Delta\Sigma$ /D コンバータへ接続し、オフセット・トリミングを行います。 |
| <a href="#">r_pga_dsad_conversion_interrupt</a> | 24 ビット $\Delta\Sigma$ /D コンバータ変換終了割り込み INTDSAD の発生に伴う処理を行います。                         |
| <a href="#">r_pga_dsad_scan_interrupt</a>       | 24 ビット $\Delta\Sigma$ /D コンバータスキャン完了割り込み INTDSADS の発生に伴う処理を行います。                      |

## R\_PGA\_DSAD\_Create

プログラマブル・ゲイン計装アンプ付き 24 ビット  $\Delta\Sigma$ /D コンバータを制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_PGA_DSAD_Create ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_PGA\_DSAD\_Create\_UserInit

プログラマブル・ゲイン計装アンプ付き 24 ビット  $\Delta\Sigma$ /D コンバータに関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_PGA\\_DSAD\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_PGA_DSAD_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

## r\_pga\_dsad\_interrupt\_conversion

24 ビット  $\Delta\Sigma/A/D$  コンバータ変換終了割り込み INTDSAD の発生に伴う処理を行います。

備考 本 API 関数は、24 ビット  $\Delta\Sigma/A/D$  コンバータ変換終了割り込み INTDSAD に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_pga_dsad_interrupt_conversion ( void );
```

CC-RL コンパイラの場合

```
static void __near r_pga_dsad_interrupt_conversion ( void );
```

### [引数]

なし

### [戻り値]

なし

## r\_pga\_dsad\_interrupt\_scan

24 ビット  $\Delta\Sigma$ /D コンバータスキャン完了割り込み INTDSADS の発生に伴う処理を行います。

備考 本 API 関数は、24 ビット  $\Delta\Sigma$ /D コンバータスキャン完了割り込み INTDSADS に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_pga_dsad_interrupt_scan ( void );
```

CC-RL コンパイラの場合

```
static void __near r_pga_dsad_interrupt_scan ( void );
```

### [引数]

なし

### [戻り値]

なし

**R\_PGA\_DSAD\_Start**

A/D 変換を開始します。

**[指定形式]**

```
void R_PGA_DSAD_Start ( void );
```

**[引数]**

なし

**[戻り値]**

なし

**R\_PGA\_DSAD\_Stop**

A/D 変換を終了します。

**[指定形式]**

```
void R_PGA_DSAD_Stop ( void );
```

**[引数]**

なし

**[戻り値]**

なし

**R\_PGA\_DSAD\_Set\_PowerOff**

プログラマブル・ゲイン計装アンプ付き 24 ビット  $\Delta\Sigma$ /D コンバータに対するクロック供給を停止します。

**[指定形式]**

```
void R_PGA_DSAD_Set_PowerOff ( void );
```

**[引数]**

なし

**[戻り値]**

なし



## R\_PGA\_DSAD\_Get\_AverageResult

A/D 変換結果の平均値を読み出します。

### [指定形式]

```
#include "r_cg_macrodriver.h"
void R_PGA_DSAD_Get_AverageResult ( uint16_t * const bufferH, uint16_t * const
bufferL );
```

### [引数]

| I/O | 引数                                | 説明  |
|-----|-----------------------------------|---|
| ○   | uint16_t * const <i>bufferH</i> ; | 読み出した A/D 変換結果 (DSADMVM レジスタと DSADMVH レジスタ) を格納する領域へのポインタ |
| ○   | uint16_t * const <i>bufferL</i> ; | 読み出した A/D 変換結果 (DSADMVC レジスタと DSADMVL レジスタ) を格納する領域へのポインタ |

### [戻り値]

なし

## R\_PGA\_DSAD\_Get\_Result

A/D 変換結果を読み出します。

### [指定形式]

```
#include "r_cg_macrodriver.h"
void R_PGA_DSAD_Get_Result ( uint16_t * const bufferH, uint16_t * const bufferL );
```

### [引数]

| I/O | 引数                        | 説明  |
|-----|---------------------------|---|
| O   | uint16_t * const bufferH; | 読み出した A/D 変換結果 (DSADCRM レジスタと DSADCRH レジスタ) を格納する領域へのポインタ |
| O   | uint16_t * const bufferL; | 読み出した A/D 変換結果 (DSADCRC レジスタと DSADCRL レジスタ) を格納する領域へのポインタ |

### [戻り値]

なし

## R\_PGA\_DSAD\_CAMP\_OffsetTrimming

コンフィギュラブル・アンプをプログラマブル・ゲイン計装アンプ付き 24 ビット  $\Delta\Sigma$ /D コンバータへ接続し、オフセット・トリミングを行います。

### [指定形式]

```
void R_PGA_DSAD_CAMP_OffsetTrimming ( void );
```

### [引数]

なし

### [戻り値]

なし

## r\_pga\_dsad\_conversion\_interrupt

24 ビット  $\Delta\Sigma$ /D コンバータ変換終了割り込み INTDSAD の発生に伴う処理を行います。

備考 本 API 関数は、24 ビット  $\Delta\Sigma$ /D コンバータ変換終了割り込み INTDSAD に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_pga_dsad_conversion_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_pga_dsad_conversion_interrupt ( void );
```

### [引数]

なし

### [戻り値]

なし

## r\_pga\_dsad\_scan\_interrupt

24 ビット  $\Delta\Sigma$ /D コンバータスキャン完了割り込み INTDSADS の発生に伴う処理を行います。

備考 本 API 関数は、24 ビット  $\Delta\Sigma$ /D コンバータスキャン完了割り込み INTDSADS に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_pga_dsad_scan_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_pga_dsad_scan_interrupt ( void );
```

### [引数]

なし

### [戻り値]

なし

## 3.2.21 A/D コンバータ

以下に、コード生成が A/D コンバータ用として出力する API 関数の一覧を示します。

表 3.21 A/D コンバータ用 API 関数

| API 関数名                | 機能概要                                       |
|------------------------|--|
| R_ADC_Create           | A/D コンバータを制御するうえで必要となる初期化処理を行います。          |
| R_ADC_Create_UserInit  | A/D コンバータに関するユーザ独自の初期化処理を行います。             |
| r_adc_interrupt        | A/D 変換終了割り込み INTAD の発生に伴う処理を行います。          |
| R_ADC_Set_OperationOn  | 電圧コンパレータを動作許可状態に設定します。                     |
| R_ADC_Set_OperationOff | 電圧コンパレータを動作停止状態に設定します。                     |
| R_ADC_Start            | A/D 変換を開始します。                              |
| R_ADC_Stop             | A/D 変換を終了します。                              |
| R_ADC_Reset            | A/D コンバータをリセットします。                         |
| R_ADC_Set_PowerOff     | A/D コンバータに対するクロック供給を停止します。                 |
| R_ADC_Set_ADChannel    | A/D 変換するアナログ電圧の入力端子を設定します。                 |
| R_ADC_Set_SnoozeOn     | STOP モードから SNOOZE モードへの切り替えを許可します。         |
| R_ADC_Set_SnoozeOff    | STOP モードから SNOOZE モードへの切り替えを禁止します。         |
| R_ADC_Set_TestChannel  | A/D コンバータの動作モードを設定します。                     |
| R_ADC_Get_Result       | A/D 変換結果（10 ビット）を読み出します。                   |
| R_ADC_Get_Result_8bit  | A/D 変換結果（8 ビット：10 ビット分解能の上位 8 ビット）を読み出します。 |

## R\_ADC\_Create

A/D コンバータを制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_ADC_Create ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_ADC\_Create\_UserInit

A/D コンバータに関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_ADC\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_ADC_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし



## r\_adc\_interrupt

A/D 変換終了割り込み INTAD の発生に伴う処理を行います。

備考 本 API 関数は、A/D 変換終了割り込み INTAD に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_adc_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_adc_interrupt ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_ADC\_Set\_OperationOn

電圧コンパレータを動作許可状態に設定します。

- 備考 1. 電圧コンパレータが動作停止状態から動作許可状態へと移行した際、約 1 $\mu$  秒の安定時間を必要とします。  
したがって、本 API 関数と [R\\_ADC\\_Start](#) の間には、約 1 $\mu$  秒の時間を空ける必要があります。
- 備考 2. [A/D コンバータ] の [コンパレータ動作設定] エリアで“許可”を選択した場合、電圧コンパレータは“常時 ON”となるため、本 API 関数の呼び出しは不要となります。

### [指定形式]

```
void R_ADC_Set_OperationOn ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_ADC\_Set\_OperationOff

電圧コンパレータを動作停止状態に設定します。

### [指定形式]

```
void R_ADC_Set_OperationOff ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_ADC\_Start

A/D 変換を開始します。

備考 電圧コンパレータが動作停止状態から動作許可状態へと移行した際、約 1 $\mu$  秒の安定時間を必要とします。  
したがって、[R\\_ADC\\_Set\\_OperationOn](#) と本 API 関数の間には、約 1 $\mu$  秒の時間を空ける必要があります。

### [指定形式]

```
void R_ADC_Start ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_ADC\_Stop

A/D 変換を終了します。

備考 電圧コンパレータは、本 API 関数の処理完了後も動作を継続しています。  
したがって、電圧コンパレータの動作を停止する場合は、本 API 関数の処理完了後、[R\\_ADC\\_Set\\_OperationOff](#) を呼び出す必要があります。

### [指定形式]

```
void R_ADC_Stop ( void );
```

### [引数]

なし

### [戻り値]

なし

**R\_ADC\_Reset**

ADコンバータをリセットします。

**[指定形式]**

```
void R_ADC_Reset ( void );
```

**[引数]**

なし

**[戻り値]**

なし

## R\_ADC\_Set\_PowerOff

A/D コンバータに対するクロック供給を停止します。

備考           本 API 関数の呼び出しにより、A/D コンバータはリセット状態へと移行します。  
このため、本 API 関数の呼び出し後、制御レジスタへの書き込みは無視されます。

### [指定形式]

```
void R_ADC_Set_PowerOff ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_ADC\_Set\_ADChannel

A/D 変換するアナログ電圧の入力端子を設定します。

備考 引数 *channel* に指定された値は、アナログ入力チャネル指定レジスタ (ADS) に設定されます。

### [指定形式]

```
#include "r_cg_macrodriver.h"
#include "r_cg_adc.h"
MD_STATUS R_ADC_Set_ADChannel ( ad_channel_t channel );
```

### [引数]

| I/O | 引数                            | 説明                                  |
|-----|-------------------------------|-------------------------------------|
| I   | ad_channel_t <i>channel</i> ; | アナログ電圧の入力端子<br>ADCHANNEL $n$ : 入力端子 |

備考 アナログ電圧の入力端子 ADCHANNEL $n$  についての詳細は、ヘッダ・ファイル r\_cg\_adc.h を参照してください。

### [戻り値]

| マクロ         | 説明       |
|-------------|----------|
| MD_OK       | 正常終了     |
| MD_ARGERROR | 引数の指定が不正 |



## R\_ADC\_Set\_SnoozeOn

STOP モードから SNOOZE モードへの切り替えを許可します。

### [指定形式]

```
void R_ADC_Set_SnoozeOn ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_ADC\_Set\_SnoozeOff

STOP モードから SNOOZE モードへの切り替えを禁止します。

### [指定形式]

```
void R_ADC_Set_SnoozeOff ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_ADC\_Set\_TestChannel

A/D コンバータの動作モードを設定します。

### [指定形式]

```
#include "r_cg_macrodriver.h"
#include "r_cg_adc.h"
MD_STATUS R_ADC_Set_TestChannel ( test_channel_t channel );
```

### [引数]

| I/O | 引数                      | 説明  |
|-----|-------------------------|---|
| I   | test_channel_t channel; | A/D コンバータの動作モード<br>ADNORMALINPUT : 通常モード (通常の A/D 変換)<br>ADAVREFM : テスト・モード (AVREFM 入力電圧)<br>ADAVREFP : テスト・モード (AVREFP 入力電圧) |

### [戻り値]

| マクロ         | 説明       |
|-------------|----------|
| MD_OK       | 正常終了     |
| MD_ARGERROR | 引数の指定が不正 |

## R\_ADC\_Get\_Result

A/D 変換結果（10ビット）を読み出します。

### [指定形式]

```
#include "r_cg_macrodriver.h"
void R_ADC_Get_Result ( uint16_t * const buffer );
```

### [引数]

| I/O | 引数                       | 説明                          |
|-----|--------------------------|-----------------------------|
| O   | uint16_t * const buffer; | 読み出した A/D 変換結果を格納する領域へのポインタ |

### [戻り値]

なし



### 3.2.22 コンフィギュラブル・アンプ

以下に、コード生成がコンフィギュラブル・アンプ用として出力する API 関数の一覧を示します。

表 3.22 コンフィギュラブル・アンプ用 API 関数

| API 関数名                                | 機能概要                                  |
|--|---------------------------------------|
| <a href="#">R_CAMP_Create</a>          | コンフィギュラブル・アンプを制御するうえで必要となる初期化処理を行います。 |
| <a href="#">R_CAMP_Create_UserInit</a> | コンフィギュラブル・アンプに関するユーザ独自の初期化処理を行います。    |
| <a href="#">R_CAMPn_Start</a>          | コンフィギュラブル・アンプ n (AMPn) の電源をオンにします。    |
| <a href="#">R_CAMPn_Stop</a>           | コンフィギュラブル・アンプ n (AMPn) の電源をオフにします。    |
| <a href="#">R_CAMP_Set_PowerOff</a>    | コンフィギュラブル・アンプに対するクロック供給を停止します。        |

## R\_CAMP\_Create

コンフィギュラブル・アンプを制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_CAMP_Create ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_CAMP\_Create\_UserInit

コンフィギュラブル・アンブに関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_CAMP\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_CAMP_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし



## R\_CAMP $n$ \_Start

コンフィギュラブル・アンプ  $n$  (AMP $n$ ) の電源をオンにします。

### [指定形式]

```
void R_CAMP $n$ _Start ( void );
```

備考  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_CAMPn\_Stop

コンフィギュラブル・アンプ *n* (AMP*n*) の電源をオフにします。

### [指定形式]

```
void R_CAMPn_Stop ( void );
```

備考 *n*は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_CAMP\_Set\_PowerOff

コンフィギュラブル・アンプに対するクロック供給を停止します。

### [指定形式]

```
void R_CAMP_Set_PowerOff ( void );
```

### [引数]

なし

### [戻り値]

なし

### 3.2.23 温度センサ

以下に、コード生成が温度センサ用として出力する API 関数の一覧を示します。

表 3.23 温度センサ用 API 関数

| API 関数名                                | 機能概要                          |
|--|-------------------------------|
| <a href="#">R_TMPS_Create</a>          | 温度センサを制御するうえで必要となる初期化処理を行います。 |
| <a href="#">R_TMPS_Create_UserInit</a> | 温度センサに関するユーザ独自の初期化処理を行います。    |
| <a href="#">R_TMPS_Start</a>           | 温度センサを利用した温度の計測を開始します。        |
| <a href="#">R_TMPS_Stop</a>            | 温度センサを利用した温度の計測を終了します。        |
| <a href="#">R_TMPS_Reset</a>           | 温度センサをリセットします。                |
| <a href="#">R_TMPS_Set_PowerOff</a>    | 温度センサに対するクロック供給を停止します。        |

## R\_TMPS\_Create

温度センサを制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_TMPS_Create ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_TMPS\_Create\_UserInit

温度センサに関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_TMPS\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_TMPS_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_TMPS\_Start

温度センサを利用した温度の計測を開始します。

### [指定形式]

```
void R_TMPS_Start ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_TMPS\_Stop

温度センサを利用した温度の計測を終了します。

### [指定形式]

```
void R_TMPS_Stop ( void );
```

### [引数]

なし

### [戻り値]

なし



## R\_TMPS\_Reset

温度センサをリセットします。

### [指定形式]

```
void R_TMPS_Reset ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_TMPS\_Set\_PowerOff

温度センサに対するクロック供給を停止します。

備考           本 API 関数の呼び出しにより、温度センサはリセット状態へと移行します。  
このため、本 API 関数の呼び出し後、制御レジスタへの書き込みは無視されます。

### [指定形式]

```
void   R_TMPS_Set_PowerOff ( void );
```

### [引数]

なし

### [戻り値]

なし

3.2.24 24 ビット  $\Delta\Sigma$ /D コンバータ

以下に、コード生成が 24 ビット  $\Delta\Sigma$ /D コンバータ用として出力する API 関数の一覧を示します。

表 3.24 24 ビット  $\Delta\Sigma$ /D コンバータ用 API 関数

| API 関数名   | 機能概要   |
|---|--|
| <a href="#">R_DSADC_Create</a>                    | 24 ビット $\Delta\Sigma$ /D コンバータを制御するうえで必要となる初期化処理を行います。 |
| <a href="#">R_DSADC_Create_UserInit</a>           | 24 ビット $\Delta\Sigma$ /D コンバータに関するユーザ独自の初期化処理を行います。    |
| <a href="#">r_dsadc_interrupt</a>                 | $\Delta\Sigma$ /D 変換終了割り込み INTDSAD の発生に伴う処理を行います。      |
| <a href="#">r_dsadzcn_interrupt</a>               | ゼロクロス検出割り込み INTDSADZCn の発生に伴う処理を行います。                  |
| <a href="#">R_DSADC_Set_OperationOn</a>           | 24 ビット $\Delta\Sigma$ /D コンバータを動作許可状態に設定します。           |
| <a href="#">R_DSADC_Set_OperationOff</a>          | 24 ビット $\Delta\Sigma$ /D コンバータを動作停止状態に設定します。           |
| <a href="#">R_DSADC_Start</a>                     | A/D 変換を開始します。  |
| <a href="#">R_DSADC_Stop</a>                      | A/D 変換を終了します。  |
| <a href="#">R_DSADC_Reset</a>                     | 24 ビット $\Delta\Sigma$ /D コンバータをリセットします。                |
| <a href="#">R_DSADC_Set_PowerOff</a>              | 24 ビット $\Delta\Sigma$ /D コンバータに対する電荷リセットを行います。         |
| <a href="#">R_DSADC_Channeln_Get_Result</a>       | A/D 変換結果 (24 ビット) を読み出します。                             |
| <a href="#">R_DSADC_Channeln_Get_Result_16bit</a> | A/D 変換結果 (16 ビット : 24 ビット分解能の上位 16 ビット) を読み出します。       |

## R\_DSADC\_Create

24 ビット  $\Delta\Sigma$ /D コンバータを制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_DSADC_Create ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_DSADC\_Create\_UserInit

24 ビット  $\Delta\Sigma$ /D コンバータに関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_DSADC\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_DSADC_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

## r\_dsadc\_interrupt

ΔΣA/D 変換終了割り込み INTDSAD の発生に伴う処理を行います。

備考 本 API 関数は、ΔΣA/D 変換終了割り込み INTDSAD に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_dsadc_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_dsadc_interrupt ( void );
```

### [引数]

なし

### [戻り値]

なし

## r\_dsadzcn\_interrupt

ゼロクロス検出割り込み INTDSADZCn の発生に伴う処理を行います。

備考 本 API 関数は、ゼロクロス検出割り込み INTDSADZCn に対応した割り込み処理として呼び出され  
ます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_dsadzcn_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_dsadzcn_interrupt ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_DSADC\_Set\_OperationOn

24 ビット  $\Delta\Sigma$ /D コンバータを動作許可状態に設定します。

### [指定形式]

```
void R_DSADC_Set_OperationOn ( void );
```

### [引数]

なし

### [戻り値]

なし



## R\_DSADC\_Set\_OperationOff

24 ビット  $\Delta\Sigma$ /D コンバータを動作停止状態に設定します。

### [指定形式]

```
void R_DSADC_Set_OperationOff ( void );
```

### [引数]

なし

### [戻り値]

なし

**R\_DSADC\_Start**

A/D 変換を開始します。

**[指定形式]**

```
void R_DSADC_Start ( void );
```

**[引数]**

なし

**[戻り値]**

なし

**R\_DSADC\_Stop**

A/D 変換を終了します。

**[指定形式]**

```
void R_DSADC_Stop ( void );
```

**[引数]**

なし

**[戻り値]**

なし

**R\_DSADC\_Reset**

24 ビット  $\Delta\Sigma$ /D コンバータをリセットします。

**[指定形式]**

```
void R_DSADC_Reset ( void );
```

**[引数]**

なし

**[戻り値]**

なし

## R\_DSADC\_Set\_PowerOff

24 ビット  $\Delta\Sigma$ /D コンバータに対する電荷リセットを行います。

備考            24 ビット  $\Delta\Sigma$ /D コンバータに対する電荷リセットを行った場合、約 1.4 $\mu$  秒の安定時間を必要とします。

### [指定形式]

```
void R_DSADC_Set_PowerOff ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_DSADC\_Channel*n*\_Get\_Result

A/D 変換結果 (24 ビット) を読み出します。

備考 本 API 関数による A/D 変換結果 (24 ビット) の読み出しは、 $\Delta\Sigma$ A/D 変換終了割り込み INTDSAD の発生から  $\Delta\Sigma$ A/D 変換結果レジスタ *n* の最大保留時間内に行う必要があります。

### [指定形式]

```
#include "r_cg_macrodriver.h"
void R_DSADC_Channeln_Get_Result ( uint32_t * const buffer );
```

備考 *n* は、チャンネル番号を意味します。

### [引数]

| I/O | 引数                               | 説明                          |
|-----|----------------------------------|-----------------------------|
| O   | uint32_t * const <i>buffer</i> ; | 読み出した A/D 変換結果を格納する領域へのポインタ |

### [戻り値]

なし

## R\_DSADC\_Channel*n*\_Get\_Result\_16bit

A/D 変換結果（16 ビット：24 ビット分解能の上位 16 ビット）を読み出します。

備考 本 API 関数による A/D 変換結果の読み出しは、 $\Delta\Sigma$ A/D 変換終了割り込み INTDSAD の発生から  $\Delta\Sigma$ A/D 変換結果レジスタ *n* の最大保留時間内に行う必要があります。

### [指定形式]

```
#include "r_cg_macrodriver.h"
void R_DSADC_Channeln_Get_Result_16bit ( uint16_t * const buffer );
```

備考 *n* は、チャンネル番号を意味します。

### [引数]

| I/O | 引数                               | 説明                          |
|-----|----------------------------------|-----------------------------|
| O   | uint16_t * const <i>buffer</i> ; | 読み出した A/D 変換結果を格納する領域へのポインタ |

### [戻り値]

なし

## 3.2.25 D/A コンバータ

以下に、コード生成がD/Aコンバータ用として出力するAPI関数の一覧を示します。

表 3.25 D/A コンバータ用 API 関数

| API 関数名   | 機能概要                              |
|---|-----------------------------------|
| <a href="#">R_DAC_Create</a>                    | D/A コンバータを制御するうえで必要となる初期化処理を行います。 |
| <a href="#">R_DAC_Create_UserInit</a>           | D/A コンバータに関するユーザ独自の初期化処理を行います。    |
| <a href="#">R_DACn_Start</a>                    | D/A 変換を開始します。                     |
| <a href="#">R_DACn_Stop</a>                     | D/A 変換を終了します。                     |
| <a href="#">R_DAC_Set_PowerOff</a>              | D/A コンバータに対するクロック供給を停止します。        |
| <a href="#">R_DACn_Set_ConversionValue</a>      | ANOn 端子に出力するアナログ電圧値を設定します。        |
| <a href="#">R_DAC_Change_OutputVoltage_8bit</a> | D/A コンバータの出力電圧を変更します。(8 ビットモード)   |
| <a href="#">R_DAC_Change_OutputVoltage</a>      | D/A コンバータの出力電圧を変更します。(12 ビットモード)  |
| <a href="#">R_DACn_Create</a>                   | D/A コンバータを制御するうえで必要となる初期化処理を行います。 |
| <a href="#">R_DAC_Reset</a>                     | D/A コンバータをリセットします。                |
| <a href="#">R_DACn_Create_UserInit</a>          | D/A コンバータに関するユーザ独自の初期化処理を行います。    |



## R\_DAC\_Create

D/A コンバータを制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_DAC_Create ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_DAC\_Create\_UserInit

D/A コンバータに関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_DAC\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_DAC_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

**R\_DACn\_Start**

D/A 変換を開始します。

**[指定形式]**

```
void R_DACn_Start ( void );
```

備考  $n$ は、チャンネル番号を意味します。

**[引数]**

なし

**[戻り値]**

なし

## R\_DACn\_Stop

D/A 変換を終了します。

### [指定形式]

```
void R_DACn_Stop ( void );
```

備考  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_DAC\_Set\_PowerOff

D/A コンバータに対するクロック供給を停止します。

備考 本 API 関数の呼び出しにより、D/A コンバータはリセット状態へと移行します。  
このため、本 API 関数の呼び出し後、制御レジスタへの書き込みは無視されます。

### [指定形式]

```
void R_DAC_Set_PowerOff ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_DACn\_Set\_ConversionValue

ANOn 端子に出力するアナログ電圧値を設定します。

### [指定形式]

```
#include "r_cg_macrodriver.h"
void R_DACn_Set_ConversionValue ( uint8_t reg_value );
```

備考  $n$ は、チャンネル番号を意味します。

### [引数]

| I/O | 引数                 | 説明                   |
|-----|--------------------|----------------------|
| I   | uint8_t reg_value; | D/A 変換値 (0x0 ~ 0xFF) |

### [戻り値]

なし

## R\_DAC\_Change\_OutputVoltage\_8bit

D/A コンバータの出力電圧を変更します。(8ビットモード)

### [指定形式]

```
#include "r_cg_macrodriver.h"
void R_DAC_Change_OutputVoltage_8bit ( uint8_t outputVoltage );
```

### [引数]

| I/O | 引数                     | 説明                |
|-----|------------------------|-------------------|
| I   | uint8_t outputVoltage; | 出力電圧 ( 下位 8 ビット ) |

### [戻り値]

なし

## R\_DAC\_Change\_OutputVoltage

D/A コンバータの出力電圧を変更します。(12 ビットモード)

### [指定形式]

```
#include "r_cg_macrodriver.h"
void R_DAC_Change_OutputVoltage ( uint16_t outputVoltage );
```

### [引数]

| I/O | 引数                      | 説明                 |
|-----|-------------------------|--------------------|
| I   | uint16_t outputVoltage; | 出力電圧 ( 下位 12 ビット ) |

### [戻り値]

なし



## R\_DACn\_Create

D/A コンバータを制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
#include "r_cg_macrodriver.h"
void R_DACn_Create ( void );
```

備考  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_DAC\_Reset

D/A コンバータをリセットします。

### [指定形式]

```
void R_DAC_Reset ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_DACn\_Create\_UserInit

D/A コンバータに関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_DACn\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_DACn_Create_UserInit ( void );
```

備考 *n* は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

### 3.2.26 プログラマブル・ゲイン・アンプ

以下に、コード生成がプログラマブル・ゲイン・アンプ用として出力する API 関数の一覧を示します。

表 3.26 プログラマブル・ゲイン・アンプ用 API 関数

| API 関数名                               | 機能概要                                    |
|---------------------------------------|---|
| <a href="#">R_PGA_Create</a>          | プログラマブル・ゲイン・アンプを制御するうえで必要となる初期化処理を行います。 |
| <a href="#">R_PGA_Create_UserInit</a> | プログラマブル・ゲイン・アンプに関するユーザ独自の初期化処理を行います。    |
| <a href="#">R_PGA_Start</a>           | プログラマブル・ゲイン・アンプの動作を開始します。               |
| <a href="#">R_PGA_Stop</a>            | プログラマブル・ゲイン・アンプの動作を停止します。               |
| <a href="#">R_PGA_Reset</a>           | プログラマブル・ゲイン・アンプをリセットします。                |
| <a href="#">R_PGA_Set_PowerOff</a>    | プログラマブル・ゲイン・アンプに対するクロック供給を停止します。        |

## R\_PGA\_Create

プログラマブル・ゲイン・アンプを制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_PGA_Create ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_PGA\_Create\_UserInit

プログラマブル・ゲイン・アンプに関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_PGA\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_PGA_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

**R\_PGA\_Start**

プログラマブル・ゲイン・アンプの動作を開始します。

**[指定形式]**

```
void R_PGA_Start ( void );
```

**[引数]**

なし

**[戻り値]**

なし

**R\_PGA\_Stop**

プログラマブル・ゲイン・アンプの動作を停止します。

**[指定形式]**

```
void R_PGA_Stop ( void );
```

**[引数]**

なし

**[戻り値]**

なし



## R\_PGA\_Reset

プログラマブル・ゲイン・アンプをリセットします。

### [指定形式]

```
void R_PGA_Reset ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_PGA\_Set\_PowerOff

プログラマブル・ゲイン・アンプに対するクロック供給を停止します。

備考 本 API 関数の呼び出しにより、プログラマブル・ゲイン・アンプはリセット状態へと移行します。  
このため、本 API 関数の呼び出し後、制御レジスタへの書き込みは無視されます。

### [指定形式]

```
void R_PGA_Set_PowerOff ( void );
```

### [引数]

なし

### [戻り値]

なし

### 3.2.27 コンパレータ

以下に、コード生成がコンパレータ用として出力する API 関数の一覧を示します。

表 3.27 コンパレータ用 API 関数

| API 関数名                                | 機能概要                              |
|--|-----------------------------------|
| <a href="#">R_COMP_Create</a>          | コンパレータを制御するうえで必要となる初期化処理を行います。    |
| <a href="#">R_COMP_Create_UserInit</a> | コンパレータに関するユーザ独自の初期化処理を行います。       |
| <a href="#">r_compn_interrupt</a>      | コンパレータ割り込み INTCMPn の発生に伴う処理を行います。 |
| <a href="#">R_COMPn_Start</a>          | リファレンス入力電圧とアナログ入力電圧の比較動作を開始します。   |
| <a href="#">R_COMPn_Stop</a>           | リファレンス入力電圧とアナログ入力電圧の比較動作を停止します。   |
| <a href="#">R_COMP_Reset</a>           | コンパレータをリセットします。                   |
| <a href="#">R_COMP_Set_PowerOff</a>    | コンパレータに対するクロック供給を停止します。           |

## R\_COMP\_Create

コンパレータを制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_COMP_Create ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_COMP\_Create\_UserInit

コンパレータに関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_COMP\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_COMP_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

## r\_compn\_interrupt

コンパレータ割り込み INTCMP $n$ の発生に伴う処理を行います。

備考 本 API 関数は、コンパレータ割り込み INTCMP $n$ に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_compn_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_compn_interrupt ( void );
```

備考  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

**R\_COMP $n$ \_Start**

リファレンス入力電圧とアナログ入力電圧の比較動作を開始します。

**[指定形式]**

```
void R_COMP $n$ _Start ( void );
```

備考  $n$ は、チャンネル番号を意味します。

**[引数]**

なし

**[戻り値]**

なし

**R\_COMPn\_Stop**

リファレンス入力電圧とアナログ入力電圧の比較動作を停止します。

**[指定形式]**

```
void R_COMPn_Stop ( void );
```

備考  $n$ は、チャンネル番号を意味します。

**[引数]**

なし

**[戻り値]**

なし



**R\_COMP\_Reset**

コンパレータをリセットします。

**[指定形式]**

```
void R_COMP_Reset ( void );
```

**[引数]**

なし

**[戻り値]**

なし

## R\_COMP\_Set\_PowerOff

コンパレータに対するクロック供給を停止します。

備考           本 API 関数の呼び出しにより、コンパレータはリセット状態へと移行します。  
このため、本 API 関数の呼び出し後、制御レジスタへの書き込みは無視されます。

### [指定形式]

```
void R_COMP_Set_PowerOff ( void );
```

### [引数]

なし

### [戻り値]

なし

### 3.2.28 コンパレータ / プログラマブル・ゲイン・アンプ

以下に、コード生成がコンパレータ / プログラマブル・ゲイン・アンプ用として出力する API 関数の一覧を示します。

表 3.28 コンパレータ / プログラマブル・ゲイン・アンプ用 API 関数

| API 関数名                                    | 機能概要  |
|--|---|
| <a href="#">R_COMPPGA_Create</a>           | コンパレータ / プログラマブル・ゲイン・アンプを制御するうえで必要となる初期化処理を行います。                          |
| <a href="#">R_COMPPGA_Set_PowerOff</a>     | コンパレータ / プログラマブル・ゲイン・アンプに対するクロック供給を停止します。                                 |
| <a href="#">R_COMPPGA_Create_UserInit</a>  | コンパレータ / プログラマブル・ゲイン・アンプに関するユーザ独自の初期化処理を行います。                             |
| <a href="#">r_compn_interrupt</a>          | コンパレータ割り込み INTCMP $n$ の発生に伴う処理を行います。                                      |
| <a href="#">R_COMP<math>n</math>_Start</a> | リファレンス入力電圧とアナログ入力電圧の比較動作を開始します。   |
| <a href="#">R_COMP<math>n</math>_Stop</a>  | リファレンス入力電圧とアナログ入力電圧の比較動作を停止します。   |
| <a href="#">R_PGA_Start</a>                | プログラマブル・ゲイン・アンプの動作を開始します。   |
| <a href="#">R_PGA_Stop</a>                 | プログラマブル・ゲイン・アンプの動作を停止します。   |
| <a href="#">R_PWMOPT_Start</a>             | 6 相 PWM オプション・ユニットに対する入力クロックを供給します。<br>また、6 相 PWM オプション・ユニットの動作モードを設定します。 |
| <a href="#">R_PWMOPT_Stop</a>              | 6 相 PWM オプション・ユニットに対するクロック供給を停止します。                                       |

## R\_COMPPGA\_Create

コンパレータ / プログラマブル・ゲイン・アンプを制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_COMPPGA_Create ( void );
```

### [引数]

なし

## R\_COMPPGA\_Set\_PowerOff

コンパレータ / プログラマブル・ゲイン・アンプに対するクロック供給を停止します。

備考           本 API 関数の呼び出しにより、コンパレータ / プログラマブル・ゲイン・アンプはリセット状態へと移行します。  
このため、本 API 関数の呼び出し後、制御レジスタへの書き込みは無視されます。

### [指定形式]

```
void R_COMPPGA_Set_PowerOff ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_COMPPGA\_Create\_UserInit

コンパレータ / プログラマブル・ゲイン・アンプに関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_COMPPGA\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_COMPPGA_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

## r\_compn\_interrupt

コンパレータ割り込み INTCMP $n$ の発生に伴う処理を行います。

備考 本 API 関数は、コンパレータ割り込み INTCMP $n$ に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_compn_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_compn_interrupt ( void );
```

備考  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

**R\_COMPn\_Start**

リファレンス入力電圧とアナログ入力電圧の比較動作を開始します。

**[指定形式]**

```
void R_COMPn_Start ( void );
```

備考  $n$ は、チャンネル番号を意味します。

**[引数]**

なし

**[戻り値]**

なし



**R\_COMP $n$ \_Stop**

リファレンス入力電圧とアナログ入力電圧の比較動作を停止します。

**[指定形式]**

```
void R_COMP $n$ _Stop ( void );
```

備考  $n$ は、チャンネル番号を意味します。

**[引数]**

なし

**[戻り値]**

なし

**R\_PGA\_Start**

プログラマブル・ゲイン・アンプの動作を開始します。

**[指定形式]**

```
void R_PGA_Start ( void );
```

**[引数]**

なし

**[戻り値]**

なし

## R\_PGA\_Stop

プログラマブル・ゲイン・アンプの動作を停止します。

### [指定形式]

```
void R_PGA_Stop ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_PWMOPT\_Start

6相PWMオプション・ユニットに対する入力クロックを供給します。  
また、6相PWMオプション・ユニットの動作モードを設定します。

### [指定形式]

```
void R_PWMOPT_Start ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_PWMOPT\_Stop

6相PWMオプション・ユニットに対するクロック供給を停止します。

### [指定形式]

```
void R_PWMOPT_Stop ( void );
```

### [引数]

なし

### [戻り値]

なし

## 3.2.29 シリアル・アレイ・ユニット

以下に、コード生成がシリアル・アレイ・ユニット用として出力する API 関数の一覧を示します。

表 3.29 シリアル・アレイ・ユニット用 API 関数

| API 関数名                          | 機能概要                                       |
|----------------------------------|--|
| R_SAUm_Create                    | シリアル・アレイ・ユニットを制御するうえで必要となる初期化処理を行います。      |
| R_SAUm_Create_UserInit           | シリアル・アレイ・ユニットに関するユーザ独自の初期化処理を行います。         |
| R_SAUm_Reset                     | シリアル・アレイ・ユニットをリセットします。                     |
| R_SAUm_Set_PowerOff              | シリアル・アレイ・ユニットに対するクロック供給を停止します。             |
| R_SAUm_Set_SnoozeOn              | STOP モードから SNOOZE モードへの切り替えを許可します。         |
| R_SAUm_Set_SnoozeOff             | STOP モードから SNOOZE モードへの切り替えを禁止します。         |
| R_UARTn_Create                   | UART 通信を行ううえで必要となる初期化処理を行います。              |
| r_uartn_interrupt_send           | UART 送信完了割り込み INTST $n$ の発生に伴う処理を行います。     |
| r_uartn_interrupt_receive        | UART 受信完了割り込み INTSR $n$ の発生に伴う処理を行います。     |
| r_uartn_interrupt_error          | 受信エラー割り込み INTSRE $n$ の発生に伴う処理を行います。        |
| R_UARTn_Start                    | UART 通信を待機状態にします。                          |
| R_UARTn_Stop                     | UART 通信を終了します。                             |
| R_UARTn_Send                     | データの UART 送信を開始します。                        |
| R_UARTn_Receive                  | データの UART 受信を開始します。                        |
| r_uartn_callback_sendend         | UART 送信完了割り込み INTST $n$ の発生に伴う処理を行います。     |
| r_uartn_callback_receiveend      | UART 受信完了割り込み INTSR $n$ の発生に伴う処理を行います。     |
| r_uartn_callback_error           | UART 受信エラー割り込み INTSRE $n$ の発生に伴う処理を行います。   |
| r_uartn_callback_softwareoverrun | オーバラン・エラーの検出に伴う処理を行います。                    |
| R_CSImn_Create                   | 3 線シリアル I/O 通信を行ううえで必要となる初期化処理を行います。       |
| r_csimn_interrupt                | CSI 通信完了割り込み INTCSIm $n$ の発生に伴う処理を行います。    |
| R_CSImn_Start                    | 3 線シリアル I/O 通信を待機状態にします。                   |
| R_CSImn_Stop                     | 3 線シリアル I/O 通信を終了します。                      |
| R_CSImn_Send                     | データの CSI 送信を開始します。                         |
| R_CSImn_Receive                  | データの CSI 受信を開始します。                         |
| R_CSImn_Send_Receive             | データの CSI 送受信を開始します。                        |
| r_csimn_callback_sendend         | CSI 送信完了割り込み INTCSIm $n$ の発生に伴う処理を行います。    |
| r_csimn_callback_receiveend      | CSI 受信完了割り込み INTCSIm $n$ の発生に伴う処理を行います。    |
| r_csimn_callback_error           | CSI 受信エラー割り込み INTSRE $n$ の発生に伴う処理を行います。    |
| R_IICmn_Create                   | 簡易 IIC 通信を行ううえで必要となる初期化処理を行います。            |
| r_iicmn_interrupt                | 簡易 IIC 通信完了割り込み INTIICm $n$ の発生に伴う処理を行います。 |
| R_IICmn_StartCondition           | スタート・コンディションを発生させます。                       |

| API 関数名  | 機能概要                                       |
|--|--|
| <a href="#">R_IICmn_StopCondition</a>              | ストップ・コンディションを発生させます。                       |
| <a href="#">R_IICmn_Stop</a>                       | 簡易 IIC 通信を終了します。                           |
| <a href="#">R_IICmn_Master_Send</a>                | 簡易 IIC マスタ送信を開始します。                        |
| <a href="#">R_IICmn_Master_Receive</a>             | 簡易 IIC マスタ受信を開始します。                        |
| <a href="#">r_iicmn_callback_master_sendend</a>    | 簡易 IIC マスタ送信完了割り込み INTIICmn の発生に伴う処理を行います。 |
| <a href="#">r_iicmn_callback_master_receiveend</a> | 簡易 IIC マスタ受信完了割り込み INTIICmn の発生に伴う処理を行います。 |
| <a href="#">r_iicmn_callback_master_error</a>      | パリティ・エラー（ACK エラー）の検出に伴う処理を行います。            |

## R\_SAUm\_Create

シリアル・アレイ・ユニットを制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_SAUm_Create ( void );
```

備考 *m*は、ユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし



## R\_SAUm\_Create\_UserInit

シリアル・アレイ・ユニットに関するユーザ独自の初期化処理を行います。

備考 本API関数は、[R\\_SAUm\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_SAUm_Create_UserInit ( void );
```

備考 *m* は、ユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_SAUm\_Reset

シリアル・アレイ・ユニットをリセットします。

### [指定形式]

```
void R_SAUm_Reset ( void );
```

備考 *m*は、ユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_SAUm\_Set\_PowerOff

シリアル・アレイ・ユニットに対するクロック供給を停止します。

備考 本 API 関数の呼び出しにより、シリアル・アレイ・ユニットはリセット状態へと移行します。  
このため、本 API 関数の呼び出し後、制御レジスタ（シリアル・クロック選択レジスタ  $n$ : SPS $n$  など）への書き込みは無視されます。

### [指定形式]

```
void R_SAUm_Set_PowerOff ( void );
```

備考  $m$  は、ユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_SAUm\_Set\_SnoozeOn

STOP モードから SNOOZE モードへの切り替えを許可します。

### [指定形式]

```
void R_SAUm_Set_SnoozeOn ( void );
```

備考 *m* は、ユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_SAUm\_Set\_SnoozeOff

STOP モードから SNOOZE モードへの切り替えを禁止します。

### [指定形式]

```
void R_SAUm_Set_SnoozeOff ( void );
```

備考 *m* は、ユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_UARTn\_Create

UART 通信を行ううえで必要となる初期化処理を行います。

備考 本 API 関数は、[R\\_SAUm\\_Create](#) の内部関数として位置づけられているため、通常、ユーザの処理プログラムから呼び出す必要はありません。

### [指定形式]

```
void R_UARTn_Create ( void );
```

備考 *n* は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## r\_uartn\_interrupt\_send

UART 送信完了割り込み INTST $n$ の発生に伴う処理を行います。

備考 本 API 関数は、UART 送信完了割り込み INTST $n$ に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_uartn_interrupt_send ( void );
```

CC-RL コンパイラの場合

```
static void __near r_uartn_interrupt_send ( void );
```

備考  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## r\_uartn\_interrupt\_receive

UART 受信完了割り込み INTSR $n$ の発生に伴う処理を行います。

備考 本 API 関数は、UART 受信完了割り込み INTSR $n$ に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_uartn_interrupt_receive ( void );
```

CC-RL コンパイラの場合

```
static void __near r_uartn_interrupt_receive ( void );
```

備考  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし



## r\_uartn\_interrupt\_error

受信エラー割り込み INTSRE $n$ の発生に伴う処理を行います。

備考 本 API 関数は、受信エラー割り込み INTSRE $n$ に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_uartn_interrupt_error ( void );
```

CC-RL コンパイラの場合

```
static void __near r_uartn_interrupt_error ( void );
```

備考  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

**R\_UARTn\_Start**

UART 通信を待機状態にします。

**[指定形式]**

```
void R_UARTn_Start ( void );
```

備考  $n$ は、チャンネル番号を意味します。

**[引数]**

なし

**[戻り値]**

なし

## R\_UARTn\_Stop

UART 通信を終了します。

### [指定形式]

```
void R_UARTn_Stop ( void );
```

備考  $n$ は、チャネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_UARTn\_Send

データの UART 送信を開始します。

備考 1. 本 API 関数では、引数 *tx\_buf* で指定されたバッファから 1 バイト単位の UART 送信を引数 *tx\_num* で指定された回数だけ繰り返し行います。

備考 2. UART 送信を行う際には、本 API 関数の呼び出し以前に [R\\_UARTn\\_Start](#) を呼び出す必要があります。

### [指定形式]

```
#include "r_cg_macrodriver.h"
MD_STATUS R_UARTn_Send ( uint8_t * const tx_buf, uint16_t tx_num );
```

備考 *n* は、チャンネル番号を意味します。

### [引数]

| I/O | 引数                              | 説明                     |
|-----|---------------------------------|------------------------|
| I   | uint8_t * const <i>tx_buf</i> ; | 送信するデータを格納したバッファへのポインタ |
| I   | uint16_t <i>tx_num</i> ;        | 送信するデータの総数             |

### [戻り値]

| マクロ         | 説明       |
|-------------|----------|
| MD_OK       | 正常終了     |
| MD_ARGERROR | 引数の指定が不正 |

## R\_UARTn\_Receive

データの UART 受信を開始します。

- 備考 1. 本 API 関数では、1 バイト単位の UART 受信を引数 *rx\_num* で指定された回数だけ繰り返し行い、引数 *rx\_buf* で指定されたバッファに格納します。
- 備考 2. 実際の UART 受信は、本 API 関数の呼び出し後、[R\\_UARTn\\_Start](#) を呼び出すことにより開始されます。

### [指定形式]

```
#include "r_cg_macrodriver.h"
MD_STATUS R_UARTn_Receive ( uint8_t * const rx_buf, uint16_t rx_num );
```

備考 *n* は、チャンネル番号を意味します。

### [引数]

| I/O | 引数                              | 説明                     |
|-----|---------------------------------|------------------------|
| O   | uint8_t * const <i>rx_buf</i> ; | 受信したデータを格納するバッファへのポインタ |
| I   | uint16_t <i>rx_num</i> ;        | 受信するデータの総数             |

### [戻り値]

| マクロ         | 説明       |
|-------------|----------|
| MD_OK       | 正常終了     |
| MD_ARGERROR | 引数の指定が不正 |

## r\_uartn\_callback\_sendend

UART 送信完了割り込み INTST $n$ の発生に伴う処理を行います。

備考 本 API 関数は、UART 送信完了割り込み INTST $n$ に対応した割り込み処理 [r\\_uartn\\_interrupt\\_send](#) のコールバック・ルーチン ([R\\_UARTn\\_Send](#) の引数  $tx\_num$  で指定された数のデータ送信が完了した際の処理) として呼び出されます。

### [指定形式]

```
static void r_uartn_callback_sendend ( void );
```

備考  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## r\_uartn\_callback\_receiveend

UART 受信完了割り込み INTSR $n$ の発生に伴う処理を行います。

備考 本 API 関数は、UART 受信完了割り込み INTSR $n$ に対応した割り込み処理 [r\\_uartn\\_interrupt\\_receive](#) のコールバック・ルーチン ([R\\_UARTn\\_Receive](#) の引数 *rx\_num* で指定された数のデータ受信が完了した際の処理) として呼び出されます。

### [指定形式]

```
static void r_uartn_callback_receiveend ( void );
```

備考  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

**r\_uartn\_callback\_error**

UART 受信エラー割り込み INTSRE $n$ の発生に伴う処理を行います。

備考 本 API 関数は、UART 受信エラー割り込み INTSRE $n$ に対応した割り込み処理 [r\\_uartn\\_interrupt\\_error](#) のコールバック・ルーチンとして呼び出されます。

**[指定形式]**

```
#include "r_cg_macrodriver.h"
static void r_uartn_callback_error ( uint8_t err_type );
```

備考  $n$ は、チャンネル番号を意味します。

**[引数]**

| I/O | 引数                | 説明   |
|-----|-------------------|--|
| O   | uint8_t err_type; | UART 受信エラー割り込みの発生要因<br>00000xx1B : オーバラン・エラー<br>00000x1xB : パリティ・エラー<br>000001xxB : フレーミング・エラー |

**[戻り値]**

なし



## r\_uartn\_callback\_softwareoverrun

オーバーラン・エラーの検出に伴う処理を行います。

備考 本 API 関数は、UART 受信完了割り込み INTSR $n$ に対応した割り込み処理 `r_uartn_interrupt_receive` のコールバック・ルーチン (`R_UARTn_Receive` の引数 `rx_num` で指定された数以上のデータを受信した際の処理) として呼び出されます。

### [指定形式]

```
#include "r_cg_macrodriver.h"
static void r_uartn_callback_softwareoverrun ( uint16_t rx_data );
```

備考  $n$ は、チャンネル番号を意味します。

### [引数]

| I/O | 引数                             | 説明  |
|-----|--------------------------------|---|
| O   | <code>uint16_t rx_data;</code> | 受信したデータ ( <code>R_UARTn_Receive</code> の引数 <code>rx_num</code> で指定された数以上に受信したデータ) |

### [戻り値]

なし

## R\_CSImn\_Create

3線シリアル I/O 通信を行ううえで必要となる初期化処理を行います。

備考 本 API 関数は、[R\\_SAUm\\_Create](#) の内部関数として位置づけられているため、通常、ユーザの処理プログラムから呼び出す必要はありません。

### [指定形式]

```
void R_CSImn_Create ( void );
```

備考  $m$  はユニット番号を、 $n$  はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## r\_csimn\_interrupt

CSI 通信完了割り込み INTCSImn の発生に伴う処理を行います。

備考 本 API 関数は、CSI 通信完了割り込み INTCSImn に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_csimn_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_csimn_interrupt ( void );
```

備考 *m* はユニット番号を、*n* はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_CSImn\_Start

3 線シリアル I/O 通信を待機状態にします。

### [指定形式]

```
void R_CSImn_Start ( void );
```

備考  $m$  はユニット番号を,  $n$  はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_CSImn\_Stop

3線シリアル I/O 通信を終了します。

### [指定形式]

```
void R_CSImn_Stop ( void );
```

備考  $m$ はユニット番号を,  $n$ はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_CSImn\_Send

データの CSI 送信を開始します。

備考 1. 本 API 関数では、引数 *tx\_buf* で指定されたバッファから 1 バイト単位の CSI 送信を引数 *tx\_num* で指定された回数だけ繰り返し行います。

備考 2. CSI 送信を行う際には、本 API 関数の呼び出し以前に [R\\_CSImn\\_Start](#) を呼び出す必要があります。

### [指定形式]

```
#include "r_cg_macrodriver.h"
MD_STATUS R_CSImn_Send ( uint8_t * const tx_buf, uint16_t tx_num );
```

備考 *m* はユニット番号を、*n* はチャネル番号を意味します。

### [引数]

| I/O | 引数                              | 説明                     |
|-----|---------------------------------|------------------------|
| I   | uint8_t * const <i>tx_buf</i> ; | 送信するデータを格納したバッファへのポインタ |
| I   | uint16_t <i>tx_num</i> ;        | 送信するデータの総数             |

### [戻り値]

| マクロ         | 説明       |
|-------------|----------|
| MD_OK       | 正常終了     |
| MD_ARGERROR | 引数の指定が不正 |

## R\_CSImn\_Receive

データの CSI 受信を開始します。

備考 1. 本 API 関数では、1 バイト単位の CSI 受信を引数 *rx\_num* で指定された回数だけ繰り返し行い、引数 *rx\_buf* で指定されたバッファに格納します。

備考 2. CSI 受信を行う際には、本 API 関数の呼び出し以前に [R\\_CSImn\\_Start](#) を呼び出す必要があります。

### [指定形式]

```
#include "r_cg_macrodriver.h"
MD_STATUS R_CSImn_Receive ( uint8_t * const rx_buf, uint16_t rx_num );
```

備考 *m* はユニット番号を、*n* はチャンネル番号を意味します。

### [引数]

| I/O | 引数                              | 説明                     |
|-----|---------------------------------|------------------------|
| O   | uint8_t * const <i>rx_buf</i> ; | 受信したデータを格納するバッファへのポインタ |
| I   | uint16_t <i>rx_num</i> ;        | 受信するデータの総数             |

### [戻り値]

| マクロ         | 説明       |
|-------------|----------|
| MD_OK       | 正常終了     |
| MD_ARGERROR | 引数の指定が不正 |

## R\_CSImn\_Send\_Receive

データの CSI 送受信を開始します。

- 備考 1. 本 API 関数では、引数 *tx\_buf* で指定されたバッファから 1 バイト単位の CSI 送信を引数 *tx\_num* で指定された回数だけ繰り返し行います。
- 備考 2. 本 API 関数では、1 バイト単位の CSI 受信を引数 *tx\_num* で指定された回数だけ繰り返し行い、引数 *rx\_buf* で指定されたバッファに格納します。
- 備考 3. CSI 送受信を行う際には、本 API 関数の呼び出し以前に [R\\_CSImn\\_Start](#) を呼び出す必要があります。

### [指定形式]

```
#include "r_cg_macrodriver.h"
MD_STATUS R_CSImn_Send_Receive ( uint8_t * const tx_buf, uint16_t tx_num, uint8_t *
const rx_buf );
```

備考 *m* はユニット番号を、*n* はチャンネル番号を意味します。

### [引数]

| I/O | 引数                              | 説明                     |
|-----|---------------------------------|------------------------|
| I   | uint8_t * const <i>tx_buf</i> ; | 送信するデータを格納したバッファへのポインタ |
| I   | uint16_t <i>tx_num</i> ;        | 送受信するデータの総数            |
| O   | uint8_t * const <i>rx_buf</i> ; | 受信したデータを格納するバッファへのポインタ |

### [戻り値]

| マクロ         | 説明       |
|-------------|----------|
| MD_OK       | 正常終了     |
| MD_ARGERROR | 引数の指定が不正 |



## r\_csimn\_callback\_sendend

CSI 送信完了割り込み INTCSImn の発生に伴う処理を行います。

備考 本 API 関数は、CSI 送信完了割り込み INTCSImn に対応した割り込み処理 [r\\_csimn\\_interrupt](#) のコールバック・ルーチン ([R\\_CSImn\\_Send](#), または [R\\_CSImn\\_Send\\_Receive](#) の引数 *tx\_num* で指定された数のデータ送信が完了した際の処理) として呼び出されます。

### [指定形式]

```
static void r_csimn_callback_sendend ( void );
```

備考 *m* はユニット番号を, *n* はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## r\_csimn\_callback\_receiveend

CSI 受信完了割り込み INTCSImn の発生に伴う処理を行います。

**備考** 本 API 関数は、CSI 受信完了割り込み INTCSImn に対応した割り込み処理 `r_csimn_interrupt` のコールバック・ルーチン (`R_CSImn_Receive`, または `R_CSImn_Send_Receive` の引数 `rx_num` で指定された数のデータ受信が完了した際の処理) として呼び出されます。

### [指定形式]

```
static void r_csimn_callback_receiveend ( void );
```

**備考** `m` はユニット番号を, `n` はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

**r\_csimn\_callback\_error**

CSI 受信エラー割り込み INTSREn の発生に伴う処理を行います。

備考 本 API 関数は、CSI 受信エラー割り込み INTSREn に対応した割り込み処理 `r_uartn_interrupt_error` のコールバック・ルーチンとして呼び出されます。

**[指定形式]**

```
#include "r_cg_macrodriver.h"
static void r_csimn_callback_error ( uint8_t err_type );
```

備考 *m* はユニット番号を、*n* はチャンネル番号を意味します。

**[引数]**

| I/O | 引数                             | 説明  |
|-----|--------------------------------|---|
| O   | <code>uint8_t err_type;</code> | CSI 受信エラー割り込みの発生要因<br>00000xx1B : オーバラン・エラー |

**[戻り値]**

なし

## R\_IICmn\_Create

簡易 IIC 通信を行ううえで必要となる初期化処理を行います。

備考 本 API 関数は、[R\\_SAUm\\_Create](#) の内部関数として位置づけられているため、通常、ユーザの処理プログラムから呼び出す必要はありません。

### [指定形式]

```
void R_IICmn_Create ( void );
```

備考 *m* はユニット番号を、*n* はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## r\_iicmn\_interrupt

簡易 IIC 通信完了割り込み INTIICmn の発生に伴う処理を行います。

備考 本 API 関数は、簡易 IIC 通信完了割り込み INTIICmn に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_iicmn_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_iicmn_interrupt ( void );
```

備考  $m$  はユニット番号を、 $n$  はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_IICmn\_StartCondition

スタート・コンディションを発生させます。

備考 本 API 関数は、[R\\_IICmn\\_Master\\_Send](#)、および [R\\_IICmn\\_Master\\_Receive](#) の内部関数として位置づけられているため、通常、ユーザの処理プログラムから呼び出す必要はありません。

### [指定形式]

```
void R_IICmn_StartCondition ( void );
```

備考 *m* はユニット番号を、*n* はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_IICmn\_StopCondition

ストップ・コンディションを発生させます。

### [指定形式]

```
void R_IICmn_StopCondition ( void );
```

備考  $m$ はユニット番号を,  $n$ はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_IICmn\_Stop

簡易 IIC 通信を終了します。

### [指定形式]

```
void R_IICmn_Stop ( void );
```

備考  $m$ はユニット番号を,  $n$ はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし



## R\_IICmn\_Master\_Send

簡易 IIC マスタ送信を開始します。

備考 本 API 関数では、引数 *tx\_buf* で指定されたバッファから 1 バイト単位の簡易 IIC マスタ送信を引数 *tx\_num* で指定された回数だけ繰り返し行います。

### [指定形式]

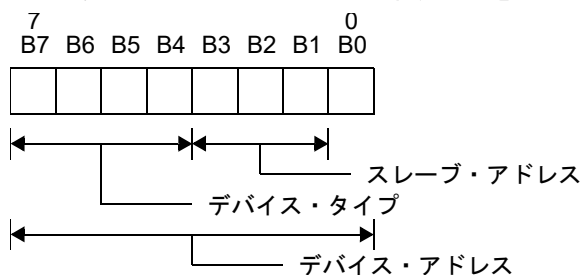
```
#include "r_cg_macrodriver.h"
void R_IICmn_Master_Send ( uint8_t adr, uint8_t * const tx_buf, uint16_t tx_num );
```

備考 *m* はユニット番号を、*n* はチャンネル番号を意味します。

### [引数]

| I/O | 引数                              | 説明                     |
|-----|---------------------------------|------------------------|
| I   | uint8_t <i>adr</i> ;            | デバイス・アドレス              |
| I   | uint8_t * const <i>tx_buf</i> ; | 送信するデータを格納したバッファへのポインタ |
| I   | uint16_t <i>tx_num</i> ;        | 送信するデータの総数             |

備考 以下に、デバイス・アドレス *adr* の指定形式を示します。



### [戻り値]

なし

## R\_IICmn\_Master\_Receive

簡易 IIC マスタ受信を開始します。

備考 本 API 関数では、1 バイト単位の簡易 IIC マスタ受信を引数 *rx\_num* で指定された回数だけ繰り返し行い、引数 *rx\_buf* で指定されたバッファに格納します。

### [指定形式]

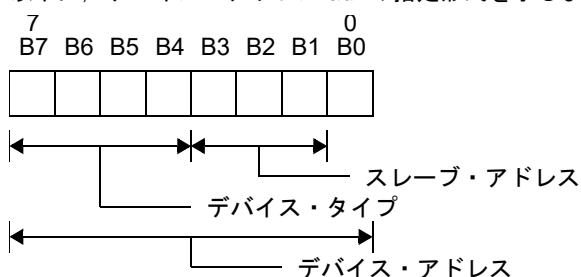
```
#include "r_cg_macrodriver.h"
void R_IICmn_Master_Receive ( uint8_t adr, uint8_t * const rx_buf, uint16_t rx_num
);
```

備考 *m* はユニット番号を、*n* はチャンネル番号を意味します。

### [引数]

| I/O | 引数                              | 説明                     |
|-----|---------------------------------|------------------------|
| I   | uint8_t <i>adr</i> ;            | デバイス・アドレス              |
| O   | uint8_t * const <i>rx_buf</i> ; | 受信したデータを格納するバッファへのポインタ |
| I   | uint16_t <i>rx_num</i> ;        | 受信するデータの総数             |

備考 以下に、デバイス・アドレス *adr* の指定形式を示します。



### [戻り値]

なし

## r\_iicmn\_callback\_master\_sendend

簡易 IIC マスタ送信完了割り込み INTIICmn の発生に伴う処理を行います。

備考 本 API 関数は、簡易 IIC マスタ送信完了割り込み INTIICmn に対応した割り込み処理 [r\\_iicmn\\_interrupt](#) のコールバック・ルーチン ([R\\_IICmn\\_Master\\_Send](#) の引数 *tx\_num* で指定された数のデータ送信が完了した際の処理) として呼び出されます。

### [指定形式]

```
static void r_iicmn_callback_master_sendend ( void );
```

備考 *m* はユニット番号を、*n* はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## r\_iicmn\_callback\_master\_receiveend

簡易 IIC マスタ受信完了割り込み INTIICmn の発生に伴う処理を行います。

**備考** 本 API 関数は、簡易 IIC マスタ受信完了割り込み INTIICmn に対応した割り込み処理 [r\\_iicmn\\_interrupt](#) のコールバック・ルーチン ([R\\_IICmn\\_Master\\_Receive](#) の引数 *rx\_num* で指定された数のデータ送信が完了した際の処理) として呼び出されます。

### [指定形式]

```
static void r_iicmn_callback_master_receiveend ( void );
```

**備考** *m* はユニット番号を、*n* はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

**r\_iicmn\_callback\_master\_error**

パリティ・エラー（ACKエラー）の検出に伴う処理を行います。

## [指定形式]

```
#include "r_cg_macrodriver.h"
static void r_iicmn_callback_master_error ( MD_STATUS flag );
```

備考  $m$ はユニット番号を、 $n$ はチャンネル番号を意味します。

## [引数]

| I/O | 引数              | 説明                                 |
|-----|-----------------|------------------------------------|
| ○   | MD_STATUS flag; | 通信エラーの発生要因<br>MD_NACK : アクノリッジの未検出 |

## [戻り値]

なし

## 3.2.30 シリアル・アレイ・ユニット 4

以下に、コード生成がシリアル・アレイ・ユニット 4 用として出力する API 関数の一覧を示します。

表 3.30 シリアル・アレイ・ユニット 4 用 API 関数

| API 関数名  | 機能概要                                       |
|--|--|
| <a href="#">R_DALIn_Create</a>                   | シリアル・アレイ・ユニット 4 を制御するうえで必要となる初期化処理を行います。   |
| <a href="#">r_dalin_interrupt_send</a>           | DALI 送信完了割り込み INTSTDL $n$ の発生に伴う処理を行います。   |
| <a href="#">r_dalin_interrupt_receive</a>        | DALI 受信完了割り込み INTSRDL $n$ の発生に伴う処理を行います。   |
| <a href="#">r_dalin_interrupt_error</a>          | DALI 受信エラー割り込み INTSREDL $n$ の発生に伴う処理を行います。 |
| <a href="#">R_DALIn_Start</a>                    | DALI 通信を待機状態にします。                          |
| <a href="#">R_DALIn_Stop</a>                     | DALI 通信を終了します。                             |
| <a href="#">R_DALIn_Send</a>                     | データの DALI 送信を開始します。                        |
| <a href="#">R_DALIn_Receive</a>                  | データの DALI 受信を開始します。                        |
| <a href="#">r_dalin_callback_sendend</a>         | DALI 送信完了割り込み INTSTDL $n$ の発生に伴う処理を行います。   |
| <a href="#">r_dalin_callback_receiveend</a>      | DALI 受信完了割り込み INTSRDL $n$ の発生に伴う処理を行います。   |
| <a href="#">r_dalin_callback_error</a>           | DALI 受信エラー割り込み INTSREDL $n$ の発生に伴う処理を行います。 |
| <a href="#">r_dalin_callback_softwareoverrun</a> | オーバラン・エラーの検出に伴う処理を行います。                    |

## R\_DALIn\_Create

シリアル・アレイ・ユニット4を制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_DALIn_Create ( void );
```

備考 *n*はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## r\_dalin\_interrupt\_send

DALI 送信完了割り込み INTSTDL $n$ の発生に伴う処理を行います。

備考 本 API 関数は、DALI 送信完了割り込み INTSTDL $n$ に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_dalin_interrupt_send ( void );
```

CC-RL コンパイラの場合

```
static void __near r_dalin_interrupt_send ( void );
```

備考  $n$ はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし



## r\_dalin\_interrupt\_receive

DALI 受信完了割り込み INTSRDL $n$ の発生に伴う処理を行います。

備考 本 API 関数は、DALI 受信完了割り込み INTSRDL $n$ に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_dalin_interrupt_receive ( void );
```

CC-RL コンパイラの場合

```
static void __near r_dalin_interrupt_receive ( void );
```

備考  $n$ はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## r\_dalin\_interrupt\_error

DALI 受信エラー割り込み INTSREDL $n$ の発生に伴う処理を行います。

備考 本 API 関数は、DALI 受信エラー割り込み INTSREDL $n$ に対応した割り込み処理として呼び出され  
ます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_dalin_interrupt_error ( void );
```

CC-RL コンパイラの場合

```
static void __near r_dalin_interrupt_error ( void );
```

備考  $n$ はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_DALIn\_Start

DALI 通信を待機状態にします。

### [指定形式]

```
void R_DALIn_Start ( void );
```

備考 *n*はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_DALIn\_Stop

DALI 通信を終了します。

### [指定形式]

```
void R_DALIn_Stop ( void );
```

備考 *n*はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_DALIn\_Send

データの DALI 送信を開始します。

備考 1. 本 API 関数では、引数 *tx\_buf* で指定されたバッファから 1 バイト単位の DALI 送信を引数 *tx\_num* で指定された回数だけ繰り返し行います。

備考 2. DALI 送信を行う際には、本 API 関数の呼び出し以前に [R\\_DALIn\\_Start](#) を呼び出す必要があります。

### [指定形式]

```
#include "r_cg_macrodriver.h"
MD_STATUS R_DALIn_Send ( uint8_t * const tx_buf, uint16_t tx_num );
```

備考 *n* はチャンネル番号を意味します。

### [引数]

| I/O | 引数                              | 説明                     |
|-----|---------------------------------|------------------------|
| I   | uint8_t * const <i>tx_buf</i> ; | 送信するデータを格納したバッファへのポインタ |
| I   | uint16_t <i>tx_num</i> ;        | 送信するデータの総数             |

### [戻り値]

| マクロ         | 説明       |
|-------------|----------|
| MD_OK       | 正常終了     |
| MD_ARGERROR | 引数の指定が不正 |

## R\_DALIn\_Receive

データの DALI 受信を開始します。

備考 1. 本 API 関数では、1 バイト単位の DALI 受信を引数 *rx\_num* で指定された回数だけ繰り返し行い、引数 *rx\_buf* で指定されたバッファに格納します。

備考 2. 実際の DALI 受信は、本 API 関数の呼び出し後、[R\\_DALIn\\_Start](#) を呼び出すことにより開始されます。

### [指定形式]

```
#include "r_cg_macrodriver.h"
MD_STATUS R_DALIn_Receive ( uint8_t * const rx_buf, uint16_t rx_num );
```

備考 *n* はチャネル番号を意味します。

### [引数]

| I/O | 引数                              | 説明                     |
|-----|---------------------------------|------------------------|
| O   | uint8_t * const <i>rx_buf</i> ; | 受信したデータを格納するバッファへのポインタ |
| I   | uint16_t <i>rx_num</i> ;        | 受信するデータの総数             |

### [戻り値]

| マクロ         | 説明       |
|-------------|----------|
| MD_OK       | 正常終了     |
| MD_ARGERROR | 引数の指定が不正 |

## r\_dalin\_callback\_sendend

DALI 送信完了割り込み INTSTDL $n$ の発生に伴う処理を行います。

備考 本 API 関数は、DALI 送信完了割り込み INTSTDL $n$ に対応した割り込み処理 [r\\_dalin\\_interrupt\\_send](#) のコールバック・ルーチン ([R\\_DALIn\\_Send](#) の引数  $tx\_num$  で指定された数のデータ送信が完了した際の処理) として呼び出されます。

### [指定形式]

```
static void r_dalin_callback_sendend ( void );
```

備考  $n$ はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## r\_dalin\_callback\_receiveend

DALI 受信完了割り込み INTSRDL $n$ の発生に伴う処理を行います。

備考 本 API 関数は、DALI 受信完了割り込み INTSRDL $n$ に対応した割り込み処理 [r\\_dalin\\_interrupt\\_receive](#) のコールバック・ルーチン ([R\\_DALIn\\_Receive](#) の引数  $rx\_num$  で指定された数のデータ受信が完了した際の処理) として呼び出されます。

### [指定形式]

```
static void r_dalin_callback_receiveend ( void );
```

備考  $n$ はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし



## r\_dalin\_callback\_error

DALI 受信エラー割り込み INTSREDLnの発生に伴う処理を行います。

備考 本 API 関数は、DALI 受信エラー割り込み INTSREDLnに対応した割り込み処理 [r\\_dalin\\_interrupt\\_error](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
#include "r_cg_macrodriver.h"
static void r_dalin_callback_error ( uint8_t err_type );
```

備考 *n*はチャンネル番号を意味します。

### [引数]

| I/O | 引数                | 説明   |
|-----|-------------------|--|
| O   | uint8_t err_type; | DALI 受信エラー割り込みの発生要因<br>00000xx1B : オーバラン・エラー<br>00000x1xB : パリティ・エラー<br>000001xxB : フレーミング・エラー |

### [戻り値]

なし

## r\_dalin\_callback\_softwareoverrun

オーバーラン・エラーの検出に伴う処理を行います。

備考 本 API 関数は、DALI 受信完了割り込み INTSRDL $n$  に対応した割り込み処理 [r\\_dalin\\_interrupt\\_receive](#) のコールバック・ルーチン ([R\\_DALIn\\_Receive](#) の引数  $rx\_num$  で指定された数以上のデータを受信した際の処理) として呼び出されます。

### [指定形式]

```
#include "r_cg_macrodriver.h"
static void r_dalin_callback_softwareoverrun ( uint16_t rx_data );
```

備考  $n$  はチャンネル番号を意味します。

### [引数]

| I/O | 引数                | 説明   |
|-----|-------------------|--|
| ○   | uint16_t rx_data; | 受信したデータ ( <a href="#">R_DALIn_Receive</a> の引数 $rx\_num$ で指定された数以上に受信したデータ) |

### [戻り値]

なし

### 3.2.31 アシクロナス・シリアル・インタフェース LIN-UART

以下に、コード生成がアシクロナス・シリアル・インタフェース LIN-UART 用として出力する API 関数の一覧を示します。

表 3.31 アシクロナス・シリアル・インタフェース LIN-UART 用 API 関数

| API 関数名                           | 機能概要  |
|-----------------------------------|---|
| R_UARTFn_Create                   | アシクロナス・シリアル・インタフェース LIN-UART を制御するうえで必要となる初期化処理を行います。 |
| R_UARTFn_Create_UserInit          | アシクロナス・シリアル・インタフェース LIN-UART に関するユーザ独自の初期化処理を行います。    |
| r_uartfn_interrupt_send           | LIN-UART 送信完了割り込み INTLT の発生に伴う処理を行います。                |
| r_uartfn_interrupt_receive        | LIN-UART 受信完了割り込み INTLR の発生に伴う処理を行います。                |
| r_uartfn_interrupt_error          | LIN-UART 受信ステータス割り込み INTLS の発生に伴う処理を行います。             |
| R_UARTFn_Start                    | LIN 通信を待機状態にします。                                      |
| R_UARTFn_Stop                     | LIN 通信を終了します。   |
| R_UARTFn_Set_PowerOff             | アシクロナス・シリアル・インタフェース LIN-UART に対するクロック供給を停止します。        |
| R_UARTFn_Send                     | データの UARTF 送信を開始します。                                  |
| R_UARTFn_Receive                  | データの UARTF 受信を開始します。                                  |
| R_UARTFn_Set_DataComparisonOn     | データの比較を開始します。   |
| R_UARTFn_Set_DataComparisonOff    | データの比較を終了します。   |
| r_uartfn_callback_sendend         | LIN-UART 送信完了割り込み INTLT の発生に伴う処理を行います。                |
| r_uartfn_callback_receiveend      | LIN-UART 受信完了割り込み INTLR の発生に伴う処理を行います。                |
| r_uartfn_callback_error           | LIN-UART 受信ステータス割り込み INTLS の発生に伴う処理を行います。             |
| r_uartfn_callback_softwareoverrun | オーバラン・エラーの検出に伴う処理を行います。                               |
| r_uartfn_callback_expbitdetect    | 拡張ビットの検出に伴う処理を行います。                                   |
| r_uartfn_callback_idmatch         | ID パリティの一致に伴う処理を行います。                                 |

## R\_UARTFn\_Create

アシンクロナス・シリアル・インタフェース LIN-UART を制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_UARTFn_Create ( void );
```

備考  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_UARTFn\_Create\_UserInit

アシンクロナス・シリアル・インタフェース LIN-UART に関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_UARTFn\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_UARTFn_Create_UserInit ( void );
```

備考 *n* は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## r\_uartfn\_interrupt\_send

LIN-UART 送信完了割り込み INTLT の発生に伴う処理を行います。

備考 本 API 関数は、LIN-UART 送信完了割り込み INTLT に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_uartfn_interrupt_send ( void );
```

CC-RL コンパイラの場合

```
static void __near r_uartfn_interrupt_send ( void );
```

備考 *n* は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## r\_uartfn\_interrupt\_receive

LIN-UART 受信完了割り込み INTLR の発生に伴う処理を行います。

備考 本 API 関数は、LIN-UART 受信完了割り込み INTLR に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_uartfn_interrupt_receive ( void );
```

CC-RL コンパイラの場合

```
static void __near r_uartfn_interrupt_receive ( void );
```

備考  $n$  は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## r\_uartfn\_interrupt\_error

LIN-UART 受信ステータス割り込み INTLS の発生に伴う処理を行います。

備考 本 API 関数は、LIN-UART 受信ステータス割り込み INTLS に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_uartfn_interrupt_error ( void );
```

CC-RL コンパイラの場合

```
static void __near r_uartfn_interrupt_error ( void );
```

備考  $n$  は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし



## R\_UARTFn\_Start

LIN 通信を待機状態にします。

### [指定形式]

```
void R_UARTFn_Start ( void );
```

備考  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

**R\_UARTFn\_Stop**

LIN 通信を終了します。

**[指定形式]**

```
void R_UARTFn_Stop ( void );
```

備考  $n$ は、チャネル番号を意味します。

**[引数]**

なし

**[戻り値]**

なし

## R\_UARTFn\_Set\_PowerOff

アシンクロナス・シリアル・インタフェース LIN-UART に対するクロック供給を停止します。

備考 本 API 関数の呼び出しにより、アシンクロナス・シリアル・インタフェース LIN-UART はリセット状態へと移行します。  
このため、本 API 関数の呼び出し後、制御レジスタへの書き込みは無視されます。

### [指定形式]

```
void R_UARTFn_Set_PowerOff ( void );
```

備考  $n$  は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_UARTFn\_Send

データの UARTF 送信を開始します。

- 備考 1. 本 API 関数では、引数 *tx\_buf* で指定されたバッファから 1 バイト単位の UARTF 送信を引数 *tx\_num* で指定された回数だけ繰り返し行います。
- 備考 2. UARTF 送信を行う際には、本 API 関数の呼び出し以前に [R\\_UARTFn\\_Start](#) を呼び出す必要があります。
- 備考 3. アシンクロナス・シリアル・インタフェース LIN-UART を拡張ビット・モードで使用する場合、引数 *tx\_buf* で指定された場合には、送信するデータを以下の形式で格納します。  
 “8 ビット・データ”, “拡張ビット”, “8 ビット・データ”, “拡張ビット”, ...

### [指定形式]

```
MD_STATUS R_UARTFn_Send ( uint8_t * const tx_buf, uint16_t tx_num );
```

備考 *n* は、チャンネル番号を意味します。

### [引数]

| I/O | 引数                              | 説明                     |
|-----|---------------------------------|------------------------|
| I   | uint8_t * const <i>tx_buf</i> ; | 送信するデータを格納したバッファへのポインタ |
| I   | uint16_t <i>tx_num</i> ;        | 送信するデータの総数             |

### [戻り値]

| マクロ           | 説明       |
|---------------|----------|
| MD_OK         | 正常終了     |
| MD_ARGERROR   | 引数の指定が不正 |
| MD_DATAEXISTS | 送信処理を実行中 |

## R\_UARTFn\_Receive

データの UARTF 受信を開始します。

- 備考 1. 本 API 関数では、1 バイト単位の UARTF 受信を引数 *rx\_num* で指定された回数だけ繰り返し行い、引数 *rx\_buf* で指定されたバッファに格納します。
- 備考 2. 実際の UARTF 受信は、本 API 関数の呼び出し後、[R\\_UARTFn\\_Start](#) を呼び出すことにより開始されます。
- 備考 3. アシクロナス・シリアル・インタフェース LIN-UART を拡張ビット・モードで使用する場合、引数 *rx\_buf* で指定された場合には、受信したデータが以下の形式で格納されます。  
 “8 ビット・データ”, “拡張ビット”, “8 ビット・データ”, “拡張ビット”, ...

### [指定形式]

```
MD_STATUS R_UARTFn_Receive ( uint8_t * const rx_buf, uint16_t rx_num );
```

備考 *n* は、チャンネル番号を意味します。

### [引数]

| I/O | 引数                              | 説明                     |
|-----|---------------------------------|------------------------|
| O   | uint8_t * const <i>rx_buf</i> ; | 受信したデータを格納するバッファへのポインタ |
| I   | uint16_t <i>rx_num</i> ;        | 受信するデータの総数             |

### [戻り値]

| マクロ         | 説明       |
|-------------|----------|
| MD_OK       | 正常終了     |
| MD_ARGERROR | 引数の指定が不正 |

## R\_UARTFn\_Set\_DataComparisonOn

データの比較を開始します。

備考 本 API 関数の呼び出しにより、アシンクロナス・シリアル・インタフェース LIN-UART は拡張ビット・モード（データ比較あり）へと移行します。

### [指定形式]

```
void R_UARTFn_Set_DataComparisonOn ( void );
```

備考  $n$  は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_UARTFn\_Set\_DataComparisonOff

データの比較を終了します。

備考 本API関数の呼び出しにより、アシンクロナス・シリアル・インタフェース LIN-UART は拡張ビット・モード（データ比較なし）へと移行します。

### [指定形式]

```
void R_UARTFn_Set_DataComparisonOff ( void );
```

備考 *n*は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## r\_uartfn\_callback\_sendend

LIN-UART 送信完了割り込み INTLT の発生に伴う処理を行います。

備考 本 API 関数は、LIN-UART 送信完了割り込み INTLT に対応した割り込み処理 `r_uartfn_interrupt_send` のコールバック・ルーチン (`R_UARTFN_Send` の引数 `tx_num` で指定された数のデータ送信が完了した際の処理) として呼び出されます。

### [指定形式]

```
static void r_uartfn_callback_sendend ( void );
```

備考 `n` は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし



## r\_uartfn\_callback\_receiveend

LIN-UART 受信完了割り込み INTLR の発生に伴う処理を行います。

備考 本 API 関数は、LIN-UART 受信完了割り込み INTLR に対応した割り込み処理 [r\\_uartfn\\_interrupt\\_receive](#) のコールバック・ルーチン ([R\\_UARTFn\\_Receive](#) の引数 *rx\_num* で指定された数のデータ受信が完了した際の処理) として呼び出されます。

### [指定形式]

```
static void r_uartfn_callback_receiveend ( void );
```

備考 *n* は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

**r\_uartfn\_callback\_error**

LIN-UART 受信ステータス割り込み INTLS の発生に伴う処理を行います。

備考 本 API 関数は、LIN-UART 受信ステータス割り込み INTLS に対応した割り込み処理 [r\\_uartfn\\_interrupt\\_error](#) のコールバック・ルーチンとして呼び出されます。

**[指定形式]**

```
static void r_uartfn_callback_error ( uint8_t err_type );
```

備考 *n* は、チャンネル番号を意味します。

**[引数]**

| I/O | 引数                | 説明   |
|-----|-------------------|--|
| O   | uint8_t err_type; | LIN-UART 受信ステータス割り込みの発生要因<br>00000xx1B : オーバラン・エラー<br>00000x1xB : パリティ・エラー<br>000001xxB : フレーミング・エラー |

**[戻り値]**

なし

## r\_uartfn\_callback\_softwareoverrun

オーバーラン・エラーの検出に伴う処理を行います。

備考 本 API 関数は、LIN-UART 受信完了割り込み INTLR に対応した割り込み処理 [r\\_uartfn\\_interrupt\\_receive](#) のコールバック・ルーチン ([R\\_UARTFn\\_Receive](#) の引数 *rx\_num* で指定された数のデータ受信が完了した際の処理) として呼び出されます。

### [指定形式]

```
static void r_uartfn_callback_softwareoverrun ( void );
```

備考 *n* は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## r\_uartfn\_callback\_expbitdetect

拡張ビットの検出に伴う処理を行います。

備考 本 API 関数は、LIN-UART 受信ステータス割り込み INTLS に対応した割り込み処理 [r\\_uartfn\\_interrupt\\_error](#) のコールバック・ルーチン（拡張ビットを検出した際の処理）として呼び出されます。

### [指定形式]

```
static void r_uartfn_callback_expbitdetect ( void );
```

備考 *n* は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## r\_uartfn\_callback\_idmatch

ID パリティの一致に伴う処理を行います。

備考 本 API 関数は、LIN-UART 受信ステータス割り込み INTLS に対応した割り込み処理 [r\\_uartfn\\_interrupt\\_error](#) のコールバック・ルーチン (ID パリティが一致した際の処理) として呼び出されます。

### [指定形式]

```
static void r_uartfn_callback_idmatch ( void );
```

備考 *n* は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## 3.2.32 シリアル・インタフェース IICA

以下に、コード生成がシリアル・インタフェース IICA 用として出力する API 関数の一覧を示します。

表 3.32 シリアル・インタフェース IICA 用 API 関数

| API 関数名                            | 機能概要                                       |
|------------------------------------|--|
| R_IICAn_Create                     | シリアル・インタフェース IICA を制御するうえで必要となる初期化処理を行います。 |
| R_IICAn_Create_UserInit            | シリアル・インタフェース IICA に関するユーザ独自の初期化処理を行います。    |
| r_iican_interrupt                  | IICA 通信完了割り込み INTIICAn の発生に伴う処理を行います。      |
| R_IICAn_StopCondition              | ストップ・コンディションを発生させます。                       |
| R_IICAn_Stop                       | IICA 通信を終了します。                             |
| R_IICAn_Reset                      | シリアル・インタフェース IICA をリセットします。                |
| R_IICAn_Set_PowerOff               | シリアル・インタフェース IICA に対するクロック供給を停止します。        |
| R_IICAn_Master_Send                | IICA マスタ送信を開始します。                          |
| R_IICAn_Master_Receive             | IICA マスタ受信を開始します。                          |
| r_iican_callback_master_sendend    | IICA マスタ送信完了割り込み INTIICAn の発生に伴う処理を行います。   |
| r_iican_callback_master_receiveend | IICA マスタ受信完了割り込み INTIICAn の発生に伴う処理を行います。   |
| r_iican_callback_master_error      | IICA マスタ通信エラーの検出に伴う処理を行います。                |
| R_IICAn_Slave_Send                 | IICA スレーブ送信を開始します。                         |
| R_IICAn_Slave_Receive              | IICA スレーブ受信を開始します。                         |
| r_iican_callback_slave_sendend     | IICA スレーブ送信完了割り込み INTIICAn の発生に伴う処理を行います。  |
| r_iican_callback_slave_receiveend  | IICA スレーブ受信完了割り込み INTIICAn の発生に伴う処理を行います。  |
| r_iican_callback_slave_error       | IICA スレーブ通信エラーの検出に伴う処理を行います。               |
| r_iican_callback_getstopcondition  | ストップ・コンディションの検出に伴う処理を行います。                 |
| R_IICAn_Set_SnoozeOn               | STOP モード時のアドレス一致ウエイクアップ機能の動作を許可します。        |
| R_IICAn_Set_SnoozeOff              | STOP モード時のアドレス一致ウエイクアップ機能の動作を禁止します。        |
| R_IICAn_Set_WakeupOn               | STOP モード時のアドレス一致ウエイクアップ機能の動作を許可します。        |
| R_IICAn_Set_WakeupOff              | STOP モード時のアドレス一致ウエイクアップ機能の動作を禁止します。        |

## R\_IICAn\_Create

シリアル・インタフェース IICA を制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_IICAn_Create ( void );
```

備考  $n$ はチャネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_IICAn\_Create\_UserInit

シリアル・インタフェース IICA に関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_IICAn\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_IICAn_Create_UserInit ( void );
```

備考 *n* はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし



## r\_iican\_interrupt

IICA 通信完了割り込み INTIICAn の発生に伴う処理を行います。

備考 本 API 関数は、IICA 通信完了割り込み INTIICAn に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_iican_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_iican_interrupt ( void );
```

備考 *n* はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_IICAn\_StopCondition

ストップ・コンディションを発生させます。

備考 本 API 関数を呼出し後 IICA の動作を停止する前には、SPD0 ビットでストップコンディション検出されたことを確認してください。

### [指定形式]

```
void R_IICAn_StopCondition ( void );
```

備考 *n* はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

**R\_IICAn\_Stop**

IICA 通信を終了します。

**[指定形式]**

```
void R_IICAn_Stop ( void );
```

備考 *n*はチャネル番号を意味します。

**[引数]**

なし

**[戻り値]**

なし

## R\_IICAn\_Reset

シリアル・インタフェース IICA をリセットします。

### [指定形式]

```
void R_IICAn_Reset ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_IICAn\_Set\_PowerOff

シリアル・インタフェース IICA に対するクロック供給を停止します。

備考 本 API 関数の呼び出しにより、シリアル・インタフェース IICA はリセット状態へと移行します。  
このため、本 API 関数の呼び出し後、制御レジスタ（IICA コントロール・レジスタ  $n$ : IICCTL $n$  など）への書き込みは無視されます。

### [指定形式]

```
void R_IICAn_Set_PowerOff ( void );
```

備考  $n$  はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_IICAn\_Master\_Send

IICA マスタ送信を開始します。

備考 本 API 関数では、引数 *tx\_buf* で指定されたバッファから 1 バイト単位の IICA マスタ送信を引数 *tx\_num* で指定された回数だけ繰り返し行います。

### [指定形式]

```
#include "r_cg_macrodriver.h"
MD_STATUS R_IICAn_Master_Send ( uint8_t adr, uint8_t * const tx_buf, uint16_t tx_num,
uint8_t wait );
```

備考 *n* はチャネル番号を意味します。

### [引数]

| I/O | 引数                              | 説明                     |
|-----|---------------------------------|------------------------|
| I   | uint8_t <i>adr</i> ;            | スレーブ・アドレス              |
| I   | uint8_t * const <i>tx_buf</i> ; | 送信するデータを格納したバッファへのポインタ |
| I   | uint16_t <i>tx_num</i> ;        | 送信するデータの総数             |
| I   | uint8_t <i>wait</i> ;           | スタート・コンディションのセットアップ時間  |

### [戻り値]

| マクロ       | 説明      |
|-----------|---------|
| MD_OK     | 正常終了    |
| MD_ERROR1 | バス通信状態  |
| MD_ERROR2 | バス未解放状態 |

## R\_IICAn\_Master\_Receive

IICA マスタ受信を開始します。

備考 本 API 関数では、1 バイト単位の IICA マスタ受信を引数 *rx\_num* で指定された回数だけ繰り返し行い、引数 *rx\_buf* で指定されたバッファに格納します。

### [指定形式]

```
#include "r_cg_macrodriver.h"
MD_STATUS R_IICAn_Master_Receive ( uint8_t adr, uint8_t * const rx_buf, uint16_t
rx_num, uint8_t wait );
```

備考 *n* はチャネル番号を意味します。

### [引数]

| I/O | 引数                              | 説明                     |
|-----|---------------------------------|------------------------|
| I   | uint8_t <i>adr</i> ;            | スレーブ・アドレス              |
| O   | uint8_t * const <i>rx_buf</i> ; | 受信したデータを格納するバッファへのポインタ |
| I   | uint16_t <i>rx_num</i> ;        | 受信するデータの総数             |
| I   | uint8_t <i>wait</i> ;           | スタート・コンディションのセットアップ時間  |

### [戻り値]

| マクロ       | 説明      |
|-----------|---------|
| MD_OK     | 正常終了    |
| MD_ERROR1 | バス通信状態  |
| MD_ERROR2 | バス未解放状態 |

## r\_iican\_callback\_master\_sendend

IICA マスタ送信完了割り込み INTIICAn の発生に伴う処理を行います。

備考 本 API 関数は、IICA マスタ送信完了割り込み INTIICAn に対応した割り込み処理 [r\\_iican\\_interrupt](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
static void r_iican_callback_master_sendend ( void );
```

備考 *n* はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし



## r\_iican\_callback\_master\_receiveend

IICA マスタ受信完了割り込み INTIICAn の発生に伴う処理を行います。

備考 本 API 関数は、IICA マスタ受信完了割り込み INTIICAn に対応した割り込み処理 `r_iican_interrupt` のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
static void r_iican_callback_master_receiveend ( void );
```

備考 *n* はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

**r\_iican\_callback\_master\_error**

IICA マスタ通信エラーの検出に伴う処理を行います。

**[指定形式]**

```
#include "r_cg_macrodriver.h"
static void r_iican_callback_master_error ( MD_STATUS flag );
```

備考 *n* はチャネル番号を意味します。

**[引数]**

| I/O | 引数                      | 説明  |
|-----|-------------------------|---|
| I   | MD_STATUS <i>flag</i> ; | 通信エラーの発生要因<br>MD_SPT : ストップ・コンディションの検出<br>MD_NACK : アクノリッジの未検出<br>(アドレス一致のスレーブがない、またはスレーブがデータ受信できない / 次のデータを必要としない場合) |

**[戻り値]**

なし

## R\_IICAn\_Slave\_Send

IICA スレーブ送信を開始します。

備考 本 API 関数では、引数 *tx\_buf* で指定されたバッファから 1 バイト単位の IICA スレーブ送信を引数 *tx\_num* で指定された回数だけ繰り返し行います。

### [指定形式]

```
#include "r_cg_macrodriver.h"
void R_IICAn_Slave_Send ( uint8_t * const tx_buf, uint16_t tx_num );
```

備考 *n* はチャンネル番号を意味します。

### [引数]

| I/O | 引数                              | 説明                     |
|-----|---------------------------------|------------------------|
| I   | uint8_t * const <i>tx_buf</i> ; | 送信するデータを格納したバッファへのポインタ |
| I   | uint16_t <i>tx_num</i> ;        | 送信するデータの総数             |

### [戻り値]

なし

## R\_IICAn\_Slave\_Receive

IICA スレーブ受信を開始します。

備考 本 API 関数では、1 バイト単位の IICA スレーブ受信を引数 *rx\_num* で指定された回数だけ繰り返し行い、引数 *rx\_buf* で指定されたバッファに格納します。

### [指定形式]

```
#include "r_cg_macrodriver.h"
void R_IICAn_Slave_Receive ( uint8_t * const rx_buf, uint16_t rx_num );
```

備考 *n* はチャンネル番号を意味します。

### [引数]

| I/O | 引数                              | 説明                     |
|-----|---------------------------------|------------------------|
| O   | uint8_t * const <i>rx_buf</i> ; | 受信したデータを格納するバッファへのポインタ |
| I   | uint16_t <i>rx_num</i> ;        | 受信するデータの総数             |

### [戻り値]

なし

## r\_iican\_callback\_slave\_sendend

IICA スレーブ送信完了割り込み INTIICAn の発生に伴う処理を行います。

備考 本 API 関数は、IICA スレーブ送信完了割り込み INTIICAn に対応した割り込み処理 `r_iican_interrupt` のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
static void r_iican_callback_slave_sendend ( void );
```

備考 *n* はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## r\_iican\_callback\_slave\_receiveend

IICA スレーブ受信完了割り込み INTIICAn の発生に伴う処理を行います。

備考 本 API 関数は、IICA スレーブ受信完了割り込み INTIICAn に対応した割り込み処理 `r_iican_interrupt` のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
static void r_iican_callback_slave_receiveend ( void );
```

備考 *n* はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

**r\_iican\_callback\_slave\_error**

IICA スレーブ通信エラーの検出に伴う処理を行います。

## [指定形式]

```
#include "r_cg_macrodriver.h"
static void r_iican_callback_slave_error ( MD_STATUS flag );
```

備考 *n*はチャンネル番号を意味します。

## [引数]

| I/O | 引数                      | 説明   |
|-----|-------------------------|--|
| I   | MD_STATUS <i>flag</i> ; | 通信エラーの発生要因<br>MD_ERROR : アドレス不一致の検出<br>MD_NACK : アクノリッジの未検出<br>(マスタ受信終了) |

## [戻り値]

なし

**r\_iican\_callback\_getstopcondition**

ストップ・コンディションの検出に伴う処理を行います。

**[指定形式]**

```
static void r_iican_callback_getstopcondition ( void );
```

備考 *n*はチャネル番号を意味します。

**[引数]**

なし

**[戻り値]**

なし



## R\_IICAn\_Set\_SnoozeOn

STOP モード時のアドレス一致ウエイクアップ機能の動作を許可します。

### [指定形式]

```
void R_IICAn_Set_SnoozeOn ( void );
```

備考 *n*はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_IICAn\_Set\_SnoozeOff

STOP モード時のアドレス一致ウエイクアップ機能の動作を禁止します。

### [指定形式]

```
void R_IICAn_Set_SnoozeOff ( void );
```

備考 *n*はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_IICAn\_Set\_WakeupOn

STOP モード時のアドレス一致ウエイクアップ機能の動作を許可します。

### [指定形式]

```
void R_IICAn_Set_WakeupOn ( void );
```

備考 *n*はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_IICAn\_Set\_WakeupOff

STOP モード時のアドレス一致ウエイクアップ機能の動作を禁止します。

### [指定形式]

```
void R_IICAn_Set_WakeupOff ( void );
```

備考 *n*はチャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

### 3.2.33 LCD コントローラ／ドライバ

以下に、コード生成がLCDコントローラ／ドライバ用として出力するAPI関数の一覧を示します。

表 3.33 LCD コントローラ／ドライバ用 API 関数

| API 関数名               | 機能概要                                    |
|-----------------------|---|
| R_LCD_Create          | LCD コントローラ／ドライバを制御するうえで必要となる初期化処理を行います。 |
| R_LCD_Create_UserInit | LCD コントローラ／ドライバに関するユーザ独自の初期化処理を行います。    |
| r_lcd_interrupt       | LCD フレーム割り込み INTLCD の発生に伴う処理を行います。      |
| R_LCD_Start           | LCD コントローラ／ドライバを表示オン状態にします。             |
| R_LCD_Stop            | LCD コントローラ／ドライバを表示オフ状態にします。             |
| R_LCD_Set_VoltageOn   | 内部昇圧回路、および容量分割回路を動作許可状態にします。            |
| R_LCD_Set_VoltageOff  | 内部昇圧回路、および容量分割回路を動作停止状態にします。            |
| R_LCD_Set_PowerOff    | LCD コントローラ／ドライバに対するクロック供給を停止します。        |
| R_LCD_VoltageOn       | 内部昇圧回路、および容量分割回路を動作許可状態にします。            |
| R_LCD_VoltageOff      | 内部昇圧回路、および容量分割回路を動作停止状態にします。            |

**R\_LCD\_Create**

LCDコントローラ／ドライバを制御するうえで必要となる初期化処理を行います。

**[指定形式]**

```
void R_LCD_Create ( void );
```

**[引数]**

なし

**[戻り値]**

なし

## R\_LCD\_Create\_UserInit

LCD コントローラ／ドライバに関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_LCD\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_LCD_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

## r\_lcd\_interrupt

LCD フレーム割り込み INTLCD の発生に伴う処理を行います。

備考 本 API 関数は、LCD フレーム割り込み INTLCD に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_lcd_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_lcd_interrupt ( void );
```

### [引数]

なし

### [戻り値]

なし



**R\_LCD\_Start**

LCD コントローラ／ドライバを表示オン状態にします。

**[指定形式]**

```
void R_LCD_Start ( void );
```

**[引数]**

なし

**[戻り値]**

なし

## R\_LCD\_Stop

LCDコントローラ／ドライバを表示オフ状態にします。

### [指定形式]

```
void R_LCD_Stop ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_LCD\_Set\_VoltageOn

内部昇圧回路、および容量分割回路を動作可能状態にします。

### [指定形式]

```
void R_LCD_Set_VoltageOn ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_LCD\_Set\_VoltageOff

内部昇圧回路、および容量分割回路を動作停止状態にします。

### [指定形式]

```
void R_LCD_Set_VoltageOff ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_LCD\_Set\_PowerOff

LCD コントローラ／ドライバに対するクロック供給を停止します。

- 備考 1. 本 API 関数の呼び出しにより、LCD コントローラ／ドライバはリセット状態へと移行します。このため、本 API 関数の呼び出し後、制御レジスタへの書き込みは無視されます。
- 備考 2. 本 API 関数では、周辺イネーブル・レジスタ *n* の RTCEN ビットを操作することにより、LCD コントローラ／ドライバに対するクロック供給の停止を実現しています。このため、本 API 関数の呼び出しを行った際には、RTCEN ビットを共用している他の周辺装置（リアルタイム・クロックなど）に対するクロック供給も停止することになります。

### [指定形式]

```
void R_LCD_Set_PowerOff ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_LCD\_VoltageOn

内部昇圧回路、および容量分割回路を動作可能状態にします。

### [指定形式]

```
void R_LCD_VoltageOn ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_LCD\_VoltageOff

内部昇圧回路、および容量分割回路を動作停止状態にします。

### [指定形式]

```
void R_LCD_VoltageOff ( void );
```

### [引数]

なし

### [戻り値]

なし

### 3.2.34 サウンド・ジェネレータ

以下に、コード生成がサウンド・ジェネレータ用として出力する API 関数の一覧を示します。

表 3.34 サウンド・ジェネレータ用 API 関数

| API 関数名                              | 機能概要  |
|--------------------------------------|---|
| <a href="#">R_SG_Create</a>          | サウンド・ジェネレータを制御するうえで必要となる初期化処理を行います。             |
| <a href="#">R_SG_Create_UserInit</a> | サウンド・ジェネレータに関するユーザ独自の初期化処理を行います。                |
| <a href="#">r_sg_interrupt</a>       | 対数減衰率のスレッシュ・ホールド値検出による割り込み INTSG の発生に伴う処理を行います。 |
| <a href="#">R_SG_Start</a>           | サウンド・ジェネレータを動作許可状態にします。                         |
| <a href="#">R_SG_Stop</a>            | サウンド・ジェネレータを動作停止状態にします。                         |



## R\_SG\_Create

サウンド・ジェネレータを制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_SG_Create ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_SG\_Create\_UserInit

サウンド・ジェネレータに関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_SG\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_SG_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

## r\_sg\_interrupt

対数減衰率のスレッシュ・ホールド値検出による割り込み INTSG の発生に伴う処理を行います。

備考 本 API 関数は、対数減衰率のスレッシュ・ホールド値検出による割り込み INTSG に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_sg_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_sg_interrupt ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_SG\_Start

サウンド・ジェネレータを動作許可状態にします。

### [指定形式]

```
void R_SG_Start ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_SG\_Stop

サウンド・ジェネレータを動作停止状態にします。

### [指定形式]

```
void R_SG_Stop ( void );
```

### [引数]

なし

### [戻り値]

なし

### 3.2.35 DMA コントローラ

以下に、コード生成がDMAコントローラ用として出力するAPI関数の一覧を示します。

表 3.35 DMA コントローラ用 API 関数

| API 関数名                                       | 機能概要                                   |
|---|--|
| <a href="#">R_DMACn_Create</a>                | DMA コントローラを制御するうえで必要となる初期化処理を行います。     |
| <a href="#">R_DMACn_Create_UserInit</a>       | DMA コントローラに関するユーザ独自の初期化処理を行います。        |
| <a href="#">R_DMAC_Create</a>                 | DMA コントローラを制御するうえで必要となる初期化処理を行います。     |
| <a href="#">R_DMAC_Create_UserInit</a>        | DMA コントローラに関するユーザ独自の初期化処理を行います。        |
| <a href="#">r_dmacn_interrupt</a>             | DMA 転送終了割り込み INTDMA $n$ の発生に伴う処理を行います。 |
| <a href="#">R_DMACn_Start</a>                 | チャンネル $n$ を動作許可状態に設定します。               |
| <a href="#">R_DMACn_Stop</a>                  | チャンネル $n$ を動作停止状態に設定します。               |
| <a href="#">R_DMACn_Set_SoftwareTriggerOn</a> | DMA 転送を開始します。                          |

## R\_DMAn\_Create

DMA コントローラを制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_DMAn_Create ( void );
```

備考 *n*は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_DMAc<sub>n</sub>\_Create\_UserInit

DMA コントローラに関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_DMAc<sub>n</sub>\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_DMAcn_Create_UserInit ( void );
```

備考 *n*は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし



## R\_DMAC\_Create

DMA コントローラを制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_DMAC_Create ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_DMACE\_Create\_UserInit

DMA コントローラに関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_DMACE\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_DMACE_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

## r\_dmacn\_interrupt

DMA 転送終了割り込み INTDMAn の発生に伴う処理を行います。

備考 本 API 関数は、DMA 転送終了割り込み INTDMAn に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_dmacn_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_dmacn_interrupt ( void );
```

備考 *n* は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_DMAn\_Start

チャンネル  $n$  を動作許可状態に設定します。

### [指定形式]

```
void R_DMAn_Start ( void );
```

備考  $n$  は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_DMAn\_Stop

チャンネル  $n$  を動作停止状態に設定します。

備考 1. 本 API 関数は、DMA 転送を強制終了させるものではありません。

備考 2. 本 API 関数は、“転送終了”の確認後に呼び出す必要があります。

### [指定形式]

```
void R_DMAn_Stop ( void );
```

備考  $n$  は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_DMAn\_Set\_SoftwareTriggerOn

DMA 転送を開始します。

### [指定形式]

```
void R_DMAn_Set_SoftwareTriggerOn ( void );
```

備考  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

### 3.2.36 データ・トランスファ・コントローラ

以下に、コード生成がデータ・トランスファ・コントローラ用として出力する API 関数の一覧を示します。

表 3.36 データ・トランスファ・コントローラ用 API 関数

| API 関数名                               | 機能概要                                      |
|---------------------------------------|---|
| <a href="#">R_DTC_Create</a>          | データ・トランスファ・コントローラを制御するうえで必要となる初期化処理を行います。 |
| <a href="#">R_DTC_Create_UserInit</a> | データ・トランスファ・コントローラに関するユーザ独自の初期化処理を行います。    |
| <a href="#">R_DTCn_Start</a>          | データ・トランスファ・コントローラを動作可能状態にします。             |
| <a href="#">R_DTCn_Stop</a>           | データ・トランスファ・コントローラを動作停止状態にします。             |
| <a href="#">R_DTC_Set_PowerOff</a>    | データ・トランスファ・コントローラに対するクロック供給を停止します。        |
| <a href="#">R_DTCDn_Start</a>         | データ・トランスファ・コントローラを動作可能状態にします。             |
| <a href="#">R_DTCDn_Stop</a>          | データ・トランスファ・コントローラを動作停止状態にします。             |

## R\_DTC\_Create

データ・トランスファ・コントローラを制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_DTC_Create ( void );
```

### [引数]

なし

### [戻り値]

なし



## R\_DTC\_Create\_UserInit

データ・トランスファ・コントローラに関するユーザ独自の初期化処理を行います。

備考 本API関数は、[R\\_DTC\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_DTC_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_DTCn\_Start

データ・トランスファ・コントローラを動作可能状態にします。

### [指定形式]

```
void R_DTCn_Start ( void );
```

備考  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_DTCn\_Stop

データ・トランスファ・コントローラを動作停止状態にします。

### [指定形式]

```
void R_DTCn_Stop ( void );
```

備考  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_DTC\_Set\_PowerOff

データ・トランスファ・コントローラに対するクロック供給を停止します。

**備考**           本 API 関数の呼び出しにより、データ・トランスファ・コントローラはリセット状態へと移行します。  
このため、本 API 関数の呼び出し後、制御レジスタへの書き込みは無視されます。

### [指定形式]

```
void R_DTC_Set_PowerOff ( void );
```

### [引数]

なし

### [戻り値]

なし

**R\_DTCDn\_Start**

データ・トランスファ・コントローラを動作可能状態にします。

**[指定形式]**

```
void R_DTCDn_Start ( void );
```

備考  $n$ は、チャンネル番号を意味します。

**[引数]**

なし

**[戻り値]**

なし

## R\_DTCDn\_Stop

データ・トランスファ・コントローラを動作停止状態にします。

### [指定形式]

```
void R_DTCDn_Stop ( void );
```

備考  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

### 3.2.37 イベントリンクコントローラ

以下に、コード生成がイベントリンクコントローラ用として出力する API 関数の一覧を示します。

表 3.37 イベントリンクコントローラ用 API 関数

| API 関数名                               | 機能概要                                  |
|---------------------------------------|---------------------------------------|
| <a href="#">R_ELC_Create</a>          | イベントリンクコントローラを制御するうえで必要となる初期化処理を行います。 |
| <a href="#">R_ELC_Create_UserInit</a> | イベントリンクコントローラに関するユーザ独自の初期化処理を行います。    |
| <a href="#">R_ELC_Stop</a>            | イベントリンクコントローラを動作停止状態にします。             |

## R\_ELC\_Create

イベントリンクコントローラを制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_ELC_Create ( void );
```

### [引数]

なし

### [戻り値]

なし



## R\_ELC\_Create\_UserInit

イベントリンクコントローラに関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_ELC\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_ELC_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

**R\_ELC\_Stop**

イベントリンクコントローラを動作停止状態にします。

**[指定形式]**

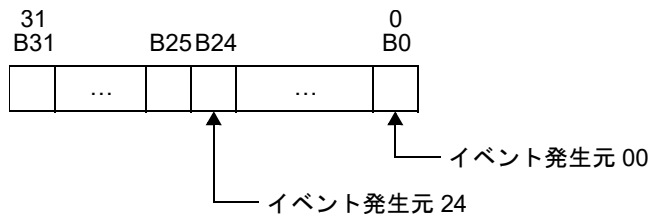
```
void R_ELC_Stop ( uint32_t event );
```

**[引数]**

| I/O | 引数              | 説明          |
|-----|-----------------|-------------|
| I   | uint32_t event; | 停止するイベント発生元 |

**備考**

以下に、停止するイベント発生元 *event* の指定形式を示します。  
 なお、*event* に 0x01010101 を設定した場合、イベント発生元 00, 08, 16, 24 のイベントリンク動作が禁止されます。

**[戻り値]**

なし

## 3.2.38 割り込み機能

以下に、コード生成が割り込み機能用として出力する API 関数の一覧を示します。

表 3.38 割り込み機能用 API 関数

| API 関数名                | 機能概要  |
|------------------------|---|
| R_INTC_Create          | 割り込み機能を制御するうえで必要となる初期化処理を行います。                    |
| R_INTC_Create_UserInit | 割り込み機能に関するユーザ独自の初期化処理を行います。                       |
| r_intcn_interrupt      | 外部マスカブル割り込み INTP <sub>n</sub> の発生に伴う処理を行います。      |
| R_INTCn_Start          | 外部マスカブル割り込み INTP <sub>n</sub> の受け付けを許可します。        |
| R_INTCn_Stop           | 外部マスカブル割り込み INTP <sub>n</sub> の受け付けを禁止します。        |
| r_intclrn_interrupt    | 外部マスカブル割り込み INTPLR <sub>n</sub> の発生に伴う処理を行います。    |
| R_INTCLRn_Start        | 外部マスカブル割り込み INTPLR <sub>n</sub> の受け付けを許可します。      |
| R_INTCLRn_Stop         | 外部マスカブル割り込み INTPLR <sub>n</sub> の受け付けを禁止します。      |
| r_intrtcicn_interrupt  | 外部マスカブル割り込み INTRTCIC <sub>n</sub> の発生に伴う処理を行います。  |
| R_INTRTCICn_Start      | 外部マスカブル割り込み INTRTCIC <sub>n</sub> の受け付けを許可します。    |
| R_INTRTCICn_Stop       | 外部マスカブル割り込み INTRTCIC <sub>n</sub> の受け付けを禁止します。    |
| R_INTFO_Start          | 外部マスカブル割り込み INTFO の受け付けを許可します。                    |
| R_INTFO_Stop           | 外部マスカブル割り込み INTFO の受け付けを禁止します。                    |
| R_INTFO_ClearFlag      | 割り込みフラグ出力制御レジスタ (INTFOCTL1) の INTFCLR フラグをセットします。 |
| r_intfo_interrupt      | 外部マスカブル割り込み INTFO の発生に伴う処理を行います。                  |

## R\_INTC\_Create

割り込み機能を制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_INTC_Create ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_INTC\_Create\_UserInit

割り込み機能に関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_INTC\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_INTC_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

## r\_intcn\_interrupt

外部マスカブル割り込み INTP $n$ の発生に伴う処理を行います。

備考 本 API 関数は、外部マスカブル割り込み INTP $n$ に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_intcn_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_intcn_interrupt ( void );
```

備考  $n$ は、割り込み要因番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_INTCn\_Start

外部マスクブル割り込み INTP $n$ の受け付けを許可します。

### [指定形式]

```
void R_INTCn_Start ( void );
```

備考  $n$ は、割り込み要因番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_INTC $n$ \_Stop

外部マスカブル割り込み INTC $n$ の受け付けを禁止します。

### [指定形式]

```
void R_INTC $n$ _Stop ( void );
```

備考  $n$ は、割り込み要因番号を意味します。

### [引数]

なし

### [戻り値]

なし



## r\_intclrn\_interrupt

外部マスカブル割り込み INTPLR $n$ の発生に伴う処理を行います。

備考 本 API 関数は、外部マスカブル割り込み INTPLR $n$ に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_intclrn_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_intclrn_interrupt ( void );
```

備考  $n$ は、割り込み要因番号を意味します。

### [引数]

なし

### [戻り値]

なし

**R\_INTCLR $n$ \_Start**

外部マスカブル割り込み INTPLR $n$ の受け付けを許可します。

**[指定形式]**

```
void R_INTCLR $n$ _Start ( void );
```

備考  $n$ は、割り込み要因番号を意味します。

**[引数]**

なし

**[戻り値]**

なし

## R\_INTCLRn\_Stop

外部マスクブル割り込み INTPLR $n$ の受け付けを禁止します。

### [指定形式]

```
void R_INTCLRn_Stop ( void );
```

備考  $n$ は、割り込み要因番号を意味します。

### [引数]

なし

### [戻り値]

なし

## r\_intrtcicn\_interrupt

外部マスカブル割り込み INTRTCIC $n$ の発生に伴う処理を行います。

備考 本 API 関数は、外部マスカブル割り込み INTRTCIC $n$ に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_intrtcicn_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_intrtcicn_interrupt ( void );
```

備考  $n$ は、割り込み要因番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_INTRTCICn\_Start

外部マスクブル割り込み INTRTCICnの受け付けを許可します。

### [指定形式]

```
void R_INTRTCICn_Start ( void );
```

備考  $n$ は、割り込み要因番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_INTRTCICn\_Stop

外部マスカブル割り込み INTRTCICnの受け付けを禁止します。

### [指定形式]

```
void R_INTRTCICn_Stop ( void );
```

備考  $n$ は、割り込み要因番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_INTFO\_Start

外部マスカブル割り込み INTFO の受け付けを許可します。

### [指定形式]

```
void R_INTFO_Start ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_INTFO\_Stop

外部マスカブル割り込み INTFO の受け付けを禁止します。

### [指定形式]

```
void R_INTFO_Stop ( void );
```

### [引数]

なし

### [戻り値]

なし



## R\_INTFO\_ClearFlag

割り込みフラグ出力制御レジスタ (INTFOCTL1) の INTFCLR フラグをセットします。

### [指定形式]

```
void R_INTFO_ClearFlag ( void );
```

### [引数]

なし

### [戻り値]

なし

## r\_intfo\_interrupt

外部マスカブル割り込み INTFO の発生に伴う処理を行います。

備考 本 API 関数は、外部マスカブル割り込み INTFO に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_intfo_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_intfo_interrupt ( void );
```

### [引数]

なし

### [戻り値]

なし

### 3.2.39 キー割り込み機能

以下に、コード生成がキー割り込み機能用として出力する API 関数の一覧を示します。

表 3.39 キー割り込み機能用 API 関数

| API 関数名                               | 機能概要                             |
|---------------------------------------|----------------------------------|
| <a href="#">R_KEY_Create</a>          | キー割り込み機能を制御するうえで必要となる初期化処理を行います。 |
| <a href="#">R_KEY_Create_UserInit</a> | キー割り込み機能に関するユーザ独自の初期化処理を行います。    |
| <a href="#">r_key_interrupt</a>       | キー割り込み INTKR の発生に伴う処理を行います。      |
| <a href="#">R_KEY_Start</a>           | キー割り込み INTKR の受け付けを許可します。        |
| <a href="#">R_KEY_Stop</a>            | キー割り込み INTKR の受け付けを禁止します。        |

## R\_KEY\_Create

キー割り込み機能を制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_KEY_Create ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_KEY\_Create\_UserInit

キー割り込み機能に関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_KEY\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_KEY_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

## r\_key\_interrupt

キー割り込み INTKR の発生に伴う処理を行います。

備考 本 API 関数は、キー割り込み INTKR に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_key_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_key_interrupt ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_KEY\_Start

キー割り込み INTKR の受け付けを許可します。

### [指定形式]

```
void R_KEY_Start ( void );
```

### [引数]

なし

### [戻り値]

なし

**R\_KEY\_Stop**

キー割り込み INTKR の受け付けを禁止します。

**[指定形式]**

```
void R_KEY_Stop ( void );
```

**[引数]**

なし

**[戻り値]**

なし



## 3.2.40 電圧検出回路

以下に、コード生成が電圧検出回路用として出力する API 関数の一覧を示します。

表 3.40 電圧検出回路用 API 関数

| API 関数名                   | 機能概要  |
|---------------------------|---|
| R_LVD_Create              | 電圧検出回路を制御するうえで必要となる初期化処理を行います。              |
| R_LVD_Create_UserInit     | 電圧検出回路に関するユーザ独自の初期化処理を行います。                 |
| r_lvd_interrupt           | 電圧検出割り込み INTLVI の発生に伴う処理を行います。              |
| r_lvd_vddinterrupt        | VDD 端子電圧検出割り込み INTLVDVDD の発生に伴う処理を行います。     |
| r_lvd_vbatinterrupt       | VBAT 端子電圧検出割り込み INTLVDVBAT の発生に伴う処理を行います。   |
| r_lvd_vrtcinterrupt       | VRTC 端子電圧検出割り込み INTLVDVRTC の発生に伴う処理を行います。   |
| r_lvd_exlvdinterrupt      | EXLVD 端子電圧検出割り込み INTLVDEXLVD の発生に伴う処理を行います。 |
| R_LVD_InterruptMode_Start | 電圧検出動作を開始します（割り込みモード時、および割り込み & リセット・モード時）。 |
| R_LVD_Start_VDD           | VDD 端子の電圧検出機能を動作許可状態に設定します。                 |
| R_LVD_Start_VBAT          | VBAT 端子の電圧検出機能を動作許可状態に設定します。                |
| R_LVD_Start_VRTC          | VRTC 端子の電圧検出機能を動作許可状態に設定します。                |
| R_LVD_Start_EXLVD         | EXLVD 端子の電圧検出機能を動作許可状態に設定します。               |
| R_LVD_Stop_VDD            | VDD 端子の電圧検出機能を動作禁止状態に設定します。                 |
| R_LVD_Stop_VBAT           | VBAT 端子の電圧検出機能を動作禁止状態に設定します。                |
| R_LVD_Stop_VRTC           | VRTC 端子の電圧検出機能を動作禁止状態に設定します。                |
| R_LVD_Stop_EXLVD          | EXLVD 端子の電圧検出機能を動作禁止状態に設定します。               |
| R_LVI_Create              | 電圧検出回路を制御するうえで必要となる初期化処理を行います。              |
| R_LVI_Create_UserInit     | 電圧検出回路に関するユーザ独自の初期化処理を行います。                 |
| r_lvi_interrupt           | 電圧検出割り込み INTLVI の発生に伴う処理を行います。              |
| R_LVI_InterruptMode_Start | 電圧検出動作を開始します（割り込みモード時、および割り込み & リセット・モード時）。 |

## R\_LVD\_Create

電圧検出回路を制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_LVD_Create ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_LVD\_Create\_UserInit

電圧検出回路に関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_LVD\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_LVD_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

## r\_lvd\_interrupt

電圧検出割り込み INTLVI の発生に伴う処理を行います。

備考 本 API 関数は、電圧検出割り込み INTLVI に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_lvd_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_lvd_interrupt ( void );
```

### [引数]

なし

### [戻り値]

なし

## r\_lvd\_vddinterrupt

VDD 端子電圧検出割り込み INTLVDVDD の発生に伴う処理を行います。

備考 本 API 関数は、VDD 端子電圧検出割り込み INTLVDVDD に対応した割り込み処理として呼び出され  
ます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_lvd_vddinterrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_lvd_vddinterrupt ( void );
```

### [引数]

なし

### [戻り値]

なし

## r\_lvd\_vbatinterrupt

VBAT 端子電圧検出割り込み INTLVDVBAT の発生に伴う処理を行います。

備考 本 API 関数は、VBAT 端子電圧検出割り込み INTLVDVBAT に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_lvd_vbatinterrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_lvd_vbatinterrupt ( void );
```

### [引数]

なし

### [戻り値]

なし

## r\_lvd\_vrtcinterrupt

VRTC 端子電圧検出割り込み INTLVDVRTC の発生に伴う処理を行います。

備考 本 API 関数は、VRTC 端子電圧検出割り込み INTLVDVRTC に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_lvd_vrtcinterrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_lvd_vrtcinterrupt ( void );
```

### [引数]

なし

### [戻り値]

なし

## r\_lvd\_exlvdinterrupt

EXLVD 端子電圧検出割り込み INTLVDEXLVD の発生に伴う処理を行います。

備考 本 API 関数は、EXLVD 端子電圧検出割り込み INTLVDEXLVD に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_lvd_exlvdinterrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_lvd_exlvdinterrupt ( void );
```

### [引数]

なし

### [戻り値]

なし



## R\_LVD\_InterruptMode\_Start

電圧検出動作を開始します（割り込みモード時，および割り込み & リセット・モード時）。

### [指定形式]

```
void R_LVD_InterruptMode_Start ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_LVD\_Start\_VDD

VDD 端子の電圧検出機能を動作許可状態に設定します。

### [指定形式]

```
void R_LVD_Start_VDD ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_LVD\_Start\_VBAT

VBAT 端子の電圧検出機能を動作許可状態に設定します。

### [指定形式]

```
void R_LVD_Start_VBAT ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_LVD\_Start\_VRTC

VRTC 端子の電圧検出機能を動作許可状態に設定します。

### [指定形式]

```
void R_LVD_Start_VRTC ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_LVD\_Start\_EXLVD

EXLVD 端子の電圧検出機能を動作許可状態に設定します。

### [指定形式]

```
void R_LVD_Start_EXLVD ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_LVD\_Stop\_VDD

VDD 端子の電圧検出機能を動作禁止状態に設定します。

### [指定形式]

```
void R_LVD_Stop_VDD ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_LVD\_Stop\_VBAT

VBAT 端子の電圧検出機能を動作禁止状態に設定します。

### [指定形式]

```
void R_LVD_Stop_VBAT ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_LVD\_Stop\_VRTC

VRTC 端子の電圧検出機能を動作禁止状態に設定します。

### [指定形式]

```
void R_LVD_Stop_VRTC ( void );
```

### [引数]

なし

### [戻り値]

なし



## R\_LVD\_Stop\_EXLVD

EXLVD 端子の電圧検出機能を動作禁止状態に設定します。

### [指定形式]

```
void R_LVD_Stop_EXLVD ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_LVI\_Create

電圧検出回路を制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_LVI_Create ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_LVI\_Create\_UserInit

電圧検出回路に関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_LVI\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_LVI_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

## r\_lvi\_interrupt

電圧検出割り込み INTLVI の発生に伴う処理を行います。

備考 本 API 関数は、電圧検出割り込み INTLVI に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_lvi_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_lvi_interrupt ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_LVI\_InterruptMode\_Start

電圧検出動作を開始します（割り込みモード時，および割り込み & リセット・モード時）。

### [指定形式]

```
void R_LVI_InterruptMode_Start ( void );
```

### [引数]

なし

### [戻り値]

なし

### 3.2.41 バッテリ・バックアップ機能

以下に、コード生成がバッテリ・バックアップ機能用として出力する API 関数の一覧を示します。

表 3.41 バッテリ・バックアップ機能用 API 関数

| API 関数名                               | 機能概要                                  |
|---------------------------------------|---------------------------------------|
| <a href="#">R_BUP_Create</a>          | バッテリ・バックアップ機能を制御するうえで必要となる初期化処理を行います。 |
| <a href="#">R_BUP_Create_UserInit</a> | バッテリ・バックアップ機能に関するユーザ独自の初期化処理を行います。    |
| <a href="#">r_bup_interrupt</a>       | 電源切り替え検出割り込み INTVBAT の発生に伴う処理を行います。   |
| <a href="#">R_BUP_Start</a>           | バッテリ・バックアップ機能を動作許可状態に設定します。           |
| <a href="#">R_BUP_Stop</a>            | バッテリ・バックアップ機能を動作停止状態に設定します。           |

## R\_BUP\_Create

バッテリー・バックアップ機能を制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_BUP_Create ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_BUP\_Create\_UserInit

バッテリー・バックアップ機能に関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_BUP\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_BUP_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし



## r\_bup\_interrupt

電源切り替え検出割り込み INTVBAT の発生に伴う処理を行います。

備考 本 API 関数は、電源切り替え検出割り込み INTVBAT に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_bup_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_bup_interrupt ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_BUP\_Start

バッテリー・バックアップ機能を動作許可状態に設定します。

### [指定形式]

```
void R_BUP_Start ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_BUP\_Stop

バッテリー・バックアップ機能を動作停止状態に設定します。

### [指定形式]

```
void R_BUP_Stop ( void );
```

### [引数]

なし

### [戻り値]

なし

### 3.2.42 発振停止検出回路

以下に、コード生成が発振停止検出回路用として出力する API 関数の一覧を示します。

表 3.42 発振停止検出回路用 API 関数

| API 関数名                                | 機能概要                              |
|--|-----------------------------------|
| <a href="#">R_OSDC_Create</a>          | 発振停止検出回路を制御するうえで必要となる初期化処理を行います。  |
| <a href="#">R_OSDC_Create_UserInit</a> | 発振停止検出回路に関するユーザ独自の初期化処理を行います。     |
| <a href="#">r_osdc_interrupt</a>       | 発振停止検出割り込み INTOSDC の発生に伴う処理を行います。 |
| <a href="#">R_OSDC_Start</a>           | 発振停止検出回路を動作許可状態に設定します。            |
| <a href="#">R_OSDC_Stop</a>            | 発振停止検出回路を動作停止状態に設定します。            |
| <a href="#">R_OSDC_Set_PowerOff</a>    | 発振停止検出回路に対するクロック供給を停止します。         |
| <a href="#">R_OSDC_Reset</a>           | 発振停止検出回路をリセットします。                 |

## R\_OSDC\_Create

発振停止検出回路を制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_OSDC_Create ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_OSDC\_Create\_UserInit

発振停止検出回路に関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_OSDC\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_OSDC_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

## r\_osdc\_interrupt

発振停止検出割り込み INTOSDC の発生に伴う処理を行います。

備考 本 API 関数は、発振停止検出割り込み INTOSDC に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_osdc_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_osdc_interrupt ( void );
```

### [引数]

なし

### [戻り値]

なし

**R\_OSDC\_Start**

発振停止検出回路を動作許可状態に設定します。

**[指定形式]**

```
void R_OSDC_Start ( void );
```

**[引数]**

なし

**[戻り値]**

なし



## R\_OSDC\_Stop

発振停止検出回路を動作停止状態に設定します。

### [指定形式]

```
void R_OSDC_Stop ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_OSDC\_Set\_PowerOff

発振停止検出回路に対するクロック供給を停止します。

備考           本 API 関数の呼び出しにより、発振停止検出回路はリセット状態へと移行します。  
このため、本 API 関数の呼び出し後、制御レジスタへの書き込みは無視されます。

### [指定形式]

```
void   R_OSDC_Set_PowerOff ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_OSDC\_Reset

発振停止検出回路をリセットします。

### [指定形式]

```
void R_OSDC_Reset ( void );
```

### [引数]

なし

### [戻り値]

なし

### 3.2.43 SPI インタフェース

以下に、コード生成が SPI インタフェース用として出力する API 関数の一覧を示します。

表 3.43 SPI インタフェース用 API 関数

| API 関数名                                | 機能概要                                |
|--|-------------------------------------|
| <a href="#">R_SAIC_Create</a>          | SPI インタフェースを制御するうえで必要となる初期化処理を行います。 |
| <a href="#">R_SAIC_Create_UserInit</a> | SPI インタフェースに関するユーザ独自の初期化処理を行います。    |
| <a href="#">R_SAIC_Write</a>           | データの SPI 送信を開始します。                  |
| <a href="#">R_SAIC_Read</a>            | データの SPI 受信を開始します。                  |
| <a href="#">R_SPI_Create</a>           | SPI インタフェースを制御するうえで必要となる初期化処理を行います。 |
| <a href="#">R_SPI_Create_UserInit</a>  | SPI インタフェースに関するユーザ独自の初期化処理を行います。    |
| <a href="#">R_SPI_Start</a>            | SPI 通信を待機状態にします。                    |
| <a href="#">R_SPI_Stop</a>             | SPI 通信を終了します。                       |

## R\_SAIC\_Create

SPI インタフェースを制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_SAIC_Create ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_SAIC\_Create\_UserInit

SPI インタフェースに関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_SAIC\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_SAIC_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

**R\_SAIC\_Write**

データの SPI 送信を開始します。

**[指定形式]**

```
void R_SAIC_Write ( const smartanalog_t * p_saic_data );
```

**[引数]**

| I/O | 引数                                    | 説明                   |
|-----|---------------------------------------|----------------------|
| I   | const smartanalog_t<br>* p_saic_data; | 送信するデータを格納した領域へのポインタ |

**[戻り値]**

なし

## R\_SAIC\_Read

データの SPI 受信を開始します。

### [指定形式]

```
void R_SAIC_Read ( const smartanalog_t * p_saic_data, smartanalog_t *  
p_saic_read_buf );
```

### [引数]

| I/O | 引数                                    | 説明                     |
|-----|---------------------------------------|------------------------|
| O   | const smartanalog_t<br>* p_saic_data; | 受信したデータを格納する領域へのポインタ   |
| O   | smartanalog_t<br>* p_saic_read_buf;   | 受信したデータを格納するバッファへのポインタ |

### [戻り値]

なし



## R\_SPI\_Create

SPI インタフェースを制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_SPI_Create ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_SPI\_Create\_UserInit

SPI インタフェースに関するユーザ独自の初期化処理を行います。

備考           本 API 関数は、[R\\_SPI\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void   R_SPI_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

**R\_SPI\_Start**

SPI 通信を待機状態にします。

**[指定形式]**

```
void R_SPI_Start ( void );
```

**[引数]**

なし

**[戻り値]**

なし

**R\_SPI\_Stop**

SPI 通信を終了します。

**[指定形式]**

```
void R_SPI_Stop ( void );
```

**[引数]**

なし

**[戻り値]**

なし

### 3.2.44 オペアンプ

以下に、コード生成がオペアンプ用として出力する API 関数の一覧を示します。

表 3.44 オペアンプ用 API 関数

| API 関数名   | 機能概要                                   |
|---|--|
| <a href="#">R_OPAMP_Create</a>                  | オペアンプを制御するうえで必要となる初期化処理を行います。          |
| <a href="#">R_OPAMP_Create_UserInit</a>         | オペアンプに関するユーザ独自の初期化処理を行います。             |
| <a href="#">R_OPAMP_Set_ReferenceCircuitOn</a>  | オペアンプ・リファレンス電流回路を動作許可します。              |
| <a href="#">R_OPAMP_Set_ReferenceCircuitOff</a> | オペアンプ・リファレンス電流回路を停止します。                |
| <a href="#">R_OPAMPn_Start</a>                  | ユニット n のオペアンプを動作します。                   |
| <a href="#">R_OPAMPn_Stop</a>                   | ユニット n のオペアンプを停止します。                   |
| <a href="#">R_OPAMPn_Set_PrechargeOn</a>        | ユニット n のオペアンプの外部コンデンサのプリチャージを許可状態にします。 |
| <a href="#">R_OPAMPn_Set_PrechargeOff</a>       | ユニット n のオペアンプの外部コンデンサのプリチャージを停止状態にします。 |

## R\_OPAMP\_Create

オペアンプを制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_OPAMP_Create ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_OPAMP\_Create\_UserInit

オペアンプに関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_OPAMP\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_OPAMP_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_OPAMP\_Set\_ReferenceCircuitOn

オペアンプ・リファレンス電流回路を動作します。

### [指定形式]

```
void R_OPAMP_ReferenceCircuitOn ( void );
```

### [引数]

なし

### [戻り値]

なし



## R\_OPAMP\_Set\_ReferenceCircuitOff

オペアンプ・リファレンス電流回路を停止します。

### [指定形式]

```
void R_OPAMP_ReferenceCircuitOff ( void );
```

### [引数]

なし

### [戻り値]

なし

**R\_OPAMP $n$ \_Start**

ユニット  $n$  のオペアンプを動作します。

**[指定形式]**

```
void R_OPAMP $n$ _Start ( void );
```

備考  $n$  は、チャンネル番号を意味します。

**[引数]**

なし

**[戻り値]**

なし

**R\_OPAMP $n$ \_Stop**

ユニット  $n$  のオペアンプを停止します。

**[指定形式]**

```
void R_OPAMP $n$ _Stop ( void );
```

備考  $n$  は、チャンネル番号を意味します。

**[引数]**

なし

**[戻り値]**

なし

## R\_OPAMPn\_Set\_PrechargeOn

ユニット  $n$  のオペアンプの外部コンデンサのプリチャージを許可状態にします。

### [指定形式]

```
void R_OPAMPn_Set_PrechargeOn ( void );
```

備考  $n$  は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

**R\_OPAMP $n$ \_Set\_PrechargeOff**

ユニット  $n$  のオペアンプの外部コンデンサのプリチャージを停止状態にします。

**[指定形式]**

```
void R_OPAMP $n$ _Set_PrechargeOff ( void );
```

備考  $n$  は、チャンネル番号を意味します。

**[引数]**

なし

**[戻り値]**

なし

### 3.2.45 データ演算回路

以下に、コード生成がデータ演算回路用として出力するAPI関数の一覧を示します。

表 3.45 データ演算用API関数

| API関数名                                | 機能概要                                 |
|---------------------------------------|--------------------------------------|
| <a href="#">R_DOC_Create</a>          | データ演算回路を制御するうえで必要となる初期化処理を行います。      |
| <a href="#">R_DOC_Create_UserInit</a> | データ演算回路に関するユーザ独自の初期化処理を行います。         |
| <a href="#">r_doc_interrupt</a>       | データ演算結果検出割り込み INTDOC の発生に伴う処理を行います。  |
| <a href="#">R_DOC_SetMode</a>         | データ演算回路の動作モードを設定します。                 |
| <a href="#">R_DOC_WriteData</a>       | 演算対象の16ビットのデータを設定します。                |
| <a href="#">R_DOC_GetResult</a>       | データの加算結果または減算結果を取得します。               |
| <a href="#">R_DOC_ClearFlag</a>       | DOCコントロールレジスタ(DOCR)のDOPCFフラグをクリアします。 |
| <a href="#">R_DOC_Set_PowerOff</a>    | データ演算回路へのクロック供給を停止します。               |
| <a href="#">R_DOC_Reset</a>           | データ演算回路をリセットします。                     |

## R\_DOC\_Create

データ演算回路を制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_DOC_Create ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_DOC\_Create\_UserInit

データ演算回路に関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_DOC\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_DOC_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし



## r\_doc\_interrupt

DOC 演算結果検出割り込み INTDOC の発生に伴う処理を行います。

備考 本 API 関数は、DOC 演算結果検出割り込み INTDOC に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_doc_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_doc_interrupt ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_DOC\_SetMode

データ演算回路の動作モードを設定します。

### [指定形式]

```
#include "r_cg_macrodriver.h"
#include "r_cg_doc.h"
MD_STATUS R_DOC_SetMode ( doc_mode_t mode, unit16_t value );
```

### [引数]

| I/O | 引数               | 説明   |
|-----|------------------|--|
| I   | doc_mode_t mode; | データ演算回路の動作モード<br>ADDITION : データ加算モード<br>SUBTRACTION : データ減算モード<br>COMPARE_MATCH : データ比較モード<br>(結果の検出条件 : 一致を検出)<br>COMPARE_MISMATCH : データ比較モード<br>(結果の検出条件 : 不一致を検出) |
| I   | Unit16_t value;  | データ加算モードおよびデータ減算モードの場合 : 演算結果<br>データ比較モードの場合 : 基準となるデータ  |

### [戻り値]

| マクロ         | 説明       |
|-------------|----------|
| MD_OK       | 正常終了     |
| MD_ARGERROR | 引数の指定が不正 |

## R\_DOC\_WriteData

演算対象の 16 ビットのデータを設定します。

### [指定形式]

```
#include    "r_cg_macrodriver.h"
void        R_DOC_WriteData ( unit16_tdata );
```

### [引数]

| I/O | 引数             | 説明              |
|-----|----------------|-----------------|
| O   | Unit16_t data; | 演算対象の 16 ビットデータ |

### [戻り値]

なし

## R\_DOC\_GetResult

データの加算結果または減算結果を取得します。

### [指定形式]

```
#include    "r_cg_macrodriver.h"
void        R_DOC_GetResult ( unit16_t*const data );
```

### [引数]

| I/O | 引数                   | 説明                |
|-----|----------------------|-------------------|
| O   | Unit16_t*const data; | 演算結果を格納した領域へのポインタ |

### [戻り値]

なし

**R\_DOC\_ClearFlag**

DOC コントロールレジスタ (DOCR) の DOPCF フラグをクリアします。

**[指定形式]**

```
void R_DOC_ClearFlag ( void );
```

**[引数]**

なし

**[戻り値]**

なし

## R\_DOC\_Set\_PowerOff

データ演算回路へのクロック供給を停止します。

### [指定形式]

```
void R_DOC_Set_PowerOff ( void );
```

### [引数]

なし

### [戻り値]

なし

**R\_DOC\_Reset**

データ演算回路をリセットします。

**[指定形式]**

```
void R_DOC_Reset ( void );
```

**[引数]**

なし

**[戻り値]**

なし

### 3.2.46 32 ビット積和演算回路

以下に、コード生成が32ビット積和演算回路用として出力するAPI関数の一覧を示します。

表 3.46 32 ビット積和演算回路用 API 関数

| API 関数名                                    | 機能概要   |
|--|--|
| <a href="#">R_MAC32Bit_Create</a>          | 32 ビット積和演算回路を制御するうえで必要となる初期化処理を行います。                     |
| <a href="#">R_MAC32Bit_Create_UserInit</a> | 32 ビット積和演算回路に関するユーザ独自の初期化処理を行います。                        |
| <a href="#">r_mac32bit_interrupt_flow</a>  | 32 ビット積和演算オーバーフロー / アンダーフロー割り込み INTMACLOF の発生に伴う処理を行います。 |
| <a href="#">R_MAC32Bit_Reset</a>           | 32 ビット積和演算回路をリセットします。                                    |
| <a href="#">R_MAC32Bit_Set_PowerOff</a>    | 32 ビット積和演算回路に対するクロック供給を停止します。                            |
| <a href="#">R_MAC32Bit_MULUnsigned</a>     | 符号なし乗算を行います。   |
| <a href="#">R_MAC32Bit_MULSigned</a>       | 符号あり乗算を行います。   |
| <a href="#">R_MAC32Bit_MACUnsigned</a>     | 符号なし積和演算を行います。   |
| <a href="#">R_MAC32Bit_MACSigned</a>       | 符号あり積和演算を行います。   |



## R\_MAC32Bit\_Create

32ビット積和演算回路を制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_MAC32Bit_Create ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_MAC32Bit\_Create\_UserInit

32 ビット積和演算回路回路に関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_MAC32Bit\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_MAC32Bit_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

## r\_mac32bit\_interrupt\_flow

32 ビット積和演算オーバーフロー/アンダーフロー割り込み INTMACLOF の発生に伴う処理を行います。

備考 本 API 関数は、32 ビット積和演算オーバーフロー/アンダーフロー割り込み INTMACLOF に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_mac32bit_interrupt_flow ( void );
```

CC-RL コンパイラの場合

```
static void __near r_mac32bit_interrupt_flow ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_MAC32Bit\_Reset

32 ビット積和演算回路をリセットします。

### [指定形式]

```
void R_MAC32Bit_Reset ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_MAC32Bit\_Set\_PowerOff

32 ビット積和演算回路に対するクロック供給を停止します。

備考 本 API 関数の呼び出しにより、32 ビット積和演算回路はリセット状態へと移行します。  
このため、本 API 関数の呼び出し後、制御レジスタへの書き込みは無視されます。

### [指定形式]

```
void R_MAC32Bit_Set_PowerOff ( void );
```

### [引数]

なし

### [戻り値]

なし

**R\_MAC32Bit\_MULUnsigned**

符号なし乗算を行います。

**[指定形式]**

```
#include "r_cg_macrodriver.h"
#include "r_cg_mac32bit.h"
void R_MAC32Bit_MULUnsigned(uint32_t data_a, uint32_t data_b, mac32bit_uint64_t
* buffer_64bit);
```

**[引数]**

| I/O | 引数                                  | 説明   |
|-----|-------------------------------------|------|
| I   | uint32_t data_a                     | 被乗数値 |
| I   | uint32_t data_b                     | 乗数値  |
| O   | mac32bit_uint64_t<br>* buffer_64bit | 演算結果 |

備考 以下に、演算結果 mac32bit\_uint64\_t の構成を示します。

```
typedef struct
{
    uint16_t low_low;
    uint16_t low_high;
    uint16_t high_low;
    uint16_t high_high;
} mac32bit_uint64_t;
```

**[戻り値]**

なし

## R\_MAC32Bit\_MULSigned

符号あり乗算を行います。

### [指定形式]

```
#include "r_cg_macrodriver.h"
#include "r_cg_mac32bit.h"
void R_MAC32Bit_MULSigned(int32_t data_a, int32_t data_b, mac32bit_int64_t *
buffer_64bit);
```

### [引数]

| I/O | 引数                                 | 説明   |
|-----|------------------------------------|------|
| I   | int32_t data_a                     | 被乗数値 |
| I   | int32_t data_b                     | 乗数値  |
| O   | mac32bit_int64_t<br>* buffer_64bit | 演算結果 |

備考 以下に、演算結果 mac32bit\_int64\_t の構成を示します。

```
typedef struct
{
    int16_t low_low;
    int16_t low_high;
    int16_t high_low;
    int16_t high_high;
} mac32bit_int64_t;
```

### [戻り値]

なし

## R\_MAC32Bit\_MACUnsigned

符号なし積和演算を行います。

### [指定形式]

```
#include "r_cg_macrodriver.h"
#include "r_cg_mac32bit.h"
void R_MAC32Bit_SetMAC_Unsigned ( uint32_t data_a, uint32_t data_b,
mac32bit_uint64_t * buffer_64bit );
```

### [引数]

| I/O | 引数                                  | 説明         |
|-----|-------------------------------------|------------|
| I   | uint32_t data_a                     | 被乗数値       |
| I   | uint32_t data_b                     | 乗数値        |
| O   | mac32bit_uint64_t<br>* buffer_64bit | 累計初期値／演算結果 |

備考 累計初期値／演算結果 mac32bit\_uint64\_t についての詳細は、[R\\_MAC32Bit\\_MULUnsigned](#) を参照してください。

### [戻り値]

なし



## R\_MAC32Bit\_MACSigned

符号あり積和演算を行います。

### [指定形式]

```
#include "r_cg_macrodriver.h"
#include "r_cg_mac32bit.h"
void R_MAC32Bit_MACSigned ( int32_t data_a, int32_t data_b, mac32bit_int64_t *
buffer_64bit );
```

### [引数]

| I/O | 引数                                 | 説明         |
|-----|------------------------------------|------------|
| I   | int32_t data_a;                    | 被乗数値       |
| I   | int32_t data_b                     | 乗数値        |
| O   | mac32bit_int64_t<br>* buffer_64bit | 累計初期値／演算結果 |

備考 累計初期値／演算結果 mac32bit\_int64\_t についての詳細は、[R\\_MAC32Bit\\_MULSigned](#) を参照してください。

### [戻り値]

なし

## 3.2.47 12 ビット A/D コンバータ

以下に、コード生成が 12 ビット A/D コンバータ用として出力する API 関数の一覧を示します。

表 3.47 12 ビット A/D コンバータ用 API 関数

| API 関数名  | 機能概要                                     |
|--|--|
| <a href="#">R_12ADC_Create</a>                       | 12 ビット A/D コンバータを制御するうえで必要となる初期化処理を行います。 |
| <a href="#">R_12ADC_Create_UserInit</a>              | 12 ビット A/D コンバータに関するユーザ独自の初期化処理を行います。    |
| <a href="#">r_12adc_interrupt</a>                    | A/D 変換終了割り込み INTAD の発生に伴う処理を行います。        |
| <a href="#">R_12ADC_Start</a>                        | A/D 変換を開始します。                            |
| <a href="#">R_12ADC_Stop</a>                         | A/D 変換を終了します。                            |
| <a href="#">R_12ADC_Get_ValueResult</a>              | A/D 変換結果 (12 ビット) を読み出します。               |
| <a href="#">R_12ADC_Set_ADChannel</a>                | A/D 変換するアナログ電圧の入力端子を設定します。               |
| <a href="#">R_12ADC_TemperatureSensorOutput_On</a>   | 12 ビット A/D コンバータの温度センサー回路を動作します。         |
| <a href="#">R_12ADC_TemperatureSensorOutput_Off</a>  | 12 ビット A/D コンバータの温度センサー回路を停止します。         |
| <a href="#">R_12ADC_InternalReferenceVoltage_On</a>  | 12 ビット A/D コンバータのリファレンス電圧回路を動作します。       |
| <a href="#">R_12ADC_InternalReferenceVoltage_Off</a> | 12 ビット A/D コンバータのリファレンス電圧回路を停止します。       |
| <a href="#">R_12ADC_Set_PowerOff</a>                 | 12 ビット A/D コンバータに対するクロック供給を停止します。        |

## R\_12ADC\_Create

12ビット A/D コンバータを制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_12ADC_Create ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_12ADC\_Create\_UserInit

12ビット A/D コンバータに関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_12ADC\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_12ADC_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

## r\_12adc\_interrupt

A/D 変換終了割り込み INTAD の発生に伴う処理を行います。

備考 本 API 関数は、A/D 変換終了割り込み INTAD に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_12adc_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_12adc_interrupt ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_12ADC\_Start

A/D 変換を開始します。

備考 電圧コンパレータが動作停止状態から動作許可状態へと移行した際、約 1μ 秒の安定時間を必要とします。  
したがって、[R\\_12ADC\\_Create](#) と本 API 関数の間には、約 1μ 秒の時間を空ける必要があります。

### [指定形式]

```
void R_12ADC_Start ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_12ADC\_Stop

A/D 変換を終了します。

備考 電圧コンパレータは、本 API 関数の処理完了後も動作を継続しています。  
したがって、電圧コンパレータの動作を停止する場合は、本 API 関数の処理完了後、[R\\_12ADC\\_Set\\_PowerOff](#) を呼び出す必要があります。

### [指定形式]

```
void R_12ADC_Stop ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_12ADC\_Get\_ValueResult

A/D 変換結果（12ビット）を読み出します。

### [指定形式]

```
#include "r_cg_macrodriver.h"
void R_12ADC_Get_ValueResult ( ad_channel_t channel, uint16_t * const buffer );
```

### [引数]

| I/O | 引数                       | 説明                          |
|-----|--------------------------|-----------------------------|
| I   | ad_channel_t channel     | チャンネル番号                     |
| O   | uint16_t * const buffer; | 読み出した A/D 変換結果を格納する領域へのポインタ |

### [戻り値]

| マクロ         | 説明       |
|-------------|----------|
| MD_OK       | 正常終了     |
| MD_ARGERROR | 引数の指定が不正 |



## R\_12ADC\_Set\_ADChannel

A/D 変換するアナログ電圧の入力端子を設定します。

備考 引数 channel, data に指定された値は、A/D チャンネル選択レジスタ A0 (ADANSA0) または A/D 変換拡張入力コントロールレジスタ (ADEXICR) に設定されます。

### [指定形式]

```
#include "r_cg_macrodriver.h"
#include "r_cg_12adc.h"
MD_STATUS R_12ADC_Set_ADChannel ( ad_sel_register_t register, uint16_t data );
```

### [引数]

| I/O | 引数                          | 説明   |
|-----|-----------------------------|--|
| I   | ad_sel_register_t register; | 設定するレジスタ<br>SEL_ADANSA0 : A/D チャンネル選択レジスタ A0 (ADANSA0)<br>SEL_ADEXICR : A/D 変換拡張入力コントロールレジスタ (ADEXICR) |
| I   | uint16_t data;              | 制御レジスタに設定する値   |

### [戻り値]

| マクロ         | 説明       |
|-------------|----------|
| MD_OK       | 正常終了     |
| MD_ARGERROR | 引数の指定が不正 |

## R\_12ADC\_TemperatureSensorOutput\_On

12ビットA/Dコンバータの温度センサー回路を動作します。

### [指定形式]

```
MD_STATUS R_12ADC_TemperatureSensorOutput_On( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_12ADC\_TemperatureSensorOutput\_Off

12ビット A/D コンバータの温度センサー回路を停止します。

### [指定形式]

```
void R_12ADC_TemperatureSensorOutput_Off ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_12ADC\_InternalReferenceVoltage\_On

12ビットA/Dコンバータのリファレンス電圧回路を動作します。

### [指定形式]

```
void R_12ADC_InternalReferenceVoltage_On( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_12ADC\_InternalReferenceVoltage\_Off

12ビット A/D コンバータのリファレンス電圧回路を停止します。

### [指定形式]

```
void R_12ADC_InternalReferenceVoltage_Off ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_12ADC\_Set\_PowerOff

12ビット A/D コンバータに対するクロック供給を停止します。

備考           本 API 関数の呼び出しにより、12ビット A/D コンバータはリセット状態へと移行します。  
このため、本 API 関数の呼び出し後、制御レジスタへの書き込みは無視されます。

### [指定形式]

```
void R_12ADC_Set_PowerOff ( void );
```

### [引数]

なし

### [戻り値]

なし

### 3.2.48 12 ビット D/A コンバータ

以下に、コード生成が D/A コンバータ用として出力する API 関数の一覧を示します。

表 3.48 D/A コンバータ用 API 関数

| API 関数名                                     | 機能概要                                     |
|---|--|
| <a href="#">R_12DA_Create</a>               | 12 ビット D/A コンバータを制御するうえで必要となる初期化処理を行います。 |
| <a href="#">R_12DA_Create_UserInit</a>      | 12 ビット D/A コンバータに関するユーザ独自の初期化処理を行います。    |
| <a href="#">R_12DAn_Start</a>               | D/A 変換を開始します。                            |
| <a href="#">R_12DAn_Stop</a>                | D/A 変換を終了します。                            |
| <a href="#">R_12DAn_Set_ConversionValue</a> | ANOn 端子に出力するアナログ電圧値を設定します。               |
| <a href="#">R_12DA_Set_PowerOff</a>         | D/A コンバータに対するクロック供給を停止します。               |

## R\_12DA\_Create

12ビットD/Aコンバータを制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_12DA_Create ( void );
```

### [引数]

なし

### [戻り値]

なし



## R\_12DA\_Create\_UserInit

12ビット D/A コンバータに関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_12DA\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_12DA_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

**R\_12DAn\_Start**

D/A 変換を開始します。

**[指定形式]**

```
void R_12DAn_Start ( void );
```

備考  $n$ は、チャンネル番号を意味します。

**[引数]**

なし

**[戻り値]**

なし

**R\_12DAn\_Stop**

D/A 変換を終了します。

**[指定形式]**

```
void R_12DAn_Stop ( void );
```

備考  $n$ は、チャンネル番号を意味します。

**[引数]**

なし

**[戻り値]**

なし

## R\_12DA\_Set\_PowerOff

12ビット D/A コンバータに対するクロック供給を停止します。

備考           本 API 関数の呼び出しにより、12ビット D/A コンバータはリセット状態へと移行します。  
このため、本 API 関数の呼び出し後、制御レジスタへの書き込みは無視されます。

### [指定形式]

```
void   R_12DA_Set_PowerOff ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_12DAn\_Set\_ConversionValue

ANOn 端子に出力するアナログ電圧値を設定します。

### [指定形式]

```
#include "r_cg_macrodriver.h"
void R_12DAn_Set_ConversionValue ( uint16_t reg_value );
```

備考  $n$ は、チャンネル番号を意味します。

### [引数]

| I/O | 引数                  | 説明      |
|-----|---------------------|---------|
| I   | uint16_t reg_value; | D/A 変換値 |

### [戻り値]

なし

### 3.2.49 オペアンプ&アナログスイッチ

以下に、コード生成がオペアンプ&アナログスイッチ用として出力するAPI関数の一覧を示します。

表 3.49 オペアンプ&アナログスイッチ用API関数

| API関数名   | 機能概要                                   |
|--|--|
| <a href="#">R_AMPANSW_Create</a>                 | オペアンプ&アナログスイッチを制御するうえで必要となる初期化処理を行います。 |
| <a href="#">R_AMPANSW_Create_UserInit</a>        | オペアンプ&アナログスイッチに関するユーザ独自の初期化処理を行います。    |
| <a href="#">R_OPAMPm_Set_ReferenceCircuitOn</a>  | ユニット m のオペアンプ・リファレンス電流回路を動作許可します。      |
| <a href="#">R_OPAMPm_Set_ReferenceCircuitOff</a> | ユニット m のオペアンプ・リファレンス電流回路を停止します。        |
| <a href="#">R_OPAMPm_Start</a>                   | ユニット m のオペアンプを動作します。                   |
| <a href="#">R_OPAMPm_Stop</a>                    | ユニット m のオペアンプを停止します。                   |
| <a href="#">R_ANSW_ChargePumpm_On</a>            | ユニット m のアナログスイッチ回路を動作します。              |
| <a href="#">R_ANSW_ChargePumpm_Off</a>           | ユニット m のアナログスイッチ回路を停止します。              |

## R\_AMPANSW\_Create

オペアンプ&アナログスイッチを制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_AMPANSW_Create ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_AMPANSW\_Create\_UserInit

オペアンプ&アナログスイッチに関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_AMPANSW\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_AMPANSW_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし



## R\_OPAMP $m$ \_Set\_ReferenceCircuitOn

ユニット  $m$  のオペアンプ・リファレンス電流回路を動作します。

### [指定形式]

```
void R_OPAMP $m$ _Set_ReferenceCircuitOn ( void );
```

備考  $m$  は、ユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_OPAMPm\_Set\_ReferenceCircuitOff

ユニット *m* のオペアンプ・リファレンス電流回路を停止します。

### [指定形式]

```
void R_OPAMPm_Set_ReferenceCircuitOff ( void );
```

備考 *m* は、ユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし

**R\_OPAMPm\_Start**

ユニット *m* のオペアンプを動作します。

**[指定形式]**

```
void R_OPAMPm_Start ( void );
```

備考 *m* は、ユニット番号を意味します。

**[引数]**

なし

**[戻り値]**

なし

## R\_OPAMPm\_Stop

ユニット *m* のオペアンプを停止します。

### [指定形式]

```
void R_OPAMPm_Stop ( void );
```

備考 *m* は、ユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_ANSW\_ChargePumpm\_On

ユニット *m* のアナログスイッチ回路を動作します。

### [指定形式]

```
void R_ANSW_ChargePumpm_On ( void );
```

備考 *m* は、ユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_ANSW\_ChargePumpm\_Off

ユニット *m* のアナログスイッチ回路を停止します。

### [指定形式]

```
void R_ANSW_ChargePumpm_Off ( void );
```

備考 *m* は、ユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし

### 3.2.50 ボルテージ・リファレンス

以下に、コード生成が電圧検出回路用として出力する API 関数の一覧を示します。

表 3.50 電圧検出回路用 API 関数

| API 関数名                              | 機能概要                           |
|--------------------------------------|--------------------------------|
| <a href="#">R_VR_Create</a>          | 電圧検出回路を制御するうえで必要となる初期化処理を行います。 |
| <a href="#">R_VR_Create_UserInit</a> | 電圧検出回路に関するユーザ独自の初期化処理を行います。    |
| <a href="#">R_VR_Start</a>           | VDD 端子の電圧検出機能を動作許可状態に設定します。    |
| <a href="#">R_VR_Stop</a>            | VDD 端子の電圧検出機能を動作禁止状態に設定します。    |

**R\_VR\_Create**

ポルテージ・リファレンスを制御するうえで必要となる初期化処理を行います。

**[指定形式]**

```
void R_VR_Create ( void );
```

**[引数]**

なし

**[戻り値]**

なし



## R\_VR\_Create\_UserInit

ポルテージ・リファレンスに関するユーザ独自の初期化処理を行います。

備考 本API関数は、[R\\_VR\\_Create](#)のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_VR_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

**R\_VR\_Start**

ボルテージ・リファレンスを動作します。

**[指定形式]**

```
void R_VR_Start ( void );
```

**[引数]**

なし

**[戻り値]**

なし

**R\_VR\_Stop**

ボルテージ・リファレンスを停止します。

**[指定形式]**

```
void R_VR_Stop ( void );
```

**[引数]**

なし

**[戻り値]**

なし

### 3.2.51 サンプリング出力タイマ/ディテクタ

以下に、サンプリング出力タイマ/ディテクタ用として出力するAPI関数の一覧を示します。

表 3.51 サンプリング出力タイマ/ディテクタ用API関数

| API 関数名                                    | 機能概要  |
|--|---|
| <a href="#">R_SMOTD_Create</a>             | サンプリング出力タイマ/ディテクタを制御するうえで必要となる初期化処理を行います。       |
| <a href="#">R_SMOTD_Create_UserInit</a>    | サンプリング出力タイマ/ディテクタに関するユーザ独自の初期化処理を行います。          |
| <a href="#">r_smotd_counterA_interrupt</a> | サンプリング出力タイマインターバル割り込み (INTSMOTA) の発生に伴う処理を行います。 |
| <a href="#">r_smotd_counterB_interrupt</a> | サンプリング出力タイマインターバル割り込み (INTSMOTB) の発生に伴う処理を行います。 |
| <a href="#">r_smotd_smpn_interrupt</a>     | サンプリング・ディテクタ検出割り込みの発生に伴う処理を行います。                |
| <a href="#">R_SMOTD_Start</a>              | サンプリング出力タイマ/ディテクタの動作を開始します。                     |
| <a href="#">R_SMOTD_Stop</a>               | サンプリング出力タイマ/ディテクタの動作を終了します。                     |
| <a href="#">R_SMOTD_Set_PowerOFF</a>       | サンプリング出力タイマ/ディテクタに対するクロック供給を停止します。              |

## R\_SMOTD\_Create

サンプリング出力タイマ/ディテクタを制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_SMOTD_Create ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_SMOTD\_Create\_UserInit

サンプリング出力タイマ/ディテクタに関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_SMOTD\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_SMOTD_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

## r\_smotd\_counterA\_interrupt

サンプリング出力タイムインターバル割り込み (INTSMOTA) の発生に伴う処理を行います。

備考 本 API 関数は、サンプリング出力タイムインターバル割り込み (INTSMOTA) に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_smotd_counterA_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_smotd_counterA_interrupt ( void );
```

### [引数]

なし

### [戻り値]

なし

## r\_smotd\_counterB\_interrupt

サンプリング出力タイムインターバル割り込み (INTSMOTB) の発生に伴う処理を行います。

備考 本 API 関数は、サンプリング出力タイムインターバル割り込み (INTSMOTB) に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_smotd_counterB_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_smotd_counterB_interrupt ( void );
```

### [引数]

なし

### [戻り値]

なし



**r\_smotd\_smpn\_interrupt**

サンプリング・ディテクタ検出割り込みの発生に伴う処理を行います。

**[指定形式]**

CA78K0R コンパイラの場合

```
__interrupt static void r_smotd_smpn_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_smotd_smpn_interrupt ( void );
```

備考  $n$ は、入力端子 (SMP0-SMP5) の番号を意味します。

**[引数]**

なし

**[戻り値]**

なし

**R\_SMOTD\_Start**

サンプリング出力タイマ/ディテクタの動作を開始します。

**[指定形式]**

```
void R_SMOTD_Start ( void );
```

**[引数]**

なし

**[戻り値]**

なし

## R\_SMOTD\_Stop

サンプリング出力タイマ/ディテクタの動作を停止します。

### [指定形式]

```
void R_SMOTD_Stop ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_SMOTD\_Set\_PowerOFF

サンプリング出力タイマ／ディテクタに対するクロック供給を停止します。

備考 本 API 関数の呼び出しにより、サンプリング出力タイマ／ディテクタはリセット状態へと移行します。  
このため、本 API 関数の呼び出し後、制御レジスタへの書き込みは無視されます。

### [指定形式]

```
void R_SMOTD_Set_PowerOFF ( void );
```

### [引数]

なし

### [戻り値]

なし

### 3.2.52 外部サンプリング

以下に、コード生成が外部サンプリング用として出力する API 関数の一覧を示します。

表 3.52 外部サンプリング用 API 関数

| API 関数名                                | 機能概要                             |
|--|----------------------------------|
| <a href="#">R_EXSD_Create</a>          | 外部サンプリングを制御するうえで必要となる初期化処理を行います。 |
| <a href="#">R_EXSD_Create_UserInit</a> | 外部サンプリングに関するユーザ独自の初期化処理を行います。    |
| <a href="#">r_exsd_interrupt</a>       | 外部サンプリングエッジ検出割り込みの発生に伴う処理を行います。  |
| <a href="#">R_EXSD_Start</a>           | 外部サンプリングの動作を開始します。               |
| <a href="#">R_EXSD_Stop</a>            | 外部サンプリングの動作を終了します。               |
| <a href="#">R_EXSD_Set_PowerOff</a>    | 外部サンプリングに対するクロック供給を停止します。        |

## R\_EXSD\_Create

外部サンプリングを制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_EXSD_Create ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_EXSD\_Create\_UserInit

外部サンプリングに関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_EXSD\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_EXSD_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

## r\_exsd\_interrupt

外部サンプリングエッジ検出割り込みの発生に伴う処理を行います。

備考 本 API 関数は、外部サンプリングエッジ検出割り込みに対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_exsd_interrupt ( void );
```

CC-RL コンパイラの場合

```
static void __near r_exsd_interrupt ( void );
```

### [引数]

なし

### [戻り値]

なし



**R\_EXSD\_Start**

外部サンプリングの動作を開始します。

**[指定形式]**

```
void R_EXSD_Start ( void );
```

**[引数]**

なし

**[戻り値]**

なし

## R\_EXSD\_Stop

外部サンプリングの動作を終了します。

### [指定形式]

```
void R_EXSD_Stop ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_EXSD\_Set\_PowerOff

外部サンプリングに対するクロック供給を停止します。

備考           本 API 関数の呼び出しにより、外部サンプリングはリセット状態へと移行します。  
このため、本 API 関数の呼び出し後、制御レジスタへの書き込みは無視されます。

### [指定形式]

```
void R_EXSD_Set_PowerOff ( void );
```

### [引数]

なし

### [戻り値]

なし

## 3.2.53 シリアル・インタフェース UARTMG

以下に、コード生成がシリアル・インタフェース UARTMG 用として出力する API 関数の一覧を示します。

表 3.53 シリアル・インタフェース UARTMG 用 API 関数

| API 関数名  | 機能概要   |
|--|--|
| <a href="#">R_UARTMGn_Create</a>                   | シリアル・インタフェース UARTMG を制御するうえで必要となる初期化処理を行います。       |
| <a href="#">R_UARTMGn_Create_UserInit</a>          | シリアル・インタフェース UARTMG に関するユーザ独自の初期化処理を行います。          |
| <a href="#">r_uartmgn_interrupt_send</a>           | UARTMG 送信完了 / 送信バッファ空き割り込み INTSTMGn の発生に伴う処理を行います。 |
| <a href="#">r_uartmgn_interrupt_receive</a>        | UARTMG 受信完了割り込み INTSRMGn の発生に伴う処理を行います。            |
| <a href="#">r_uartmgn_interrupt_error</a>          | UARTMG 受信転送完了通信エラー発生割り込み INTSREMGn の発生に伴う処理を行います。  |
| <a href="#">R_UARTMGn_Start</a>                    | UART 通信を待機状態にします。                                  |
| <a href="#">R_UARTMGn_Stop</a>                     | UART 通信を終了します。                                     |
| <a href="#">R_UARTMGn_Set_PowerOff</a>             | シリアル・インタフェース UARTMG に対するクロック供給を停止します。              |
| <a href="#">R_UARTMGn_Send</a>                     | データの UART 送信を開始します。                                |
| <a href="#">R_UARTMGn_Receive</a>                  | データの UART 受信を開始します。                                |
| <a href="#">r_uartmgn_callback_sendend</a>         | UARTMG 送信完了 / 送信バッファ空き割り込み INTSTMGn の発生に伴う処理を行います。 |
| <a href="#">r_uartmgn_callback_receiveend</a>      | UART 受信完了割り込み INTSRMGn の発生に伴う処理を行います。              |
| <a href="#">r_uartmgn_callback_error</a>           | UART 受信エラー割り込み INTSRMGn / INTSREMGn の発生に伴う処理を行います。 |
| <a href="#">r_uartmgn_callback_softwareoverrun</a> | オーバラン・エラーの検出に伴う処理を行います。                            |

## R\_UARTMGn\_Create

シリアル・インタフェース UARTMG を制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_UARTMGn_Create ( void );
```

備考 *n*はユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_UARTMGn\_Create\_UserInit

シリアル・インタフェース UARTMG に関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[R\\_UARTMGn\\_Create](#) のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_UARTMGn_Create_UserInit ( void );
```

備考 *n* はユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし

## r\_uartmgn\_interrupt\_send

UARTMG 送信完了／送信バッファ空き割り込みの発生に伴う処理を行います。

備考 本 API 関数は、UARTMG 送信完了／送信バッファ空き割り込み INTSTMGn に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_uartmgn_interrupt_send ( void );
```

CC-RL コンパイラの場合

```
static void __near r_uartmgn_interrupt_send ( void );
```

備考 *n* はユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし

## r\_uartmgn\_interrupt\_receive

UARTMG 受信完了空き割り込み INTSRMG $n$ の発生に伴う処理を行います。

備考 本 API 関数は、UARTMG 受信完了割り込み INTSRMG $n$ に対応した割り込み処理として呼び出され  
ます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_uartmgn_interrupt_receive ( void );
```

CC-RL コンパイラの場合

```
static void __near r_uartmgn_interrupt_receive ( void );
```

備考  $n$ はユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし



## r\_uartmgn\_interrupt\_error

UARTMG 受信完了空き割り込み INTSREMG $n$ の発生に伴う処理を行います。

備考 本 API 関数は、UARTMG 受信完了エラー発生割り込み INTSREMG $n$ に対応した割り込み処理として呼び出されます。

### [指定形式]

CA78K0R コンパイラの場合

```
__interrupt static void r_uartmgn_interrupt_error ( void );
```

CC-RL コンパイラの場合

```
static void __near r_uartmgn_interrupt_error ( void );
```

備考  $n$ はユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_UARTMGn\_Start

UART 通信を待機状態にします。

### [指定形式]

```
void R_UARTMGn_Start ( void );
```

備考  $n$ はユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_UARTMGn\_Stop

UART 通信を終了します。

### [指定形式]

```
void R_UARTMGn_Stop ( void );
```

備考  $n$ はユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_UARTMGn\_Set\_PowerOff

シリアル・インタフェース UARTMG に対するクロック供給を停止します。

備考 本 API 関数の呼び出しにより、シリアル・インタフェース UARTMG はリセット状態へと移行します。  
このため、本 API 関数の呼び出し後、制御レジスタへの書き込みは無視されます。

### [指定形式]

```
void R_UARTMGn_Set_PowerOff ( void );
```

備考  $n$  は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## R\_UARTM $G_n$ \_Send

データの UART 送信を開始します。

- 備考 1. 本 API 関数では、引数 *tx\_buf* で指定されたバッファから 1 バイト単位の UART 送信を引数 *tx\_num* で指定された回数だけ繰り返し行います。
- 備考 2. UART 送信を行う際には、本 API 関数の呼び出し以前に [R\\_UARTM \$G\_n\$ \\_Start](#) を呼び出す必要があります。

### [指定形式]

```
#include "r_cg_macrodriver.h"
MD_STATUS R_UARTM $G_n$ _Send ( uint8_t * const tx_buf, uint16_t tx_num );
```

備考  $n$  はユニット番号を意味します。

### [引数]

| I/O | 引数                              | 説明                     |
|-----|---------------------------------|------------------------|
| I   | uint8_t * const <i>tx_buf</i> ; | 送信するデータを格納したバッファへのポインタ |
| I   | uint16_t <i>tx_num</i> ;        | 送信するデータの総数             |

### [戻り値]

| マクロ         | 説明       |
|-------------|----------|
| MD_OK       | 正常終了     |
| MD_ARGERROR | 引数の指定が不正 |

## R\_UARTMGn\_Receive

データの UART 受信を開始します。

- 備考 1. 本 API 関数では、1 バイト単位の UART 受信を引数 *rx\_num* で指定された回数だけ繰り返し行い、引数 *rx\_buf* で指定されたバッファに格納します。
- 備考 2. 実際の UART 受信は、本 API 関数の呼び出し後、[R\\_UARTMGn\\_Start](#) を呼び出すことにより開始されます。

### [指定形式]

```
#include "r_cg_macrodriver.h"
MD_STATUS R_UARTMGn_Receive ( uint8_t * const rx_buf, uint16_t rx_num );
```

備考 *n* はユニット番号を意味します。

### [引数]

| I/O | 引数                              | 説明                     |
|-----|---------------------------------|------------------------|
| O   | uint8_t * const <i>rx_buf</i> ; | 受信したデータを格納するバッファへのポインタ |
| I   | uint16_t <i>rx_num</i> ;        | 受信するデータの総数             |

### [戻り値]

| マクロ         | 説明       |
|-------------|----------|
| MD_OK       | 正常終了     |
| MD_ARGERROR | 引数の指定が不正 |

## r\_uartmgn\_callback\_sendend

UART 送信完了／送信バッファ空き割り込み INTSTMGNの発生に伴う処理を行います。

備考 本 API 関数は、UART 送信完了割り込み INTSTMGNに対応した割り込み処理 [r\\_uartmgn\\_interrupt\\_send](#) のコールバック・ルーチン ([R\\_UARTMGn\\_Send](#) の引数 *rx\_num* で指定された数のデータ受信が完了した際の処理) として呼び出されます。

### [指定形式]

```
static void r_uartmgn_callback_sendend ( void );
```

備考 *n*はユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし

## r\_uartmgn\_callback\_receiveend

UART 受信完了割り込み INTSRMG $n$  の発生に伴う処理を行います。

備考 本 API 関数は、UART 受信完了割り込み INTSRMG $n$  に対応した割り込み処理 [r\\_uartmgn\\_interrupt\\_receive](#) のコールバック・ルーチン ([R\\_UARTMG \$n\$ \\_Receive](#) の引数  $rx\_num$  で指定された数のデータ受信が完了した際の処理) として呼び出されます。

### [指定形式]

```
static void r_uartmgn_callback_receiveend ( void );
```

備考  $n$  はユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし



**r\_uartmgn\_callback\_error**

UART 受信エラーの検出に伴う処理を行います。

備考 本 API 関数は、UART 受信完了割り込み INTSRMGn に対応した割り込み処理 [r\\_uartmgn\\_interrupt\\_receive](#) または、UART 受信完了エラー発生割り込み INTSREMGn に対応した割り込み処理 [r\\_uartmgn\\_interrupt\\_error](#) のコールバック・ルーチン [R\\_UARTMGn\\_Receive](#) として呼び出されます。

**[指定形式]**

```
#include "r_cg_macrodriver.h"
static void r_uartmgn_callback_error ( uint8_t err_type );
```

備考 *n* はユニット番号を意味します。

**[引数]**

| I/O | 引数                | 説明   |
|-----|-------------------|--|
| I   | uint8_t err_type; | UART 受信エラー割り込みの発生要因<br>00000xx1B : オーバラン・エラー<br>00000x1xB : フレーミング・エラー<br>000001xxB : パリティ・エラー |

**[戻り値]**

なし

**r\_uartmgn\_callback\_softwareoverrun**

オーバーラン・エラーの検出に伴う処理を行います。

備考 本 API 関数は、UART 受信完了割り込み INTSRMG $n$  に対応した割り込み処理 [r\\_uartmgn\\_interrupt\\_receive](#) のコールバック・ルーチン ([R\\_UARTMGn\\_Receive](#) の引数  $rx\_num$  で指定された数以上のデータを受信した際の処理) として呼び出されます。

**[指定形式]**

```
#include "r_cg_macrodriver.h"
static void r_uartmgn_callback_softwareoverrun ( uint16_t rx_data );
```

備考  $n$  はユニット番号を意味します。

**[引数]**

| I/O | 引数                      | 説明   |
|-----|-------------------------|--|
| I   | uint8_t * const tx_buf; | 受信したデータ ( <a href="#">R_UARTMGn_Receive</a> の引数 $rx\_num$ で指定された数以上に受信したデータ) |

**[戻り値]**

なし

### 3.2.54 アンプ・ユニット

以下に、コード生成がアンプ・ユニット用として出力する API 関数の一覧を示します。

表 3.54 アンプ・ユニット用 API 関数

| API 関数名                               | 機能概要                             |
|---------------------------------------|----------------------------------|
| <a href="#">R_AMP_Create</a>          | アンプ・ユニットを制御するうえで必要となる初期化処理を行います。 |
| <a href="#">R_AMP_Create_UserInit</a> | アンプ・ユニットに関するユーザ独自の初期化処理を行います。    |
| <a href="#">R_AMP_Set_PowerOn</a>     | アンプ・ユニット部の電源を投入します。              |
| <a href="#">R_AMP_Set_PowerOff</a>    | アンプ・ユニット部の電源を切断します。              |
| <a href="#">R_PGA1_Start</a>          | アンプ・ユニット (PGA1) を待機状態にします。       |
| <a href="#">R_PGA1_Stop</a>           | アンプ・ユニット (PGA1) を停止します。          |
| <a href="#">R_AMPn_Start</a>          | アンプ・ユニット (AMPn) を待機状態にします。       |
| <a href="#">R_AMPn_Stop</a>           | アンプ・ユニット (AMPn) を停止します。          |

## R\_AMP\_Create

アンプ・ユニットを制御するうえで必要となる初期化処理を行います。

### [指定形式]

```
void R_AMP_Create ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_AMP\_Create\_UserInit

アンブ・ユニットに関するユーザ独自の初期化処理を行います。

備考 本API関数は、[R\\_AMP\\_Create](#)のコールバック・ルーチンとして呼び出されます。

### [指定形式]

```
void R_AMP_Create_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

**R\_AMP\_Set\_PowerOn**

アンプ・ユニット部の電源を投入します。

**[指定形式]**

```
void R_AMP_Set_PowerOn ( void );
```

**[引数]**

なし

**[戻り値]**

なし

## R\_AMP\_Set\_PowerOff

アンプ・ユニット部の電源を切断します。

### [指定形式]

```
void R_AMP_Set_PowerOff ( void );
```

### [引数]

なし

### [戻り値]

なし

**R\_PGA1\_Start**

アンプ・ユニット (PGA1) を待機状態にします。

**[指定形式]**

```
void R_PGA1_Start ( void );
```

**[引数]**

なし

**[戻り値]**

なし



**R\_PGA1\_Stop**

アンプ・ユニット (PGA1) を停止します。

**[指定形式]**

```
void R_PGA1_Stop ( void );
```

**[引数]**

なし

**[戻り値]**

なし

**R\_AMPn\_Start**

アンプ・ユニット (AMP $n$ ) を待機状態にします。

**[指定形式]**

```
void R_AMPn_Start ( void );
```

備考  $n$ は、オペアンプユニット番号を意味します。

**[引数]**

なし

**[戻り値]**

なし

## R\_AMPn\_Stop

アンプ・ユニット (AMP $n$ ) を停止します。

### [指定形式]

```
void R_AMPn_Stop ( void );
```

備考  $n$ は、オペアンプユニット番号を意味します。

### [引数]

なし

### [戻り値]

なし

### 3.2.55 データ・フラッシュ・ライブラリ

以下に、コード生成がデータ・フラッシュ・ライブラリ用として出力する API 関数の一覧を示します。

表 3.55 データ・フラッシュ・ライブラリ用 API 関数

| API 関数名                      | 機能概要                                   |
|------------------------------|--|
| <a href="#">R_FDL_Create</a> | データ・フラッシュライブラリを制御するうえで必要となる初期化処理を行います。 |
| <a href="#">R_FDL_Open</a>   | データ・フラッシュ・ライブラリの使用を開始します。              |
| <a href="#">R_FDL_Close</a>  | データ・フラッシュ・ライブラリの使用を終了します。              |
| <a href="#">R_FDL_Write</a>  | データをデータ・フラッシュに書き込みます。                  |
| <a href="#">R_FDL_Read</a>   | データをデータ・フラッシュから読み込みます。                 |
| <a href="#">R_FDL_Erase</a>  | データ・フラッシュのデータを消去します。                   |

## R\_FDL\_Create

データ・フラッシュ・ライブラリ Type04 を制御するうえで必要となる変数の代入のみを行います。

備考            スタートアップ処理から自動で呼ばれる関数です。ユーザが呼ぶ必要はありません。

### [指定形式]

```
void R_FDL_Create ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_FDL\_Open

データ・フラッシュ・ライブラリを使用するためのドライバをオープンします。  
オープン状態で、のコマンドが動作します。

### [指定形式]

```
void R_FDL_Open ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_FDL\_Close

データ・フラッシュ・ライブラリのドライバをクローズします。  
再度データ・フラッシュ・ライブラリを使用するためには、オープン処理 () が必要です。

### [指定形式]

```
void R_FDL_interrupt ( void );
```

### [引数]

なし

### [戻り値]

なし

## R\_FDL\_Write

データ・フラッシュへの書き込みを行います。  
 フランクチェックは行いません。  
 データ・フラッシュ・メモリの制御状態が、コマンド実行中 (PFDL\_BUSY) の場合、コマンド実行完了まで  
 書き込みは実行されません。

### [指定形式]

```
pfdl_status_t R_FDL_Write ( pfdl_u16 index, __near pfdl_u08 * buffer, pfdl_u16 bytecount );
```

### [引数]

| I/O | 引数                  | 説明                               |
|-----|---------------------|----------------------------------|
| I   | pfdl_u16 index;     | 書き込むデータ・フラッシュのアドレス 0000h ~ 0FFFh |
| I   | pfdl_u08 * buffer;  | 書き込むデータを格納したバッファへのポインタ           |
| I   | pfdl_u16 bytecount; | 書き込むデータのバイト数                     |

### [戻り値]

| マクロ                | 説明       |
|--------------------|----------|
| PFDL_OK            | 正常終了     |
| PFDL_BUSY          | コマンド実行中  |
| PFDL_ERR_WRITE     | 書き込みエラー  |
| PFDL_ERR_PARAMETER | 引数の指定が不正 |



**R\_FDL\_Read**

データ・フラッシュからの読み込みを行います。

**[指定形式]**

```
pfdl_status_t R_FDL_Read ( pfdl_u16 index, __near pfdl_u08 * buffer, pfdl_u16 bytecount );
```

**[引数]**

| I/O | 引数                  | 説明                               |
|-----|---------------------|----------------------------------|
| I   | pfdl_u16 index;     | 読み込むデータ・フラッシュのアドレス 0000h ~ 0FFFh |
| O   | pfdl_u08 * buffer;  | 読み込むデータを格納するバッファへのポインタ           |
| I   | pfdl_u16 bytecount; | 読み込むデータのバイト数                     |

**[戻り値]**

| マクロ                | 説明       |
|--------------------|----------|
| PFDL_OK            | 正常終了     |
| PFDL_BUSY          | コマンド実行中  |
| PFDL_ERR_PARAMETER | 引数の指定が不正 |

**R\_FDL\_Erase**

指定した番号のデータ・フラッシュのブロックを消去します。  
 戻り値がエラーの場合、対象ブロックに対して書き込みは行えません。  
 再度、この API を実行し消去が正常に実行されることを確認してください。

**[指定形式]**

```
pdf1_status_t R_FDL_Erase ( pdf1_u16 blockno );
```

**[引数]**

| I/O | 引数                | 説明              |
|-----|-------------------|-----------------|
| I   | pdf1_u16 blockno; | 消去するブロックの番号 0～3 |

**[戻り値]**

| マクロ                | 説明       |
|--------------------|----------|
| PFDL_OK            | 正常終了     |
| PFDL_ERR_ERASE     | 消去エラー    |
| PFDL_ERR_PARAMETER | 引数の指定が不正 |

## 改訂記録

| Rev. | 発行日        | 改定内容                            |  |
|------|------------|---------------------------------|--|
|      |            | ページ                             | ポイント   |
| 1.00 | 2014.08.01 | -                               | 初版発行   |
| 1.01 | 2014.12.01 | 8                               | 2. 出力ファイル<br>タイマ RJ、タイマ RD API 追加  |
|      |            | 12                              | 2. 出力ファイル<br>コンパレータ / プログラマブル・ゲイン・アンプ API 追加   |
|      |            | 64 –<br>70                      | 3.2.5 タイマ RJ<br>API 追加   |
|      |            | 81 –<br>89                      | 3.2.6 タイマ RD<br>API 追加   |
|      |            | 253 –<br>263                    | 3.2.24 コンパレータ / プログラマブル・ゲイン・アンプ<br>章の追加  |
| 1.02 | 2015.08.01 | 8, 11,<br>12, 16                | 2. 出力ファイル<br>API 追加<br>・タイマ RX<br>・PGA+ $\Delta\Sigma$ /D コンバータ<br>・コンフィギュラブル・アンプ<br>・D/A コンバータ<br>・電圧検出回路   |
|      |            | 98 –<br>105                     | 3.2.8 タイマ RX<br>章の追加   |
|      |            | 208 –<br>218                    | 3.2.19 PGA+ $\Delta\Sigma$ /D コンバータ<br>章の追加  |
|      |            | 235 –<br>240                    | 3.2.21 コンフィギュラブル・アンプ<br>章の追加   |
|      |            | 265 –<br>266                    | 3.2.24 D/A コンバータ<br>API 追加   |
|      |            | 439 –<br>442                    | 3.2.39 電圧検出回路<br>API 追加  |
| 1.03 | 2016.03.01 | 7,11,<br>13,16,<br>17,18,<br>19 | 2. 出力ファイル<br>API 追加<br>・高速オンチップ・オシレータ・クロック周波数補正機能<br>・リアルタイムクロック<br>・温度センサ<br>・24 ビット $\Delta\Sigma$ /D コンバータ<br>・シリアルインタフェース<br>・割り込み機能<br>・電圧検出回路<br>・発振停止検出回路<br>・32 ビット積和演算回路の機能 |
|      |            | 44 –<br>49                      | 3.2.4 高速オンチップ・オシレータ・クロック周波数補正機能<br>章の追加  |
|      |            | 166 –<br>190                    | 3.2.13 リアルタイムクロック<br>API 追加  |
|      |            | 271                             | 3.2.23 温度センサ<br>API 追加   |

| Rev. | 発行日        | 改定内容                              |   |
|------|------------|-----------------------------------|---|
|      |            | ページ                               | ポイント  |
|      |            | 282                               | 3.2.24 24 ビット $\Delta\Sigma$ /D コンバータ<br>API 追加   |
|      |            | 323                               | 3.2.32 シリアルインターフェース<br>API 追加   |
|      |            | 457 –<br>459                      | 3.2.38 割り込み機能<br>API 追加   |
|      |            | 470 –<br>482                      | 3.2.40 電圧検出回路<br>API 追加   |
|      |            | 500                               | 3.2.42 発振停止検出回路<br>API 追加   |
|      |            | 525 –<br>534                      | 3.2.46 32 ビット積和演算回路<br>章の追加   |
| 1.04 | 2016.10.01 | 8, 9,<br>11, 13,<br>14,<br>16, 17 | 2. 出力ファイル<br>API 追加<br>・ タイマ・アレイ・ユニット (R_TAUmReset)<br>・ 16 ビット・タイマ KB(R_TMR_KBm_ForcedOutput_mn_Start,<br>R_TMR_KBm_ForcedOutput_mn_Stop, R_TMR_KBm_Reset)<br>・ 12 ビット・インターバル・タイマ (R_IT_Reset)<br>・ A/D コンバータ (R_ADC_Stop)<br>・ D/A コンバータ (R_DACn_Create, R_DAC_Reset)<br>・ プログラマブル・ゲイン・アンプ (R_PGA_Reset)<br>・ コンパレータ (R_COMP_Reset)<br>・ LCD コントローラ/ドライバ (R_LCD_VoltageOn, R_LCD_VoltageOff)<br>・ 割り込み機能 (R_INTFO_Start, R_INTFO_Stop, R_INTFO_ClearFlag,<br>r_intfo_interrupt) |
|      |            | 9, 17                             | 2. 出力ファイル<br>ファイル名追加<br>・ 16 ビット・タイマ KB(r_cg_tmkb.c, r_cg_tmb_user.c, r_cg_tmkb.h)<br>・ キー割り込み機能 (r_cg_key.c, r_cg_key_user.c, r_cg_key.h)  |
|      |            | 19-20                             | 2. 出力ファイル<br>機能追加<br>・ 12 ビット A/D コンバータ<br>・ 12 ビット D/A コンバータ<br>・ オペアンプ&アナログスイッチ<br>・ ボルテージ・リファレンス   |
|      |            | 116,<br>126-<br>128               | 3.2.10.16 ビット・タイマ KB<br>API 追加<br>・ R_TMR_KBm_ForcedOutput_mn_Start<br>・ R_TMR_KBm_ForcedOutput_mn_Stop<br>・ R_TMR_KBm_Reset  |
|      |            | 290,<br>299-<br>300               | 3.2.25.D/A コンバータ<br>API 追加<br>・ R_DACn_Create<br>・ R_DAC_Reset  |
|      |            | 301,<br>306                       | 3.2.26. プログラマブル・ゲイン・アンプ<br>API 追加<br>・ R_PGA_Reset  |

| Rev. | 発行日 | 改定内容                |  |
|------|-----|---------------------|--|
|      |     | ページ                 | ポイント   |
|      |     | 421,<br>430-<br>431 | 3.2.33.LCD コントローラ／ドライバ<br>API 追加<br>・ R_LCD_VoltageOn<br>・ R_LCD_VoltageOff  |
|      |     | 457,<br>469-<br>472 | 3.2.38. 割り込み機能<br>API 追加<br>・ R_INTFO_Start<br>・ R_INTFO_Stop<br>・ R_INTFO_ClearFlag<br>・ r_intfo_interrupt  |
|      |     | 548-<br>560         | 3.2.47.12 ビット A/D コンバータ<br>章の追加<br>API 追加<br>・ R_12ADC_Create<br>・ R_12ADC_Create_UserInit<br>・ r_12adc_interrupt<br>・ R_12ADC_Start<br>・ R_12ADC_Stop<br>・ R_12ADC_Get_ValueResult<br>・ R_12ADC_Set_ADChannel<br>・ R_12ADC_TemperatureSensorOutput_On<br>・ R_12ADC_TemperatureSensorOutput_Off<br>・ R_12ADC_InternalReferenceVoltage_On<br>・ R_12ADC_InternalReferenceVoltage_Off<br>・ R_12ADC_Set_PowerOff |
|      |     | 561-<br>567         | 3.2.48.12 ビット D/A コンバータ<br>章の追加<br>API 追加<br>・ R_12DA_Create<br>・ R_12DA_Create_UserInit<br>・ R_12DAn_Start<br>・ R_12DAn_Stop<br>・ R_12DAn_Set_ConversionValue<br>・ R_12DA_Set_PowerOff  |
|      |     | 568-<br>576         | 3.2.49. オペアンプ&アナログスイッチ<br>章の追加<br>API 追加<br>・ R_AMPANSW_Create<br>・ R_AMPANSW_Create_UserInit<br>・ R_OPAMPm_Set_ReferenceCircuitOn<br>・ R_OPAMPm_Set_ReferenceCircuitOff<br>・ R_OPAMPm_Start<br>・ R_OPAMPm_Stop<br>・ R_ANSW_ChargePumpm_On<br>・ R_ANSW_ChargePumpm_Off   |
|      |     | 577-<br>581         | 3.2.50. ボルテージ・リファレンス<br>章の追加<br>API 追加<br>・ R_VR_Create<br>・ R_VR_Create_UserInit<br>・ R_VR_Start<br>・ R_VR_Stop   |

| Rev. | 発行日        | 改定内容                |   |
|------|------------|---------------------|---|
|      |            | ページ                 | ポイント  |
|      |            | 11,<br>161,<br>197  | 3.2.13. リアルタイム・クロック<br>API 追加<br>・ r_rtc_callback_periodeic   |
|      |            | 14,<br>302,<br>308  | 3.2.26. プログラマブル・ゲイン・アンプ<br>API 追加<br>・ R_PGA_Set_PowerOff   |
|      |            | 19,<br>521,<br>528  | 3.2.44. オペアンプ<br>API 名称変更<br>・ R_OPANPn_Set_PrechargeOn → R_OPAMPn_Set_PrechargeOn  |
|      |            | 19                  | 2. 出力ファイル<br>API 変更<br>・ 2 番目の R_OPAMP_Set_ReferenceCircuitOff → R_OPAMPn_Set_PrechargeOff  |
|      |            | 412,<br>417,<br>404 | 3.2.32. シリアル・インタフェース IICA<br>API 説明更新<br>・ r_iican_callback_master_error<br>・ r_iican_callback_slave_error<br>・ R_IICAn_StopCondition   |
| 1.05 | 2018.02.01 | 67-81               | 3.2.6 タイマ RJ<br>API 名称変更<br>・ R_TMR_RJ0_Create → R_TMR_RJn_Create<br>・ R_TMR_RJ0_Start → R_TMR_RJn_Start<br>・ R_TMR_RJ0_Stop → R_TMR_RJn_Stop<br>・ R_TMR_RJ0_Set_PowerOff → R_TMR_RJn_Set_PowerOff<br>・ R_TMR_RJ0_Get_PulseWidth → R_TMR_RJn_Get_PulseWidth<br>・ R_TMR_RJ0_Create_UserInit → R_TMR_RJn_Create_UserInit<br>・ r_tmr_rj0_interrupt → r_tmr_rjn_interrupt<br>・ R_TMRJ0_Create → R_TMRJn_Create<br>・ R_TMRJ0_Start → R_TMRJn_Start<br>・ R_TMRJ0_Stop → R_TMRJn_Stop<br>・ R_TMRJ0_Set_PowerOff → R_TMRJn_Set_PowerOff<br>・ R_TMRJ0_Get_PulseWidth → R_TMRJn_Get_PulseWidth<br>・ R_TMRJ0_Create_UserInit → R_TMRJn_Create_UserInit<br>・ r_tmj0_interrupt → r_tmjrn_interrupt |
|      |            | 82,<br>101          | 3.2.7 タイマ RD<br>API 追加<br>・ R_TMRD_Set_PowerOff   |
|      |            | 215,<br>222         | 3.2.16 8 ビット・インターバル・タイマ<br>API 追加<br>・ R_IT8bitm_Set_PowerOff   |
|      |            | 241,<br>252,<br>253 | 3.2.20 プログラマブル・ゲイン計装アンプ付き 24 ビット $\Delta\Sigma/D$ コンバータ<br>API 追加<br>・ r_pga_dsad_conversion_interrupt<br>・ r_pga_dsad_scan_interrupt   |
|      |            | 296,<br>307         | 3.2.25 D/A コンバータ<br>API 追加<br>・ R_DACn_Create_UserInit  |
|      |            | 455,<br>461,<br>462 | 3.2.36 データ・トランスファ・コントローラ<br>API 追加<br>・ R_DTCDn_Start<br>・ R_DTCDn_Stop   |

| Rev. | 発行日 | 改定内容                |   |
|------|-----|---------------------|---|
|      |     | ページ                 | ポイント  |
|      |     | 524,<br>529-<br>532 | 3.2.43 SPI インタフェース<br>API 追加<br><ul style="list-style-type: none"> <li>・ R_SPI_Create</li> <li>・ R_SPI_Start</li> <li>・ R_SPI_Stop</li> <li>・ R_SPI_Create_UserInit</li> </ul>  |
|      |     | 596-<br>604         | 3.2.51 サンプリング出カタイマ/ディテクタ<br>章追加<br>API 追加<br><ul style="list-style-type: none"> <li>・ R_SMOTD_Create</li> <li>・ R_SMOTD_Start</li> <li>・ R_SMOTD_Stop</li> <li>・ R_SMOTD_Set_PowerOff</li> <li>・ R_SMOTD_Create_UserInit</li> <li>・ r_smotd_counterA_Interrupt</li> <li>・ r_smotd_counterB_Interrupt</li> <li>・ r_smotd_smpn_interrupt</li> </ul>   |
|      |     | 605-<br>611         | 3.2.52 外部サンプリング<br>章追加<br>API 追加<br><ul style="list-style-type: none"> <li>・ R_EXSD_Create</li> <li>・ R_EXSD_Start</li> <li>・ R_EXSD_Stop</li> <li>・ R_EXSD_PowerOff</li> <li>・ R_EXSD_Create_UserInit</li> <li>・ r_exsd_interrupt</li> </ul>   |
|      |     | 612-<br>626         | 3.2.53 シリアル・インタフェース UARTMG<br>章追加<br>API 追加<br><ul style="list-style-type: none"> <li>・ R_UARTMGn_Create</li> <li>・ R_UARTMGn_Start</li> <li>・ R_UARTMGn_Stop</li> <li>・ R_UARTMGn_Send</li> <li>・ R_UARTMGn_Receive</li> <li>・ R_UARTMGn_Set_PowerOff</li> <li>・ R_UARTMGn_Create_UserInit</li> <li>・ r_uartmgn_interrupt_send</li> <li>・ r_uartmgn_interrupt_receive</li> <li>・ r_uartmgn_interrupt_error</li> <li>・ r_uartmgn_callback_sendend</li> <li>・ r_uartmgn_callback_receiveend</li> <li>・ r_uartmgn_callback_error</li> <li>・ r_uartmgn_callback_softwareoverrun</li> </ul> |
|      |     | 627-<br>635         | 3.2.54 アンプ・ユニット<br>章追加<br>API 追加<br><ul style="list-style-type: none"> <li>・ R_AMP_Create</li> <li>・ R_PGA1_Start</li> <li>・ R_PGA1_Stop</li> <li>・ R_AMPn_Start</li> <li>・ R_AMPn_Stop</li> <li>・ R_AMP_Set_PowerOn</li> <li>・ R_AMP_Set_PowerOff</li> <li>・ R_AMP_CreateInit</li> </ul>   |

| Rev. | 発行日 | 改定内容    |  |
|------|-----|---------|--|
|      |     | ページ     | ポイント   |
|      |     | 636-642 | 3.2.55 データ・フラッシュ・ライブラリ<br>章追加<br>API 追加<br>・ R_FDL_Create<br>・ R_FDL_Open<br>・ R_FDL_Close<br>・ R_FDL_Write<br>・ R_FDL_Read<br>・ R_FDL_Erase |



---

e<sup>2</sup> studio コード生成ツール ユーザーズマニュアル  
RL78 APIリファレンス編

発行年月日 2014年 8月 1日 Rev.1.00  
2018年 2月 1日 Rev.1.05

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)

---



ルネサスエレクトロニクス株式会社

営業お問合せ窓口

<http://www.renesas.com>

営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

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)

技術的なお問合せおよび資料のご請求は下記どうぞ。  
総合お問合せ窓口：<https://www.renesas.com/contact/>

# e<sup>2</sup> studio コード生成ツール



ルネサスエレクトロニクス株式会社

R20UT3127JJ0105