User's Manual

スマート・コンフィグレータ

ユーザーズマニュアル　RH850 API リファレンス編

対象デバイス

RH850 ファミリ

ルネサス エレクトロニクス
www.renesas.com

Rev.1.60　2024.1

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。回路、ソフトウェアおよびこれらに関連する情報を使用する場合、お客様の責任において、お客様の機器・システムを設計ください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品または本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
6. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。
   標準水準：　コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット等
   高品質水準：輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等
   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（「脆弱性問題」といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因しまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
10. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
12. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
13. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
14. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

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

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

(Rev.5.0-1　2020.10)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

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

www.renesas.com/contact/

商標について

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

製品ご使用上の注意事項

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

1. 静電気対策

   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。

2. 電源投入時の処置

   電源投入時は、製品の状態は不定です。電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。

3. 電源オフ時における入力信号

   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。

4. 未使用端子の処理

   未使用端子は、「未使用端子の処理」に従って処理してください。CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。

5. クロックについて

   リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。

6. 入力端子の印加波形

   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V_{IL}（Max.）から V_{IH}（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}（Max.）から V_{IH}（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。

7. リザーブアドレス（予約領域）のアクセス禁止

   リザーブアドレス（予約領域）のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。

8. 製品間の相違について

   型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違うと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ幅射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

このマニュアルの使い方

対象者　　このマニュアルは、スマート・コンフィグレータのドライバコード生成の機能を理解し、それ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、スマート・コンフィグレータのドライバコード生成の持つソフトウエ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エア、ソフトウエ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。

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。

凡例			
	データ表記の重み	:	左が上位桁、右が下位桁
	アクティブ・ロウの表記	:	$\overline{\text{XXX}}$　（端子、信号名称に上線）
	注	:	本文中につけた注の説明
	注意	:	気をつけて読んでいただきたい内容
	備考	:	本文中の補足説明
	数の表記	:	10 進数　...　XXXX
			16 進数　...　0xXXXX

目次

1. 概　説

本章では、スマート・コンフィグレータのドライバコード生成（以下、コード生成と略す）の概要について説明します。

1.1 概　要

本ツールは、GUI ベースで各種情報を設定することにより、デバイスが提供している周辺機能（クロック発生回路、電圧検出回路など）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 関数呼び出しを含んだサンプルプログラム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

  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ォルダ名、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 for *xxxx*. Do not edit comment generated here */

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

  “*xxxx*”は、ユーザコードにより異なります。

  - “global” - グローバル変数と関数を追加します。
  - “function” - 関数宣言を追加できます。
  - “user init” - 初期化コードを追加できます。
  - *割り込み関数名* - 割り込み関数のコードを追加できます。
  - “adding” - 関数を追加できます。

  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(1//21)

周辺機能	ファイル名	API 関数名
共　通	r_cg_main.c	main R_MAIN_UserInit
	r_cg_systeminit.c	R_Systeminit
	r_cg_cgc.c	R_CGC_Create
	r_cg_cgc_user.c	R_CGC_Create_UserInit
	r_cg_cgc.h	―
	r_smc_clock_info.h	―
	r_cg_intvector.c	―
	r_cg_intvector_PEk.c	―
	r_cg_intc_PEk.c	R_Interrupt_Initialize_ForPE
	r_cg_macrodriver.h	―
	r_cg_userdefine.h	―
	r_cg_ad.h	―
	r_cg_ad_air.h	―
	r_cg_ad_common.h	―
	r_cg_ad_common.c	R_ADC_SyncStart R_ADC_TimerSyncStart
	r_cg_abfg.h	―
	r_cg_csig.h	―
	r_cg_csih.h	―
	r_cg_dcra.h	―
	r_cg_dma.h	―
	r_cg_dma_common.c	R_DMAC_Create R_DMA_Suspend R_DMA_Resume R_PDMAn_Suspend R_PDMAn_Resume R_DMAC_Start_Interrupt_Error_PEk R_DMAC_Stop_Interrupt_Error_PEk R_DMA_Start_Transfer_Error_Interrupt R_DMA_Stop_Transfer_Error_Interrupt R_DTS_CreateR_DTS_Resume R_DTSg_Start_Transfer_End_Interrupt R_DTSg_Stop_Transfer_End_Interrupt R_DTSg_Start_Transfer_Count_Match_Interrupt R_DTSg_Stop_Transfer_Count_Match_Interrupt

表 2-2　　出力ファイル(2//21)

周辺機能	ファイル名	API 関数名
共　　通	r_cg_dma_common_user.c	r_sdmac_address_error_interrupt_pek r_dmac_error_interrupt_pe r_dtsg_transfer_end_interrupt r_dtsg_transfer_count_match_interrupt
	r_cg_dtm.h	―
	r_cg_dts.h	―
	r_cg_dts_common.c	R_DTS_Suspend R_DTS_Resume R_DTSg_Start_Transfer_End_Interrupt R_DTSg_Stop_Transfer_End_Interrupt R_DTSg_Start_Transfer_Count_Match_Interrupt R_DTSg_Stop_Transfer_Count_Match_Interrupt R_DTS_Start_PEk_Transfer_Error_Interrupt R_DTS_Stop_PEk_Transfer_Error_Interrupt
	r_cg_dts_common_user.c	r_dtsg_transfer_end_interrupt r_dtsg_transfer_count_match_interrupt r_dts_transfer_error_interrupt_pek
	r_cg_ecm.h	―
	r_cg_gtm_clock.h	―
	r_cg_gtm_common.h	―
	r_cg_gtm_common.c	R_GTM_Start_Interrupt_Error R_GTM_Stop_Interrupt_Error
	r_cg_gtm_common_user.c	r_gtm_error_interrupt
	r_cg_atom.h	―
	r_cg_atom_common.c	R_ATOMn_Start_Interrupt_IRQk R_ATOMn_Stop_Interrupt_IRQk
	r_cg_atom_common_user.c	r_atomn_irqk_interrupt
	r_cg_intc.h	―
	r_cg_kcrc.h	―
	r_cg_key.h	―

表 2-3　　出力ファイル(3//21)

周辺機能	ファイル名	API 関数名
共　　通	r_cg_mspi.h	―
	r_cg_mspi_common.c	R_MSPI_Master_Create R_MSPI_Start_Interrupt_MSPInTX R_MSPI_Stop_Interrupt_MSPInTX R_MSPI_Start_Interrupt_MSPInRX R_MSPI_Stop_Interrupt_MSPInRX R_MSPI_Start_Interrupt_MSPInFE R_MSPI_Stop_Interrupt_MSPInFE R_MSPI_Start_Interrupt_MSPInERR R_MSPI_Stop_Interrupt_MSPInERR
	r_cg_mspi_common_user.c	r_mspin_interrupt_send r_mspin_interrupt_receive r_mspin_interrupt_frameend r_mspin_interrupt_error
	r_cg_ostm.h	―
	r_cg_port.h	―
	r_cg_riic.h	―
	r_cg_rtc.h	―
	r_cg_sci3.h	―
	r_cg_stbc.h	―
	r_cg_taub.h	―
	r_cg_taud.h	―
	r_cg_tauj.h	―
	r_cg_tbu.h	―
	r_cg_tim.h	―
	r_cg_uart.h	―
	r_cg_wdt.h	―
	r_smc_interrup.c	R_Interrupt_Create
	r_smc_interrupt.h	―
	r_smc_intprg.c	―
	r_smc_entry.h	―
	Pin.c	R_Pins_Create
	Pin.h	―

表 2-4　　出力ファイル(4//21)

周辺機能	ファイル名	API 関数名
A/D コンバータ	Config_ADCA*n*.c	R_<Config_ADCXn>_Create R_<Config_ADCXn>_Halt R_<Config_ADCXn>_ SelfDiagnostic_VoltageCircuitOn R_<Config_ADCXn>_ SelfDiagnostic_VoltageCircuitOff R_<Config_ADCXn>_ScanGroupx_Start R_<Config_ADCXn>_ScanGroupx_OperationOn R_<Config_ADCXn>_ScanGroupx_OperationOff R_<Config_ADCXn>_ScanGroupx_GetResult R_<Config_ADCXn>_ScanGroupx_GetPWMDiagResult R_<Config_ADCXn>_ScanGroupx_GetFloatingPointResult R_<Config_ADCXn>_TH_Groupk_Start R_<Config_ADCXn>_TH_Sampling_Start R_<Config_ADCXn>_TH_Stop R_<Config_ADCXn>_SetMultiplexerCommand R_<Config_ADCXn>_ScanGroupx_TimerStart R_<Config_ADCXn>_ScanGroupx_TimerStop R_<Config_ADCXn>_VoltageDivider_Start R_<Config_ADCXn>_VoltageDivider_Stop R_<Config_ADCXn>_StrongPullDown_Start R_<Config_ADCXn>_StrongPullDown_Stop R_<Config_ADCXn>_SGDiag_Start R_<Config_ADCXn>_SGDiag_OperationOn R_<Config_ADCXn>_SGDiag_OperationOff R_<Config_ADCXn>_SGDiag_GetResult R_<Config_ADCXn>_SGDiag_GetSubtractionResult R_<Config_ADCXn>_ASF_Start R_<Config_ADCXn>_ASF_Stop R_<Config_ADCXn>_ASF_GetResult R_<Config_ADCXn>_ASF_GetRealTimeResult
	<Config_ADCAn>_user.c	R_<Config_ADCXn>_Create_UserInit r_<Config_ADCXn>_error_interrupt r_<Config_ADCXn>_scan_groupx_end_interrupt r_<Config_ADCXn>_sg_diag_end_interrupt r_<Config_ADCXn>_mpx_request_interrupt r_<Config_ADCXn>_asf_channelk_end_interrupt r_<Config_ADCXn>_monitor_error_interrupt
	Config_ADCA*n*.h	―

表 2-5　出力ファイル(5//21)

周辺機能	ファイル名	API 関数名
CSI スレーブ	Config_CSI*Xn*.c	R_<Config_CSIXn>_Create R_<Config_CSIXn>_Start R_<Config_CSIXn>_Stop R_<Config_CSIXn>_Send R_<Config_CSIXn>_Receive R_<Config_CSIXn>_Send_Receive
	Config_CSI*Xn*_user.c	R_<Config_CSIXn>_Create_UserInit r_<Config_CSIXn>_interrupt_send r_<Config_CSIXn>_interrupt_receive r_<Config_CSIXn>_interrupt_error
	Config_CSI*Xn*.h	－
CSI マスタ	Config_CSIG*n*.c	R_<Config_CSIXn>_Create R_<Config_CSIXn>_Start R_<Config_CSIXn>_Stop R_<Config_CSIXn>_Send R_<Config_CSIXn>_Receive R_<Config_CSIXn>_Send_Receive
	Config_CSIG*n*_user.c	R_<Config_CSIXn>_Create_UserInit r_<Config_CSIXn>_interrupt_send r_<Config_CSIXn>_interrupt_receive r_<Config_CSIXn>_interrupt_error
	Config_CSIG*n*.h	－

表 2-6　　出力ファイル(6//21)

周辺機能	ファイル名	API 関数名
割り込み	Config_INTC.c	R_<Config_INTC>_Create R_<Config_INTC>_INTPn_Start R_<Config_INTC>_INTPn_Stop R_<Config_INTC>_IRQn_Start R_<Config_INTC>_IRQn_Stop R_<Config_INTC>_NMI_Start R_<Config_INTC>_NMI_Stop R_<Config_INTC>_SINTn_Start R_<Config_INTC>_SINTn_Stop R_<Config_INTC>_SINTn_TriggerOn
	Config_INTC_user.c	R_<Config_INTC>_Create_UserInit r_<Config_INTC>_intpn_interrupt r_<Config_INTC>_intpn_interrupt_pek r_<Config_INTC>_irqn_interrupt r_<Config_INTC>_nmi_interrupt r_<Config_INTC>_sintn_interrupt_pek
	Config_ICU.h	－
入力インターバルタイマ	Config_TAU*Xn_m*.c	R_<Config_TAUXn_m>_Create R_<Config_TAUXn_m>_Start R_<Config_TAUXn_m>_Stop
	Config_TAU*Xn_m*_user.c	R_<Config_TAUXn_m>_Create_UserInit r_<Config_TAUXn_m>_interrupt r_<Config_TAUXn_m>_interrupt_pek
	Config_TAU*Xn_m*.h	－
入力パルスインターバル測定	Config_TAU*Xn_m*.c	R_<Config_TAUXn_m>_Create R_<Config_TAUXn_m>_Start R_<Config_TAUXn_m>_Stop R_<Config_TAUXn_m>_Get_PulseWidth
	Config_TAU*Xn_m*_user.c	R_<Config_TAUXn_m>_Create_UserInit R_<Config_TAUXn_m>_interrupt r_<Config_TAUXn_m>_interrupt_pek
	Config_TAU*Xn_m*.h	－

表 2-7　　出力ファイル(7//21)

周辺機能	ファイル名	API 関数名
インターバルタイマ	Config_TAU*Xn_m*.c	R_<Config_TAUXn_m>_Create R_<Config_TAUXn_m>_Start R_<Config_TAUXn_m>_Stop
	Config_TAU*Xn_m*_user.c	R_<Config_TAUXn_m>_Create_UserInit r_<Config_TAUXn_m>_interrupt r_<Config_TAUXn_m>_interrupt_pek
	Config_TAU*Xn_m*.h	－
三角波 PWM 出力	Config_TAU*Xn*.c	R_<Config_TAUXn>_Create R_<Config_TAUXn>_Start R_<Config_TAUXn>_Stop
	Config_TAU*Xn*_user.c	R_<Config_TAUXn>_Create_UserInit r_<Config_TAUXn>_channelm_interrupt r_<Config_TAUXn>_channelm_interrupt_pek
	Config_TAU*Xn*.h	－
三角波 PWM 出力（デッドタイム付き）	Config_TAU*Xn*.c	R_<Config_TAUXn>_Create R_<Config_TAUXn>_Start R_<Config_TAUXn>_Stop
	Config_TAU*Xn*_user.c	R_<Config_TAUXn>_Create_UserInit r_<Config_TAUXn>_channelm_interrupt r_<Config_TAUXn>_channelm_interrupt_pek
	Config_TAU*Xn*.h	－
OS タイマ	Config_OSTM*n*.c	R_<Config_OSTMn>_Create R_<Config_OSTMn>_Start R_<Config_OSTMn>_Stop R_<Config_OSTMn>_Set_CompareValue
	Config_OSTM*n*_user.c	R_<Config_OSTMn>_Create_UserInit r_<Config_OSTMn>_interrupt
	Config_OSTM*n*.h	－
ポート	Config_PORT.c	R_<Config_PORT>_Create
	Config_PORT_user.c	R_<Config_PORT>_Create_UserInit
	Config_PORT.h	－
PWM 出力	Config_TAU*Xn*.c	R_<Config_TAUXn>_Create R_<Config_TAUXn>_Start R_<Config_TAUXn>_Stop
	Config_TAU*Xn*_user.c	R_<Config_TAUXn>_Create_UserInit r_<Config_TAUXn>_channelm_interrupt r_<Config_TAUXn_m>_interrupt_pek
	Config_TAU*Xn*.h	－

表 2-8　　出力ファイル(8//21)

周辺機能	ファイル名	API 関数名
スタンバイコントローラ	Config_STBC.c	R_<Config_STBC>_Prepare_Stop_Mode R_<Config_STBC>_Start_Stop_Mode R_<Config_STBC>_Prepare_Deep_Stop_Mode R_<Config_STBC>_Start_Deep_Stop_Mode R_<Config_STBC>_Deep_Stop_Loop R_<Config_STBC>_Set_Module_*XXXX*_Standby_Mode R_<Config_STBC>_Cancel_Module_*XXXX*_Standby_Mode
	Config_STBC_user.c	R_<Config_STBC>_Prepare_Stop_Mode_Set_Peripheral R_<Config_STBC>_Prepare_Stop_Mode_Set_Interrupt R_<Config_STBC>_Prepare_Stop_Mode_Set_Clock_Mask R_<Config_STBC>_Prepare_Stop_Mode_Set_Clock_Source R_<Config_STBC>_Prepare_Stop_Mode_Set_CPUCLK R_<Config_STBC>_Prepare_Deep_Stop_Mode_Set_Peripheral R_<Config_STBC>_Prepare_Deep_Stop_Mode_Set_Interrupt R_<Config_STBC>_Prepare_Deep_Stop_Mode_Set_Clock_Mask R_<Config_STBC>_Prepare_Deep_Stop_Mode_Set_Clock_Source R_<Config_STBC>_Check_Module_*XXXX*_Idle_State
	Config_STBC.h	－
UART インタフェース	Config_UART*n*.c	R_<Config_UARTn>_Create R_<Config_UARTn>_Start R_<Config_UARTn>_Stop R_<Config_UARTn>_Send R_<Config_UARTn>_Receive
	Config_UART*n*_user.c	R_<Config_UARTn>_Create_UserInit r_<Config_UARTn>_interrupt_send r_<Config_UARTn>_interrupt_receive r_<Config_UARTn>_interrupt_error
	Config_UART*n*.h	－
ウォッチドッグタイマ	Config_WDT*n*.c Config_WDT*Xn*.c	R_<Config_WDTn>_Create R_<Config_WDTn>_Restart R_<Config_WDTXn_Create R_<Config_WDTXn_Restart
	Config_WDT*n*_user.c Config_WDT*Xn*_user.c	R_<Config_WDTn>_Create_UserInit r_<Config_WDTn>_interrupt R_<Config_WDTXn_Create_UserInit r_<Config_WDTXn_interrupt
	Config_WDT*n*.h	－

表 2-9　　出力ファイル(9//21)

周辺機能	ファイル名	API 関数名
クロック分周	Config_TAU*Xn*.c	R_<Config_TAUXn_m>_Create R_<Config_TAUXn_m>_Start R_<Config_TAUXn_m>_Stop
	Config_TAU*Xn*_user.c	r_<Config_TAUXn_m>_Create_UserInit r_<Config_TAUXn_m>_interrupt r_<Config_TAUXn_m>_interrupt_pek
	Config_TAU*Xn*.h	－
データ CRC	Config_DCRA*n*.c Config_KCRA*n*.c	R_<Config_DCRAn>_Create R_<Config_DCRAn>_InitializeCRCData R_<Config_DCRAn>_Input32bitData R_<Config_DCRAn>_Input16bitData R_<Config_DCRAn>_Input8bitData R_<Config_DCRAn>_GetResult_32bitData R_<Config_DCRAn>_GetResult_16bitData R_<Config_KCRCn>_Create R_<Config_KCRCn>_InitializeCRCData R_<Config_KCRCn>_Input32bitData R_<Config_KCRCn>_Input16bitData R_<Config_KCRCn>_Input8bitData R_<Config_KCRCn>_GetResult_64bitData R_<Config_KCRCn>_GetResult_32bitData R_<Config_KCRCn>_GetResult_16bitData R_<Config_KCRCn>_GetResult_8bitData
	Config_DCRA*n*_user.c	R_<Config_DCRAn>_Create_UserInit R_<Config_KCRCn>_Create_UserInit
	Config_DCRA*n*.h	－
ディレイカウント	Config_TAU*Xn_m*.c	R_<Config_TAUXn_m>_Create R_<Config_TAUXn_m>_Start R_<Config_TAUXn_m>_Stop
	Config_TAU*Xn_m*_user.c	R_<Config_TAUXn_m>_Create_UserInit r_<Config_TAUXn_m>_interrupt r_<Config_TAUXn_m>_interrupt_pek
	Config_TAU*Xn_m*.h	－

表 2-10　　出力ファイル(10//21)

周辺機能	ファイル名	API 関数名
DMA コントローラ	Config_DMAC*n*.c	R_<Config_DMACn>_Create R_<Config_DMACnm>_Start R_<Config_DMACnm>_Stop R_<Config_DMACnm>_Set_SoftwareTrigger R_<Config_DMACnm>_Suspend R_<Config_DMACnm>_Resume R_<Config_SDMACnm>_Create R_<Config_SDMACnm>_Start R_<Config_SDMACnm>_Stop R_<Config_SDMACnm>_Resume R_<Config_SDMACnm>_Reset R_<Config_SDMACnm>_Set_DescriptorMemory
	Config_DMAC*n*_user.c	R_<Config_DMACn_Create_UserInit r_<Config_DMACnm>_dmacnm_interrupt r_<Config_DMACnm>_callback_transfer_completion r_<Config_DMACnm>_callback_transfer_count_match R_<Config_SDMACnm>_Create_UserInit r_<Config_SDMACnm>_end_interrupt r_<Config_SDMACnm>_Callback_PEk_Address_Error
	Config_DMAC*n*.h	－
外部イベントカウント	Config_TAU*Xn_m*.c	R_<Config_TAUXn_m>_Create R_<Config_TAUXn_m>_Start R_<Config_TAUXn_m>_Stop
	Config_TAU*Xn_m*_user.c	R_<Config_TAUXn_m>_Create_UserInit r_<Config_TAUXn_m>_interrupt r_<Config_TAUXn_m>_interrupt_pek
	Config_TAU*Xn_m*.h	－
入力期間カウント検出	Config_TAU*Xn_m*.c	R_<Config_TAUXn_m>_Create R_<Config_TAUXn_m>_Start R_<Config_TAUXn_m>_Stop R_<Config_TAUXn_m>_Get_PulseWidth
	Config_TAU*Xn_m*_user.c	R_<Config_TAUXn_m>_Create_UserInit r_<Config_TAUXn_m>_interrupt r_<Config_TAUXn_m>_interrupt_pek
	Config_TAU*Xn_m*.h	－

表 2-11　　出力ファイル(11//21)

周辺機能	ファイル名	API 関数名
入力位置検出	Config_TAU*Xn_m*.c	R_<Config_TAUXn_m>_Create R_<Config_TAUXn_m>_Start R_<Config_TAUXn_m>_Stop R_<Config_TAUXn_m>_Get_PulseWidth
	Config_TAU*Xn_m*_user.c	R_<Config_TAUXn_m>_Create_UserInit r_<Config_TAUXn_m>_interrupt r_<Config_TAUXn_m>_interrupt_pek
	Config_TAU*Xn_m*.h	－
入力パルスインターバル判定	Config_TAU*Xn_m*.c	R_<Config_TAUXn_m>_Create R_<Config_TAUXn_m>_Start R_<Config_TAUXn_m>_Stop
	Config_TAU*Xn_m*_user.c	R_<Config_TAUXn_m>_Create_UserInit r_<Config_TAUXn_m>_interrupt r_<Config_TAUXn_m>_interrupt_pek
	Config_TAU*Xn_m*.h	－
入力信号幅判定	Config_TAU*Xn_m*.c	R_<Config_TAUXn_m>_Create R_<Config_TAUXn_m>_Start R_<Config_TAUXn_m>_Stop
	Config_TAU*Xn_m*_user.c	R_<Config_TAUXn_m>_Create_UserInit r_<Config_TAUXn_m>_interrupt r_<Config_TAUXn_m>_interrupt_pek
	Config_TAU*Xn_m*.h	－
入力信号幅測定	Config_TAU*Xn_m*.c	R_<Config_TAUXn_m>_Create R_<Config_TAUXn_m>_Start R_<Config_TAUXn_m>_Stop R_<Config_TAUXn_m>_Get_PulseWidth
	Config_TAU*Xn_m*_user.c	R_<Config_TAUXn_m>_Create_UserInit r_<Config_TAUXn_m>_interrupt r_<Config_TAUXn_m>_interrupt_pek
	Config_TAU*Xn_m*.h	－
キー割り込み機能	Config_KEY.c	R_<Config_KEY>_Create R_<Config_KEY>_Start R_<Config_KEY>_Stop
	Config_KEY_user.c	R_<Config_KEY>_Create_UserInit r_<Config_KEY>_interrupt
	Config_KEY.h	－

表 2-12　　出力ファイル(12//21)

周辺機能	ファイル名	API 関数名
ワンパルス出力	Config_TAU*Xn_m*.c	R_<Config_TAUXn_m>_Create R_<Config_TAUXn_m>_Start R_<Config_TAUXn_m>_Stop
	Config_TAU*Xn_m*_user.c	R_<Config_TAUXn_m>_Create_UserInit r_<Config_TAUXn_m>_interrupt r_<Config_TAUXn_m>_interrupt_pek
	Config_TAU*Xn_m*.h	－
ワンショットパルス出力	Config_TAU*Xn_m*.c	R_<Config_TAUXn>_Create R_<Config_TAUXn>_Start R_<Config_TAUXn>_Stop R_<Config_TAUXn>_ SoftwareTriggerOn
	Config_TAU*Xn_m*_user.c	R_<Config_TAUXn>_Create_UserInit r_<Config_TAUXn>_channelm_interrupt r_<Config_TAUXn>_channelm_interrupt_pek
	Config_TAU*Xn_m*.h	－
オーバーフロー割り込み出力機能（入力期間カウント検出）	Config_TAU*Xn_m*.c	R_<Config_TAUXn_m>_Create R_<Config_TAUXn_m>_Start R_<Config_TAUXn_m>_Stop
	Config_TAU*Xn_m*_user.c	R_<Config_TAUXn_m>_Create_UserInit r_<Config_TAUXn_m>_interrupt r_<Config_TAUXn_m>_interrupt_pek
	Config_TAU*Xn_m*.h	－
オーバーフロー割り込み出力機能（幅測定）	Config_TAU*Xn_m*.c	R_<Config_TAUXn_m>_Create R_<Config_TAUXn_m>_Start R_<Config_TAUXn_m>_Stop
	Config_TAU*Xn_m*_user.c	R_<Config_TAUXn_m>_Create_UserInit r_<Config_TAUXn_m>_interrupt r_<Config_TAUXn_m>_interrupt_pek
	Config_TAU*Xn_m*.h	－

表 2-13　出力ファイル(13//21)

周辺機能	ファイル名	API 関数名
IIC マスタモード	Config_RIIC*n*.c	R_<Config_RIICn>_Create R_<Config_RIICn>_Start R_<Config_RIICn>_Stop R_<Config_RIICn>_Master_Send R_<Config_RIICn>_Master_Send_Without_Stop R_<Config_RIICn>_Master_Receive R_<Config_RIICn>_StartCondition R_<Config_RIICn>_StopCondition
	Config_RIIC*n*_user.c	R_<Config_RIICn>_Create_UserInit r_<Config_RIICn>_error_interrupt r_<Config_RIICn>_receive_interrupt r_<Config_RIICn>_transmit_interrupt r_<Config_RIICn>_transmitend_interrupt r_<Config_RIICn>_callback_transmitend r_<Config_RIICn>_callback_receiveend r_<Config_RIICn>_callback_receiveerror
	Config_RIIC*n*.h	－
IIC スレーブモード	Config_RIIC*n*.c	R_<Config_RIICn>_Create R_<Config_RIICn>_Start R_<Config_RIICn>_Stop R_<Config_RIICn>_Slave_Send R_<Config_RIICn>_Slave_Receive R_<Config_RIICn>_StartCondition R_<Config_RIICn>_StopCondition
	Config_RIIC*n*_user.c	R_<Config_RIICn>_Create_UserInit r_<Config_RIICn>_error_interrupt r_<Config_RIICn>_receive_interrupt r_<Config_RIICn>_transmit_interrupt r_<Config_RIICn>_transmitend_interrupt r_<Config_RIICn>_callback_transmitend r_<Config_RIICn>_callback_receiveend r_<Config_RIICn>_callback_receiveerror
	Config_RIIC*n*.h	－

表 2-14　出力ファイル(14//21)

周辺機能	ファイル名	API 関数名
SCI3 クロック同期式モード	Config_SCI3*n*.c	R_<Config_SCI3n>_Create R_<Config_SCI3n>_Start R_<Config_SCI3n>_Stop R_<Config_SCI3n>_Send R_<Config_SCI3n>_Receive
	Config_SCI3_user *n*.c	R_<Config_SCI3n>_Create_UserInit r_<Config_SCI3n>_interrupt_send r_<Config_SCI3n>_interrupt_receive r_<Config_SCI3n>_interrupt_error r_<Config_SCI3n>_interrupt_sendend r_<Config_SCI3n>_callback_sendend r_<Config_SCI3n>_callback_receiveend r_<Config_SCI3n>_callback_error
	Config_SCI3*n*.h	－
SCI3 調歩同期式モード	Config_SCI3*n*.c	R_<Config_SCI3n>_Create R_<Config_SCI3n>_Start R_<Config_SCI3n>_Stop R_<Config_SCI3n>_Send R_<Config_SCI3n>_Receive R_<Config_SCI3n>_Multiprocessor_Send R_<Config_SCI3n>_Multiprocessor_Receive
	Config_SCI3_user *n*.c	R_<Config_SCI3n>_Create_UserInit r_<Config_SCI3n>_interrupt_send r_<Config_SCI3n>_interrupt_receive r_<Config_SCI3n>_interrupt_error r_<Config_SCI3n>_interrupt_sendend r_<Config_SCI3n>_callback_sendend r_<Config_SCI3n>_callback_receiveend r_<Config_SCI3n>_callback_error
	Config_SCI3*n*.h	－

表 2-15　出力ファイル(15//21)

周辺機能	ファイル名	API 関数名
MSPI マスタモード	Config_MSPI*nm*.c	R_<Config_MSPInm>_Create R_<Config_MSPInm>_Start R_<Config_MSPInm>_Stop R_<Config_MSPInm>_Send R_<Config_MSPInm>_Receive R_<Config_MSPInm>_Software_Trigger
	Config_MSPI*nm*_user n.c	R_<Config_MSPInm>_Create_UserInit r_<Config_MSPInm>_channelnm_interrupt_send r_<Config_MSPInm>_channelnm_interrupt_receive r_<Config_MSPInm>_Callback_Interrupt_Send r_<Config_MSPInm>_Callback_Interrupt_Receive r_<Config_MSPInm>_Callback_Interrupt_Error r_<Config_MSPInm>_Callback_Interrupt_Frameend r_<Config_MSPInm>_callback_sendend r_<Config_MSPInm>_callback_receiveend r_<Config_MSPInm>_callback_error
	Config_MSPI*nm*.h	―
MSPI スレーブモード	Config_MSPI*nm*.c	R_<Config_MSPIn>_Create R_<Config_MSPIn>_Start R_<Config_MSPIn>_Stop R_<Config_MSPIn>_Send R_<Config_MSPIn>_Receive R_<Config_MSPIn>_Software_Trigger
	Config_MSPI*nm*_user n.c	R_<Config_MSPIn>_Create_UserInit r_<Config_MSPIn>_channeln0_interrupt_send r_<Config_MSPIn>_channeln0_interrupt_receive r_<Config_MSPIn>_interrupt_send r_<Config_MSPIn>_interrupt_receive r_<Config_MSPIn>_interrupt_error r_<Config_MSPIn>_interrupt_frameend r_<Config_MSPIn>_callback_sendend r_<Config_MSPIn>_callback_receiveend r_<Config_MSPIn>_callback_error
	Config_MSPI*nm*.h	―

表 2-16　出力ファイル(16//21)

周辺機能	ファイル名	API 関数名
リアルタイムクロック	Config_RTC.c	R_<Config_RTCAn>_Create R_<Config_RTCAn>_Start R_<Config_RTCAn>_Stop R_<Config_RTCAn>_Set_HourSystem R_<Config_RTCAn>_Get_CounterBufferValue R_<Config_RTCAn>_Get_CounterDirectValue R_<Config_RTCAn>_Set_CounterValue R_<Config_RTCAn>_Get_SubCounterValue R_<Config_RTCAn>_Set_ErrorCorrectionValue R_<Config_RTCAn>_Set_SubCounterCompareValue R_<Config_RTCAn>_Get_AlarmValue R_<Config_RTCAn>_Set_AlarmValue R_<Config_RTCAn>_Set_AlarmOn R_<Config_RTCAn>_Set_AlarmOff R_<Config_RTCAn>_Set_ConstPeriodInterruptOn R_<Config_RTCAn>_Set_ConstPeriodInterruptOff R_<Config_RTCAn>_Set_1secondInterruptOn R_<Config_RTCAn>_Set_1secondInterruptOff R_<Config_RTCAn>_Set_RTCA1HZOn R_<Config_RTCAn>_Set_RTCA1HZOff
	Config_RTC_user.c	R_<Config_RTCAn>_Create_UserInit r_<Config_RTCAn>_interrupt_alarm r_<Config_RTCAn>_interrupt_periodic r_<Config_RTCAn>_interrupt_1second
	Config_RTC.h	－
エラーコントロールモジュール	Config_ECM.c	R_<Config_ECM>_Create R_<Config_ECM>_Start R_<Config_ECM>_Stop R_<Config_ECM>_Set_Master_Error R_<Config_ECM>_Clear_Master_Error R_<Config_ECM>_Set_Checker_Error R_<Config_ECM>_Clear_Checker_Error R_<Config_ECM>_Delay_Timer_Start R_<Config_ECM>_Delay_Timer_Stop R_<Config_ECM>_Pseud_XXXX_Start R_<Config_ECM>_Pseudo_*XXXX*_Stop
	Config_ECM_user.c	R_<Config_ECM>_Create_UserInit r_<Config_ECM>_ecmmi_interrupt_pek r_<Config_ECM>_mi_interrupt_pek r_<Config_ECM>_dclsmi_interrupt_pek
	Config_ECM.h	－

表 2-17　出力ファイル(17//21)

周辺機能	ファイル名	API 関数名
GTM Clock	Config_GTM_Clock.c	R_<Config_GTM>_Clock_Create R_<Config_GTM>_CCMn_Create
	Config_GTM_Clock_user.c	R_<Config_GTM>_Clock_Create_UserInit
	Config_GTM_Clock.h	―
Time Base Unit	Config_TBU.c	R_<Config_TBU>_Create R_<Config_TBU>_Start R_<Config_TBU>_Stop
	Config_TBU_user.c	R_<Config_TBU>_Create_UserInit
	Config_TBU.h	―
TIM Bit Compression Mode	Config_TIMn.c	R_<Config_TIMn>_Create R_<Config_TIMn>_Start R_<Config_TIMn>_Stop R_<Config_TIMn>_SetDetectedEdge R_<Config_TIMn>_Software_Reset R_<Config_TIMn>_NEWVAL_TriggerOn R_<Config_TIMn>_ECNTOFL_TriggerOn R_<Config_TIMn>_CNTOFL_TriggerOn R_<Config_TIMn>_GPROFL_TriggerOn R_<Config_TIMn>_TODETOFL_TriggerOn R_<Config_TIMn>_ChannelIm_GLITCHDETOFL_TriggerOn
	Config_TIMn_user.c	R_<Config_TIMn>_Create_UserInit r_<Config_TIMn>_share_interrupt r_<Config_TIMn>_share_interrupt r_<Config_TIMn>_chm_share_interrupt
	Config_TIMn.h	―
TIM Gated Periodic Sampling Mode	Config_TIMn_m.c	R_<Config_TIMn_m>_Create R_<Config_TIMn_m>_Start R_<Config_TIMn_m>_Stop R_<Config_TIMn_m>_UpdateElapsedClock R_<Config_TIMn_m>_Software_Reset R_<Config_TIMn_m>_NEWVAL_TriggerOn R_<Config_TIMn_m>_ECNTOFL_TriggerOn R_<Config_TIMn_m>_CNTOFL_TriggerOn R_<Config_TIMn_m>_GPROFL_TriggerOn R_<Config_TIMn_m>_TODETOFL_TriggerOn R_<Config_TIMn_m>_GLITCHDETOFL_TriggerOn
	Config_TIMn_m_user.c	R_<Config_TIMn_m>_Create_UserInit r_<Config_TIMn_m>_ Callback_GTM_Error r_<Config_TIMn_m>_share_interrupt
	Config_TIMn_m.h	―

表 2-18　出力ファイル(18//21)

周辺機能	ファイル名	API 関数名
TIM Input Prescaler Mode	Config_TIMn_m.c	R_<Config_TIMn_m>_Create R_<Config_TIMn_m>_Start R_<Config_TIMn_m>_Stop R_<Config_TIMn_m>_SetDetectedEdge R_<Config_TIMn_m>_Software_Reset R_<Config_TIMn_m>_NEWVAL_TriggerOn R_<Config_TIMn_m>_ECNTOFL_TriggerOn R_<Config_TIMn_m>_CNTOFL_TriggerOn R_<Config_TIMn_m>_GPROFL_TriggerOn R_<Config_TIMn_m>_TODETOFL_TriggerOn R_<Config_TIMn_m>_GLITCHDETOFL_TriggerOn
	Config_TIMn_m_user.c	R_<Config_TIMn_m>_Create_UserInit r_<Config_TIMn_m>_Callback_GTM_Error r_<Config_TIMn_m>_share_interrupt
	Config_TIMn_m.h	－
TIM Input Event Mode	Config_TIMn_m.c	R_<Config_TIMn_m>_Create R_<Config_TIMn_m>_Start R_<Config_TIMn_m>_Stop R_<Config_TIMn_m>_Software_Reset R_<Config_TIMn_m>_NEWVAL_TriggerOn R_<Config_TIMn_m>_ECNTOFL_TriggerOn R_<Config_TIMn_m>_CNTOFL_TriggerOn R_<Config_TIMn_m>_GPROFL_TriggerOn R_<Config_TIMn_m>_TODETOFL_TriggerOn R_<Config_TIMn_m>_GLITCHDETOFL_TriggerOn
	Config_TIMn_m_user.c	R_<Config_TIMn_m>_Create_UserInit r_<Config_TIMn_m>_Callback_GTM_Error r_<Config_TIMn_m>_share_interrupt
	Config_TIMn_m.h	－

表 2-19　出力ファイル(19//21)

周辺機能	ファイル名	API 関数名
TIM Pulse Integration Mode	Config_TIMn_m.c	R_<Config_TIMn_m>_Create R_<Config_TIMn_m>_Start R_<Config_TIMn_m>_Stop R_<Config_TIMn_m>_Software_Reset R_<Config_TIMn_m>_NEWVAL_TriggerOn R_<Config_TIMn_m>_ECNTOFL_TriggerOn R_<Config_TIMn_m>_CNTOFL_TriggerOn R_<Config_TIMn_m>_GPROFL_TriggerOn R_<Config_TIMn_m>_TODETOFL_TriggerOn R_<Config_TIMn_m>_GLITCHDETOFL_TriggerOn
	Config_TIMn_m_user.c	R_<Config_TIMn_m>_Create_UserInit r_<Config_TIMn_m>_Callback_GTM_Error r_<Config_TIMn_m>_share_interrupt
	Config_TIMn_m.h	－
TIM PWM Measurement Mode	Config_TIMn_m.c	R_<Config_TIMn_m>_Create R_<Config_TIMn_m>_Start R_<Config_TIMn_m>_Stop R_<Config_TIMn_m>_Software_Reset R_<Config_TIMn_m>_NEWVAL_TriggerOn R_<Config_TIMn_m>_ECNTOFL_TriggerOn R_<Config_TIMn_m>_CNTOFL_TriggerOn R_<Config_TIMn_m>_GPROFL_TriggerOn R_<Config_TIMn_m>_TODETOFL_TriggerOn R_<Config_TIMn_m>_GLITCHDETOFL_TriggerOn
	Config_TIMn_m_user.c	R_<Config_TIMn_m>_Create_UserInit r_<Config_TIMn_m>_Callback_GTM_Error r_<Config_TIMn_m>_share_interrupt
	Config_TIMn_m.h	－

表 2-20　出力ファイル(20//21)

周辺機能	ファイル名	API 関数名
TIM Serial Shift Mode	Config_TIMn_m.c	R_<Config_TIMn_m>_Create R_<Config_TIMn_m>_Start R_<Config_TIMn_m>_Stop R_<Config_TIMn_m>_UpdateElapsedClock R_<Config_TIMn_m>_UpdateShiftClock R_<Config_TIMn_m>_UpdateCaptrueSource R_<Config_TIMn_m>_UpdateTssmOut R_<Config_TIMn_m>_SetGPR0AsShadowRegister R_<Config_TIMn_m>_Software_Reset R_<Config_TIMn_m>_NEWVAL_TriggerOn R_<Config_TIMn_m>_ECNTOFL_TriggerOn R_<Config_TIMn_m>_CNTOFL_TriggerOn R_<Config_TIMn_m>_GPROFL_TriggerOn R_<Config_TIMn_m>_TODETOFL_TriggerOn R_<Config_TIMn_m>_GLITCHDETOFL_TriggerOn
	Config_TIMn_m_user.c	R_<Config_TIMn_m>_Create_UserInit r_<Config_TIMn_m>_Callback_GTM_Error r_<Config_TIMn_m>_share_interrupt
	Config_TIMn_m.h	－
ATOM Signal Output Mode Compare	Config_ATOMn_m.c	R_<Config_ATOMn_m>_Create R_<Config_ATOMn_m>_Start R_<Config_ATOMn_m>_Stop R_<Config_ATOMn_m>_Output_Start R_<Config_ATOMn_m>_Output_Stop R_<Config_ATOMn_m>_Channel_Reset R_<Config_ATOMn_m>_LateUpdate_Request
	Config_ATOMn_m_user.c	R_<Config_ATOMn_m>_Create_UserInit R_<Config_ATOMn_m>_Callback_Shared_
	Config_ATOMn_m.h	－
ATOM Signal Output Mode Immediate	Config_ATOMn_m.c	R_<Config_ATOMn_m>_Create R_<Config_ATOMn_m>_Start R_<Config_ATOMn_m>_Stop R_<Config_ATOMn_m>_Output_Start R_<Config_ATOMn_m>_Output_Stop R_<Config_ATOMn_m>_Channel_Reset
	Config_ATOMn_m_user.c	R_<Config_ATOMn_m>_Create_UserInit
	Config_ATOMn_m.h	－

表 2-21　出力ファイル(21//21)

周辺機能	ファイル名	API 関数名
ATOM Signal Output Mode PWM	Config_ATOMn_m.c	R_<Config_ATOMn_m>_Create R_<Config_ATOMn_m>_Start R_<Config_ATOMn_m>_Stop R_<Config_ATOMn_m>_Output_Start R_<Config_ATOMn_m>_Output_Stop R_<Config_ATOMn_m>_Channel_Reset
	Config_ATOMn_m_user.c	R_<Config_ATOMn_m>_Create_UserInit R_<Config_ATOMn_m>_Callback_Shared_IRQk
	Config_ATOMn_m.h	－
ATOM Signal Output Mode Serial	Config_ATOMn_m.c	R_<Config_ATOMn_m>_Create R_<Config_ATOMn_m>_Start R_<Config_ATOMn_m>_Stop R_<Config_ATOMn_m>_Output_Start R_<Config_ATOMn_m>_Output_Stop R_<Config_ATOMn_m>_Channel_Reset
	Config_ATOMn_m_user.c	R_<Config_ATOMn_m>_Create_UserInit R_<Config_ATOMn_m>_Callback_Shared_IRQk
	Config_ATOMn_m.h	－
Dead Time Module	Config_CDTMn_m.c	R_<Config_CDTMn_m>_Create
	Config_CDTMn_m_user.c	R_<Config_CDTMn_m>_Create_UserInit
	Config_CDTMn_m.h	－
DTS コントローラ	Config_DTSn.c	R_<Config_DTSn>_Create R_<Config_DTSn>_Start R_<Config_DTSn>_Stop R_<Config_DTSn>_Set_SoftwareTrigger
	Config_DTSn_user.c	R_<Config_DTSn>_Create_UserInit r_<Config_DTSn>_Callback_Dtsg_Transfer_End r_<Config_DTSn>_Callback_Dtsg_Transfer_Count_Match r_<Config_DTSn>_Callback_PEk_Transfer_Error r_<Config_DTSn>_Callback_Transfer_Error
	Config_DTSn.h	－
ADC Boundary Flag Generator	ABFG.c	R_<Config_ABFG>_Create R_<Config_ABFG>_Start R_<Config_ABFG>_Stop
	ABFG_user.c	R_<Config_ABFG>_Create_UserInit r_<Config_ABFG>_boundary_flag_pulse_w_interrupt
	ABFG.h	－

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。

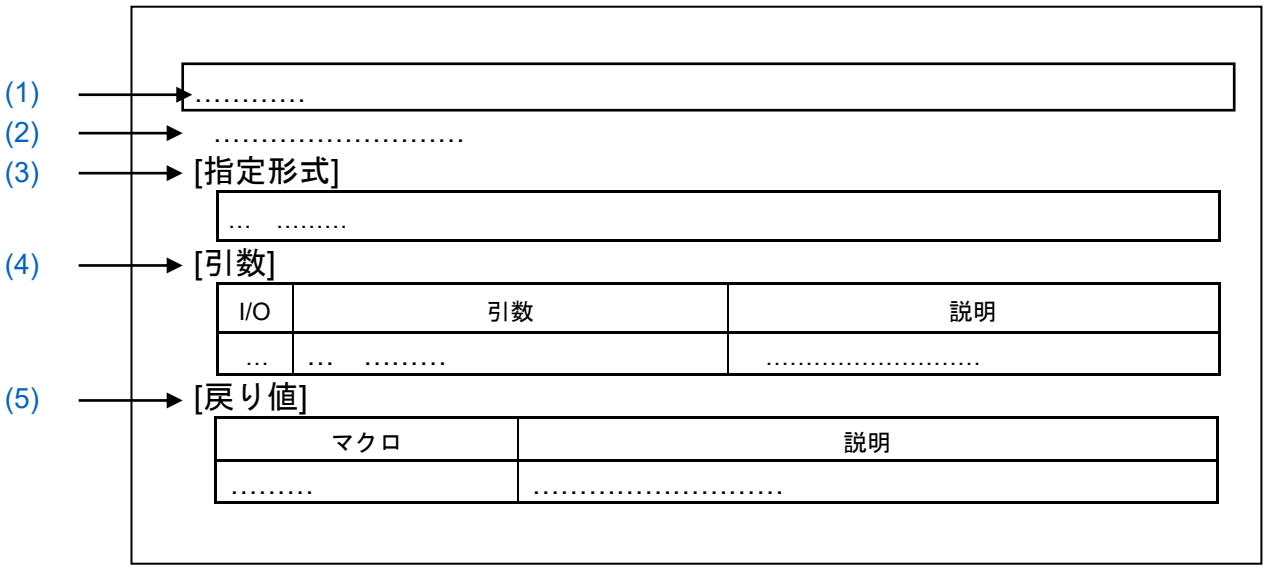
- 列挙指定子 enum の要素名
  すべて大文字。

備考 コード生成ツールの生成するコードには、レジスタの反映待ち処理等で for 文、while 文、do while 文（ループ処理）を使用しています。
無限ループに対するフェールセーフ処理が必要な場合は、生成されたコードを確認の上、処理を追加してください。

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、コード生成ツールが共通用に出力する 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
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。
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。
R_CGC_Create	クロック発生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_CGC_Create_UserInit	クロック発生回路に関するユーザ独自の初期化処理を行います。
R_Interrupt_Create	（予約関数）
R_Pins_Create	端子機能の設定のリファレンスです。
R_Interrupt_Initialize_ForPE	各 CPU コアで使用されるすべての割り込みを初期化します。
R_ADC_SyncStart	全ての A/D コンバータの各スキャングループを同時に開始します。
R_ADC_TimerSyncStart	全ての A/D コンバータの A/D タイマのカウントを同時に開始します。
R_DMAC_Create	全ての DMA チャネルを初期化します。
R_DMA_Suspend	全ての DMA チャネルの転送をサスペンドします。
R_DMA_Resume	全ての DMA チャネルの転送を再開します。
R_PDMAn_Suspend	対象の DMA ユニットの全てのチャネルの転送をサスペンドします。
R_PDMAn_Resume	対象の DMA ユニットの全てのチャネルの転送を再開します。
R_DMAC_Start_Interrupt_Error_PEk	PEk の DMA エラー割り込みを開始します。
R_DMAC_Stop_Interrupt_Error_PEk	PEk の DMA エラー割り込みを停止します。
r_sdmac_address_error_interrupt_pek	DMA アドレスエラー割り込みに伴う処理を行います
R_DMA_Start_Transfer_Error_Interrupt	DMAC および DTS の全てのチャネルの DMA 転送エラー割り込みを開始します。
R_DMA_Stop_Transfer_Error_Interrupt	DMAC および DTS の全てのチャネルの DMA 転送エラー割り込みを停止します。
r_dmac_error_interrupt_pe	PE1 の DMA アドレスエラー割り込みに伴う処理を行います。

API 関数名	機能概要
R_DTS_Suspend	DTS 転送を一時停止します。
R_DTS_Resume	DTS 転送を再開します。
R_DTSg_Start_Transfer_End_Interrupt	DTS 転送終了割り込みを開始します。
R_DTSg_Stop_Transfer_End_Interrupt	DTS 転送終了割り込みを停止します。
R_DTSg_Start_Transfer_Count_Match_Interrupt	DTS 転送回数一致割り込みを開始します。
R_DTSg_Stop_Transfer_Count_Match_Interrupt	DTS 転送回数一致割り込みを停止します。
R_DTS_Start_PEk_Transfer_Error_Interrupt	DTS 転送エラー一致割り込みを開始します。
R_DTS_Stop_PEk_Transfer_Error_Interrupt	DTS 転送エラー一致割り込みを停止します。
r_dtsg_transfer_end_interrupt	DTS チャネル転送終了割り込みに伴う処理を行います。
r_dtsg_transfer_count_match_interrupt	DTS チャネル転送回数一致割り込みに伴う処理を行います。
r_dts_transfer_error_interrupt_pek	DTS 転送エラー割り込みに伴う処理を行います。
R_GTM_Start_Interrupt_Error	GTM エラー割り込みを有効にします。
R_GTM_Stop_Interrupt_Error	GTM エラー割り込みを無効にします。
r_gtm_error_interrupt	GTM エラー割り込みに伴う処理を行います。
R_ATOMn_Start_Interrupt_IRQk	INTGTMA0ATOMnk 割り込みを開始します。
R_ATOMn_Stop_Interrupt_IRQk	INTGTMA0ATOMnk 割り込みを停止します。
r_atomn_irqk_interrupt	INTGTMA0ATOMnk 割り込みに伴う処理を行います。
R_MSPI_Master_Create	全ての MSPI チャネルを初期化します。
R_MSPI_Start_Interrupt_MSPInTX	MSPIn 送信割り込みを有効にします。
R_MSPI_Stop_Interrupt_MSPInTX	MSPIn 送信割り込みを無効にします。
R_MSPI_Start_Interrupt_MSPInRX	MSPIn 受信割り込みを有効にします。
R_MSPI_Stop_Interrupt_MSPInRX	MSPIn 受信割り込みを無効にします。
R_MSPI_Start_Interrupt_MSPInFE	MSPIn フレーム終了割り込みを有効にします。
R_MSPI_Stop_Interrupt_MSPInFE	MSPIn フレーム終了割り込みを無効にします。
R_MSPI_Start_Interrupt_MSPInERR	MSPIn エラー割り込みを有効にします。
R_MSPI_Stop_Interrupt_MSPInERR	MSPIn エラー割り込みを無効にします。
r_mspin_interrupt_send	MSPIn 送信割り込みに伴う処理を行います。
r_mspin_interrupt_receive	MSPIn 受信割り込みに伴う処理を行います。

main

main 関数です。

[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

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、R_MAIN_UserInit 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

R_CGC_Create

クロック発生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。

備考　　本 API 関数は、main()関数内にて、R_Systeminit から呼び出されます。

[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_Interrupt_Create

予約関数です。（処理コードは出力されません。）

[指定形式]

```
void        R_Interrupt_Create ( void );
```

[引数]

なし

[戻り値]

なし

R_Pins_Create

端子機能の初期リファレンス設定をします（I/O ポート以外）。
スマート・コンフィグレータの[端子]ページで設定した端子機能について、設定コードをリファレンスとして出力しています。リファレンスのため、どの関数からも呼び出されません。

[指定形式]

```
void        R_Pins_Create ( void );
```

[引数]
なし

[戻り値]
なし

R_Interrupt_Initialize_ForPE

各 CPU コアで使用されるすべての割り込みを初期化します。
備考　　本 API 関数は、R_Systeminit から呼び出されます。

[指定形式]

```
void    R_Interrupt_Initialize_ForPE ( void );
```

[引数]

なし

[戻り値]

なし

R_ADC_SyncStart

全ての A/D コンバータの各スキャングループを同時に開始します。
備考　　本 API 関数は、ユーザが呼び出します。

[指定形式]

```
void    R_ADC_SyncStart(void);
```

[引数]

なし

[戻り値]

なし

R_ADC_TimerSyncStart

全ての A/D コンバータの A/D タイマのカウントを同時に開始します。
備考　　本 API 関数は、ユーザが呼び出します。

[指定形式]

```
void    R_ADC_TimerSyncStart(void);
```

[引数]

なし

[戻り値]

なし

R_DMAC_Create

全ての DMA チャネルを初期化します。

備考　　本 API 関数は、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

```
void    R_DMAC_Create(void)
```

[引数]

なし

[戻り値]

なし

R_DMA_Suspend

全ての DMA チャネルの転送をサスペンドします。
備考　　本 API 関数は、ユーザが呼び出します。

[指定形式]

```
void    R_DMA_Suspend(void)
```

[引数]

なし

[戻り値]

なし

R_DMA_Resume

全ての DMA チャネルの転送を再開します。
備考　　本 API 関数は、ユーザが呼び出します。

[指定形式]

```
void    R_DMA_Resume(void)
```

[引数]
なし

[戻り値]
なし

R_PDMA*n*_Suspend

対象の DMA ユニット *n* の全てのチャネルの転送をサスペンドします。
備考　　本 API 関数は、ユーザが呼び出します。

[指定形式]

void　R_PDMA*n*_Suspend(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_PDMA*n*_Resume

対象の DMA ユニット *n* の全てのチャネルの転送を再開します。

備考　　本 API 関数は、ユーザが呼び出します。

[指定形式]

void　R_PDMA*n*_Resume(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_DMAC_Start_Interrupt_Error_PE*k*

PE*k* の DMA エラー割り込みを開始します。

備考　本 API 関数は、ユーザが呼び出します。

[指定形式]

void	R_DMAC_Start_Interrupt_Error_PE*k* (void);

備考　*k* は CPU コア番号を意味します。

[引数]

なし

[戻り値]

なし

R_DMAC_Stop_Interrupt_Error_PE*k*

PE*k* の DMA エラー割り込みを停止します。

備考　本 API 関数は、ユーザが呼び出します。

[指定形式]

void	R_DMAC_Stop_Interrupt_Error_PE*k* (void);

備考　*k* は CPU コア番号を意味します。

[引数]

なし

[戻り値]

なし

r_sdmac_address_error_interrupt_pe*k*

DMA アドレスエラー割り込みに伴う処理を行います。

[指定形式]

void	r_sdmac_address_error_interrupt_pe*k* (void);

備考　*k* は CPU コア番号を意味します。

[引数]

なし

[戻り値]

なし

R_DMA_Start_Transfer_Error_Interrupt

DMAC および DTS の全てのチャネルの DMA 転送エラー割り込みを開始します。
備考　　本 API 関数は、ユーザが呼び出します。

[指定形式]

```
void    R_DMA_Start_Transfer_Error_Interrupt(void);
```

[引数]

なし

[戻り値]

なし

R_DMA_Stop_Transfer_Error_Interrupt

DMAC および DTS の全てのチャネルの DMA 転送エラー割り込みを停止します。
備考　　本 API 関数は、ユーザが呼び出します。

[指定形式]

```
void    R_DMA_Stop_Transfer_Error_Interrupt(void);
```

[引数]

なし

[戻り値]

なし

r_dmac_error_interrupt_pe*k*

PE*k* の DMA アドレスエラー割り込みに伴う処理を行います。

[指定形式]

void	r_dmac_error_interrupt_pe*k*(void);

備考　　*k* は CPU コア番号を意味します。(RH850/C1M は PE1 のみサポート)

[引数]

なし

[戻り値]

なし

R_DTS_Create

DTS を初期化します。

備考　　本 API 関数は、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

```
void    R_DTS_Create(void)
```

[引数]

なし

[戻り値]

なし

R_DTS_Suspend

DTS 転送を一時停止します。

備考　　本 API 関数は、ユーザが呼び出します。

[指定形式]

```
void    R_DTS_Suspend ( void );
```

[引数]

なし

[戻り値]

なし

R_DTS_Resume

DTS 転送を再開します。

備考　　本 API 関数は、ユーザが呼び出します。

[指定形式]

```
void    R_DTS_Resume ( void );
```

[引数]

なし

[戻り値]

なし

R_DTS*g*_Start_Transfer_End_Interrupt

DTS 転送終了割り込みを開始します。

備考　　本 API 関数は、ユーザが呼び出します。

[指定形式]

void	R_DTS*g*_Start_Transfer_End_Interrupt (void);

備考　　*g* は”31_0”, “63_32”, “95_64”, “127_96”の割り込みグループ番号です。

[引数]

なし

[戻り値]

なし

R_DTS*g*_Stop_Transfer_End_Interrupt

DTS 転送終了割り込みを停止します。

備考　　本 API 関数は、ユーザが呼び出します。

[指定形式]

void	R_DTS*g*_Stop_Transfer_End_Interrupt (void);

備考　　*g* は”31_0”, “63_32”, “95_64”, “127_96”の割り込みグループ番号です。

[引数]

なし

[戻り値]

なし

R_DTS*g*_Start_Transfer_Count_Match_Interrupt

DTS 転送回数一致割り込みを開始します。

備考　　本 API 関数は、ユーザが呼び出します。

[指定形式]

void	R_DTS*g*_Start_Transfer_Count_Match_Interrupt (void);

備考　　*g* は”31_0”, “63_32”, “95_64”, “127_96”の割り込みグループ番号です。

[引数]

なし

[戻り値]

なし

R_DTS*g*_Stop_Transfer_Count_Match_Interrupt

DTS 転送回数一致割り込みを停止します。

備考　　本 API 関数は、ユーザが呼び出します。

[指定形式]

void	R_DTS*g*_Stop_Transfer_Count_Match_Interrupt (void);

備考　　*g* は”31_0”, “63_32”, “95_64”, “127_96”の割り込みグループ番号です。

[引数]

なし

[戻り値]

なし

R_DTS_Start_PE*k*_Transfer_Error_Interrupt

DTS 転送エラー割り込みを開始します。

備考　　本 API 関数は、ユーザが呼び出します。

[指定形式]

void	R_DTS_Start_PE*k*_Transfer_Error_Interrupt (void);

備考　　*k* は CPU コア番号を意味します。

[引数]

なし

[戻り値]

なし

R_DTS_Stop_PE*k*_Transfer_Error_Interrupt

DTS 転送エラー割り込みを停止します。

備考　本 API 関数は、ユーザが呼び出します。

[指定形式]

void	R_DTS_Stop_PE*k*_Transfer_Error_Interrupt (void);

備考　*k* は CPU コア番号を意味します。

[引数]

なし

[戻り値]

なし

r_dts*g*_transfer_end_interrupt

DTS チャネル転送終了割り込みに伴う処理を行います。

[指定形式]

void	r_dts*g*_transfer_end_interrupt (void);

備考 *g* は”31_0”, “63_32”, “95_64”, “127_96”の割り込みグループ番号です。

[引数]

なし

[戻り値]

なし

r_dts*g*_transfer_count_match_interrupt

DTS チャネル転送カウント一致割り込みに伴う処理を行います。

[指定形式]

void	r_dts*g*_transfer_count_match_interrupt (void);

備考 *g* は”31_0”, “63_32”, “95_64”, “127_96”の割り込みグループ番号です。

[引数]

なし

[戻り値]

なし

r_dts_transfer_error_interrupt_pe*k*

PE*k* の DTS 転送エラー割り込みに伴う処理を行います。

[指定形式]

void	r_dts_transfer_error_interrupt_pe*k* (void);

備考　　*k* は CPU コア番号を意味します。

[引数]

なし

[戻り値]

なし

R_GTM_Start_Interrupt_Error

GTM エラー割り込みを有効にします。

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

R_<Config_TIMn>_Start (TIM Bit Compression Mode)

R_<Config_TIMn_m>_Start (TIM Gated Periodic Sampling Mode)

R_<Config_TIMn_m>_Start (TIM Input Prescaler Mode)

R_<Config_TIMn_m>_Start (TIM input event mode)

R_<Config_TIMn_m>_Start(TIM pulse integration mode)

R_<Config_TIMn_m>_Start (TIM pwm measurement mode)

R_<Config_TIMn_m>_Start (TIM serial shift mode)

[指定形式]

```
void    R_GTM_Start_Interrupt_Error ( void );
```

[引数]

なし

[戻り値]

なし

R_GTM_Stop_Interrupt_Error

GTM エラー割り込みを無効にします。

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

R_<Config_TIMn>_Stop (TIM Bit Compression Mode)
R_<Config_TIMn_m>_Stop (TIM Gated Periodic Sampling Mode)
R_<Config_TIMn_m>_Stop (TIM Input Prescaler Mode)
R_<Config_TIMn_m>_Stop (TIM input event mode)
R_<Config_TIMn_m>_Stop (TIM pulse integration mode)
R_<Config_TIMn_m>_Stop (TIM pwm measurement mode)
R_<Config_TIMn_m>_Stop (TIM serial shift mode)

[指定形式]

```
void    R_GTM_Stop_Interrupt_Error ( void );
```

[引数]

なし

[戻り値]

なし

r_gtm_error_interrupt

GTM エラー割り込みに伴う処理を行います。

[指定形式]

```
void        r_gtm_error_interrupt ( void );
```

[引数]

なし

[戻り値]

なし

R_ATOM*n*_Start_Interrupt_IRQ*k*

INTGTMA0ATOM*nk* 割り込みを開始します。

備考　本 API 関数は、から呼び出されます。
R_<Config_ATOMn_m>_Start (ATOM Signal Output Mode Compare)
R_<Config_ATOMn_m>_Start (ATOM Signal Output Mode PWM)
R_<Config_ATOMn_m>_Start (ATOM Signal Output Mode Serial) function.

[指定形式]

void	R_ATOM*n*_Start_Interrupt_IRQ*k* (void);

備考　*n* はユニット番号を、*k* は割り込みチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_ATOM*n*_Stop_Interrupt_IRQ*k*

INTGTMA0ATOM*nk* 割り込みを停止します。

備考　本 API 関数は、下記関数から呼び出されます。
R_<Config_ATOMn_m>_Stop (ATOM Signal Output Mode Compare)
R_<Config_ATOMn_m>_Stop (ATOM Signal Output Mode PWM)
R_<Config_ATOMn_m>_Stop (ATOM Signal Output Mode Serial) function.

[指定形式]

void	R_ATOM*n*_Stop_Interrupt_IRQ*k* (void);

備考　*n* はユニット番号を、*k* は割り込みチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_atom*n*_irq*k*_interrupt

INTGTMA0ATOM*nk* 割り込みに伴う処理を行います。

[指定形式]

void	r_atom*n*_irq*k*_interrupt (void);
備考	*n* はユニット番号を、*k* は割り込みチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_MSPI_Master_Create

全ての MSPI チャネルを初期化します。

備考　　本 API 関数は、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

```
void    R_MSPI_Master_Create(void);
```

[引数]

なし

[戻り値]

なし

R_MSPI_Start_Interrupt_MSPI*n*TX

MSPI*n* 送信割り込みを有効にします。

備考　　本 API 関数は、ユーザが呼び出します。

[指定形式]

void	R_MSPI_Start_Interrupt_MSPI*n*TX (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_MSPI_Stop_Interrupt_MSPI*n*TX

MSPI*n* 送信割り込みを無効にします。

備考　　本 API 関数は、ユーザが呼び出します。

[指定形式]

void　　R_MSPI_Stop_Interrupt_MSPI*n*TX (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_MSPI_Start_Interrupt_MSPI*n*RX

MSPI*n* 受信割り込みを有効にします。

備考　　本 API 関数は、ユーザが呼び出します。

[指定形式]

void	R_MSPI_Start_Interrupt_MSPI*n*RX (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_MSPI_Stop_Interrupt_MSPI*n*RX

MSPI*n* 受信割り込みを無効にします。

備考　　本 API 関数は、ユーザが呼び出します。

[指定形式]

```
void    R_MSPI_Stop_Interrupt_MSPInRX ( 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_MSPI_Start_Interrupt_MSPI*n*FE

MSPI*n* フレーム終了割り込みを有効にします。

備考　　本 API 関数は、ユーザが呼び出します。

[指定形式]

void	R_MSPI_Start_Interrupt_MSPI*n*FE (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_MSPI_Stop_Interrupt_MSPI*n*FE

MSPI*n* フレーム終了割り込みを無効にします。

備考　　本 API 関数は、ユーザが呼び出します。

[指定形式]

void　　R_MSPI_Stop_Interrupt_MSPI*n*FE (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_MSPI_Start_Interrupt_MSPI*n*ERR

MSPI*n* エラー割り込みを有効にします。

備考　　本 API 関数は、ユーザが呼び出します。

[指定形式]

void	R_MSPI_Start_Interrupt_MSPI*n*ERR (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_MSPI_Stop_Interrupt_MSPI*n*ERR

MSPI*n* エラー割り込みを無効にします。

備考　　本 API 関数は、ユーザが呼び出します。

[指定形式]

void	R_MSPI_Stop_Interrupt_MSPI*n*ERR (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_mspi*n*_interrupt_send

MSPI*n* 送信割り込みに伴う処理を行います。

[指定形式]

void	r_mspi*n*_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_mspi*n*_interrupt_receive

MSPI*n* 受信割り込みに伴う処理を行います。

[指定形式]

void	r_mspi*n*_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_mspi*n*_interrupt_frameend

MSPI*n* フレーム終了割り込みに伴う処理を行います。

[指定形式]

void	r_mspi*n*_interrupt_frameend (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_mspi*n*_interrupt_error

MSPI*n* エラー割り込みに伴う処理を行います。

[指定形式]

void	r_mspi*n*_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

3.2.2 A/D コンバータ

以下に、コード生成ツールが A/D コンバータ用に出力する API 関数の一覧を示します。

表 3-2 グループスキャンモード ADCA 用 API 関数

API 関数名	機能概要
R_<Config_ADCXn>_Create	グループスキャンモード A/D 変換を制御するうえで必要となる初期化処理を行います。
R_<Config_ADCXn>_Halt	ADCA を停止します。
R_<Config_ADCXn>_ SelfDiagnostic_VoltageCircuitOn	自己診断電圧回路を ON、または基準電圧の更新を行います。
R_<Config_ADCXn>_ SelfDiagnostic_VoltageCircuitOff	自己診断電圧回路を OFF にします。
R_<Config_ADCXn>_ScanGroupx_Start	スキャングループの A/D 変換を開始します。
R_<Config_ADCXn>_ScanGroupx_OperationOn	スキャングループのスキャンを開始します。
R_<Config_ADCXn>_ScanGroupx_OperationOff	スキャングループのスキャンを停止します。
R_<Config_ADCXn>_ScanGroupx_GetResult	スキャングループの A/D 変換結果を取得します。
R_<Config_ADCXn>_ScanGroupx_GetPWMDiagResult	スキャングループの A/D 変換結果を取得します。(PWM-Diag)
R_<Config_ADCXn>_ScanGroupx_GetFloatingPointResult	スキャングループの A/D 変換結果を浮動小数点形式で取得します。
R_<Config_ADCXn>_TH_Groupk_Start	T&H グループのホールドを開始します。
R_<Config_ADCXn>_TH_Sampling_Start	T&H のサンプリングを開始します。
R_<Config_ADCXn>_TH_Stop	T&H を停止します。
R_<Config_ADCXn>_VoltageDivider_Start	分圧器の電圧モニタリングを有効にします。
R_<Config_ADCXn>_VoltageDivider_Stop	分圧器の電圧モニタリングを無効にします。
R_<Config_ADCXn>_StrongPullDown_Start	物理チャネルの Strong Pull-Down を有効にします。
R_<Config_ADCXn>_StrongPullDown_Stop	物理チャネルの Strong Pull-Down を無効にします。
R_<Config_ADCXn>_SGDiag_Start	SG-Diag.の A/D 変換を有効にします。
R_<Config_ADCXn>_SGDiag_OperationOn	SG-Diag.の A/D 変換を開始します。
R_<Config_ADCXn>_SGDiag_OperationOff	SG-Diag.の A/D 変換を停止します。
R_<Config_ADCXn>_SGDiag_GetResult	SG-Diag.の A/D 変換結果を取得します。
R_<Config_ADCXn>_SGDiag_GetSubtractionResult	SG-Diag. subtraction の結果を取得します。
R_<Config_ADCXn>_ASF_Start	ASF 累積カウンタを開始します。
R_<Config_ADCXn>_ASF_Stop	ASF 累積カウンタを停止します。
R_<Config_ADCXn>_ASF_GetResult	ASF.の累積結果を取得します。
R_<Config_ADCXn>_ASF_GetRealTimeResult	ASF.の中間累積結果を取得します。
R_<Config_ADCXn>_SetMultiplexerCommand	マルチプレクサコマンドを設定します。
R_<Config_ADCXn>_ScanGroupx_TimerStart	スキャングループのタイマを開始します。
R_<Config_ADCXn>_ScanGroupx_TimerStop	スキャングループのタイマを停止します。
R_<Config_ADCXn>_Create_UserInit	グループスキャンモード ADCA に関するユーザ独自の初期化処理を行います。

API 関数名	機能概要
r_<Config_ADCXn>_error_interrupt	AD エラー割り込みの発生に伴う処理を行います。
r_<Config_ADCXn>_scan_groupx_end_interrupt	スキャングループのスキャン終了割り込みの発生に伴う処理を行います。
r_<Config_ADCXn>_monitor_error_interrupt	モニターエラー割り込みの発生に伴う処理を行います。
r_<Config_ADCXn>_sg_diag_end_interrupt	SG-Diag 終了割り込みの発生に伴う処理を行います。
r_<Config_ADCXn>_mpx_request_interrupt	MPX DMA トリガ要求割り込みの発生に伴う処理を行います。
r_<Config_ADCXn>_asf_channelk_end_interrupt	ASF の加算終了割り込みの発生に伴う処理を行います。

R_<Config_ADCXn>_Create

グループスキャンモード A/D 変換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、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

void	R_<Config_ADCXn>_Create (void);

備考　ADC*X* は A/D コンバータの名前(ADCA, ADCC, ADCJ, ADCK)を、*n* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ADCXn>_Halt

ADCA を停止します。

[指定形式]

void	R_<Config_ADCXn>_Halt (void);

備考 ADC*X* は A/D コンバータの名前(ADCA, ADCC, ADCJ, ADCK)を、*n* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ADCXn>_ SelfDiagnostic_VoltageCircuitOn

自己診断電圧回路を ON、または基準電圧の更新を行います。

[指定形式]

void	R_<Config_ADCXn>_ SelfDiagnostic_VoltageCircuitOn(void);

備考 ADC*X* は A/D コンバータの名前(ADCA, ADCC, ADCJ, ADCK)を、*n* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ADCXn>_ SelfDiagnostic_VoltageCircuitOff

自己診断電圧回路を OFF にします。

[指定形式]

void	R_<Config_ADCXn>_ SelfDiagnostic_VoltageCircuitOff(void);

備考 ADC*X* は A/D コンバータの名前(ADCA, ADCC, ADCJ, ADCK)を、*n* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ADC*X*n>_ScanGroup*x*_Start

スキャングループの AD 変換を開始します。

[指定形式]

void	R_<Config_ADC*X*n>_ScanGroup*x*_Start (void);

備考 ADC*X* は A/D コンバータの名前(ADCA, ADCC, ADCJ, ADCK)を、*n* はユニット番号を、*x* はスキャングループ番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ADC*X*n>_ScanGroup*x*_OperationOn

SW トリガによってスキャングループのスキャンを開始します。

[指定形式]

void	R_<Config_ADC*X*n>_ScanGroup*x*_OperationOn (void);

備考 ADC*X* は A/D コンバータの名前(ADCA, ADCC, ADCJ, ADCK)を、*n* はユニット番号を、*x* はスキャングループ番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ADC*Xn*>_ScanGroup*x*_OperationOff

SW トリガによってスキャングループのスキャンを停止します。

[指定形式]

void	R_<Config_ADC*Xn*>_ScanGroup*x*_OperationOff(void);

備考 ADC*X* は A/D コンバータの名前(ADCA, ADCC, ADCJ, ADCK)を *n* はユニット番号を、*x* はスキャングループ番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ADC*X*n>_ScanGroup*x*_GetResult

スキャングループの AD 変換結果を取得します。

[指定形式]

void	R_<Config_ADC*X*n>_ScanGroup*x*_GetResult(uint16_t * const *buffer*, uint8_t buffer_size);

備考 ADC*X* は A/D コンバータの名前(ADCA, ADCC, ADCJ, ADCK)を、*n* はユニット番号を、*x* はスキャングループ番号を意味します。

[引数]

I/O	引数	説明
O	uint16_t * const *buffer*;	A/D 変換結果を格納する領域へのポインタ
I	uint8_t *buffer_size*	バッファサイズ

[戻り値]

戻り値	説明
MD_OK	通常終了
MD_ARGERROR	変換データ数が *buffer_size* をオーバー

R_<Config_ADC*Xn*>_ScanGroup*x*_GetPWMDiagResult

PWM-Diag のスキャングループの A/D 変換結果を取得します。

[指定形式]

void	R_<Config_ADC*Xn*>_ScanGroup*x*_GetPWMDiagResult (uint32_t * const buffer);

備考 ADC*X* は A/D コンバータの名前(ADCA, ADCC, ADCJ, ADCK)を、*n* はユニット番号を、*x* はスキャングループ番号を意味します。

[引数]

I/O	引数	説明
O	uint32_t * const *buffer*	A/D 変換結果を格納する領域へのポインタ

[戻り値]

戻り値	説明
MD_OK	通常終了

R_<Config_ADC*Xn*>_ScanGroup*x*_GetFloatingPointResult

スキャングループの A/D 変換結果を浮動小数点形式で取得します。

[指定形式]

void	R_<Config_ADC*Xn*>_ScanGroup*x*_GetFloatingPointResult(uint32_t * const buffer);

備考　ADC*X* は A/D コンバータの名前(ADCA, ADCC, ADCJ, ADCK)を、*n* はユニット番号を、*x* はスキャングループ番号を意味します。

[引数]

I/O	引数	説明
O	uint32_t * const *buffer*	A/D 変換結果を格納する領域へのポインタ

[戻り値]

戻り値	説明
MD_OK	通常終了
MD_ARGERROR	変換データ数が *buffer_size* をオーバー

R_<Config_ADC*X*n>_SetMultiplexerCommand

マルチプレクサコマンドを設定します。

[指定形式]

void	R_<Config_ADC*X*n>_SetMultiplexerCommand (uint8_t cmdData);

備考　ADC*X* は A/D コンバータの名前(ADCA, ADCC, ADCJ, ADCK)を、*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 * *cmdData*	設定するマルチプレクサコマンド

[戻り値]

なし

R_<Config_ADC*X*n>_ScanGroup*x*_TimerStart

スキャングループのタイマを開始します。

[指定形式]

void	R_<Config_ADC*X*n>_ScanGroup*x*_TimerStart (void);

備考 ADC*X* は A/D コンバータの名前(ADCA, ADCC, ADCJ, ADCK)を、*n* はユニット番号を、*x* はスキャングループ番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ADC*X*n>_ScanGroup*x*_TimerStop

スキャングループのタイマを停止します。

[指定形式]

void	R_<Config_ADC*X*n>_ScanGroup*x*_TimerStop (void);

備考 ADC*X* は A/D コンバータの名前(ADCA, ADCC, ADCJ, ADCK)を、*n* はユニット番号を、*x* はスキャングループ番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ADC*X*n>_TH_Group*k*_Start

T&H グループのホールドを開始します。

[指定形式]

void	R_<Config_ADC*X*n>_TH_Group*k*_Start (void);

備考 ADC*X* は A/D コンバータの名前(ADCA, ADCC, ADCJ, ADCK)を、*n* はユニット番号を、*k* は T&H グループ番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ADCXn>_TH_Sampling_Start

T&H のサンプリングを開始します。

[指定形式]

void	R_<Config_ADCXn>_TH_Sampling_Start (void);

備考 ADC*X* は A/D コンバータの名前(ADCA, ADCC, ADCJ, ADCK)を、*n* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ADC*Xn*>_TH_Stop

全ての T&H の動作を停止します。

[指定形式]

void	R_<Config_ADC*Xn*>_TH_Stop(void);

備考 ADC*X* は A/D コンバータの名前(ADCA, ADCC, ADCJ, ADCK)を、*n* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ADC*Xn*>_VoltageDivider_Start

分圧器の電圧モニタリングを有効にします。

[指定形式]

void	R_<Config_ADC*Xn*>_VoltageDivider_Start(void);

備考 ADC*X* は A/D コンバータの名前(ADCA, ADCC, ADCJ, ADCK)を、*n* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ADC*Xn*>_VoltageDivider_Stop

分圧器の電圧モニタリングを無効にします。

[指定形式]

void	R_<Config_ADC*Xn*>_VoltageDivider_Stop(void);

備考 ADC*X* は A/D コンバータの名前(ADCA, ADCC, ADCJ, ADCK)を、*n* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ADC*Xn*>_StrongPullDown_Start

物理チャネルの Strong Pull-Down を有効にします。

[指定形式]

void	R_<Config_ADC*Xn*>_StrongPullDown_Start(void);

備考 ADC*X* は A/D コンバータの名前(ADCA, ADCC, ADCJ, ADCK)を、*n* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ADC*Xn*>_StrongPullDown_Stop

物理チャネルの Strong Pull-Down を無効にします。

[指定形式]

void	R_<Config_ADC*Xn*>_StrongPullDown_Stop(void);

備考　ADC*X* は A/D コンバータの名前(ADCA, ADCC, ADCJ, ADCK)を、*n* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ADC*Xn*>_SGDiag_Start

SG-Diag.の A/D 変換を有効にします。

[指定形式]

void	R_<Config_ADC*Xn*>_SGDiag_Start(void);

備考　ADC*X* は A/D コンバータの名前(ADCA, ADCC, ADCJ, ADCK)を、*n* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ADC*Xn*>_SGDiag_OperationOn

SG-Diag.の A/D 変換を開始します。

[指定形式]

void	R_<Config_ADC*Xn*>_SGDiag_OperationOn(void);

備考 ADC*X* は A/D コンバータの名前(ADCA, ADCC, ADCJ, ADCK)を、*n* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ADC*Xn*>_SGDiag_OperationOff

SG-Diag.の A/D 変換を停止します。

[指定形式]

void	R_<Config_ADC*Xn*>_SGDiag_OperationOff(void);

備考 ADC*X* は A/D コンバータの名前(ADCA, ADCC, ADCJ, ADCK)を、*n* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ADC*Xn*>_SGDiag_GetResult

SG-Diag.の A/D 変換結果を取得します。

[指定形式]

void	R_<Config_ADC*Xn*>_SGDiag_GetResult(uint16_t * const *buffer*);

備考　ADC*X* は A/D コンバータの名前(ADCA, ADCC, ADCJ, ADCK)を、*n* はユニット番号を意味します。

[引数]

I/O	引数	説明
O	uint16_t * const *buffer*;	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_<Config_ADC*Xn*>_SGDiag_GetSubtractionResult

SG-Diag. subtraction の結果を取得します。

[指定形式]

void	R_<Config_ADC*Xn*>_SGDiag_GetSubtractionResult(uint32_t * const buffer);

備考　ADC*X* は A/D コンバータの名前(ADCA, ADCC, ADCJ, ADCK)を、*n* はユニット番号を意味します。

[引数]

I/O	引数	説明
O	uint32_t * const *buffer*;	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_<Config_ADC*Xn*>_ASF_Start

ASF 累積カウンタを開始します。

[指定形式]

void	R_<Config_ADC*Xn*>_ASF_Start(void);

備考　ADC*X* は A/D コンバータの名前(ADCA, ADCC, ADCJ, ADCK)を、*n* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ADC*Xn*>_ASF_Stop

ASF 累積カウンタを停止します。

[指定形式]

void	R_<Config_ADC*Xn*>_ASF_Stop(void);

備考　ADC*X* は A/D コンバータの名前(ADCA, ADCC, ADCJ, ADCK)を、*n* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ADC*Xn*>_ASF_GetResult

ASF.の累積結果を取得します。

[指定形式]

MD_STATUS R_<Config_ADC*Xn*>_ASF_GetResult(uint8_t *channel*, uint32_t* const *data*);

備考 ADC*X* は A/D コンバータの名前(ADCA, ADCC, ADCJ, ADCK)を、*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 *channel*	累積チャネル i (i=0～15)
O	uint32_t * const *data*	累積データを格納する領域へのポインタ

[戻り値]

戻り値	説明
MD_OK	通常終了
MD_ARGERROR	*channel* が累積チャネルの範囲（0～15）をオーバー

R_<Config_ADC*Xn*>_ASF_GetRealTimeResult

ASF.の中間累積結果を取得します。

[指定形式]

MD_STATUS	R_<Config_ADC*Xn*>_ASF_GetRealTimeResult(uint8_t *channel*, uint32_t* const *data*, uint16_t* const *counter*);

備考　ADC*X* は A/D コンバータの名前(ADCA, ADCC, ADCJ, ADCK)を、*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 channel	累積チャネル i (i=0～15)
O	uint32_t * const data	累積データを格納する領域へのポインタ
O	uint16_t* const *counter*	累積カウンタへのポインタ

[戻り値]

戻り値	説明
MD_OK	通常終了
MD_ARGERROR	*channel* が累積チャネルの範囲（0～15）をオーバー

R_<Config_ADCXn>_Create_UserInit

グループスキャンモード ADCA 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_<Config_ADCXn>_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_<Config_ADCXn>_Create_UserInit (void);

備考　ADC*X* は A/D コンバータの名前(ADCA, ADCC, ADCJ, ADCK)を、*n* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_ADCXn>_error_interrupt

AD エラー割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_ADC*X*n>_error_interrupt (void);

備考 ADC*X* は A/D コンバータの名前(ADCA, ADCC, ADCJ, ADCK)を、*n* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_ADC*X*n>_scan_group*x*_end_interrupt

スキャングループのスキャン終了割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_ADC*X*n>_scan_group*x*_end_interrupt (void);

備考 ADC*X* は A/D コンバータの名前(ADCA, ADCC, ADCJ, ADCK)を、*n* はユニット番号を、*x* はスキャングループ番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_ADCXn>_monitor_error_interrupt

モニターエラー割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_ADCXn>_monitor_error_interrupt (void);

備考 ADC*X* は A/D コンバータの名前(ADCA, ADCC, ADCJ, ADCK)を、*n* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_ADC*Xn*>_sg_diag_end_interrupt

SG-Diag 終了割り込みの発生に伴う処理を行います。

[指定形式]

void	r_<Config_ADC*Xn*>_sg_diag_end_interrupt(void);

備考　ADC*X* は A/D コンバータの名前(ADCA, ADCC, ADCJ, ADCK)を、*n* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_ADC*Xn*>_mpx_request_interrupt

MPX DMA トリガ要求割り込みの発生に伴う処理を行います。

[指定形式]

void	r_<Config_ADC*Xn*>_mpx_request_interrupt(void);

備考 ADC*X* は A/D コンバータの名前(ADCA, ADCC, ADCJ, ADCK)を、*n* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_ADC*Xn*>_asf_channel*k*_end_interrupt

累積チャネルの終了割り込みの発生に伴う処理を行います。

[指定形式]

void	r_<Config_ADC*Xn*>_asf_channel*k*_end_interrupt(void);

備考 ADC*X* は A/D コンバータの名前(ADCA, ADCC, ADCJ, ADCK)を、*n* はユニット番号、*k* は累積チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

3.2.3 CSI スレーブ

以下に、コード生成ツールが CSI スレーブ用に出力する API 関数の一覧を示します。

表 3-3 CSI スレーブ用 API 関数

API 関数名	機能概要
R_<Config_CSIXn>_Create	CSI スレーブを制御するうえで必要となる初期化処理を行います。
R_<Config_CSIXn>_Start	CSI の動作を許可します。
R_<Config_CSIXn>_Stop	CSI の動作を禁止します。
R_<Config_CSIXn>_Send	スレーブモードでデータ送信を行います。
R_<Config_CSIXn>_Receive	スレーブモードでデータ受信を行います。
R_<Config_CSIXn>_Send_Receive	スレーブモードでデータ送受信を行います。
R_<Config_CSIXn>_Create_UserInit	CSI スレーブに関するユーザ独自の初期化処理を行います。
r_<Config_CSIXn>_interrupt_send	通信ステータス割り込みの発生に伴う処理を行います。
r_<Config_CSIXn>_interrupt_receive	受信ステータス割り込みの発生に伴う処理を行います。
r_<Config_CSIXn>_interrupt_error	通信エラー割り込みの発生に伴う処理を行います。
r_<Config_CSIXn>_callback_sendend	通信ステータス割り込みの発生に伴う処理を行います。
r_<Config_CSIXn>_callback_receiveend	受信ステータス割り込みの発生に伴う処理を行います。
r_<Config_CSIXn>_callback_error	通信エラー割り込みの発生に伴う処理を行います。

R_<Config_CSIXn>_Create

CSI スレーブ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、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

void	R_<Config_CSIXn>_Create (void);

備考　CSI*X* は CSI 名(CSIG, CSIH)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_<Config_CSIXn>_Start

CSI の動作を許可します。

[指定形式]

void	R_<Config_CSIXn>_Start (void);

備考　CSI*X* は CSI 名(CSIG, CSIH)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_<Config_CSIXn>_Stop

CSI の動作を禁止します。

[指定形式]

void	R_<Config_CSIXn>_Stop (void);

備考 CSI*X* は CSI 名(CSIG, CSIH)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_<Config_CSIXn>_Send

スレーブモードでデータ送信を行います。

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 2 バイト単位の送信を引数 *tx_num* で指定された回数だけ繰り返し行います。

備考 2. 送信を行う際には、本 API 関数の呼び出し以前に R_<Config_CSIXn>_Start を呼び出す必要があります。

[指定形式]

```
MD_STATUS      R_<Config_CSIXn>_Send ( uint16_t* const tx_buf, uint16_t tx_num );
```

備考 CSI*X* は CSI 名(CSIG, CSIH)を、*n* はチャネル番号を意味します。

[引数]

I/O	引数	説明
I	uint16_t* const *tx_buf*;	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 *tx_num*;	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 *tx_num* の指定が不正

R_<Config_CSIXn>_Receive

スレーブモードでデータ受信を行います。

備考 1. 本 API 関数では、 2 バイト単位の受信を引数 *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. 受信を行う際には、本 API 関数の呼び出し以前に R_<Config_CSIXn>_Start を呼び出す必要があります。

[指定形式]

MD_STATUS R_<Config_CSIXn>_Receive (uint16_t* const *rx_buf*, uint16_t *rx_num*);

備考 CSI*X* は CSI 名(CSIG, CSIH)を、*n* はチャネル番号を意味します。

[引数]

I/O	引数	説明
O	uint16_t* const *rx_buf*;	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 *rx_num*;	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 *rx_num* の指定が不正

R_<Config_CSIXn>_Send_Receive

スレーブモードでデータ受信を行います。

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 2 バイト単位の送信を引数 *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、 2 バイト単位の受信を引数 *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 3. 受信を行う際には、本 API 関数の呼び出し以前に R_<Config_CSIXn>_Start を呼び出す必要があります。

[指定形式]

MD_STATUS R_<Config_CSIXn>_Send_Receive(uint16_t* const *tx_buf*, uint16_t *rx_num*, uint16_t* const *rx_buf*);

備考 CSI*X* は CSI 名(CSIG, CSIH)を、*n* はチャネル番号を意味します。

[引数]

I/O	引数	説明
I	uint16_t* const *tx_buf*	送信するデータを格納したバッファへのポインタ
O	uint16_t *rx_num*	受信するデータの総数
I	uint16_t* const *rx_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]

マクロ	説明
MD_OK	正常終了
MD_ARGERROR	引数 *rx_num* の指定が不正

R_<Config_CSIXn>_Create_UserInit

CSI スレー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_<Config_CSIXn>_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_<Config_CSIXn>_Create_UserInit (void);

備考　CSI*X* は CSI 名(CSIG, CSIH)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_<Config_CSIXn>_interrupt_send

通信ステータス割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_CSIXn>_interrupt_send (void);

備考　CSI*X* は CSI 名(CSIG, CSIH)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_<Config_CSIXn>_interrupt_receive

受信ステータス割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_CSIXn>_interrupt_receive (void);

備考　CSI*X* は CSI 名(CSIG, CSIH)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_<Config_CSIXn>_interrupt_error

通信エラー割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_CSIXn>_interrupt_error (void);

備考　CSI*X* は CSI 名(CSIG, CSIH)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_<Config_CSIXn>_callback_sendend

通信ステータス割り込みの発生に伴う処理を行います。

備考　本 API 関数は、r_<Config_CSIXn>_interrupt_send 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_<Config_CSIXn>_callback_sendend (void);

備考　CSI*X* は CSI 名(CSIG, CSIH)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_<Config_CSIXn>_callback_receiveend

受信ステータス割り込みの発生に伴う処理を行います。

備考　本 API 関数は、r_<Config_CSIXn>_interrupt_receive 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_<Config_CSIXn>_callback_receiveend (void);

備考　CSI*X* は CSI 名(CSIG, CSIH)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_<Config_CSIXn>_callback_error

通信エラー割り込みの発生に伴う処理を行います。

備考　本 API 関数は、r_<Config_CSIXn>_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]

void	R_<Config_CSIXn>_callback_error (uint32_t *err_type*);

備考　CSI*X* は CSI 名(CSIG, CSIH)を、*n* はチャネル番号を意味します。

[引数]

I/O	引数	説明
I	uint32_t *err_type*;	通信エラー割り込みの発生要因 0000x0x1B : オーバラン・エラー 0000x01xB : パリティ・エラー 000010xxB : データ整合性チェック・エラー

[戻り値]

なし

使用例

CSI マスタの使用例を参照してください。

3.2.4 CSI マスタ

以下に、コード生成ツールが CSI マスタ用に出力する API 関数の一覧を示します。

表 3-4 CSI マスタ用 API 関数

API 関数名	機能概要
R_<Config_CSIXn>_Create	CSI マス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_<Config_CSIXn>_Start	CSI の動作を許可します。
R_<Config_CSIXn>_Stop	CSI の動作を禁止します。
R_<Config_CSIXn>_Send	マスタモードでデータ送信を行います。
R_<Config_CSIXn>_Receive	マスタモードでデータ受信を行います。
R_<Config_CSIXn>_Send_Receive	マスタモードでデータ送受信を行います。
R_<Config_CSIXn>_Create_UserInit	CSI マスタに関するユーザ独自の初期化処理を行います。
r_<Config_CSIXn>_interrupt_send	通信ステータス割り込みの発生に伴う処理を行います。
r_<Config_CSIXn>_interrupt_receive	受信ステータス割り込みの発生に伴う処理を行います。
r_<Config_CSIXn>_interrupt_error	通信エラー割り込みの発生に伴う処理を行います。
r_<Config_CSIXn>_callback_sendend	通信ステータス割り込みの発生に伴う処理を行います。
r_<Config_CSIXn>_callback_receiveend	受信ステータス割り込みの発生に伴う処理を行います。
r_<Config_CSIXn>_callback_error	通信エラー割り込みの発生に伴う処理を行います。

R_<Config_CSIXn>_Create

CSI マスタ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、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

void	R_<Config_CSIXn>_Create (void);

備考　CSI*X* は CSI 名(CSIG, CSIH)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_<Config_CSIXn>_Start

CSI の動作を許可します。

[指定形式]

void	R_<Config_CSIXn>_Start (void);

備考 CSI*X* は CSI 名(CSIG, CSIH)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_<Config_CSIXn>_Stop

CSI の動作を禁止します。

[指定形式]

void	R_<Config_CSIXn>_Stop (void);

備考 CSI*X* は CSI 名(CSIG, CSIH)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_<Config_CSIXn>_Send

マスタモードでデータ送信を行います。

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 2 バイト単位の送信を引数 *tx_num* で指定された回数だけ繰り返し行います。

備考 2. 送信を行う際には、本 API 関数の呼び出し以前に R_<Config_CSIXn>_Start を呼び出す必要があります。

備考 3. 使用リソースにより、引数が異なります。

[指定形式]

CSIG の場合

```
MD_STATUS      R_<Config_CSIGn_Send( uint16_t* const tx_buf, uint16_t tx_num );
```

CSIH の場合

```
MD_STATUS      R_<Config_CSIHn_Send( uint16_t* const tx_buf, uint16_t tx_num, uint32_t chip_id );
```

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	uint16_t* const *tx_buf*;	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 *tx_num*;	送信するデータの総数
I	uint32_t *chip_id*	スレーブチャネルの ID（"|"演算子使用で複数選択可） _CSIH_SELECT_CHIP_*n* ：チャネル n

[戻り値]

マクロ	説明
MD_OK	正常終了
MD_ARGERROR	引数 *tx_num* または *chip_id* の指定が不正

R_<Config_CSIXn>_Receive

マスタモードでデータ受信を行います。

備考 1. 本 API 関数では、 2 バイト単位の受信を引数 *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. 受信を行う際には、本 API 関数の呼び出し以前に R_<Config_CSIXn>_Start を呼び出す必要があります。

備考 3. 使用リソースにより、引数が異なります。

[指定形式]

CSIG の場合

```
MD_STATUS      R_<Config_CSIGn_Receive( uint16_t* const rx_buf, uint16_t rx_num );
```

CSIH の場合

```
MD_STATUS      R_<Config_CSIHn_Receive( uint16_t* const rx_buf, uint16_t rx_num, uint32_t chip_id );
```

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	uint16_t* const *rx_buf*;	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 *rx_num*;	受信するデータの総数
I	uint32_t *chip_id*	スレーブチャネルの ID（"|"演算子使用で複数選択可） _CSIH_SELECT_CHIP_n : チャネル n

[戻り値]

マクロ	説明
MD_OK	正常終了
MD_ARGERROR	引数 *rx_num* または *chip_id* の指定が不正

R_<Config_CSIXn>_Send_Receive

スレーブモードでデータ受信を行います。

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 2 バイト単位の送信を引数 *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、 2 バイト単位の受信を引数 *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 3. 受信を行う際には、本 API 関数の呼び出し以前に R_<Config_CSIXn>_Start を呼び出す必要があります。

[指定形式]

MD_STATUS R_<Config_CSIX*n*>_Send_Receive(uint16_t* const *tx_buf*, uint16_t *tx_num*, uint16_t* const *rx_buf*, uint32_t *chip_id)*;

備考 CSI*X* は CSI 名(CSIG, CSIH)を、*n* はチャネル番号を意味します。

[引数]

I/O	引数	説明
I	uint16_t* const *tx_buf*	送信するデータを格納したバッファへのポインタ
O	uint16_t *rx_num*	受信するデータの総数
I	uint16_t* const *rx_buf*	受信したデータを格納するバッファへのポインタ
I	uint32_t *chip_id*	スレーブチャネル _CSIH_SELECT_CHIP_*n*: チャネル n

[戻り値]

マクロ	説明
MD_OK	正常終了
MD_ARGERROR	引数 *rx_num* の指定が不正

R_<Config_CSIXn>_Create_UserInit

CSI マス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_<Config_CSIXn>_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_<Config_CSIXn>_Create_UserInit (void);

備考 CSI*X* は CSI 名(CSIG, CSIH)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_<Config_CSIXn>_interrupt_send

通信ステータス割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_CSIXn>_interrupt_send (void);

備考　CSI*X* は CSI 名(CSIG, CSIH)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_<Config_CSIXn>_interrupt_receive

受信ステータス割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_CSIXn>_interrupt_receive (void);

備考　CSI*X* は CSI 名(CSIG, CSIH)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_<Config_CSIXn>_interrupt_error

通信エラー割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_CSIXn>_interrupt_error (void);

備考　CSI*X* は CSI 名(CSIG, CSIH)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_<Config_CSIXn>_callback_sendend

通信ステータス割り込みの発生に伴う処理を行います。

備考　本 API 関数は、r_<Config_CSIXn>_interrupt_send 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_<Config_CSIXn>_callback_sendend ( void );
```

備考　CSI*X* は CSI 名(CSIG, CSIH)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_<Config_CSIXn>_callback_receiveend

受信ステータス割り込みの発生に伴う処理を行います。

備考　本 API 関数は、r_<Config_CSIXn>_interrupt_receive 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_<Config_CSIXn>_callback_receiveend ( void );
```

備考　CSI*X* は CSI 名(CSIG, CSIH)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_<Config_CSIXn>_callback_error

通信エラー割り込みの発生に伴う処理を行います。

備考　本 API 関数は、r_<Config_CSIXn>_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]

void	R_<Config_CSIXn>_callback_error (uint32_t *err_type*);

備考　CSI*X* は CSI 名(CSIG, CSIH)を、*n* はチャネル番号を意味します。

[引数]

I/O	引数	説明
I	uint32_t *err_type*;	通信エラー割り込みの発生要因 0000x0x1B : オーバラン・エラー 0000x01xB : パリティ・エラー 000010xxB : データ整合性チェック・エラー

[戻り値]

なし

使用例

CSIH1 をマスタ、CSIH0 をスレーブとしてデータを送受信する場合 (青字はユーザコード)

r_cg_main.c

```
/* Start user code for global. Do not edit comment generated here */
uint16_t tx_buf_h1[] = {0xA5A5, 0x5A5A};
uint16_t rx_buf1_h1[] = {0x8888, 0x8888};
uint16_t rx_buf2_h1[] = {0x9999, 0x9999};

uint16_t tx_buf_h0[] = {0xCACA, 0xBABA};
uint16_t rx_buf1_h0[] = {0x8888, 0x8888};
uint16_t rx_buf2_h0[] = {0x9999, 0x9999};
/* End user code. Do not edit comment generated here */

void main(void)
{
    r_main_userinit();
    /* Start user code for main. Do not edit comment generated here */
    R_Config_CSIH0_Start();
    R_Config_CSIH1_Start();

    R_Config_CSIH0_Send_Receive(tx_buf_h0, 2, rx_buf1_h0);
    R_Config_CSIH1_Send_Receive(tx_buf_h1, 2, rx_buf1_h1, _CSIH_SELECT_CHIP_1);
    while(transmitend_flag1 != 1);
    while(receiveend_flag1 != 1);
    while(transmitend_flag0 != 1);
    while(receiveend_flag0 != 1);
    transmitend_flag1 = 0;
    receiveend_flag1 = 0;
    transmitend_flag0 = 0;
    receiveend_flag0 = 0;

    R_Config_CSIH0_Send_Receive(tx_buf_h0, 2, rx_buf2_h0);
    R_Config_CSIH1_Send_Receive(tx_buf_h1, 2, rx_buf2_h1, _CSIH_SELECT_CHIP_1);
    while(transmitend_flag1 != 1);
    while(receiveend_flag1 != 1);
    while(transmitend_flag0 != 1);
    while(receiveend_flag0 != 1);
    transmitend_flag1 = 0;
    receiveend_flag1 = 0;
    transmitend_flag0 = 0;
    receiveend_flag0 = 0;
    while(1);
    /* End user code. Do not edit comment generated here */
}
```

Config_CSIH1_user.c

```
/* Start user code for global. Do not edit comment generated here */
volatile uint8_t transmitend_flag_0 = 0;
volatile uint8_t receiveend_flag_0 = 0;
/* End user code. Do not edit comment generated here */

void r_Config_CSIH1_callback_sendend(void)
{
    /* Start user code for r_Config_CSIH1_callback_sendend. Do not edit comment generated here */
    transmitend_flag1 = 1;
    /* End user code. Do not edit comment generated here */
```

```
}

void r_Config_CSIH1_callback_receiveend(void)
{
    /* Start user code for r_Config_CSIH1_callback_receiveend. Do not edit comment generated here */
    receiveend_flag1 = 1;
    /* End user code. Do not edit comment generated here */
}
```

Config_CSIH0_user.c

```
/* Start user code for global. Do not edit comment generated here */
volatile uint8_t transmitend_flag0 = 0;
volatile uint8_t receiveend_flag0 = 0;
/* End user code. Do not edit comment generated here */

void r_Config_CSIH0_callback_sendend(void)
{
    /* Start user code for r_Config_CSIH0_callback_sendend. Do not edit comment generated here */
    transmitend_flag0 = 1;
    /* End user code. Do not edit comment generated here */
}
void r_Config_CSIH0_callback_receiveend(void)
{
    /* Start user code for r_Config_CSIH0_callback_receiveend. Do not edit comment generated here */
    receiveend_flag0 = 1;
    /* End user code. Do not edit comment generated here */
}
```

3.2.5 割り込み

以下に、コード生成ツールが割り込み用に出力する API 関数の一覧を示します。

表 3-5 割り込み用 API 関数

API 関数名	機能概要
R_<Config_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_<Config_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_<Config_INTC>_INTPn_Start	INTP*n* 割り込みを許可します。
R_<Config_INTC>_INTPn_Stop	INTP*n* 割り込み要求をマスク状態にします。
R_<Config_INTC>_IRQn_Start	IRQn 割り込みを許可します。
R_<Config_INTC>_IRQn_Stop	IRQn 割り込み要求をマスク状態にします。
R_<Config_INTC>_NMI_Start	NMI 割り込みを許可します。
R_<Config_INTC>_NMI_Stop	NMI 割り込み要求をマスク状態にします。
R_<Config_INTC>_SINTn_Start	SINTn 割り込みを許可します。
R_<Config_INTC>_SINTn_Stop	SINTn 割り込み要求をマスク状態にします。
R_<Config_INTC>_SINTn_TriggerOn	SINTn 割り込み要求を発生します。
r_<Config_INTC>_intpn_interrupt	INTPn 割り込みの発生に伴うユーザ独自の処理を行います。
r_<Config_INTC>_intpn_interrupt_pek	PE k の INTPn 割り込みの発生に伴うユーザ独自の処理を行います。
r_<Config_INTC>_irqn_interrupt	INTQn 割り込みの発生に伴うユーザ独自の処理を行います。
r_<Config_INTC>_nmi_interrupt	NMI の発生に伴う処理を行います。
r_<Config_INTC>_sintn_interrupt_pek	PE k の SINTn 割り込みの発生に伴う処理を行います。

R_<Config_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。

備考　　本 API 関数は、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

```
void    R_<Config_INTC>_Create ( void );
```

[引数]

なし

[戻り値]

なし

R_<Config_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_<Config_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_<Config_INTC>_Create_UserInit ( void );
```

[引数]

なし

[戻り値]

なし

R_<Config_INTC>_INTP*n*_Start

INTP*n* 割り込みを許可します。

[指定形式]

void	R_<Config_INTC>_INTP*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_<Config_INTC>_INTP*n*_Stop

INTP*n* 割り込み要求をマスク状態にします。

[指定形式]

void	R_<Config_INTC>_INTP*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_<Config_INTC>_IRQ*n*_Start

IRQn 割り込みを許可します。

[指定形式]

void	R_<Config_INTC>_IRQ*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_<Config_INTC>_IRQ*n*_Stop

IRQn 割り込み要求をマスク状態にします。

[指定形式]

```
void    R_<Config_INTC>_IRQn_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_<Config_INTC>_NMI_Start

NMI 割り込みを許可します。

[指定形式]

```
void        R_<Config_INTC>_NMI_Start ( void );
```

[引数]

なし

[戻り値]

なし

R_<Config_INTC>_NMI_Stop

NMI 割り込み要求をマスク状態にします。

[指定形式]

```
void        R_<Config_INTC>_NMI_Stop ( void );
```

[引数]

なし

[戻り値]

なし

R_<Config_INTC>_SINT*n*_Start

SINT*n* 割り込みを許可します。

[指定形式]

void	R_<Config_INTC>_SINT*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_<Config_INTC>_SINT*n*_Stop

SINT*n* 割り込み要求をマスク状態にします。

[指定形式]

void	R_<Config_INTC>_SINT*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_<Config_INTC>_SINT*n*_TriggerOn

SINT*n* 割り込み要求を発生します。

[指定形式]

void	R_<Config_INTC>_SINT*n*_TriggerOn (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_<Config_INTC>_intp*n*_interrupt

INTP*n* 割り込みの発生に伴うユーザ独自の処理を行います。

[指定形式]

void	r_<Config_INTC>_intp*n*_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_<Config_INTC>_intp*n*_interrupt_pe*k*

PE *k* の INTP*n* 割り込みの発生に伴うユーザ独自の処理を行います。

[指定形式]

```
void    r_<Config_INTC>_intpn_interrupt_pek(void);
```

備考　　*n* は割り込み要因番号、*k* は CPU コア番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_INTC>_irq*n*_interrupt

INTQ*n* 割り込みの発生に伴うユーザ独自の処理を行います。

[指定形式]

void	r_<Config_INTC>_irq*n*_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_<Config_INTC>_nmi_interrupt

nmi 割り込みの発生に伴うユーザ独自の処理を行います。

[指定形式]

```
void        r_<Config_INTC>_nmi_interrupt ( void );
```

[引数]

なし

[戻り値]

なし

r_<Config_INTC>_sint*n*_interrupt_pe*k*

PE *k* の SINT*n* 割り込みの発生に伴うユーザ独自の処理を行います。

[指定形式]

void	r_<Config_INTC>_sint*n*_interrupt_pe*k*(void);

備考　*n* は割り込み要因番号、*k* は CPU コア番号を意味します。

[引数]

なし

[戻り値]

なし

3.2.6 入力インターバルタイマ

以下に、コード生成ツールが入力インターバルタイマ用に出力する API 関数の一覧を示します。

表 3-6 入力インターバルタイマ用 API 関数

API 関数名	機能概要
R_<Config_TAUXn_m>_Create	入力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。
R_<Config_TAUXn_m>_Start	タイマのカウントを開始します。
R_<Config_TAUXn_m>_Stop	タイマのカウントを停止します。
R_<Config_TAUXn_m>_Create_UserInit	入力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。
r_<Config_TAUXn_m>_interrupt	タイマ割り込みの発生に伴う処理を行います。
r_<Config_TAUXn_m>_interrupt_pek	タイマ割り込みの発生に伴う処理を行います。(CPU コア別)

R_<Config_TAU*Xn_m*>_Create

入力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。

備考　本 API 関数は、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

void	R_<Config_TAU*Xn_m*>_Create (void);

備考　TAU*X* は TAU 名(TAUB, TAUD, TAUJ)を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAU*Xn_m*>_Start

タイマのカウントを開始します。

[指定形式]

void	R_<Config_TAU*Xn_m*>_Start (void);

備考 TAU*X* は TAU 名(TAUB, TAUD, TAUJ)を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAU*Xn_m*>_Stop

タイマのカウントを終了します。

[指定形式]

void	R_<Config_TAU*Xn_m*>_Stop (void);

備考 TAU*X* は TAU 名(TAUB, TAUD, TAUJ)を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAU*Xn_m*>_Create_UserInit

入力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_<Config_TAUXn_m>_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_<Config_TAU*Xn_m*>_Create_UserInit (void);

備考　TAU*X* は TAU 名(TAUB, TAUD, TAUJ)を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TAU*Xn_m*>_interrupt

タイマ割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_TAU*Xn_m*>_interrupt (void);

備考 TAU*X* は TAU 名(TAUB, TAUD, TAUJ)を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TAU*Xn_m*>_interrupt_pe*k*

タイマ割り込みの発生に伴う処理を行います。(CPU コア別)

[指定形式]

void	R_<Config_TAU*Xn_m*>_interrupt_pe*k* (void);

備考 TAU*X* は TAU 名(TAUB, TAUD, TAUJ)を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

3.2.7 入力パルスインターバル測定

以下に、コード生成ツールが入力パルスインターバル測定用に出力する API 関数の一覧を示します。

表 3-7 入力パルスインターバル測定用 API 関数

API 関数名	機能概要
R_<Config_TAUXn_m>_Create	入力パルスインターバル測定を制御するうえで必要となる初期化処理を行います。
R_<Config_TAUXn_m>_Start	タイマのカウントを開始します。
R_<Config_TAUXn_m>_Stop	タイマのカウントを停止します。
R_<Config_TAUXn_m>_Get_PulseWidth	タイマのパルス幅を測定します。
R_<Config_TAUXn_m>_Create_UserInit	入力パルスインターバル測定に関するユーザ独自の初期化処理を行います。
R_<Config_TAUXn_m>_interrupt	タイマ割り込みの発生に伴う処理を行います。
r_<Config_TAUXn_m>_interrupt_pek	タイマ割り込みの発生に伴う処理を行います。(CPU コア別)

R_<Config_TAU*Xn_m*>_Create

入力パルスインターバル測定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、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

void	R_<Config_TAU*Xn_m*>_Create (void);

備考　TAU*X* は TAU 名(TAUB, TAUD, TAUJ)を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAU*Xn_m*>_Start

タイマのカウントを開始します。

[指定形式]

void	R_<Config_TAU*Xn_m*>_Start (void);

備考 TAU*X* は TAU 名(TAUB, TAUD, TAUJ)を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAU*Xn_m*>_Stop

タイマのカウントを終了します。

[指定形式]

void	R_<Config_TAU*Xn_m*>_Stop (void);

備考 TAU*X* は TAU 名(TAUB, TAUD, TAUJ)を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAU*Xn_m*>_Get_PulseWidth

タイマのパルス幅を測定します。

[指定形式]

void	R_<Config_TAU*Xn_m*>_Get_PulseWidth (uint32_t * const *width*);

備考　TAU*X* は TAU 名(TAUB, TAUD, TAUJ)を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

I/O	引数	説明
O	uint32_t * const *width*;	パルス幅を格納する領域へのポインタ

[戻り値]

なし

R_<Config_TAU*Xn_m*>_Create_UserInit

入力パルスインターバル測定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_<Config_TAUXn_m>_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_<Config_TAU*Xn_m*>_Create_UserInit (void);

備考　TAU*X* は TAU 名(TAUB, TAUD, TAUJ)を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAU*Xn_m*>_interrupt

タイマ割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_TAU*Xn_m*>_interrupt (void);

備考 TAU*X* は TAU 名(TAUB, TAUD, TAUJ)を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TAU*Xn_m*>_interrupt_pe*k*

タイマ割り込みの発生に伴う処理を行います。(CPU コア別)

[指定形式]

void	R_<Config_TAU*Xn_m*>_interrupt_pe*k* (void);

備考 TAU*X* は TAU 名(TAUB, TAUD, TAUJ)を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

3.2.8 インターバルタイマ

以下に、コード生成ツールがインターバルタイマ用に出力する API 関数の一覧を示します。

表 3-8 インターバルタイマ用 API 関数

API 関数名	機能概要
R_<Config_TAUXn_m>_Create	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。
R_<Config_TAUXn_m>_Start	タイマのカウントを開始します。
R_<Config_TAUXn_m>_Stop	タイマのカウントを停止します。
R_<Config_TAUXn_m>_Create_UserInit	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。
r_<Config_TAUXn_m>_interrupt	タイマ割り込みの発生に伴う処理を行います。
r_<Config_TAUXn_m>_interrupt_pek	タイマ割り込みの発生に伴う処理を行います。(CPU コア別)

R_<Config_TAU*Xn_m*>_Create

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。

備考　本 API 関数は、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

void	R_<Config_TAU*Xn_m*>_Create (void);

備考　TAU*X* は TAU 名(TAUB, TAUD, TAUJ)を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAU*Xn_m*>_Start

タイマのカウントを開始します。

[指定形式]

void	R_<Config_TAU*Xn_m*>_Start (void);

備考 TAU*X* は TAU 名(TAUB, TAUD, TAUJ)を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAU*Xn_m*>_Stop

タイマのカウントを終了します。

[指定形式]

void	R_<Config_TAU*Xn_m*>_Stop (void);

備考 TAU*X* は TAU 名(TAUB, TAUD, TAUJ)を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAU*Xn_m*>_Create_UserInit

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_<Config_TAUXn_m>_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_<Config_TAU*Xn_m*>_Create_UserInit (void);

備考　TAU*X* は TAU 名(TAUB, TAUD, TAUJ)を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TAU*Xn_m*>_interrupt

タイマ割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_TAU*Xn_m*>_interrupt (void);

備考 TAU*X* は TAU 名(TAUB, TAUD, TAUJ)を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TAU*Xn_m*>_interrupt_pe*k*

タイマ割り込みの発生に伴う処理を行います。(CPU コア別)

[指定形式]

void	R_<Config_TAU*Xn_m*>_interrupt_pe*k* (void);

備考 TAU*X* は TAU 名(TAUB, TAUD, TAUJ)を、*n* はユニット番号を、*m* はチャネル番号を、*k* は PE 番号を意味します。

[引数]

なし

[戻り値]

なし

3.2.9 三角波 PWM 出力

以下に、コード生成ツールが三角波 PWM 出力用に出力する API 関数の一覧を示します。

表 3-9 三角波 PWM 出力用 API 関数

API 関数名	機能概要
R_<Config_TAUXn>_Create	三角波 PWM 出力を制御するうえで必要となる初期化処理を行います。
R_<Config_TAUXn>_Start	タイマのカウントを開始します。
R_<Config_TAUXn>_Stop	タイマのカウントを停止します。
R_<Config_TAUXn>_Create_UserInit	三角波 PWM 出力に関するユーザ独自の初期化処理を行います。
r_<Config_TAUXn>_channelm_interrupt	タイマ割り込みの発生に伴う処理を行います。
r_<Config_TAUXn>_channelm_interrupt_pek	タイマ割り込みの発生に伴う処理を行います。(CPU コア別)

R_<Config_TAU*Xn*>_Create

三角波 PWM 出力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、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

void	R_<Config_TAU*Xn*>_Create (void);

備考　TAU*X* は TAU 名(TAUB, TAUD, TAUJ)を、*n* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAU*Xn*>_Start

タイマのカウントを開始します。

[指定形式]

void	R_<Config_TAU*Xn*>_Start (void);

備考　TAU*X* は TAU 名(TAUB, TAUD, TAUJ)を、*n* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAU*Xn*>_Stop

タイマのカウントを終了します。

[指定形式]

void	R_<Config_TAU*Xn*>_Stop (void);

備考　TAU*X* は TAU 名(TAUB, TAUD, TAUJ)を、*n* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAU*Xn*>_Create_UserInit

三角波 PWM 出力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_<Config_TAUXn>_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_<Config_TAU*Xn*>_Create_UserInit (void);

備考　TAU*X* は TAU 名(TAUB, TAUD, TAUJ)を、*n* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TAU*Xn*>_channel*m*_interrupt

タイマ割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_TAU*Xn*>_channel*m*_interrupt (void);

備考 TAU*X* は TAU 名(TAUB, TAUD, TAUJ)を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TAU*Xn*>_channel*m*_interrupt_pe*k*

タイマ割り込みの発生に伴う処理を行います。(CPU コア別)

[指定形式]

void	R_<Config_TAU*Xn*>_channel*m*_interrupt_pe*k* (void);

備考 TAU*X* は TAU 名(TAUB, TAUD, TAUJ)を、*n* はユニット番号を、*m* はチャネル番号を、*k* は PE 番号を意味します。

[引数]

なし

[戻り値]

なし

3.2.10 三角波 PWM 出力（デッドタイム付き）

以下に、コード生成ツールが三角波 PWM 出力（デッドタイム付き）用に出力する API 関数の一覧を示します。

表 3-10 三角波 PWM 出力用（デッドタイム付き） API 関数

API 関数名	機能概要
R_<Config_TAUXn>_Create	三角波 PWM 出力（デッドタイム付き）を制御するうえで必要となる初期化処理を行います。
R_<Config_TAUXn>_Start	タイマのカウントを開始します。
R_<Config_TAUXn>_Stop	タイマのカウントを停止します。
R_<Config_TAUXn>_Create_UserInit	三角波 PWM 出力（デッドタイム付き）に関するユーザ独自の初期化処理を行います。
r_<Config_TAUXn>_channelm_interrupt	タイマ割り込みの発生に伴う処理を行います。
r_<Config_TAUXn>_channelm_interrupt_pek	タイマ割り込みの発生に伴う処理を行います。(CPU コア別)

R_<Config_TAU*Xn*>_Create

三角波 PWM 出力（デッドタイム付き）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、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

void	R_<Config_TAU*Xn*>_Create (void);

備考　TAU*X* は TAU 名(TAUB, TAUD, TAUJ)を、*n* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAU*Xn*>_Start

タイマのカウントを開始します。

[指定形式]

void	R_<Config_TAU*Xn*>_Start (void);

備考　TAU*X* は TAU 名(TAUB, TAUD, TAUJ)を、*n* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAU*Xn*>_Stop

タイマのカウントを終了します。

[指定形式]

void	R_<Config_TAU*Xn*>_Stop (void);

備考　TAU*X* は TAU 名(TAUB, TAUD, TAUJ)を、*n* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAU*Xn*>_Create_UserInit

三角波 PWM 出力（デッドタイム付き）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_<Config_TAUXn>_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_<Config_TAU*Xn*>_Create_UserInit (void);

備考　TAU*X* は TAU 名(TAUB, TAUD, TAUJ)を、*n* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TAU*Xn*>_channel*m*_interrupt

タイマ割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_TAU*Xn*>_channel*m*_interrupt (void);

備考　TAU*X* は TAU 名(TAUB, TAUD, TAUJ)を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TAU*Xn*>_channel*m*_interrupt_pe*k*

タイマ割り込みの発生に伴う処理を行います。(CPU コア別)

[指定形式]

void	R_<Config_TAU*Xn*>_channel*m*_interrupt_pe*k* (void);

備考 TAU*X* は TAU 名(TAUB, TAUD, TAUJ)を、*n* はユニット番号を、*m* はチャネル番号を、*k* は PE 番号を意味します。

[引数]

なし

[戻り値]

なし

3.2.11 OS タイマ

以下に、コード生成ツールが OS タイマ用に出力する API 関数の一覧を示します。

表 3-11 OS タイマ用 API 関数

API 関数名	機能概要
R_<Config_OSTMn>_Create	OS 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。
R_<Config_OSTMn>_Start	OS タイマのカウントを開始します。
R_<Config_OSTMn>_Stop	OS タイマのカウントを終了します。
R_<Config_OSTMn>_Set_CompareValue	インターバルタイマモードの場合、ダウンカウンタの開始値を設定します。 フリーランニングコンペアモードの場合、コンペア値を設定します。
R_<Config_OSTMn>_Create_UserInit	OS タイマに関するユーザ独自の初期化処理を行います。
R_<Config_OSTMn>_interrupt	OSTM の割り込みの発生に伴う処理を行います。

R_<Config_OSTM*n*>_Create

OS 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。

備考　本 API 関数は、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

void	R_<Config_OSTM*n*>_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_<Config_OSTM*n*>_Start

OS タイマのカウントを開始します。

[指定形式]

void	R_<Config_OSTM*n*>_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_<Config_OSTM*n*>_Stop

OS タイマのカウントを終了します。

[指定形式]

void	R_<Config_OSTM*n*>_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_<Config_OSTM*n*>_Set_CompareValue

インターバルタイマモードの場合、ダウンカウンタの開始値を設定します。
フリーランニングコンペアモードの場合、コンペア値を設定します。

[指定形式]

void	R_<Config_OSTM*n*>_Set_CompareValue (uint32_t *value*);

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	uint32_t *value*;	ダウンカウンタの開始値またはコンペア値

[戻り値]

なし

R_<Config_OSTM*n*>_Create_UserInit

OS 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_<Config_OSTMn>_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_<Config_OSTM*n*>_Create_UserInit (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_<Config_OSTM*n*>_interrupt

OSTM の割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_OSTM*n*>_interrupt (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

3.2.12 ポート

以下に、コード生成ツールがポート用に出力する API 関数の一覧を示します。

表 3-12 ポート用 API 関数

API 関数名	機能概要
R_<Config_PORT>_Create	I/O ポー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_<Config_PORT>_Create_UserInit	I/O ポートに関するユーザ独自の初期化処理を行います。

R_<Config_PORT>_Create

I/O ポート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、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

```
void    R_<Config_PORT>_Create ( void );
```

[引数]

なし

[戻り値]

なし

R_<Config_PORT>_Create_UserInit

I/O ポー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_<Config_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_<Config_PORT>_Create_UserInit (void);

[引数]

なし

[戻り値]

なし

3.2.13 PWM 出力

以下に、コード生成ツールが PWM 出力用に出力する API 関数の一覧を示します。

表 3-13 PWM 出力用 API 関数

API 関数名	機能概要
R_<Config_TAUXn>_Create	PWM 出力を制御するうえで必要となる初期化処理を行います。
R_<Config_TAUXn>_Start	タイマのカウントを開始します。
R_<Config_TAUXn>_Stop	タイマのカウントを終了します。
R_<Config_TAUXn>_Create_UserInit	PWM 出力に関するユーザ独自の初期化処理を行います。
r_<Config_TAUXn>_channelm_interrupt	タイマ割り込みの発生に伴う処理を行います。
r_<Config_TAUXn_m>_interrupt_pek	タイマ割り込みの発生に伴う処理を行います。(CPU コア別)

R_<Config_TAUXn>_Create

PWM 出力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、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

void	R_<Config_TAUXn>_Create (void);

備考　*X* は TAU 名を、*n* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAUXn>_Start

タイマのカウントを開始します。

[指定形式]

void	R_<Config_TAUXn>_Start (void);

備考　*X* は TAU 名を、*n* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAUXn>_Stop

タイマのカウントを終了します。

[指定形式]

void	R_<Config_TAUXn>_Stop (void);

備考　*X* は TAU 名を、*n* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAUXn>_Create_UserInit

PWM 出力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_<Config_TAUXn>_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_<Config_TAUXn>_Create_UserInit ( void );
```

備考　*X* は TAU 名を、*n* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TAUXn>_channel*m*_interrupt

タイマ割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_TAUXn>_channel*m*_interrupt (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TAUXn_m>_interrupt_pe*k*

タイマ割り込みの発生に伴う処理を行います。(CPU コア別)

[指定形式]

void	R_<Config_TAUXn_m>_interrupt_pe*k* (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を、*k* は PE 番号を意味します。

[引数]

なし

[戻り値]

なし

3.2.14 スタンバイコントローラ

以下に、コード生成ツールがスタンバイコントローラ用に出力する API 関数の一覧を示します。

表 3-14 スタンバイコントローラ用 API 関数

API 関数名	機能概要
R_<Config_STBC>_Prepare_Stop_Mode	スタンバイ（STOP モード）を開始するための準備に関するユーザ独自の処理を行います。
R_<Config_STBC>_Start_Stop_Mode	スタンバイ（STOP モード）を開始します。
R_<Config_STBC>_Prepare_Deep_Stop_Mode	スタンバイ（DeepSTOP モード）を開始するための準備に関するユーザ独自の処理を行います。
R_<Config_STBC>_Start_Deep_Stop_Mode	スタンバイ（DeepSTOP モード）を開始します。
R_<Config_STBC>_Deep_Stop_Loop	スタンバイ（DeepSTOP モード）の待ち処理を行います。
R_<Config_STBC>_Prepare_Stop_Mode_Set_Peripheral	スタンバイ（STOP モード）の準備（周辺機能の停止）に関するユーザ独自の処理を行います。
R_<Config_STBC>_Prepare_Stop_Mode_Set_Interrupt	スタンバイ（STOP モード）の準備（割り込み制御レジスタ）に関するユーザ独自の処理を行います。
R_<Config_STBC>_Prepare_Stop_Mode_Set_Clock_Mask	スタンバイ（STOP モード）の準備（クロック停止マスクレジスタを設定）に関するユーザ独自の処理を行います。
R_<Config_STBC>_Prepare_Stop_Mode_Set_Clock_Source	スタンバイ（STOP モード）の準備（各クロックソースの発振／停止）に関するユーザ独自の処理を行います。
R_<Config_STBC>_Prepare_Stop_Mode_Set_CPUCLK	スタンバイ（STOP モード）の準備（CPUCLK）に関するユーザ独自の処理を行います。
R_<Config_STBC>_Prepare_Deep_Stop_Mode_Set_Peripheral	スタンバイ（DeepSTOP モード）の準備（周辺機能の停止）に関するユーザ独自の処理を行います。
R_<Config_STBC>_Prepare_Deep_Stop_Mode_Set_Interrupt	スタンバイ（DeepSTOP モード）の準備（割り込み制御レジスタ）に関するユーザ独自の処理を行います。
R_<Config_STBC>_Prepare_Deep_Stop_Mode_Set_Clock_Mask	スタンバイ（DeepSTOP モード）の準備（クロック停止マスクレジスタ）に関するユーザ独自の処理を行います。
R_<Config_STBC>_Prepare_Deep_Stop_Mode_Set_Clock_Source	スタンバイ（DeepSTOP モード）の準備（各クロックソースの発振または停止）に関するユーザ独自の処理を行います。
R_<Config_STBC>_Set_Module_*XXXX*_Standby_Mode	対象モジュールの指定されたチャネルのスタンバイモードを設定します。
R_<Config_STBC>_Cancel_Module_*XXXX*_Standby_Mode	対象モジュールの指定されたチャネルのスタンバイモードをキャンセルします。
R_<Config_STBC>_Check_Module_*XXXX*_Idle_State	対象モジュールのアイドル状態を確認します。

R_<Config_STBC>_Prepare_Stop_Mode

スタンバイ（STOP モード）を開始するための準備に関するユーザ独自の処理を行います。

[指定形式]

```
void        R_<Config_STBC>_Prepare_Stop_Mode ( void );
```

[引数]

なし

[戻り値]

なし

R_<Config_STBC>_Start_Stop_Mode

スタンバイ（STOP モード）を開始します。

備考　本 API 関数の呼び出し以前に R_<Config_STBC>_Prepare_Stop_Mode を呼び出す必要があります。

[指定形式]

```
void    R_<Config_STBC>_Start_Stop_Mode ( void );
```

[引数]

なし

[戻り値]

なし

R_<Config_STBC>_Prepare_Deep_Stop_Mode

スタンバイ（DeepSTOP モード）を開始するための準備に関するユーザ独自の処理を行います。

[指定形式]

```
void        R_<Config_STBC>_Prepare_Deep_Stop_Mode ( void );
```

[引数]

なし

[戻り値]

なし

R_<Config_STBC>_Start_Deep_Stop_Mode

スタンバイ（DeepSTOP モード）を開始します。

備考　本 API 関数の呼び出し以前に R_<Config_STBC>_Prepare_Deep_Stop_Mode を呼び出す必要があります。

[指定形式]

```
void    R_<Config_STBC>_Start_Deep_Stop_Mode ( void );
```

[引数]

なし

[戻り値]

なし

R_<Config_STBC>_Deep_Stop_Loop

スタンバイ（DeepSTOP モード）の待ち処理を行います。

[指定形式]

```
void        R_<Config_STBC>_Deep_Stop_Loop ( void );
```

[引数]

なし

[戻り値]

なし

R_<Config_STBC>_Prepare_Stop_Mode_Set_Peripheral

スタンバイ（STOP モード）の準備（周辺機能の停止）に関するユーザ独自の処理を行います。

[指定形式]

```
void        R_<Config_STBC>_Prepare_Stop_Mode_Set_Peripheral ( void );
```

[引数]

なし

[戻り値]

なし

R_<Config_STBC>_Prepare_Stop_Mode_Set_Interrupt

スタンバイ（STOP モード）の準備（割り込み制御レジスタ）に関するユーザ独自の処理を行います。

[指定形式]

```
void        R_<Config_STBC>_Prepare_Stop_Mode_Set_Interrupt ( void );
```

[引数]

なし

[戻り値]

なし

R_<Config_STBC>_Prepare_Stop_Mode_Set_Clock_Mask

スタンバイ（STOP モード）の準備（クロック停止マスクレジスタを設定）に関するユーザ独自の処理を行います。

[指定形式]

```
void        R_<Config_STBC>_Prepare_Stop_Mode_Set_Clock_Mask ( void );
```

[引数]

なし

[戻り値]

なし

R_<Config_STBC>_Prepare_Stop_Mode_Set_Clock_Source

スタンバイ（STOP モード）の準備（各クロックソースの発振／停止）に関するユーザ独自の処理を行います。

[指定形式]

```
void        R_<Config_STBC>_Prepare_Stop_Mode_Set_Clock_Source ( void );
```

[引数]

なし

[戻り値]

なし

R_<Config_STBC>_Prepare_Stop_Mode_Set_CPUCLK

スタンバイ（STOP モード）の準備（CPUCLK）に関するユーザ独自の処理を行います。

[指定形式]

```
void    R_<Config_STBC>_Prepare_Stop_Mode_Set_CPUCLK(void);
```

[引数]

なし

[戻り値]

なし

R_<Config_STBC>_Prepare_Deep_Stop_Mode_Set_Peripheral

スタンバイ（DeepSTOP モード）の準備（周辺機能の停止）に関するユーザ独自の処理を行います。

[指定形式]

```
void        R_<Config_STBC>_Prepare_Deep_Stop_Mode_Set_Peripheral ( void );
```

[引数]

なし

[戻り値]

なし

R_<Config_STBC>_Prepare_Deep_Stop_Mode_Set_Interrupt

スタンバイ（DeepSTOP モード）の準備（割り込み制御レジスタ）に関するユーザ独自の処理を行います。

[指定形式]

```
void        R_<Config_STBC>_Prepare_Deep_Stop_Mode_Set_Interrupt ( void );
```

[引数]

なし

[戻り値]

なし

R_<Config_STBC>_Prepare_Deep_Stop_Mode_Set_Clock_Mask

スタンバイ（DeepSTOP モード）の準備（クロック停止マスクレジスタ）に関するユーザ独自の処理を行います。

[指定形式]

```
void      R_<Config_STBC>_Prepare_Deep_Stop_Mode_Set_Clock_Mask(void);
```

[引数]

なし

[戻り値]

なし

R_<Config_STBC>_Prepare_Deep_Stop_Mode_Set_Clock_Source

スタンバイ（DeepSTOP モード）の準備（各クロックソースの発振または停止）に関するユーザ独自の処理を行います。

[指定形式]

```
void      R_<Config_STBC>_Prepare_Deep_Stop_Mode_Set_Clock_Source(void);
```

[引数]

なし

[戻り値]

なし

R_<Config_STBC>_Set_Module_*XXXX*_Standby_Mode

対象モジュールの指定されたチャネルのスタンバイモードを設定します。

[指定形式]

MD_STATUS R_<Config_STBC>_Set_Module_XXXX_Standby_Mode (uint8_t channel);

備考 *XXXX* は対象モジュール名を意味します。対象モジュール名については、表 3.2.14-1STBC モジュール名一覧を参照してください。

[引数]

I/O	引数	説明
I	uint8_t channel	チャネルの指定

[戻り値]

マクロ	説明
MD_OK	正常終了
MD_BUSY1	指定されたチャネルのリセット実行が処理されています。

R_<Config_STBC>_Cancel_Module_*XXXX*_Standby_Mode

対象モジュールの指定されたチャネルのスタンバイモードをキャンセルします。

[指定形式]

```
MD_STATUS      R_<Config_STBC>_Cancel_Module_XXXX_Standby_Mode ( uint8_t channel );
```

備考　　*XXXX* は対象モジュール名を意味します。対象モジュール名については、表 3.2.14-1STBC モジュール名一覧を参照してください。

[引数]

I/O	引数	説明
I	uint8_t channel	チャネルの指定

[戻り値]

マクロ	説明
MD_OK	正常終了
MD_BUSY1	指定されたチャネルのリセット実行が処理されています。

R_<Config_STBC>_Check_Module_*XXXX*_Idle_State

対象モジュールのアイドル状態を確認します。

[指定形式]

```
MD_STATUS      R_<Config_STBC>_Check_Module_XXXX_Idle_State ( void );
```

備考　*XXXX* は対象モジュール名を意味します。対象モジュール名については、表 3.2.14-1STBC モジュール名一覧を参照してください。

[引数]

なし

[戻り値]

なし

表 3.2.14-1 STBCモジュール名一覧

	モジュール名(*XXXX*)		モジュール名(*XXXX*)		モジュール名(*XXXX*)
1	RSCFD	10	MMCA	19	TAUD
2	FLXA	11	ENCA	20	TAUJ_ISO
3	GTM	12	PSI5	21	TPBA
4	ETNB	13	PSI5S	22	TSG3
5	RSENT	14	PWMD	23	OSTM
6	MSPI	15	RHSIF	24	ADCJ_AWO
7	RLIN3	16	RIIC	25	RTCA
8	ADCJ_ISO	17	SCI3	26	TAUJ_AWO
9	CXPI	18	TAPA		

使用例

ウェイクアップ要因に INTP12 を使用して STBC を STOP モードに遷移する場合（青字はユーザコード）

r_cg_main.c

```
void main(void)
{
    r_main_userinit();
    /* Start user code for main. Do not edit comment generated here */
    R_Config_INTC_INTP12_Start();
    R_Config_STBC_Prepare_Stop_Mode();
    R_Config_STBC_Start_Stop_Mode();

    while(1);
    /* End user code. Do not edit comment generated here */
}
```

3.2.15 UART インタフェース

以下に、コード生成ツールが UART インタフェース用に出力する API 関数の一覧を示します。

表 3-15 UART インタフェース用 API 関数

API 関数名	機能概要
R_<Config_UARTn>_Create	UART 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。
R_<Config_UARTn>_Start	UART の動作を許可します。
R_<Config_UARTn>_Stop	UART の動作を禁止します。
R_<Config_UARTn>_Send	データ送信を行います。
R_<Config_UARTn>_Receive	データ受信を行います。
R_<Config_UARTn>_Create_UserInit	UART 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。
r_<Config_UARTn>_interrupt_send	送信割り込みの発生に伴う処理を行います。
r_<Config_UARTn>_interrupt_receive	受信完了割り込みの発生に伴う処理を行います。
r_<Config_UARTn>_interrupt_error	通信エラー割り込みの発生に伴う処理を行います。
r_<Config_UARTn>_callback_sendend	送信割り込みの発生に伴う処理を行います。
r_<Config_UARTn>_callback_receiveend	受信完了割り込みの発生に伴う処理を行います。
r_<Config_UARTn>_callback_error	通信エラー割り込みの発生に伴う処理を行います。

R_<Config_UARTn>_Create

UART 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。

備考　本 API 関数は、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

void	R_<Config_UARTn>_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_<Config_UARTn>_Start

UART の動作を許可します。

[指定形式]

void	R_<Config_UARTn>_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_<Config_UARTn>_Stop

UART の動作を禁止します。

[指定形式]

void	R_<Config_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_<Config_UARTn>_Send

データ送信を行います。

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. 送信を行う際には、本 API 関数の呼び出し以前に R_<Config_UARTn>_Start を呼び出す必要があります。

備考 3. 連続して送信を行う際には、以前の送信が終了した後に本 API 関数を呼び出す必要があります。

[指定形式]

```
MD_STATUS      R_<Config_UARTn>_Send ( uint8_t * const tx_buf, uint16_t tx_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
I	uint8_t * const *tx_buf*;	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 *tx_num*;	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 *tx_num* の指定が不正
MD_ERROR	送信処理を実行中

R_<Config_UARTn>_Receive

データ受信を行います。

備考 1. 本 API 関数では、 1 バイト単位の受信を引数 *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. 受信を行う際には、本 API 関数の呼び出し以前に R_<Config_UARTn>_Start を呼び出す必要があります。

備考 3. 連続して受信を行う際には、以前の受信が終了した後に本 API 関数を呼び出す必要があります。

[指定形式]

```
MD_STATUS      R_<Config_UARTn>_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 *rx_buf*;	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 *rx_num*;	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 *rx_num* の指定が不正

R_<Config_UARTn>_Create_UserInit

UART 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_<Config_UARTn>_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_<Config_UARTn>_Create_UserInit ( 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_<Config_UARTn>_interrupt_send

送信割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_UARTn>_interrupt_send (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_<Config_UARTn>_interrupt_receive

受信完了割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_UARTn>_interrupt_receive (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_<Config_UARTn>_interrupt_error

通信エラー割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_UARTn>_interrupt_error (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_<Config_UARTn>_callback_sendend

送信割り込みの発生に伴う処理を行います。

備考　本 API 関数は、r_<Config_UARTn>_interrupt_send 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_<Config_UARTn>_callback_sendend (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_<Config_UARTn>_callback_receiveend

受信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、r_<Config_UARTn>_interrupt_receive 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_<Config_UARTn>_callback_receiveend (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_<Config_UARTn>_callback_error

通信エラー割り込みの発生に伴う処理を行います。

備考　本 API 関数は、r_<Config_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]

void	R_<Config_UARTn>_callback_error (uint32_t *err_type*);

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	uint32_t *err_type*;	通信エラー割り込みの発生要因 0x00xx01B　：ビット・エラー 0x00x10xB　：オーバラン・エラー 0x001x0xB　：フレーミング・エラー 0100xx0xB　：パリティ・エラー

[戻り値]

なし

使用例

UART0 と UART1 が互いに全二重通信でデータを送受信する場合（青字はユーザコード）

r_cg_main.c

```
extern uint8_t transmitend_flag_0;
extern uint8_t transmitend_flag_1;
extern uint8_t receiveend_flag_0;
extern uint8_t receiveend_flag_1;
uint8_t tx_buf0[3] = {0x11, 0x22, 0x33};
uint8_t tx_buf1[3] = {0x44, 0x55, 0x66};
uint8_t rx_buf0[3];
uint8_t rx_buf1[3];

main()
{
    r_main_userinit();
    /* Start user code for main. Do not edit comment generated here */
    R_Config_UART0_Start();
    R_Config_UART1_Start();

    R_Config_UART1_Receive(rx_buf1, 3);
    R_Config_UART0_Send(tx_buf0, 3);
    while(transmitend_flag_0 != 1);
    while(receiveend_flag_1 != 1);

    R_Config_UART0_Receive(rx_buf0, 3);
    R_Config_UART1_Send(tx_buf1, 3);
    while(transmitend_flag_1 != 1);
    while(receiveend_flag_0 != 1);

    while(1);
    /* End user code. Do not edit comment generated here */
}
```

Config_UARTA0_user.c

```
/* Start user code for global. Do not edit comment generated here */
volatile uint8_t transmitend_flag_0 = 0;
volatile uint8_t receiveend_flag_0 = 0;
/* End user code. Do not edit comment generated here */

void r_Config_UART0_callback_sendend(void)
{
    /* Start user code for r_Config_UART0_callback_sendend. Do not edit comment generated here */
    transmitend_flag_0 = 1;
    /* End user code. Do not edit comment generated here */
}

void r_Config_UART0_callback_receiveend(void)
{
    /* Start user code for r_Config_UART0_callback_receiveend. Do not edit comment generated here
*/
    receiveend_flag_0 = 1;
    /* End user code. Do not edit comment generated here */
}
```

Config_UARTA1_user.c

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

```
volatile uint8_t transmitend_flag_1 = 0;
volatile uint8_t receiveend_flag_1 = 0;
/* End user code. Do not edit comment generated here */

void r_Config_UART1_callback_sendend(void)
{
    /* Start user code for r_Config_UART1_callback_sendend. Do not edit comment generated here */
    transmitend_flag_1 = 1;
    /* End user code. Do not edit comment generated here */
}

void r_Config_UART1_callback_receiveend(void)
{
    /* Start user code for r_Config_UART1_callback_receiveend. Do not edit comment generated here
*/
    receiveend_flag_1 = 1;
    /* End user code. Do not edit comment generated here */
}
```

3.2.16 ウォッチドッグタイマ

以下に、コード生成ツールがウォッチドッグタイマ用に出力する API 関数の一覧を示します。

表 3-16 ウォッチドッグタイマ用 API 関数

API 関数名	機能概要
R_<Config_WDTn>_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。
R_<Config_WDTn>_Restart	WDT トリガを発生し、WDT カウンタをスタート / リスタート処理をします。
R_<Config_WDTn>_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。
r_<Config_WDTn>_interrupt	WDT インターバルタイマ割り込みの発生に伴う処理を行います。
R_<Config_WDTXn_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。
R_<Config_WDTXn_Restart	WDT トリガを発生し、WDT カウンタをスタート / リスタート処理をします。
R_<Config_WDTXn_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。
r_<Config_WDTXn_interrupt	WDT インターバルタイマ割り込みの発生に伴う処理を行います。

R_<Config_WDTn>_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。

備考　本 API 関数は、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

void	R_<Config_WDTn>_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_<Config_WDTn>_Restart

WDT トリガを発生し、WDT カウンタをスタート / リスタート処理をします。

[指定形式]

void	R_<Config_WDTn>_Restart (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_<Config_WDTn>_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_<Config_WDTn>_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_<Config_WDTn>_Create_UserInit ( 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_<Config_WDTn>_interrupt

WDT インターバルタイマ割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_WDTn>_interrupt (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_<Config_WDT*Xn*_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。

備考　本 API 関数は、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

void	R_<Config_WDT*Xn*_Create (void);

備考　*X* は WDT 名を、*n* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_WDT*Xn*_Restart

WDT トリガを発生し、WDT カウンタをスタート / リスタート処理をします。

[指定形式]

void	R_<Config_WDT*Xn*_Restart (void);

備考 *X* は WDT 名を、*n* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_WDT*Xn*_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_<Config_WDTXn_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_<Config_WDT*Xn*_Create_UserInit (void);

備考　*X* は WDT 名を、*n* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_WDT*Xn*_interrupt

WDT インターバルタイマ割り込みの発生に伴う処理を行います。

[指定形式]

void	r_<Config_WDT*Xn*_interrupt (void);

備考 *X* は WDT 名を、*n* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

3.2.17 クロック分周

以下に、コード生成ツールがクロック分周用に出力する API 関数の一覧を示します。

表 3-17 クロック分周用 API 関数

API 関数名	機能概要
R_<Config_TAUXn_m>_Create	クロック分周を制御するうえで必要となる初期化処理を行います。
R_<Config_TAUXn_m>_Start	タイマのカウントを開始します。
R_<Config_TAUXn_m>_Stop	タイマのカウントを終了します。
r_<Config_TAUXn_m>_Create_UserInit	クロック分周に関するユーザ独自の初期化処理を行います。
r_<Config_TAUXn_m>_interrupt	タイマ割り込みの発生に伴う処理を行います。
r_<Config_TAUXn_m>_interrupt_pek	タイマ割り込みの発生に伴う処理を行います。(CPU コア別)

R_<Config_TAUXn_m>_Create

クロック分周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、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

void	R_<Config_TAUXn_m>_Create (void);

備考　*X* は TAU 名を、*n* はユニット番号を、m はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAUXn_m>_Start

タイマのカウントを開始します。

[指定形式]

void	R_<Config_TAUXn_m>_Start (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAUXn_m>_Stop

タイマのカウントを終了します。

[指定形式]

void	R_<Config_TAUXn_m>_Stop (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TAUXn_m>_Create_UserInit

クロック分周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_<Config_TAUXn_m>_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_<Config_TAUXn_m>_Create_UserInit ( void );
```

備考　*X* は TAU 名を、*n* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TAUXn_m>_interrupt_pe*k*

タイマ割り込みの発生に伴う処理を行います。(CPU コア別)

[指定形式]

void	r_<Config_TAUXn_m>_interrupt_pe*k* (void);

備考 *X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を、*k* は PE 番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TAUXn_m>_interrupt

タイマ割り込みの発生に伴う処理を行います。

[指定形式]

void	r_<Config_TAUXn_m>_interrupt (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

3.2.18 データ CRC

以下に、コード生成ツールがデータ CRC 用に出力する API 関数の一覧を示します。

表 3-18 データ CRC 用 API 関数

API 関数名	機能概要
R_<Config_DCRAn>_Create	データ CRC を制御するうえで必要となる初期化処理を行います。
R_<Config_DCRAn>_InitializeCRCData	CRC データレジスタの初期値設定を行います。
R_<Config_DCRAn>_Input32bitData	32bit 幅の演算用入力データを設定します。
R_<Config_DCRAn>_Input16bitData	16bit 幅の演算用入力データを設定します。
R_<Config_DCRAn>_Input8bitData	8bit 幅の演算用入力データを設定します。
R_<Config_DCRAn>_GetResult_32bitData	32bit 幅の CRC 演算結果を読み出します。
R_<Config_DCRAn>_GetResult_16bitData	16bit 幅の CRC 演算結果を読み出します。
R_<Config_DCRAn>_Create_UserInit	データ CRC に関するユーザ独自の初期化処理を行います。
R_<Config_KCRCn>_Create	データ CRC を制御するうえで必要となる初期化処理を行います。
R_<Config_KCRCn>_InitializeCRCData	CRC データレジスタの初期値設定を行います。
R_<Config_KCRCn>_Input32bitData	32bit 幅の演算用入力データを設定します。
R_<Config_KCRCn>_Input16bitData	16bit 幅の演算用入力データを設定します。
R_<Config_KCRCn>_Input8bitData	8bit 幅の演算用入力データを設定します。
R_<Config_KCRCn>_GetResult_64bitData	64bit 幅の CRC 演算結果を読み出します。
R_<Config_KCRCn>_GetResult_32bitData	32bit 幅の CRC 演算結果を読み出します。
R_<Config_KCRCn>_GetResult_16bitData	16bit 幅の CRC 演算結果を読み出します。
R_<Config_KCRCn>_GetResult_8bitData	8bit 幅の CRC 演算結果を読み出します。
R_<Config_KCRCn>_Create_UserInit	データ CRC に関するユーザ独自の初期化処理を行います。

R_<Config_DCRAn>_Create

データ CRC 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、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

void	R_<Config_DCRAn>_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_<Config_DCRA*n*>_InitializeCRCData

CRC データレジスタの初期値設定を行います。

[指定形式]

void R_<Config_DCRA*n*>_InitializeCRCData(uint32_t *crc_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	const uint32_t *crc_data*;	DCRA 初期値

[戻り値]

なし

R_<Config_DCRAn>_Input32bitData

32bit 幅の演算用入力データを設定します。

[指定形式]

void	R_<Config_DCRAn>_Input32bitData (const uint32_t * *data*, uint32_t *data_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	const uint32_t * *data*;	演算データを格納した領域へのポインタ
I	uint32_t *data_num*;	演算データの総数

[戻り値]

なし

R_<Config_DCRAn>_Input16bitData

16bit 幅の演算用入力データを設定します。

[指定形式]

void	R_<Config_DCRAn>_Input16bitData (const uint16_t * *data*, uint32_t *data_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	const uint16_t * *data*;	演算データを格納した領域へのポインタ
I	uint32_t *data_num*;	演算データの総数

[戻り値]

なし

R_<Config_DCRAn>_Input8bitData

8bit 幅の演算用入力データを設定します。

[指定形式]

void	R_<Config_DCRAn>_Input8bitData (const uint8_t * *data*, uint32_t *data_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	const uint8_t * *data*;	演算データを格納した領域へのポインタ
I	uint32_t *data_num*;	演算データの総数

[戻り値]

なし

R_<Config_DCRAn>_GetResult_32bitData

32bit 幅の CRC 演算結果を読み出します。

[指定形式]

void	R_<Config_DCRAn>_GetResult_32bitData (uint32_t * *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
O	uint32_t * *data*;	演算データの結果を格納する領域へのポインタ

[戻り値]

なし

R_<Config_DCRAn>_GetResult_16bitData

16bit 幅の CRC 演算結果を読み出します。

[指定形式]

void	R_<Config_DCRAn>_GetResult_16bitData (uint16_t * *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
O	uint16_t * *data*;	演算データの結果を格納する領域へのポインタ

[戻り値]

なし

R_<Config_DCRAn>_Create_UserInit

データ CRC 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_<Config_DCRAn>_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_<Config_DCRAn>_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_<Config_KCRCn>_Create

データ CRC 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、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

void	R_<Config_KCRCn>_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_<Config_KCRC*n*>_InitializeCRCData

CRC データレジスタの初期値設定を行います。

[指定形式]

void R_<Config_KCRC*n*>_InitializeCRCData(uint32_t *crcout0_data*, uint32_t *crcout1_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	uint32_t *crcout0_data*	DCRA 初期値(Lower)
I	uint32_t *crcout1_data*	DCRA 初期値(Upper)

[戻り値]

なし

R_<Config_KCRCn>_Input32bitData

32bit 幅の演算用入力データを設定します。

[指定形式]

void	R_<Config_KCRCn>_Input32bitData (const uint32_t * *data*, uint32_t *data_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	const uint32_t * *data*;	演算データを格納した領域へのポインタ
I	uint32_t *data_num*;	演算データの総数

[戻り値]

なし

R_<Config_KCRCn>_Input16bitData

16bit 幅の演算用入力データを設定します。

[指定形式]

void	R_<Config_KCRCn>_Input16bitData (const uint16_t * *data*, uint32_t *data_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	const uint16_t * *data*;	演算データを格納した領域へのポインタ
I	uint32_t *data_num*;	演算データの総数

[戻り値]

なし

R_<Config_KCRCn>_Input8bitData

8bit 幅の演算用入力データを設定します。

[指定形式]

void	R_<Config_KCRCn>_Input8bitData (const uint8_t * *data*, uint32_t *data_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	const uint8_t * *data*;	演算データを格納した領域へのポインタ
I	uint32_t *data_num*;	演算データの総数

[戻り値]

なし

R_<Config_KCRCn>_GetResult_64bitData

64bit 幅の CRC 演算結果を読み出します。

[指定形式]

void	R_<Config_KCRCn>_GetResult_64bitData (uint64_t * *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
O	uint64_t * *data*;	演算データの結果を格納する領域へのポインタ

[戻り値]

なし

R_<Config_KCRCn>_GetResult_32bitData

32bit 幅の CRC 演算結果を読み出します。

[指定形式]

void	R_<Config_KCRCn>_GetResult_32bitData (uint32_t * *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
O	uint32_t * *data*;	演算データの結果を格納する領域へのポインタ

[戻り値]

なし

R_<Config_KCRCn>_GetResult_16bitData

16bit 幅の CRC 演算結果を読み出します。

[指定形式]

void	R_<Config_KCRCn>_GetResult_16bitData (uint16_t * *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
O	uint16_t * *data*;	演算データの結果を格納する領域へのポインタ

[戻り値]

なし

R_<Config_KCRCn>_GetResult_8bitData

8bit 幅の CRC 演算結果を読み出します。

[指定形式]

void	R_<Config_KCRCn>_GetResult_8bitData (uint8_t * *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
O	uint8_t * *data*;	演算データの結果を格納する領域へのポインタ

[戻り値]

なし

R_<Config_KCRCn>_Create_UserInit

データ CRC 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_<Config_KCRCn>_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_<Config_KCRCn>_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

3.2.19 ディレイカウント

以下に、コード生成ツールがディレイカウント用に出力する API 関数の一覧を示します。

表 3-19 ディレイカウント用 API 関数

API 関数名	機能概要
R_<Config_TAUXn_m>_Create	ディレイカウン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_<Config_TAUXn_m>_Start	タイマのカウントを開始します。
R_<Config_TAUXn_m>_Stop	タイマのカウントを終了します。
R_<Config_TAUXn_m>_Create_UserInit	ディレイカウントに関するユーザ独自の初期化処理を行います。
r_<Config_TAUXn_m>_interrupt	タイマ割り込みの発生に伴う処理を行います。
r_<Config_TAUXn_m>_interrupt_pek	タイマ割り込みの発生に伴う処理を行います。(CPU コア別)

R_<Config_TAUXn_m>_Create

ディレイカウント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、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

void	R_<Config_TAUXn_m>_Create (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAUXn_m>_Start

タイマのカウントを開始します。

[指定形式]

void	R_<Config_TAUXn_m>_Start (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAUXn_m>_Stop

タイマのカウントを終了します。

[指定形式]

void	R_<Config_TAUXn_m>_Stop (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAUXn_m>_Create_UserInit

ディレイカウン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_<Config_TAUXn_m>_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_<Config_TAUXn_m>_Create_UserInit (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TAUXn_m>_interrupt

タイマ割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_TAUXn_m>_interrupt (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TAUXn_m>_interrupt_pe*k*

タイマ割り込みの発生に伴う処理を行います。(CPU コア別)

[指定形式]

void	R_<Config_TAUXn_m>_interrupt_pe*k* (void);

備考 *X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を、*k* は PE 番号を意味します。

[引数]

なし

[戻り値]

なし

3.2.20 DMA コントローラ

以下に、コード生成ツールが DMA コントローラ用に出力する API 関数の一覧を示します。

表 3-20 DMA コントローラ用 API 関数

API 関数名	機能概要
R_<Config_DMACn>_Create	DMAC*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_DMACnm_Create	DMAC*n* のチャネル*m*を制御するうえで必要となる初期化処理を行います。
R_<Config_DMACnm>_Start	DMAC*n* のチャネル*m*の転送を許可します。
R_<Config_DMACnm>_Stop	DMAC*n* のチャネル*m*の転送を停止します。
R_<Config_DMACnm>_Set_SoftwareTrigger	DMAC*n* のチャネル*m*の転送要求を発生します。
R_<Config_DMACnm>_Suspend	DMAC*n* のチャネル*m*の転送を一時停止します。
R_<Config_DMACnm>_Resume	DMAC*n* のチャネル*m*の転送の一時停止を解除します。
R_<Config_DMACn_Create_UserInit	DMAC*n* に関するユーザ独自の初期化処理を行います。
r_<Config_DMACnm>_dmacnm_interrupt	DMAC*n* のチャネル*m*の割り込みの発生に伴う処理を行います。
r_<Config_DMACnm>_Callback_DMACnm_Transfer_Error	DMAC*n* のチャネル*m*の転送エラー割り込みの発生に伴う処理を行います。
r_<Config_DMACnm>_callback_transfer_completion	DMAC*n* のチャネル*m*の転送完了割り込みの発生に伴う処理を行います。
r_<Config_DMACnm>_callback_transfer_count_match	DMAC*n* のチャネル*m*の転送回数一致割り込みの発生に伴う処理を行います。
R_<Config_SDMACnm>_Create	SDMAC*n* のチャネル*m*を制御するうえで必要となる初期化処理を行います。
R_<Config_SDMACnm>_Start	SDMAC*n* のチャネル*m*の転送を許可します。
R_<Config_SDMACnm>_Stop	SDMAC*n* のチャネル*m*の転送を停止します。
R_<Config_SDMACnm>_Resume	SDMAC*n* のチャネル*m*の転送の一時停止を解除します。
R_<Config_SDMACnm>_Set_DescriptorMemory	SDMAC*n* のチャネル*m*のディスクリプタメモリを設定します。
R_<Config_SDMACnm>_Create_UserInit	SDMAC*n* のチャネル*m*に関するユーザ独自の初期化処理を行います。
r_<Config_SDMACnm>_end_interrupt	SDMAC*n* のチャネル*m*の転送完了割り込みに伴う処理を行います。
r_<Config_SDMACnm>_Callback_PEk_Address_Error	SDMAC*n* のチャネル*m*のアドレスエラー割り込みの発生に伴う処理を行います。(CPU コア別)

R_<Config_DMAC*n*>_Create

DMAC*n* 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、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

void	R_<Config_DMAC*n*_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_DMAC*nm*_Create

DMAC*n* のチャネル*m*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、R_<Config_DMACn>_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_DMAC*nm*_Create (void);

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_DMACnm>_Start

DMAC*n* のチャネル*m*の転送を許可します。

[指定形式]

void	R_<Config_DMACnm>_Start (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_DMACnm>_Stop

DMAC*n* のチャネル*m*の転送を停止します。

[指定形式]

void	R_<Config_DMACnm>_Stop (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_DMACnm>_Set_SoftwareTrigger

DMAC*n* のチャネル*m*の転送要求を発生します。

[指定形式]

void	R_<Config_DMACnm>_Set_SoftwareTrigger (void);

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_DMACnm>_Suspend

DMAC*n* のチャネル*m*の転送を一時停止します。

[指定形式]

void	R_<Config_DMACnm>_Suspend (void);

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_DMACnm>_Resume

DMAC*n* のチャネル*m*の転送の一時停止を解除します。

[指定形式]

void	R_<Config_DMACnm>_Resume (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_DMAC*n*_Create_UserInit

DMAC*n* のチャネル*m*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_<Config_DMACn>_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_<Config_DMACnm>_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_<Config_DMAC*nm*>_dmac*nm*_interrupt

DMAC*n* のチャネル*m*の割り込みの発生に伴う処理を行います。

[指定形式]

void r_<Config_DMAC*nm*>_dmac*nm*_interrupt(void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_DMAC*nm*>_Callback_DMAC*nm*_Transfer_Error

DMAC*n* のチャネル*m*の転送エラー割り込みの発生に伴う処理を行います。

備考　本 API 関数は、r_dmac_error_interrupt_pek 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_<Config_DMAC*nm*>_Callback_DMAC*nm*_Transfer_Error(void);
```

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_DMACnm>_callback_transfer_completion

DMAC*n* のチャネル*m*の転送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、r_<Config_DMACnm>_dmacnm_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]

void	R_<Config_DMACnm>_callback_transfer_completion (void);

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_DMACnm>_callback_transfer_count_match

DMAC*n* のチャネル*m*の転送回数一致割り込みの発生に伴う処理を行います。

備考　本 API 関数は、r_<Config_DMACnm>_dmacnm_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]

void	R_<Config_DMACnm>_callback_transfer_count_match (void);

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_SDMACnm>_Create

SDMAC*n* のチャネル*m*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、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

```
void    R_<Config_SDMACnm>_Create ( void );
```

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_SDMACnm>_Start

SDMAC*n* のチャネル*m*の転送を許可します。

[指定形式]

void	R_<Config_SDMACnm>_Start (void);

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_SDMACnm>_Stop

SDMAC*n* のチャネル*m*の転送を停止します。

[指定形式]

void	R_<Config_SDMACnm>_Stop (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_SDMACnm>_Resume

SDMAC*n* のチャネル*m*の転送の一時停止を解除します。

[指定形式]

void	R_<Config_SDMACnm>_Resume (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_SDMACnm>_Reset

SDMAC*n* のチャネル*m*をリセットします。

[指定形式]

void	R_<Config_SDMAC*nm*>_Reset(void);

備考　　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_SDMACnm>_Set_DescriptorMemory

SDMAC*n* のチャネル*m*のディスクリプタメモリを設定します。

[指定形式]

```
void R_<Config_SDMACnm>_Set_DescriptorMemory ( uint32_t * const address, uint32_t * const regValues, uint8_t count_num );
```

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

I/O	引数	説明
I	uint32_t * const address	ディスクリプタメモリを登録するアドレス
I	uint32_t * const regValues	登録するデータが格納してあるポインタ
I	uint8_t count_num	登録するデータ数

[戻り値]

なし

R_<Config_SDMACnm>_Create_UserInit

SDMAC*n* のチャネル*m*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_<Config_SDMACnm>_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_<Config_SDMACnm>_Create_UserInit (void);

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_SDMACnm>_end_interrupt

SDMAC*n* のチャネル*m*の転送完了割り込みに伴う処理を行います。

[指定形式]

void	R_<Config_SDMACnm>_end_interrupt (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_SDMAC*nm*>_Callback_PE*k*_Address_Error

SDMAC*n* のチャネル*m*の アドレスエラー割り込みの発生に伴う処理を行います。(CPU コア別)

備考　本 API 関数は、r_sdmac_address_error_interrupt_pek 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_<Config_SDMACnm>_Callback_PEk_Address_Error (void);
```

備考　*n* はユニット番号を、*m* はチャネル番号を、*k* は PE 番号を意味します。

[引数]

なし

[戻り値]

なし

3.2.21 外部イベントカウント

以下に、コード生成ツールが外部イベントカウント用に出力する API 関数の一覧を示します。

表 3-21 外部イベントカウント用 API 関数

API 関数名	機能概要
R_<Config_TAUXn_m>_Create	外部イベントカウン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_<Config_TAUXn_m>_Start	タイマのカウントを開始します。
R_<Config_TAUXn_m>_Stop	タイマのカウントを終了します。
R_<Config_TAUXn_m>_Create_UserInit	外部イベントカウントに関するユーザ独自の初期化処理を行います。
r_<Config_TAUXn_m>_interrupt	タイマ割り込みの発生に伴う処理を行います。
r_<Config_TAUXn_m>_interrupt_pek	タイマ割り込みの発生に伴う処理を行います。(CPU コア別)

R_<Config_TAUXn_m>_Create

外部イベントカウント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、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

```
void    R_<Config_TAUXn_m>_Create ( void );
```

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAUXn_m>_Start

タイマのカウントを開始します。

[指定形式]

void	R_<Config_TAUXn_m>_Start (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAUXn_m>_Stop

タイマのカウントを終了します。

[指定形式]

void	R_<Config_TAUXn_m>_Stop (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAUXn_m>_Create_UserInit

外部イベントカウン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_<Config_TAUXn_m>_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_<Config_TAUXn_m>_Create_UserInit (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TAUXn_m>_interrupt

タイマ割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_TAUXn_m>_interrupt (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TAUXn_m>_interrupt_pe*k*

タイマ割り込みの発生に伴う処理を行います。(CPU コア別)

[指定形式]

void	R_<Config_TAUXn_m>_interrupt_pe*k* (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を、*k* は PE 番号を意味します。

[引数]

なし

[戻り値]

なし

3.2.22 入力期間カウント検出

以下に、コード生成ツールが入力期間カウント検出用に出力する API 関数の一覧を示します。

表 3-22 入力期間カウント検出用 API 関数

API 関数名	機能概要
R_<Config_TAUXn_m>_Create	入力期間カウント検出を制御するうえで必要となる初期化処理を行います。
R_<Config_TAUXn_m>_Start	タイマのカウントを開始します。
R_<Config_TAUXn_m>_Stop	タイマのカウントを終了します。
R_<Config_TAUXn_m>_Get_PulseWidth	タイマのパルス幅を測定します
R_<Config_TAUXn_m>_Create_UserInit	入力期間カウント検出に関するユーザ独自の初期化処理を行います。
r_<Config_TAUXn_m>_interrupt	タイマ割り込みの発生に伴う処理を行います。
r_<Config_TAUXn_m>_interrupt_pek	タイマ割り込みの発生に伴う処理を行います。(CPU コア別)

R_<Config_TAUXn_m>_Create

入力期間カウント検出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、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

void	R_<Config_TAUXn_m>_Create (void);

備考　　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAUXn_m>_Start

タイマのカウントを開始します。

[指定形式]

void	R_<Config_TAUXn_m>_Start (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAUXn_m>_Stop

タイマのカウントを終了します。

[指定形式]

void	R_<Config_TAUXn_m>_Stop (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAUXn_m>_Get_PulseWidth

タイマのパルス幅を測定します。

[指定形式]

void	R_<Config_TAUXn_m>_Get_PulseWidth (uint32_t * const *width*);

備考　　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

I/O	引数	説明
O	uint32_t * const *width*;	パルス幅を格納する領域へのポインタ

[戻り値]

なし

R_<Config_TAUXn_m>_Create_UserInit

入力期間カウント検出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_<Config_TAUXn_m>_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_<Config_TAUXn_m>_Create_UserInit (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TAUXn_m>_interrupt

タイマ割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_TAUXn_m>_interrupt (void);

備考 *X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TAUXn_m>_interrupt_pe*k*

タイマ割り込みの発生に伴う処理を行います。(CPU コア別)

[指定形式]

void	R_<Config_TAUXn_m>_interrupt_pe*k* (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を、*k* は PE 番号を意味します。

[引数]

なし

[戻り値]

なし

3.2.23 入力位置検出

以下に、コード生成ツールが入力位置検出用に出力する API 関数の一覧を示します。

表 3-23 入力位置検出用 API 関数

API 関数名	機能概要
R_<Config_TAUXn_m>_Create	入力位置検出を制御するうえで必要となる初期化処理を行います。
R_<Config_TAUXn_m>_Start	タイマのカウントを開始します。
R_<Config_TAUXn_m>_Stop	タイマのカウントを終了します。
R_<Config_TAUXn_m>_Get_PulseWidth	タイマのパルス幅を測定します
R_<Config_TAUXn_m>_Create_UserInit	入力位置検出に関するユーザ独自の初期化処理を行います。
r_<Config_TAUXn_m>_interrupt	タイマ割り込みの発生に伴う処理を行います。
r_<Config_TAUXn_m>_interrupt_pek	タイマ割り込みの発生に伴う処理を行います。(CPU コア別)

R_<Config_TAUXn_m>_Create

入力位置検出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、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

void	R_<Config_TAUXn_m>_Create (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAUXn_m>_Start

タイマのカウントを開始します。

[指定形式]

void	R_<Config_TAUXn_m>_Start (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAUXn_m>_Stop

タイマのカウントを終了します。

[指定形式]

void	R_<Config_TAUXn_m>_Stop (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAUXn_m>_Get_PulseWidth

タイマのパルス幅を測定します。

[指定形式]

void	R_<Config_TAUXn_m>_Get_PulseWidth (uint32_t * const *width*);

備考　　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

I/O	引数	説明
O	uint32_t * const *width*;	パルス幅を格納する領域へのポインタ

[戻り値]

なし

R_<Config_TAUXn_m>_Create_UserInit

入力位置検出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_<Config_TAUXn_m>_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_<Config_TAUXn_m>_Create_UserInit (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TAUXn_m>_interrupt

タイマ割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_TAUXn_m>_interrupt (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TAUXn_m>_interrupt_pe*k*

タイマ割り込みの発生に伴う処理を行います。(CPU コア別)

[指定形式]

void	R_<Config_TAUXn_m>_interrupt_pe*k* (void);

備考 *X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を、*k* は PE 番号を意味します。

[引数]

なし

[戻り値]

なし

3.2.24 入力パルスインターバル判定

以下に、コード生成ツールが入力パルスインターバル判定用に出力する API 関数の一覧を示します。

表 3-24 入力パルスインターバル判定用 API 関数

API 関数名	機能概要
R_<Config_TAUXn_m>_Create	入力パルスインターバル判定を制御するうえで必要となる初期化処理を行います。
R_<Config_TAUXn_m>_Start	タイマのカウントを開始します。
R_<Config_TAUXn_m>_Stop	タイマのカウントを終了します。
R_<Config_TAUXn_m>_Create_UserInit	入力パルスインターバル判定に関するユーザ独自の初期化処理を行います。
r_<Config_TAUXn_m>_interrupt	タイマ割り込みの発生に伴う処理を行います。
r_<Config_TAUXn_m>_interrupt_pek	タイマ割り込みの発生に伴う処理を行います。(CPU コア別)

R_<Config_TAUXn_m>_Create

入力パルスインターバル判定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、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

```
void    R_<Config_TAUXn_m>_Create ( void );
```

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAUXn_m>_Start

タイマのカウントを開始します。

[指定形式]

void	R_<Config_TAUXn_m>_Start (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAUXn_m>_Stop

タイマのカウントを終了します。

[指定形式]

void	R_<Config_TAUXn_m>_Stop (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAUXn_m>_Create_UserInit

入力パルスインターバル判定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_<Config_TAUXn_m>_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_<Config_TAUXn_m>_Create_UserInit (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TAUXn_m>_interrupt

タイマ割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_TAUXn_m>_interrupt (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TAUXn_m>_interrupt_pe*k*

タイマ割り込みの発生に伴う処理を行います。(CPU コア別)

[指定形式]

void	R_<Config_TAUXn_m>_interrupt_pe*k* (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を、*k* は PE 番号を意味します。

[引数]

なし

[戻り値]

なし

3.2.25 入力信号幅判定

以下に、コード生成ツールが入力信号幅判定用に出力する API 関数の一覧を示します。

表 3-25 入力信号幅判定用 API 関数

API 関数名	機能概要
R_<Config_TAUXn_m>_Create	入力信号幅判定を制御するうえで必要となる初期化処理を行います。
R_<Config_TAUXn_m>_Start	タイマのカウントを開始します。
R_<Config_TAUXn_m>_Stop	タイマのカウントを終了します。
R_<Config_TAUXn_m>_Create_UserInit	入力信号幅判定に関するユーザ独自の初期化処理を行います。
r_<Config_TAUXn_m>_interrupt	タイマ割り込みの発生に伴う処理を行います。
r_<Config_TAUXn_m>_interrupt_pek	タイマ割り込みの発生に伴う処理を行います。(CPU コア別)

R_<Config_TAUXn_m>_Create

入力信号幅判定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、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

void	R_<Config_TAUXn_m>_Create (void);

備考　　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAUXn_m>_Start

タイマのカウントを開始します。

[指定形式]

void	R_<Config_TAUXn_m>_Start (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAUXn_m>_Stop

タイマのカウントを終了します。

[指定形式]

void	R_<Config_TAUXn_m>_Stop (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAUXn_m>_Create_UserInit

入力信号幅判定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_<Config_TAUXn_m>_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_<Config_TAUXn_m>_Create_UserInit (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TAUXn_m>_interrupt

タイマ割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_TAUXn_m>_interrupt (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TAUXn_m>_interrupt_pe*k*

タイマ割り込みの発生に伴う処理を行います。(CPU コア別)

[指定形式]

void	R_<Config_TAUXn_m>_interrupt_pe*k* (void);

備考 *X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を、*k* は PE 番号を意味します。

[引数]

なし

[戻り値]

なし

3.2.26 入力信号幅測定

以下に、コード生成ツールが入力信号幅測定用に出力する API 関数の一覧を示します。

表 3-26 入力信号幅測定用 API 関数

API 関数名	機能概要
R_<Config_TAUXn_m>_Create	入力信号幅測定を制御するうえで必要となる初期化処理を行います。
R_<Config_TAUXn_m>_Start	タイマのカウントを開始します。
R_<Config_TAUXn_m>_Stop	タイマのカウントを終了します。
R_<Config_TAUXn_m>_Get_PulseWidth	タイマのパルス幅を測定します
R_<Config_TAUXn_m>_Create_UserInit	入力信号幅測定に関するユーザ独自の初期化処理を行います。
r_<Config_TAUXn_m>_interrupt	タイマ割り込みの発生に伴う処理を行います。
r_<Config_TAUXn_m>_interrupt_pek	タイマ割り込みの発生に伴う処理を行います。(CPU コア別)

R_<Config_TAUXn_m>_Create

入力パルスインターバル測定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、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

```
void    R_<Config_TAUXn_m>_Create ( void );
```

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAUXn_m>_Start

タイマのカウントを開始します。

[指定形式]

void	R_<Config_TAUXn_m>_Start (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAUXn_m>_Stop

タイマのカウントを終了します。

[指定形式]

void	R_<Config_TAUXn_m>_Stop (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAUXn_m>_Get_PulseWidth

タイマのパルス幅を測定します。

[指定形式]

void	R_<Config_TAUXn_m>_Get_PulseWidth (uint32_t * const *width*);

備考　　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

I/O	引数	説明
O	uint32_t * const *width*;	パルス幅を格納する領域へのポインタ

[戻り値]

なし

R_<Config_TAUXn_m>_Create_UserInit

入力パルスインターバル測定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_<Config_TAUXn_m>_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_<Config_TAUXn_m>_Create_UserInit (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TAUXn_m>_interrupt

タイマ割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_TAUXn_m>_interrupt (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TAUXn_m>_interrupt_pe*k*

タイマ割り込みの発生に伴う処理を行います。(CPU コア別)

[指定形式]

void	R_<Config_TAUXn_m>_interrupt_pe*k* (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を、*k* は PE 番号を意味します。

[引数]

なし

[戻り値]

なし

3.2.27 キー割り込み機能

以下に、コード生成ツールがキー割り込み機能用に出力する API 関数の一覧を示します。

表 3-27 キー割り込み機能用 API 関数

API 関数名	機能概要
R_<Config_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。
R_<Config_KEY>_Start	キー割り込みの受け付けを許可します。
R_<Config_KEY>_Stop	キー割り込みの受け付けを禁止します。
R_<Config_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。
r_<Config_KEY>_interrupt	キー割り込みの発生に伴う処理を行います。

R_<Config_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。

備考　　本 API 関数は、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

```
void    R_<Config_KEY>_Create ( void );
```

[引数]

なし

[戻り値]

なし

R_<Config_KEY>_Start

キー割り込みの受け付けを許可します。

[指定形式]

```
void        R_<Config_KEY>_Start ( void );
```

[引数]

なし

[戻り値]

なし

R_<Config_KEY>_Stop

キー割り込みの受け付けを禁止します。

[指定形式]

```
void        R_<Config_KEY>_Stop ( void );
```

[引数]

なし

[戻り値]

なし

R_<Config_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_<Config_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_<Config_KEY>_Create_UserInit ( void );
```

[引数]

なし

[戻り値]

なし

r_<Config_KEY>_interrupt

キー割り込みの発生に伴う処理を行います。

[指定形式]

```
void        R_<Config_KEY>_interrupt ( void );
```

[引数]

なし

[戻り値]

なし

3.2.28 ワンパルス出力

以下に、コード生成ツールがワンパルス出力用に出力する API 関数の一覧を示します。

表 3-28 ワンパルス出力用 API 関数

API 関数名	機能概要
R_<Config_TAUXn_m>_Create	ワンパルス出力を制御するうえで必要となる初期化処理を行います。
R_<Config_TAUXn_m>_Start	タイマのカウントを開始します。
R_<Config_TAUXn_m>_Stop	タイマのカウントを終了します。
R_<Config_TAUXn_m>_Create_UserInit	ワンパルス出力に関するユーザ独自の初期化処理を行います。
r_<Config_TAUXn_m>_interrupt	タイマ割り込みの発生に伴う処理を行います。
r_<Config_TAUXn_m>_interrupt_pek	タイマ割り込みの発生に伴う処理を行います。(CPU コア別)

R_<Config_TAUXn_m>_Create

ワンパルス出力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、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

void	R_<Config_TAUXn_m>_Create (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAUXn_m>_Start

タイマのカウントを開始します。

[指定形式]

void	R_<Config_TAUXn_m>_Start (void);

備考　X は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAUXn_m>_Stop

タイマのカウントを終了します。

[指定形式]

void	R_<Config_TAUXn_m>_Stop (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAUXn_m>_Create_UserInit

ワンパルス出力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_<Config_TAUXn_m>_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_<Config_TAUXn_m>_Create_UserInit (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TAUXn_m>_interrupt

タイマ割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_TAUXn_m>_interrupt (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TAUXn_m>_interrupt_pe*k*

タイマ割り込みの発生に伴う処理を行います。(CPU コア別)

[指定形式]

void	R_<Config_TAUXn_m>_interrupt_pe*k* (void);

備考 *X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を、*k* は PE 番号を意味します。

[引数]

なし

[戻り値]

なし

3.2.29 ワンショットパルス出力

以下に、コード生成ツールがワンショットパルス出力用に出力する API 関数の一覧を示します。

表 3-29 ワンショットパルス出力用 API 関数

API 関数名	機能概要
R_<Config_TAUXn>_Create	ワンショットパルス出力を制御するうえで必要となる初期化処理を行います。
R_<Config_TAUXn>_Start	タイマのカウントを開始します。
R_<Config_TAUXn>_Stop	タイマのカウントを終了します。
	タイマのチャネルの開始要求(ソフトウェアトリガ)を発生します
R_<Config_TAUXn>_ SoftwareTriggerOn	ワンショットパルス出力に関するユーザ独自の初期化処理を行います。
r_<Config_TAUXn>_channelm_interrupt	タイマ割り込みの発生に伴う処理を行います。
r_<Config_TAUXn>_channelm_interrupt_pek	タイマ割り込みの発生に伴う処理を行います。(CPU コア別)

R_<Config_TAUX*n*>_Create

ワンショットパルス出力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、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

void	R_<Config_TAUX*n*>_Create (void);

備考　*X* は TAU 名を、*n* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAUX*n*>_Start

タイマのカウントを開始します。

[指定形式]

void	R_<Config_TAUX*n*>_Start (void);

備考　*X* は TAU 名を、*n* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAUX*n*>_Stop

タイマのカウントを終了します。

[指定形式]

void	R_<Config_TAUX*n*>_Stop (void);

備考　*X* は TAU 名を、*n* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAUX*n*>_ SoftwareTriggerOn

タイマのチャネルの開始要求(ソフトウェアトリガ)を発生します。

[指定形式]

void	R_<Config_TAUX*n* >_ SoftwareTriggerOn (void);

備考　*X* は TAU 名を、*n* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAUX*n*>_Create_UserInit

ワンショットパルス出力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_<Config_TAUXn>_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_<Config_TAUX*n* >_Create_UserInit (void);

備考　*X* は TAU 名を、*n* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TAUX*n*>_channel*m*_interrupt

タイマ割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_TAUX*n*>_channel*m*_interrupt (void);

備考　*X* は TAU 名を、*n* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TAUX*n*>_channel*m*_interrupt_pe*k*

タイマ割り込みの発生に伴う処理を行います。(CPU コア別)

[指定形式]

void	R_<Config_TAUX*n*>_channel*m*_interrupt_pe*k* (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*k* は PE 番号を意味します。

[引数]

なし

[戻り値]

なし

3.2.30 オーバーフロー割り込み出力機能（入力期間カウント検出）

以下に、コード生成ツールがオーバーフロー割り込み出力機能（入力期間カウント検出）用に出力する API 関数の一覧を示します。

表 3-30 オーバーフロー割り込み出力機能（入力期間カウント検出）用 API 関数

API 関数名	機能概要
R_<Config_TAUXn_m>_Create	オーバーフロー割り込み出力機能（入力期間カウント検出）を制御するうえで必要となる初期化処理を行います。
R_<Config_TAUXn_m>_Start	タイマのカウントを開始します。
R_<Config_TAUXn_m>_Stop	タイマのカウントを終了します。
R_<Config_TAUXn_m>_Create_UserInit	オーバーフロー割り込み出力機能（入力期間カウント検出）に関するユーザ独自の初期化処理を行います。
r_<Config_TAUXn_m>_interrupt	タイマ割り込みの発生に伴う処理を行います。
r_<Config_TAUXn_m>_interrupt_pek	タイマ割り込みの発生に伴う処理を行います。(CPU コア別)

R_<Config_TAUXn_m>_Create

オーバーフロー割り込み出力機能（入力期間カウント検出）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、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

void	R_<Config_TAUXn_m>_Create (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAUXn_m>_Start

タイマのカウントを開始します。

[指定形式]

void	R_<Config_TAUXn_m>_Start (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAUXn_m>_Stop

タイマのカウントを終了します。

[指定形式]

void	R_<Config_TAUXn_m>_Stop (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAUXn_m>_Create_UserInit

オーバーフロー割り込み出力機能（入力期間カウント検出）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_<Config_TAUXn_m>_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_<Config_TAUXn_m>_Create_UserInit (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TAUXn_m>_interrupt

タイマ割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_TAUXn_m>_interrupt (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TAUXn_m>_interrupt_pe*k*

タイマ割り込みの発生に伴う処理を行います。(CPU コア別)

[指定形式]

void	R_<Config_TAUXn_m>_interrupt_pe*k* (void);

備考 *X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を、*k* は PE 番号を意味します。

[引数]

なし

[戻り値]

なし

3.2.31 オーバーフロー割り込み出力機能（幅測定）

以下に、コード生成ツールがオーバーフロー割り込み出力機能（幅測定）用に出力する API 関数の一覧を示します。

表 3-31 オーバーフロー割り込み出力機能（幅測定）用 API 関数

API 関数名	機能概要
R_<Config_TAUXn_m>_Create	オーバーフロー割り込み出力機能（幅測定）を制御するうえで必要となる初期化処理を行います。
R_<Config_TAUXn_m>_Start	タイマのカウントを開始します。
R_<Config_TAUXn_m>_Stop	タイマのカウントを終了します。
R_<Config_TAUXn_m>_Create_UserInit	オーバーフロー割り込み出力機能（幅測定）に関するユーザ独自の初期化処理を行います。
r_<Config_TAUXn_m>_interrupt	タイマ割り込みの発生に伴う処理を行います。
r_<Config_TAUXn_m>_interrupt_pek	タイマ割り込みの発生に伴う処理を行います。(CPU コア別)

R_<Config_TAUXn_m>_Create

オーバーフロー割り込み出力機能（幅測定）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、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

void	R_<Config_TAUXn_m>_Create (void);

備考　*X* は TAU 名を、T*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAUXn_m>_Start

タイマのカウントを開始します。

[指定形式]

void	R_<Config_TAUXn_m>_Start (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAUXn_m>_Stop

タイマのカウントを終了します。

[指定形式]

void	R_<Config_TAUXn_m>_Stop (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TAUXn_m>_Create_UserInit

オーバーフロー割り込み出力機能（幅測定）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_<Config_TAUXn_m>_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_<Config_TAUXn_m>_Create_UserInit (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TAUXn_m>_interrupt

タイマ割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_TAUXn_m>_interrupt (void);

備考　*X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TAUXn_m>_interrupt_pe*k*

タイマ割り込みの発生に伴う処理を行います。(CPU コア別)

[指定形式]

void	R_<Config_TAUXn_m>_interrupt_pe*k* (void);

備考 *X* は TAU 名を、*n* はユニット番号を、*m* はチャネル番号を、*k* は PE 番号を意味します。

[引数]

なし

[戻り値]

なし

3.2.32 IIC マスタモード

以下に、コード生成ツールが IIC マスタモード用に出力する API 関数の一覧を示します。

表 3-32 IIC マスタモード用 API 関数

API 関数名	機能概要
R_<Config_RIICn>_Create	IIC マスタモードを制御するうえで必要となる初期化処理を行います。
R_<Config_RIICn>_Start	IIC の動作を許可します。
R_<Config_RIICn>_Stop	IIC の動作を禁止します。
R_<Config_RIICn>_Master_Send	マスタモードでデータ送信を行います。
R_<Config_RIICn>_Master_Send_Without_Stop	マスタ送信を開始します。（送信終了時にストップコンディションを発行しない）
R_<Config_RIICn>_Master_Receive	マスタモードでデータ受信を行います。
R_<Config_RIICn>_StartCondition	スタートコンディションを発行します。
R_<Config_RIICn>_StopCondition	ストップコンディションを発行します。
R_<Config_RIICn>_Create_UserInit	IIC マスタモードに関するユーザ独自の初期化処理を行います。
r_<Config_RIICn>_error_interrupt	通信エラー／通信イベント発生割り込みの発生に伴う処理を行います。
r_<Config_RIICn>_receive_interrupt	受信データフル割り込みの発生に伴う処理を行います。
r_<Config_RIICn>_transmit_interrupt	送信データエンプティ割り込みの発生に伴う処理を行います。
r_<Config_RIICn>_transmitend_interrupt	送信終了割り込みの発生に伴う処理を行います。
r_<Config_RIICn>_callback_transmitend	通信エラー／通信イベント発生割り込みに対応した割り込み処理のうち、マスタ送信に伴うストップコンディションの検出に特化した処理を行います。また、ストップコンディションを発行しないマスタ送信では、送信終了割り込み処理を行います。
r_<Config_RIICn>_callback_receiveend	通信イベント発生割り込みに対応した割り込み処理のうち、マスタ受信に伴うストップコンディションの検出に特化した処理を行います。
r_<Config_RIICn>_callback_receiveerror	通信エラー／通信イベント発生割り込みに対応した割り込み処理のうち、アービトレーションロスト検出、NACK 検出、タイムアウト検出に特化した処理を行います。

R_<Config_RIICn>_Create

IIC マスタモード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、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

void	R_<Config_RIICn>_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_<Config_RIICn>_Start

IIC の動作を許可します。

[指定形式]

void	R_<Config_RIICn>_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_<Config_RIICn>_Stop

IIC の動作を禁止します。

[指定形式]

void	R_<Config_RIICn>_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_<Config_RIICn>_Master_Send

マスタモードでデータ送信を行います。

備考 1. 本 API 関数では、データ（引数 *adr* で指定されたスレーブアドレスと R/W# ビット）をスレーブデバイスにマスタ送信したのち、引数 *tx_buf* で指定されたバッファから 1 バイト単位のマスタ送信を引数 *tx_num* で指定された回数だけ繰り返し行います。

備考 2. 本 API 関数では、マスタ送信の開始処理として、内部的に R_<Config_RIICn>_StartCondition の呼び出しを行っています。

備考 3. マスタ送信の終了処理として、r_<Config_RIICn>_transmit_interrupt にてストップコンディションを発行しています。

備考 4. マスタ送信を行う際には、本 API 関数の呼び出し以前に R_<Config_RIICn>_Start を呼び出す必要があります。

[指定形式]

```
MD_STATUS    R_<Config_RIICn>_Master_Send ( uint16_t adr, 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	uint16_t *adr*;	スレーブアドレス
I	uint8_t * const *tx_buf*;	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 *tx_num*;	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_ERROR1	バスビジー
MD_ERROR2	引数 *adr* の指定が不正

R_<Config_RIICn>_Master_Send_Without_Stop

マスタ送信を開始します。（送信終了時にストップコンディションを発行しない）

備考 1. 本 API 関数では、データ（引数 *adr* で指定されたスレーブアドレスと R/W# ビット）をスレーブデバイスにマスタ送信したのち、引数 *tx_buf* で指定されたバッファから 1 バイト単位のマスタ送信を引数 *tx_num* で指定された回数だけ繰り返し行います。

備考 2. 本 API 関数では、マスタ送信の開始処理として、内部的に R_<Config_RIICn>_StartCondition の呼び出しを行っています。

備考 3. マスタ送信の終了処理として、r_<Config_RIICn>_transmitend_interrupt にてストップコンディションを発行しません。

備考 4. マスタ送信を行う際には、本 API 関数の呼び出し以前に R_<Config_RIICn>_Start を呼び出す必要があります。

[指定形式]

```
MD_STATUS      R_<Config_RIICn>_Master_Send_Without_Stop ( uint16_t adr, uint8_t * const tx_buf, uint16_t tx_num );
```

[引数]

I/O	引数	説明
I	uint16_t *adr*;	スレーブアドレス
I	uint8_t * const *tx_buf*;	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 *tx_num*;	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_ERROR1	バスビジー
MD_ERROR2	引数 *adr* の指定が不正

R_<Config_RIICn>_Master_Receive

マスタモードでデータ受信を行います。

備考 1. 本 API 関数では、データ（引数 *adr* で指定されたスレーブアドレス）をスレーブデバイスにマスタ送信したのち、 1 バイト単位のマスタ受信を引数 *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。

備考 2. 本 API 関数では、マスタ受信の開始処理として、内部的に R_<Config_RIICn>_StartCondition の呼び出しを行っています。

備考 3. マスタ受信の終了処理として、r_<Config_RIICn>_receive_interrupt にてストップコンディションを発行しています。

備考 4. マスタ受信を行う際には、本 API 関数の呼び出し以前に R_<Config_RIICn>_Start を呼び出す必要があります。

[指定形式]

```
MD_STATUS R_<Config_RIICn>_Master_Receive ( uint16_t adr, 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
I	uint16_t *adr*;	スレーブアドレス
O	uint8_t * const *rx_buf*;	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 *rx_num*;	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_ERROR1	バスビジー
MD_ERROR2	引数 *adr* の指定が不正

R_<Config_RIICn>_StartCondition

スタートコンディションを発行します。

備考　本 API 関数の呼び出しに伴い、通信エラー／通信イベント発生割り込みを発生させ、r_<Config_RIICn>_error_interrupt が呼び出されます。

[指定形式]

```
void    R_<Config_RIICn>_StartCondition ( 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_<Config_RIICn>_StopCondition

ストップコンディションを発行します。

備考　本 API 関数の呼び出しに伴い、通信エラー／通信イベント発生割り込みを発生させ、r_<Config_RIICn>_error_interrupt が呼び出されます。

[指定形式]

```
void    R_<Config_RIICn>_StopCondition ( 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_<Config_RIICn>_Create_UserInit

IIC マスタモード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_<Config_RIICn>_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_<Config_RIICn>_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_<Config_RIICn>_error_interrupt

通信エラー／通信イベント発生割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_RIICn>_error_interrupt (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_<Config_RIICn>_receive_interrupt

受信データフル割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_RIICn>_receive_interrupt (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_<Config_RIICn>_transmit_interrupt

送信データエンプティ割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_RIICn>_transmit_interrupt (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_<Config_RIICn>_transmitend_interrupt

送信終了割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_RIICn>_transmitend_interrupt (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_<Config_RIICn>_callback_transmitend

通信エラー／通信イベント発生割り込みに対応した割り込み処理のうち、マスタ送信に伴うストップコンディションの検出に特化した処理を行います。

備考 1. 本 API 関数は、r_<Config_RIICn>_error_interrupt のコールバック・ルーチンとして呼び出されます。

備考 2. マスタ送信を行う際には、R_<Config_RIICn>_Master_Send を呼び出してください。

また、ストップコンディションを発行しないマスタ送信では、送信終了割り込み処理を行います。

備考 3. 本 API 関数は、r_<Config_RIICn>_transmit_interrupt のコールバック・ルーチンとして呼び出されます。

備考 4. マスタ送信を行う際には、R_<Config_RIICn>_Master_Send_Without_Stop を呼び出してください。

[指定形式]

void	R_<Config_RIICn>_callback_transmitend (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_<Config_RIICn>_callback_receiveend

通信エラー／通信イベント発生割り込みに対応した割り込み処理のうち、マスタ受信に伴うストップコンディションの検出に特化した処理を行います。

備考 1. 本 API 関数は、r_<Config_RIICn>_error_interrupt のコールバック・ルーチンとして呼び出されます。

備考 2. マスタ受信を行う際には、R_<Config_RIICn>_Master_Receive を呼び出してください。

[指定形式]

void	R_<Config_RIICn>_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_<Config_RIICn>_callback_receiveerror

通信エラー／通信イベント発生割り込みに対応した割り込み処理のうち、アービトレーションロスト検出、NACK 検出、タイムアウト検出に特化した処理を行います。

備考　本 API 関数は、r_<Config_RIICn>_error_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]

void	R_<Config_RIICn>_callback_receiveerror (MD_STATUS *status*);

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	MD_STATUS *status*;	割り込みの発生要因 MD_ERROR1　：アービトレーションロスト検出 MD_ERROR2　：タイムアウト検出 MD_ERROR3　：NACK 検出

[戻り値]

なし

使用例

RIIC0 マスタ、RIIC1 をスレーブとしてデータを送受信する場合（青字はユーザコード）

r_cg_main.c

```
/* Start user code for global. Do not edit comment generated here */
uint8_t tx_buf[6] = {0xAA, 0xFF, 0x00, 0x55, 0xCC, 0x33};
uint8_t rx_buf[6] = {0};
volatile uint8_t riic0_master_sendend_flag = 0x00U;
volatile uint8_t riic1_slave_receiveend_flag = 0x00U;
/* End user code. Do not edit comment generated here */

void main(void)
{
    r_main_userinit();
    /* Start user code for main. Do not edit comment generated here */
    R_Config_RIIC1_Start();
    R_Config_RIIC0_Start();

    R_Config_RIIC1_Slave_Receive(rx_buf, 6);
    R_Config_RIIC0_Master_Send(0x00, rx_buf, 6);

    while(riic0_master_sendend_flag == 0 || riic0_slave_reveiveend_flag == 0);

    while(1);
    /* End user code. Do not edit comment generated here */
}
```

Config_RIIC0_user.c

```
/* Start user code for global. Do not edit comment generated here */
extern volatile uint8_t riic0_master_sendend_flag;
/* End user code. Do not edit comment generated here */

void r_Config_RIIC0_callback_transmitend(void)
{
    /* Start user code for r_Config_RIIC0_callback_transmitend. Do not edit comment generated here */
    riic0_master_sendend_flag = 1;
    /* End user code. Do not edit comment generated here */
}
```

Config_RIIC1_user.c

```
/* Start user code for global. Do not edit comment generated here */
extern volatile uint8_t riic1_slave_receiveend_flag;
/* End user code. Do not edit comment generated here */

void r_Config_RIIC1_callback_receiveend(void)
{
    /* Start user code for r_Config_RIIC1_callback_receiveend. Do not edit comment generated here */
    riic1_slave_receiveend_flag = 1;
    /* End user code. Do not edit comment generated here */
}
```

3.2.33 IIC スレーブモード

以下に、コード生成ツールが IIC スレーブモード用に出力する API 関数の一覧を示します。

表 3-33 IIC スレーブモード用 API 関数

API 関数名	機能概要
R_<Config_RIICn>_Create	IIC スレーブモードを制御するうえで必要となる初期化処理を行います。
R_<Config_RIICn>_Start	IIC の動作を許可します。
R_<Config_RIICn>_Stop	IIC の動作を禁止します。
R_<Config_RIICn>_Slave_Send	スレーブモードでデータ送信を行います。
R_<Config_RIICn>_Slave_Receive	スレーブモードでデータ受信を行います。
R_<Config_RIICn>_StartCondition	スタートコンディションを発行します。
R_<Config_RIICn>_StopCondition	ストップコンディションを発行します。
R_<Config_RIICn>_Create_UserInit	IIC スレーブモードに関するユーザ独自の初期化処理を行います。
R_<Config_RIICn>_error_interrupt	通信エラー／通信イベント発生割り込みの発生に伴う処理を行います。
R_<Config_RIICn>_receive_interrupt	受信データフル割り込みの発生に伴う処理を行います。
R_<Config_RIICn>_transmit_interrupt	送信データエンプティ割り込みの発生に伴う処理を行います。
R_<Config_RIICn>_transmitend_interrupt	送信終了割り込みの発生に伴う処理を行います。
R_<Config_RIICn>_callback_transmitend	通信エラー／通信イベント発生割り込みに対応した割り込み処理のうち、スレーブ受信に伴うストップコンディションの検出に特化した処理を行います。
R_<Config_RIICn>_callback_receiveend	通信エラー／通信イベント発生割り込みに対応した割り込み処理のうち、スレーブ送信に伴うストップコンディションの検出に特化した処理を行います。
R_<Config_RIICn>_callback_receiveerror	通信エラー／通信イベント発生割り込みに対応した割り込み処理のうち、アービトレーションロスト検出、NACK 検出、タイムアウト検出に特化した処理を行います。

R_<Config_RIICn>_Create

IIC スレーブモード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、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

void	R_<Config_RIICn>_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_<Config_RIICn>_Start

IIC の動作を許可します。

[指定形式]

void	R_<Config_RIICn>_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_<Config_RIICn>_Stop

IIC の動作を禁止します。

[指定形式]

void	R_<Config_RIICn>_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_<Config_RIICn>_Slave_Send

スレーブモードでデータ送信を行います。

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。

備考 2. スレーブ送信を行う際には、本 API 関数の呼び出し以前に R_<Config_RIICn>_Start を呼び出す必要があります。

[指定形式]

```
MD_STATUS      R_<Config_RIICn>_Slave_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 *tx_buf*;	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 *tx_num*;	送信するデータの総数

[戻り値]

マクロ	説明
MD_OK	正常終了

R_<Config_RIICn>_Slave_Receive

スレーブモードでデータ受信を行います。

備考 1. 本 API 関数では、1 バイト単位の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。

備考 2. スレーブ受信を行う際には、本 API 関数の呼び出し以前に R_<Config_RIICn>_Start を呼び出す必要があります。

[指定形式]

```
MD_STATUS      R_<Config_RIICn>_Slave_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 *rx_buf*;	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 *rx_num*;	受信するデータの総数

[戻り値]

マクロ	説明
MD_OK	正常終了

R_<Config_RIICn>_StartCondition

スタートコンディションを発行します。

備考　本 API 関数の呼び出しに伴い、通信エラー／通信イベント発生割り込みを発生させ、R_<Config_RIICn>_error_interrupt が呼び出されます。

[指定形式]

```
void    R_<Config_RIICn>_StartCondition ( 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_<Config_RIICn>_StopCondition

ストップコンディションを発行します。

備考　本 API 関数の呼び出しに伴い、通信エラー／通信イベント発生割り込みを発生させ、R_<Config_RIICn>_error_interrupt が呼び出されます。

[指定形式]

```
void    R_<Config_RIICn>_StopCondition ( 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_<Config_RIICn>_Create_UserInit

IIC スレーブモード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_<Config_RIICn>_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_<Config_RIICn>_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_<Config_RIICn>_error_interrupt

通信エラー／通信イベント発生割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_RIICn>_error_interrupt (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_<Config_RIICn>_receive_interrupt

受信データフル割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_RIICn>_receive_interrupt (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_<Config_RIICn>_transmit_interrupt

送信データエンプティ割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_RIICn>_transmit_interrupt (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_<Config_RIICn>_transmitend_interrupt

送信終了割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_RIICn>_transmitend_interrupt (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_<Config_RIICn>_callback_transmitend

通信エラー／通信イベント発生割り込みに対応した割り込み処理のうち、スレーブ送信に伴うストップコンディションの検出に特化した処理を行います。

備考 1. 本 API 関数は、R_<Config_RIICn>_error_interrupt のコールバック・ルーチンとして呼び出されます。

備考 2. スレーブ送信を行う際には、R_<Config_RIICn>_Slave_Send を呼び出してください。

[指定形式]

```
void    R_<Config_RIICn>_callback_transmitend ( 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_<Config_RIICn>_callback_receiveend

通信エラー／通信イベント発生割り込みに対応した割り込み処理のうち、スレーブ受信に伴うストップコンディションの検出に特化した処理を行います。

備考 1. 本 API 関数は、R_<Config_RIICn>_error_interrupt のコールバック・ルーチンとして呼び出されます。

備考 2. スレーブ受信を行う際には、R_<Config_RIICn>_Slave_Receive を呼び出してください。

[指定形式]

void	R_<Config_RIICn>_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_<Config_RIICn>_callback_receiveerror

通信エラー／通信イベント発生割り込みに対応した割り込み処理のうち、アービトレーションロスト検出、NACK 検出、タイムアウト検出に特化した処理を行います。

備考　本 API 関数は、R_<Config_RIICn>_error_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]

void	R_<Config_RIICn>_callback_receiveerror (MD_STATUS *status*);

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	MD_STATUS *status*;	割り込みの発生要因 MD_ERROR1　：アービトレーションロスト検出 MD_ERROR2　：タイムアウト検出 MD_ERROR3　：NACK 検出

[戻り値]

なし

使用例

IIC マスタモードの使用例を参照してください。

3.2.34 SCI3 クロック同期式モード

以下に、コード生成ツールが SCI3 クロック同期式モード用に出力する API 関数の一覧を示します。

表 3-34 SCI3 クロック同期式モード用 API 関数

API 関数名	機能概要
R_<Config_SCI3n>_Create	SCI3*n* のクロック同期式モードを制御するうえで必要となる初期化処理を行います。
R_<Config_SCI3n>_Start	SCI3*n* の動作を許可します。
R_<Config_SCI3n>_Stop	SCI3*n* の動作を禁止します。
R_<Config_SCI3n>_Send	SCI3*n* のクロック同期式モードでデータ送信を行います。
R_<Config_SCI3n>_Receive	SCI3*n* のクロック同期式モードでデータ受信を行います。
R_<Config_SCI3n>_Create_UserInit	SCI3*n* のクロック同期式モードに関するユーザ独自の初期化処理を行います。
r_<Config_SCI3n>_interrupt_send	SCI3*n* の送信割り込みの発生に伴う処理を行います。
r_<Config_SCI3n>_interrupt_receive	SCI3*n* の受信割り込みの発生に伴う処理を行います。
r_<Config_SCI3n>_interrupt_error	SCI3*n* のエラー割り込みの発生に伴う処理を行います。
r_<Config_SCI3n>_interrupt_sendend	SCI3*n* の送信完了割り込みの発生に伴う処理を行います。
r_<Config_SCI3n>_callback_sendend	SCI3*n* の送信完了割り込みの発生に伴う処理を行います。
r_<Config_SCI3n>_callback_receiveend	SCI3*n* の受信割り込みの発生に伴う処理を行います。
r_<Config_SCI3n>_callback_error	SCI3*n* のエラー割り込みの発生に伴う処理を行います。

R_<Config_SCI3n>_Create

SCI3 クロック同期式モード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、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

void	R_<Config_SCI3n>_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_<Config_SCI3n>_Start

SCI3*n* の動作を許可します。

[指定形式]

void	R_<Config_SCI3n>_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_<Config_SCI3n>_Stop

SCI3*n* の動作を禁止します。

[指定形式]

void	R_<Config_SCI3n>_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_<Config_SCI3n>_Send

SCI3*n* のクロック同期式モードでデータ送信を行います。

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データ送信を引数 *tx_num* で指定された回数だけ繰り返し行います。

備考 2. データ送信を行う際には、本 API 関数の呼び出し以前に R_<Config_SCI3n>_Start を呼び出す必要があります。

[指定形式]

```
MD_STATUS      R_<Config_SCI3n>_Send ( uint8_t * const tx_buf, uint16_t tx_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
I	uint8_t* const *tx_buf*	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 *tx_num*	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_<Config_SCI3n>_Receive

SCI3*n* のクロック同期式モードでデータ受信を行います。

備考 1. 本 API 関数では、データ受信を引数 *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。

備考 2. データ受信を行う際には、本 API 関数の呼び出し以前に R_<Config_SCI3n>_Start を呼び出す必要があります。

[指定形式]

```
MD_STATUS      R_<Config_SCI3n>_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 *rx_buf*	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 *rx_num*	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_<Config_SCI3n>_Create_UserInit

SCI3*n* のクロック同期式モード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_<Config_SCI3n>_Start 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_<Config_SCI3n>_Create_UserInit (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_<Config_SCI3n>_interrupt_send

SCI3*n* の送信割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_SCI3n>_interrupt_send (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_<Config_SCI3n>_interrupt_receive

SCI3*n* の受信割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_SCI3n>_interrupt_receive (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_<Config_SCI3n>_interrupt_error

SCI3*n* のエラー割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_SCI3n>_interrupt_error (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_<Config_SCI3n>_interrupt_sendend

SCI3*n* の送信完了割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_SCI3n>_interrupt_sendend (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_<Config_SCI3n>_callback_sendend

SCI3*n* の送信完了割り込みの発生に伴う処理を行います。

備考 1. 本 API 関数は、r_<Config_SCI3n>_interrupt_sendend 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_<Config_SCI3n>_callback_sendend (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_<Config_SCI3n>_callback_receiveend

SCI3*n* の受信割り込みの発生に伴う処理を行います。

備考 1. 本 API 関数は、r_<Config_SCI3n>_interrupt_receive 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_<Config_SCI3n>_callback_receiveend (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_<Config_SCI3n>_callback_error

SCI3*n* のエラー割り込みの発生に伴う処理を行います。

備考　本 API 関数は、r_<Config_SCI3n>_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]

void	R_<Config_SCI3n>_callback_error (uint32_t err_type);

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	uint32_t err_type	エラー割り込みの発生要因 0xxxxxxxB　：送信データエンプティ x1xxxxxxB　：受信データフル xx1xxxxxB　：オーバラン・エラー xxx1xxxxB　：フレーミング・エラー xxxx1xxxB　：パリティ・エラー

[戻り値]

なし

使用例

SCI30 を用いてクロック同期式モードで送受信する場合（青字はユーザコード）

r_cg_main.c

```
/* Start user code for global. Do not edit comment generated here */
uint8_t tx_buf0[6] = {0xAA, 0xFF, 0x00, 0x55, 0xCC, 0x33};
uint8_t tx_buf1[6] = {0xAA, 0xFF, 0x00, 0x55, 0xCC, 0x33};
uint8_t rx_buf0[6] = {0x88, 0x00, 0x00, 0x00, 0x00, 0x88};
uint8_t rx_buf1[6] = {0x88, 0x00, 0x00, 0x00, 0x00, 0x88};
/* End user code. Do not edit comment generated here */

void main(void)
{
    r_main_userinit();
    /* Start user code for main. Do not edit comment generated here */
    R_Config_SCI30_Start();

    R_Config_SCI30_Receive(rx_buf0, 6);
    R_Config_SCI30_Send(tx_buf0, 6);
    while(receiveend_flag != 1);
    receiveend_flag = 0;
    while(transmitend_flag != 1);
    transmitend_flag = 0;

    R_Config_SCI30_Receive(rx_buf1, 6);
    R_Config_SCI30_Send(tx_buf1, 6);
    while(receiveend_flag != 1);
    receiveend_flag = 0;
    while(transmitend_flag != 1);
    transmitend_flag = 0;

    R_Config_SCI30_Stop();
    while(1);
    /* End user code. Do not edit comment generated here */
}
```

Config_SCI30_user.c

```
/* Start user code for global. Do not edit comment generated here */
volatile uint8_t transmitend_flag = 0;
volatile uint8_t receiveend_flag = 0;
/* End user code. Do not edit comment generated here */

void r_Config_SCI30_callback_sendend(void)
{
    /* Start user code for r_Config_SCI30_callback_sendend. Do not edit comment generated here */
    transmitend_flag = 1;
    /* End user code. Do not edit comment generated here */
}

void r_Config_SCI30_callback_receiveend(void)
{
    /* Start user code for r_Config_SCI30_callback_receiveend. Do not edit comment generated here */
    receiveend_flag = 1;
    /* End user code. Do not edit comment generated here */
}
```

3.2.35 SCI3 調歩同期式モード

以下に、コード生成ツールが SCI3 調歩同期式モード用に出力する API 関数の一覧を示します。

表 3-35 SCI3 調歩同期式モード用 API 関数

API 関数名	機能概要
R_<Config_SCI3n>_Create	SCI3*n* の調歩同期式モードを制御するうえで必要となる初期化処理を行います。
R_<Config_SCI3n>_Start	SCI3*n* の動作を許可します。
R_<Config_SCI3n>_Stop	SCI3*n* の動作を禁止します。
R_<Config_SCI3n>_Send	SCI3*n* の調歩同期式モードでデータ送信を行います。
R_<Config_SCI3n>_Receive	SCI3*n* の調歩同期式モードでデータ受信を行います。
R_<Config_SCI3n>_Multiprocessor_Send	SCI3*n* の調歩同期式モードでマルチプロセッサデータ送信を行います。
R_<Config_SCI3n>_Multiprocessor_Receive	SCI3*n* の調歩同期式モードでマルチプロセッサデータ受信を行います。
R_<Config_SCI3n>_Create_UserInit	SCI3*n* の調歩同期式モードに関するユーザ独自の初期化処理を行います。
r_<Config_SCI3n>_interrupt_send	SCI3*n* の送信割り込みの発生に伴う処理を行います。
r_<Config_SCI3n>_interrupt_receive	SCI3*n* の受信割り込みの発生に伴う処理を行います。
r_<Config_SCI3n>_interrupt_error	SCI3*n* のエラー割り込みの発生に伴う処理を行います。
r_<Config_SCI3n>_interrupt_sendend	SCI3*n* の送信完了割り込みの発生に伴う処理を行います。
r_<Config_SCI3n>_callback_sendend	SCI3*n* の送信完了割り込みの発生に伴う処理を行います。
r_<Config_SCI3n>_callback_receiveend	SCI3*n* の受信割り込みの発生に伴う処理を行います。
r_<Config_SCI3n>_callback_error	SCI3*n* のエラー割り込みの発生に伴う処理を行います。

R_<Config_SCI3n>_Create

SCI3 調歩同期式モード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、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

void	R_<Config_SCI3n>_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_<Config_SCI3n>_Start

SCI3*n* の動作を許可します。

[指定形式]

void	R_<Config_SCI3n>_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_<Config_SCI3n>_Stop

SCI3*n* の動作を禁止します。

[指定形式]

void	R_<Config_SCI3n>_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_<Config_SCI3n>_Send

SCI3*n* の調歩同期式モードでデータ送信を行います。

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データ送信を引数 *tx_num* で指定された回数だけ繰り返し行います。

備考 2. データ送信を行う際には、本 API 関数の呼び出し以前に R_<Config_SCI3n>_Start を呼び出す必要があります。

[指定形式]

```
MD_STATUS      R_<Config_SCI3n>_Send ( uint8_t * const tx_buf, uint16_t tx_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
I	uint8_ const t* *tx_buf*	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 *tx_num*	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_<Config_SCI3n>_Receive

SCI3*n* の調歩同期式モードでデータ受信を行います。

備考 1. 本 API 関数では、データ受信を引数 *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。

備考 2. データ受信を行う際には、本 API 関数の呼び出し以前に R_<Config_SCI3n>_Start を呼び出す必要があります。

[指定形式]

MD_STATUS R_<Config_SCI3n>_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 *rx_buf*	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 *rx_num*	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_<Config_SCI3n>_Multiprocessor_Send

SCI3*n* の調歩同期式モードでマルチプロセッサデータ送信を行います。

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データ送信を引数 *tx_num* で指定された回数だけ繰り返し行います。

備考 2. データ送信を行う際には、本 API 関数の呼び出し以前に R_<Config_SCI3n>_Start を呼び出す必要があります。

[指定形式]

```
MD_STATUS R_<Config_SCI3n>_Multiprocessor_Send (uint8_t* const tx_buf, uint16_t tx_num, uint8_t tx_id );
```

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* const *tx_buf*	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 *tx_num*	送信するデータの総数
I	uint8_t *tx_id*	送信 ID

[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_<Config_SCI3n>_Multiprocessor_Receive

SCI3*n* の調歩同期式モードでマルチプロセッサデータ受信を行います。

備考 1. 本 API 関数では、データ受信を引数 *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。

備考 2. データ受信を行う際には、本 API 関数の呼び出し以前に R_<Config_SCI3n>_Start を呼び出す必要があります。

[指定形式]

```
MD_STATUS R_<Config_SCI3n>_Multiprocessor_Receive (uint8_t* const rx_buf, uint16_t rx_num, uint8_t rx_id );
```

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 *rx_buf*	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 *rx_num*	受信するデータの総数
I	uint8_t *rx_id*	受信 ID

[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_<Config_SCI3n>_Create_UserInit

SCI3*n* の調歩同期式モード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_<Config_SCI3n>_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_<Config_SCI3n>_Create_UserInit ( 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_<Config_SCI3n>_interrupt_send

SCI3*n* の送信割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_SCI3n>_interrupt_send (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_<Config_SCI3n>_interrupt_receive

SCI3*n* の受信割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_SCI3n>_interrupt_receive (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_<Config_SCI3n>_interrupt_error

SCI3*n* のエラー割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_SCI3n>_interrupt_error (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_<Config_SCI3n>_interrupt_sendend

SCI3*n* の送信完了割り込みの発生に伴う処理を行います。

[指定形式]

void	R_<Config_SCI3n>_interrupt_sendend (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_<Config_SCI3n>_callback_sendend

SCI3*n* の送信完了割り込みの発生に伴う処理を行います。

備考 1. 本 API 関数は、r_<Config_SCI3n>_interrupt_sendend 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_<Config_SCI3n>_callback_sendend ( 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_<Config_SCI3n>_callback_receiveend

SCI3*n* の受信割り込みの発生に伴う処理を行います。

備考 1. 本 API 関数は、r_<Config_SCI3n>_interrupt_receive 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_<Config_SCI3n>_callback_receiveend ( 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_<Config_SCI3n>_callback_error

SCI3*n* のエラー割り込みの発生に伴う処理を行います。

備考　本 API 関数は、r_<Config_SCI3n>_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]

void	R_<Config_SCI3n>_callback_error (uint32_t err_type);

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	uint32_t err_type	エラー割り込みの発生要因 0xxxxxxxB　：送信データエンプティ x1xxxxxxB　：受信データフル xx1xxxxxB　：オーバラン・エラー xxx1xxxxB　：フレーミング・エラー xxxx1xxxB　：パリティ・エラー

[戻り値]

なし

使用例

SCI30 を用いて調歩同期式モードで送受信する場合（青字はユーザコード）

r_cg_main.c

```
/* Start user code for global. Do not edit comment generated here */
uint8_t tx_buf[6] = {0xAA, 0x55, 0x00, 0xFF, 0xCC, 0x33};
uint8_t rx_buf[6] = {0x88, 0x0, 0x00, 0x00, 0x00, 0x88};
/* End user code. Do not edit comment generated here */

void main(void)
{
    r_main_userinit();
    /* Start user code for main. Do not edit comment generated here */
    R_Config_SCI30_Start();
    R_Config_SCI30_Send(tx_buf, 6);
    while(transmitend_flag != 1);
    transmitend_flag = 0;

    R_Config_SCI30_Receive(rx_buf, 6);
    while(receiveendend_flag != 1);
    receiveendend_flag = 0;
    while(1);
    /* End user code. Do not edit comment generated here */
}
```

Config_SCI30_user.c

```
/* Start user code for global. Do not edit comment generated here */
volatile uint8_t transmitend_flag = 0;
volatile uint8_t receiveendend_flag = 0;
/* End user code. Do not edit comment generated here */

void r_Config_SCI30_callback_sendend(void)
{
    /* Start user code for r_Config_SCI30_callback_sendend. Do not edit comment generated here */
    transmitend_flag = 1;
    /* End user code. Do not edit comment generated here */
}

void r_Config_SCI30_callback_receiveend(void)
{
    /* Start user code for r_Config_SCI30_callback_receiveend. Do not edit comment generated here */
    receiveendend_flag = 1;
    /* End user code. Do not edit comment generated here */
}
```

3.2.36 MSPI マスタモード

以下に、コード生成ツールが MSPI マスタモード用に出力する API 関数の一覧を示します。

表 3-36 MSPI マスタモード用 API 関数

API 関数名	機能概要
R_<Config_MSPInm>_Create	MSPI*nm* のマスタモードを制御するうえで必要となる初期化処理を行います。
R_<Config_MSPInm>_Start	MSPI*nm* の動作を許可します。
R_<Config_MSPInm>_Stop	MSPI*nm* の動作を禁止します。
R_<Config_MSPInm>_Send	MSPI*nm* のマスタモードでデータ送信を行います。
R_<Config_MSPInm>_Receive	MSPI*nm* のマスタモードでデータ受信を行います。
R_<Config_MSPInm>_Software_Trigger	MSPInm のチャネルを有効にし、開始トリガを設定します。
R_<Config_MSPInm>_Create_UserInit	MSPI*nm* のマスタモードに関するユーザ独自の初期化処理を行います。
r_<Config_MSPInm>_channelnm_interrupt_send	MSPI*nm* の送信割り込みの発生に伴う処理を行います。
r_<Config_MSPInm>_channelnm_interrupt_receive	MSPI*nm* の受信割り込みの発生に伴う処理を行います。
r_<Config_MSPInm>_Callback_Interrupt_Send	MSPI*nm* の送信割り込みの発生に伴う処理を行います。
r_<Config_MSPInm>_Callback_Interrupt_Receive	MSPI*nm* の受信割り込みの発生に伴う処理を行います。
r_<Config_MSPInm>_Callback_Interrupt_Error	MSPI*nm* のエラー割り込みの発生に伴う処理を行います。
r_<Config_MSPInm>_Callback_Interrupt_Frameend	MSPI*nm* のフレーム完了割り込みの発生に伴う処理を行います。
r_<Config_MSPInm>_callback_sendend	MSPI*nm* の送信割り込みの発生に伴う処理を行います。
r_<Config_MSPInm>_callback_receiveend	MSPI*nm* の受信割り込みの発生に伴う処理を行います。
r_<Config_MSPInm>_callback_error	MSPI*nm* のエラー割り込みの発生に伴う処理を行います。

R_<Config_MSPInm>_Create

MSPI マスタモード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、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

```
void    R_<Config_MSPInm>_Create ( void );
```

備考　*n* はユニット番号、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_MSPInm>_Start

MSPI の動作を許可します。

[指定形式]

void	R_<Config_MSPInm>_Start (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_MSPInm>_Stop

MSPI の動作を禁止します。

[指定形式]

void	R_<Config_MSPInm>_Stop (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_MSPInm>_Send

MSPI マスタモードでデータ送信を行います。

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マスタ送信をフレーム数で指定された回数だけ繰り返し行います。

備考 2. データ送信を行う際には、本 API 関数の呼び出し以前に R_<Config_MSPInm>_Start を呼び出す必要があります。

[指定形式]

```
MD_STATUS    R_<Config_MSPInm>_Send ( const uint8_t * tx_buf );
MD_STATUS    R_<Config_MSPInm>_Send ( const uint16_t * tx_buf );
MD_STATUS    R_<Config_MSPInm>_Send ( const uint32_t * tx_buf );
```

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

I/O	引数	説明
I	const uint8_t * *tx_buf*	送信するデータを格納したバッファへのポインタ(1 バイト)
I	const uint16_t * *tx_buf*	送信するデータを格納したバッファへのポインタ(2 バイト)
I	const uint32_t * *tx_buf*	送信するデータを格納したバッファへのポインタ(4 バイト)

[戻り値]

マクロ	説明
MD_OK	正常終了

R_<Config_MSPInm>_Receive

MSPI マスタモードでデータ受信を行います。

備考 1. 本 API 関数では、マスタ受信をフレーム数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. データ受信を行う際には、本 API 関数の呼び出し以前に R_<Config_MSPInm>_Start を呼び出す必要があります。

[指定形式]

```
MD_STATUS    R_<Config_MSPInm>_Receive ( uint8_t * rx_buf );
MD_STATUS    R_<Config_MSPInm>_Receive ( uint16_t * rx_buf );
MD_STATUS    R_<Config_MSPInm>_Receive ( uint32_t * rx_buf );
```

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

I/O	引数	説明
O	uint8_t * *rx_buf*	受信したデータを格納するバッファへのポインタ(1 バイト)
O	uint16_t * *rx_buf*	受信したデータを格納するバッファへのポインタ(2 バイト)
O	uint32_t * *rx_buf*	受信したデータを格納するバッファへのポインタ(4 バイト)

[戻り値]

マクロ	説明
MD_OK	正常終了

R_<Config_MSPInm>_Software_Trigger

チャネルを有効にし、開始トリガを設定します

[指定形式]

void	R_<Config_MSPInm>_Software_Trigger (void);

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_MSPInm>_Create_UserInit

MSPI マスタモード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_<Config_MSPInm>_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_<Config_MSPInm>_Create_UserInit (void);

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_MSPInm>_channel*nm*_interrupt_send

MSPI*nm* の送信割り込みの発生に伴う処理を行います。

[指定形式]

void	r_<Config_MSPInm>_channel*nm*_interrupt_send (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_MSPInm>_channel*nm*_interrupt_receive

MSPI*nm* の受信割り込みの発生に伴う処理を行います。

[指定形式]

void	r_<Config_MSPInm>_channel*nm*_interrupt_receive (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_MSPI*nm*>_Callback_Interrupt_Send

MSPI*nm* の送信割り込みの発生に伴う処理を行います。

備考 1. 本 API 関数は、r_mspin_interrupt_send 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_<Config_MSPInm>_Callback_Interrupt_Send(void);
```

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_MSPI*nm*>_Callback_Interrupt_Receive

MSPI*nm* の受信割り込みの発生に伴う処理を行います。

備考 1. 本 API 関数は、r_mspin_interrupt_receive 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_<Config_MSPInm>_Callback_Interrupt_Receive(void);
```

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_MSPI*nm*>_Callback_Interrupt_Error

MSPI*nm* のエラー割り込みの発生に伴う処理を行います。

備考 1. 本 API 関数は、r_mspin_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]

```
void    r_<Config_MSPInm>_Callback_Interrupt_Error(void);
```

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_MSPI*nm*>_Callback_Interrupt_Frameend

MSPI*nm* のフレーム完了割り込みの発生に伴う処理を行います。

備考 1. 本 API 関数は、r_mspin_interrupt_frameend 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_<Config_MSPInm>_callback_frameend(void);
```

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_MSPInm>_callback_sendend

MSPI*nm* の送信割り込みの発生に伴う処理を行います。

備考 1. 本 API 関数は、r_<Config_MSPInm>_Callback_Interrupt_Send 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_<Config_MSPInm>_callback_sendend ( void );
```

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_MSPInm>_callback_receiveend

MSPI*nm* の受信割り込みの発生に伴う処理を行います。

備考 1. 本 API 関数は、r_<Config_MSPInm>_Callback_Interrupt_Receive 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_<Config_MSPInm>_callback_receiveend ( void );
```

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_MSPInm>_callback_error

MSPI*nm* のエラー割り込みの発生に伴う処理を行います。

備考　本 API 関数は、r_<Config_MSPInm>_Callback_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]

```
void    r_<Config_MSPInm>_callback_error ( uint32_t err_type );
```

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

I/O	引数	説明
I	uint32_t err_type	エラー割り込みの発生要因 000100xxB　：送信データの整合性エラー 000x001xB　：受信データの CRC エラー 000x00x1B　：受信データのパリティ・エラー

[戻り値]

なし

使用例

MSPI00 をマスタ、MSPI1 をスレーブとしてデータを送受信する場合（青字はユーザコード）

r_cg_main.c

```
/* Start user code for global. Do not edit comment generated here */
uint8_t transmitend_flag = 0;
uint8_t receiveend_flag = 0;

uint32_t tx_buf[4] = {0x12345678, 0x9abcdef0, 0xfedcba98, 0x76543210};
uint32_t rx_buf[4] = {0};
/* End user code. Do not edit comment generated here */

void main(void)
{
    r_main_userinit();
    /* Start user code for main. Do not edit comment generated here */
    R_Config_MSPI00_Start();
    R_MSPI_Start_Interrupt_MSPI0TX();
    R_MSPI_Start_Interrupt_MSPI0FE();
    R_MSPI_Start_Interrupt_MSPI0ERR();
    R_Config_MSPI1_Start();

    R_Config_MSPI1_Receive(rx_buf);
    R_Config_MSPI00_Send(tx_buf);

    R_Config_MSPI1_Software_Trigger();
    R_Config_MSPI00_Software_Trigger();
    while(1);
    /* End user code. Do not edit comment generated here */
}
```

3.2.37 MSPI スレーブモード

以下に、コード生成ツールが MSPI スレーブモード用に出力する API 関数の一覧を示します。

表 3-37 MSPI スレーブモード用 API 関数

API 関数名	機能概要
R_<Config_MSPIn>_Create	MSPI スレーブモードを制御するうえで必要となる初期化処理を行います。
R_<Config_MSPIn>_Start	MSPI の動作を許可します。
R_<Config_MSPIn>_Stop	MSPI の動作を禁止します。
R_<Config_MSPIn>_Send	MSPI スレーブモードでデータ送信を行います。
R_<Config_MSPIn>_Receive	MSPI スレーブモードでデータ受信を行います。
R_<Config_MSPIn>_Software_Trigger	MSPInm のチャネルを有効にし、開始トリガを設定します。
R_<Config_MSPIn>_Create_UserInit	MSPI スレーブモードに関するユーザ独自の初期化処理を行います。
r_<Config_MSPIn>_channeln0_interrupt_send	MSPIn のチャネル 0 の送信割り込みの発生に伴う処理を行います。
r_<Config_MSPIn>_channeln0_interrupt_receive	MSPIn のチャネル 0 の受信割り込みの発生に伴う処理を行います。
r_<Config_MSPIn>_interrupt_send	MSPIn のスレーブ送信割り込みの発生に伴う処理を行います。
r_<Config_MSPIn>_interrupt_receive	MSPIn のスレーブ受信割り込みの発生に伴う処理を行います。
r_<Config_MSPIn>_interrupt_error	MSPIn のエラー割り込みの発生に伴う処理を行います。
r_<Config_MSPIn>_interrupt_frameend	MSPIn のフレーム完了割り込みの発生に伴う処理を行います。
r_<Config_MSPIn>_callback_sendend	MSPIn のスレーブ送信割り込みの発生に伴う処理を行います。
r_<Config_MSPIn>_callback_receiveend	MSPIn のスレーブ受信割り込みの発生に伴う処理を行います。
r_<Config_MSPIn>_callback_error	MSPIn のエラー割り込みの発生に伴う処理を行います。

R_<Config_MSPIn>_Create

MSPI スレーブモード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、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

void	R_<Config_MSPIn>_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_<Config_MSPIn>_Start

MSPI の動作を許可します。

[指定形式]

void	R_<Config_MSPIn>_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_<Config_MSPIn>_Stop

MSPI の動作を禁止します。

[指定形式]

void	R_<Config_MSPIn>_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_<Config_MSPIn>_Send

MSPI スレーブモードでデータ送信を行います。

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スレーブ送信をフレーム数で指定された回数だけ繰り返し行います。

備考 2. データ送信を行う際には、本 API 関数の呼び出し以前に R_<Config_MSPIn>_Start を呼び出す必要があります。

[指定形式]

```
MD_STATUS    R_<Config_MSPIn>_Send ( const uint8_t * tx_buf );
MD_STATUS    R_<Config_MSPIn>_Send ( const uint16_t * tx_buf );
MD_STATUS    R_<Config_MSPIn>_Send ( const uint32_t * tx_buf );
```

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	const uint8_t * *tx_buf*	送信するデータを格納したバッファへのポインタ(1 バイト)
I	const uint16_t * *tx_buf*	送信するデータを格納したバッファへのポインタ(2 バイト)
I	const uint32_t * *tx_buf*	送信するデータを格納したバッファへのポインタ(4 バイト)

[戻り値]

マクロ	説明
MD_OK	正常終了

R_<Config_MSPIn>_Receive

MSPI スレーブモードでデータ受信を行います。

備考 1. 本 API 関数では、スレーブ受信をフレーム数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. データ受信を行う際には、本 API 関数の呼び出し以前に R_<Config_MSPIn>_Start を呼び出す必要があります。

[指定形式]

```
MD_STATUS    R_<Config_MSPIn>_Receive ( uint8_t * rx_buf );
MD_STATUS    R_<Config_MSPIn>_Receive ( uint16_t * rx_buf );
MD_STATUS    R_<Config_MSPIn>_Receive ( uint32_t * rx_buf );
```

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 * *rx_buf*	受信したデータを格納するバッファへのポインタ(1 バイト)
O	uint16_t * *rx_buf*	受信したデータを格納するバッファへのポインタ(2 バイト)
O	uint32_t * *rx_buf*	受信したデータを格納するバッファへのポインタ(4 バイト)

[戻り値]

マクロ	説明
MD_OK	正常終了

R_<Config_MSPIn>_Software_Trigger

MSPInm のチャネルを有効にし、開始トリガを設定します。

[指定形式]

void	R_<Config_MSPIn>_Software_Trigger (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_<Config_MSPIn>_Create_UserInit

MSPI スレーブモード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_<Config_MSPIn>_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_<Config_MSPIn>_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_<Config_MSPIn>_channel*n*0_interrupt_send

MSPI*n* のチャネル 0 の送信割り込みの発生に伴う処理を行います。

[指定形式]

void	r_<Config_MSPIn>_channel*n*0_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_<Config_MSPIn>_channel*n*0_interrupt_receive

MSPI*n* のチャネル 0 の受信割り込みの発生に伴う処理を行います。

[指定形式]

void	r_<Config_MSPIn>_channeln0_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_<Config_MSPIn>_interrupt_send

MSPI*n* のスレーブ送信割り込みの発生に伴う処理を行います。

[指定形式]

void	r_<Config_MSPIn>_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_<Config_MSPIn>_interrupt_receive

MSPI*n* のスレーブ受信割り込みの発生に伴う処理を行います。

[指定形式]

void	r_<Config_MSPIn>_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_<Config_MSPIn>_interrupt_error

MSPI*n* のエラー割り込みの発生に伴う処理を行います。

[指定形式]

void	r_<Config_MSPIn>_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_<Config_MSPIn>_interrupt_frameend

MSPI*n* のフレーム完了割り込みの発生に伴う処理を行います。

[指定形式]

```
void        r_<Config_MSPIn>_interrupt_frameend ( 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_<Config_MSPIn>_callback_sendend

MSPI*n* のスレーブ送信割り込みの発生に伴う処理を行います。

備考 1. 本 API 関数は、r_<Config_MSPIn>_interrupt_send 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_<Config_MSPIn>_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_<Config_MSPIn>_callback_receiveend

MSPI*n* のスレーブ受信割り込みの発生に伴う処理を行います。

備考 1. 本 API 関数は、r_<Config_MSPIn>_interrupt_receive 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_<Config_MSPIn>_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_<Config_MSPIn>_callback_error

MSPI*n* のエラー割り込みの発生に伴う処理を行います。

備考　本 API 関数は、r_<Config_MSPIn>_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]

void	r_<Config_MSPIn>_callback_error (uint32_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	uint32_t err_type	エラー割り込みの発生要因 000100xxB　：送信データの整合性エラー 000x001xB　：受信データの CRC エラー 000x00x1B　：受信データのパリティ・エラー

[戻り値]

なし

使用例

MSPI マスタモードの使用例を参照してください。

3.2.38 リアルタイムクロック

以下に、コード生成ツールがリアルタイムクロック用に出力する API 関数の一覧を示します。

表 3-38 リアルタイムクロック用 API 関数

API 関数名	機能概要
R_<Config_RTCAn>_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_<Config_RTCAn>_Start	リアルタイムクロックのカウントを開始します。
R_<Config_RTCAn>_Stop	リアルタイムクロックのカウントを終了します。
R_<Config_RTCAn>_Set_HourSystem	リアルタイムクロックの時間制（12 時間，24 時間制）を設定します。
R_<Config_RTCAn>_Get_CounterBufferValue	カウンタバッファレジスタを読み出します。
R_<Config_RTCAn>_Get_CounterDirectValue	カウンタレジスタを直接読み出します。
R_<Config_RTCAn>_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_<Config_RTCAn>_Get_SubCounterValue	サブカウンタのバッファレジスタを読み出します。
R_<Config_RTCAn>_Set_ErrorCorrectionValue	誤差補正値を設定します。
R_<Config_RTCAn>_Set_SubCounterCompareValue	サブカウンタの比較値を設定します。
R_<Config_RTCAn>_Get_AlarmValue	アラームの発生条件（曜日，時，分）を読み出します。
R_<Config_RTCAn>_Set_AlarmValue	アラームの発生条件（曜日，時，分）を設定します。
R_<Config_RTCAn>_Set_AlarmOn	アラーム割り込み機能を開始します。
R_<Config_RTCAn>_Set_AlarmOff	アラーム割り込み機能を終了します。
R_<Config_RTCAn>_Set_ConstPeriodInterruptOn	定周期割り込み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_<Config_RTCAn>_Set_ConstPeriodInterruptOff	定周期割り込み機能を終了します。
R_<Config_RTCAn>_Set_1secondInterruptOn	1 秒周期割り込みを開始します。
R_<Config_RTCAn>_Set_1secondInterruptOff	1 秒周期割り込みを終了します。
R_<Config_RTCAn>_Set_RTCA1HZOn	1Hz パルス出力機能を開始します。
R_<Config_RTCAn>_Set_RTCA1HZOff	1Hz パルス出力機能を終了します。
R_<Config_RTCAn>_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_<Config_RTCAn>_interrupt_alarm	アラーム割り込みの発生に伴う処理を行います。
r_<Config_RTCAn>_interrupt_periodic	定周期割り込みの発生に伴う処理を行います。
r_<Config_RTCAn>_interrupt_1second	1 秒周期割り込みの発生に伴う処理を行います。

R_<Config_RTCAn>_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。

備考　　本 API 関数は、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

```
void    R_<Config_RTCAn>_Create ( void );
```

[引数]

なし

[戻り値]

なし

R_<Config_RTCAn>_Start

リアルタイムクロックのカウントを開始します。

[指定形式]

```
void        R_<Config_RTCAn>_Start ( void );
```

[引数]

なし

[戻り値]

なし

R_<Config_RTCAn>_Stop

リアルタイムクロックのカウントを終了します。

[指定形式]

```
void        R_<Config_RTCAn>_Stop ( void );
```

[引数]

なし

[戻り値]

なし

R_<Config_RTCAn>_Set_HourSystem

リアルタイムクロックの時間制（12 時間，24 時間制）を設定します。

[指定形式]

MD_STATUS　　R_<Config_RTCAn>_Set_HourSystem (rtc_hour_system_t *hour_system*);

[引数]

I/O	引数	説明
I	rtc_hour_system_t *hour_system*;	時間制の種類 HOUR12　：12 時間制 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	カウント処理を実行中（設定変更後）
MD_ARGERROR	引数の指定が不正

R_<Config_RTCAn>_Get_CounterBufferValue

カウンタバッファレジスタを読み出します。

[指定形式]

```
MD_STATUS R_<Config_RTCAn>_Get_CounterBufferValue ( rtc_counter_value_t * const counter_read_val );
```

[引数]

I/O	引数	説明
O	rtc_counter_value_t * 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 以下に、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  */
    uint8_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	カウント処理を実行中（設定変更後）

R_<Config_RTCAn>_Get_CounterDirectValue

カウンタレジスタを直接読み込みます。

[指定形式]

```
MD_STATUS R_<Config_RTCAn>_Get_CounterDirectValue ( rtc_counter_value_t * const counter_read_val );
```

[引数]

I/O	引数	説明
O	rtc_counter_value_t * 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 以下に、カウント値 rtc_counter_value_t についての詳細は、R_<Config_RTCAn>_Get_CounterBufferValue 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	1 回目と 2 回目の読み出し値が不一致

R_<Config_RTCAn>_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]

```
MD_STATUS R_<Config_RTCAn>_Set_CounterValue ( rtc_counter_value_t counter_write_val );
```

[引数]

I/O	引数	説明
I	rtc_counter_value_t *counter_write_val*;	カウンタ値（秒，分，時，曜日，日，月，年）

備考 以下に、カウント値 rtc_counter_value_t についての詳細は、R_<Config_RTCAn>_Get_CounterBufferValue 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）
MD_BUSY2	カウント処理を実行中（設定変更後）

R_<Config_RTCAn>_Get_SubCounterValue

サブカウンタのバッファレジスタを読み出します。

[指定形式]

MD_STATUS R_<Config_RTCAn>_Get_SubCounterValue (uint32_t * const *subcounter_read_val*);

[引数]

I/O	引数	説明
O	uint32_t * const *subcounter_read_val*;	読み出したサブカウンタ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_BUSY	転送未完了

R_<Config_RTCAn>_Set_ErrorCorrectionValue

誤差補正値を設定します。

[指定形式]

MD_STATUS R_<Config_RTCAn>_Set_ErrorCorrectionValue (uint8_t * const *errorcorrection_write_val*);

[引数]

I/O	引数	説明
I	uint8_t * const *errorcorrection_write_val*;	誤差補正値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_BUSY1	動作停止状態
MD_BUSY2	書き込み未完了

R_<Config_RTCAn>_Set_SubCounterCompareValue

サブカウンタの比較値を設定します。

[指定形式]

```
MD_STATUS R_<Config_RTCAn>_Set_SubCounterCompareValue ( uint32_t * const subcompare_write_val );
```

[引数]

I/O	引数	説明
I	uint32_t * const *subcompare_write_val*;	比較値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_BUSY1	動作停止状態
MD_BUSY2	書き込み未完了

R_<Config_RTCAn>_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]

void	R_<Config_RTCAn>_Get_AlarmValue (rtc_alarm_value_t * const *alarm_val*);

[引数]

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

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。

```
typedef struct
{
    uint8_t alarmwm;    /*  分  */
    uint8_t alarmwh;    /*  分  */
    uint8_t alarmww;    /*  曜日  */
}rtc_alarm_value_t;
```

[戻り値]

なし

R_<Config_RTCAn>_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]

void	R_<Config_RTCAn>_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 アラームの発生条件 rtc_alarm_value_t についての詳細は、R_<Config_RTCAn>_Get_AlarmValue を参照してください。

[戻り値]

なし

R_<Config_RTCAn>_Set_AlarmOn

アラーム割り込み機能を開始します。

[指定形式]

```
void        R_<Config_RTCAn>_Set_AlarmOn ( void );
```

[引数]

なし

[戻り値]

なし

R_<Config_RTCAn>_Set_AlarmOff

アラーム割り込み機能を終了します。

[指定形式]

```
void        R_<Config_RTCAn>_Set_AlarmOff ( void );
```

[引数]

なし

[戻り値]

なし

R_<Config_RTCAn>_Set_ConstPeriodInterruptOn

定周期割り込み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]

MD_STATUS R_<Config_RTCAn>_Set_ConstPeriodInterruptOn (rtc_int_period_t *period*);

[引数]

I/O	引数	説明
I	rtc_int_period_t *period*;	周期割り込みの発生周期 QUARTERSEC : 0.25 秒に一度 HALFSEC : 0.5 秒に一度 ONESEC : 1 秒に一度 ONEMIN : 1 分に一度 ONEHOUR : 1 時間に一度 ONEDAY : 1 日に一度 ONEMONTH : 1 ヵ月に一度

[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_<Config_RTCAn>_Set_ConstPeriodInterruptOff

定周期割り込み機能を終了します。

[指定形式]

```
void        R_<Config_RTCAn>_Set_ConstPeriodInterruptOff ( void );
```

[引数]

なし

[戻り値]

なし

R_<Config_RTCAn>_Set_1secondInterruptOn

1 秒周期割り込みを開始します。

[指定形式]

```
void        R_<Config_RTCAn>_Set_1secondInterruptOn ( void );
```

[引数]

なし

[戻り値]

なし

R_<Config_RTCAn>_Set_1secondInterruptOff

1 秒周期割り込みを終了します。

[指定形式]

void	R_<Config_RTCAn>_Set_1secondInterruptOff (void);

[引数]

なし

[戻り値]

なし

R_<Config_RTCAn>_Set_RTCA1HZOn

1Hz パルス出力機能を開始します。

[指定形式]

```
void        R_<Config_RTCAn>_Set_RTCA1HZOn ( void );
```

[引数]

なし

[戻り値]

なし

R_<Config_RTCAn>_Set_RTCA1HZOff

1Hz パルス出力機能を終了します。

[指定形式]

void	R_<Config_RTCAn>_Set_RTCA1HZOff (void);

[引数]

なし

[戻り値]

なし

R_<Config_RTCAn>_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_<Config_RTCAn>_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_<Config_RTCAn>_Create_UserInit ( void );
```

[引数]

なし

[戻り値]

なし

r_<Config_RTCAn>_interrupt_alarm

アラーム割り込みの発生に伴う処理を行います。

[指定形式]

```
void        r_<Config_RTCAn>_interrupt_alarm ( void );
```

[引数]

なし

[戻り値]

なし

r_<Config_RTCAn>_interrupt_periodic

定周期割り込みの発生に伴う処理を行います。

[指定形式]

```
void        r_<Config_RTCAn>_interrupt_periodic ( void );
```

[引数]

なし

[戻り値]

なし

r_<Config_RTCAn>_interrupt_1second

1 秒周期割り込みの発生に伴う処理を行います。

[指定形式]

```
void        r_<Config_RTCAn>_interrupt_1second ( void );
```

[引数]

なし

[戻り値]

なし

3.2.39 エラーコントロールモジュール

以下に、コード生成ツールがエラーコントロールモジュール用に出力する API 関数の一覧を示します。

表 3-39 エラーコントロールモジュール用 API 関数

API 関数名	機能概要
R_<Config_ECM>_Create	エラーコントロールモジュールを制御するうえで必要となる初期化処理を行います。
R_<Config_ECM>_Start	エラーコントロールモジュールの動作を許可します。
R_<Config_ECM>_Stop	エラーコントロールモジュールの動作を禁止します。
R_<Config_ECM>_Set_Master_Error	ECM master error を設定します。
R_<Config_ECM>_Clear_Master_Error	ECM master error を解除します。
R_<Config_ECM>_Set_Checker_Error	ECM checker error を設定します。
R_<Config_ECM>_Clear_Checker_Error	ECM checker error を解除します。
R_<Config_ECM>_Pseud_*XXXX*_Start	エラーコントロールモジュールの XXX pseudo デバッグを許可します。
R_<Config_ECM>_Pseudo_*XXXX*_Stop	エラーコントロールモジュールの XXX pseudo デバッグを禁止します。
R_<Config_ECM>_Create_UserInit	エラーコントロールモジュールに関するユーザ独自の初期化処理を行います。
r_<Config_ECM>_ecmmi_interrupt_pek	ECM マスカブル割り込みに伴う処理を行います。
r_<Config_ECM>_mi_interrupt_pek	ECM マスカブル割り込みに伴う処理を行います。
r_<Config_ECM>_dclsmi_interrupt_pek	DCLS エラー割り込みに伴う処理を行います。

R_<Config_ECM>_Create

エラーコントロールモジュール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、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

```
void    R_<Config_ECM>_Create ( void );
```

[引数]

なし

[戻り値]

なし

R_<Config_ECM>_Start

エラーコントロールモジュールの動作を許可します。

[指定形式]

```
void        R_<Config_ECM>_Start ( void );
```

[引数]

なし

[戻り値]

なし

R_<Config_ECM>_Stop

エラーコントロールモジュールの動作を禁止します。

[指定形式]

```
void        R_<Config_ECM>_Stop ( void );
```

[引数]

なし

[戻り値]

なし

R_<Config_ECM>_Set_Master_Error

ECM master error を設定します。

[指定形式]

```
void    R_<Config_ECM>_Set_Master_Error(void);
```

[引数]

なし

[戻り値]

なし

R_<Config_ECM>_Clear_Master_Error

ECM master error を解除します。

[指定形式]

```
void      R_<Config_ECM>_Clear_Master_Error(void);
```

[引数]

なし

[戻り値]

なし

R_<Config_ECM>_Set_Checker_Error

ECM checker error を設定します。

[指定形式]

```
void    R_<Config_ECM>_Set_Checker_Error(void);
```

[引数]

なし

[戻り値]

なし

R_<Config_ECM>_Clear_Checker_Error

ECM checker error を解除します。

[指定形式]

```
void    R_<Config_ECM>_Clear_Checker_Error(void);
```

[引数]

なし

[戻り値]

なし

R_<Config_ECM>_Delay_Timer_Start

ECM delay timer を開始します。

[指定形式]

```
void    R_<Config_ECM>_Clear_Checker_Error(void);
```

[引数]

なし

[戻り値]

なし

R_<Config_ECM>_Delay_Timer_Stop

ECM delay timer を停止します。

[指定形式]

```
void    R_<Config_ECM>_Delay_Timer_Stop(void);
```

[引数]

なし

[戻り値]

なし

R_<Config_ECM>_Pseud_*XXXX*_Start

エラーコントロールモジュールの XXX pseudo デバッグを許可します。

[指定形式]

void	R_<Config_ECM>_Pseud_*XXXX*_Start (void);

備考 *XXXX* は表 3-40 のエラー名を意味します。

表3-40 ECM pseudo エラー名

	エラー名 (*XXXX*)		エラー名 (*XXXX*)
1	Delay_Timer_Overflow	2	ECM_Compare_Error
3	Activation_Production_Test_Mode	4	Activation_Normal_Operation_Mode
5	Deactivation_Normal_Operation_Mode	6	Serial_Programming_Mode
7	Activation_User_Boot_Mode	8	Deactivation_User_Boot_Mode
9	Mode_Check_Error	10	Flash_Access_Error
11	FACI_Reset_Transfer_Error	12	BIST_Parameter_Transfer_Error
13	DTS_Compare_Error	14	BUS_Bridge_Compare_Error_SDMAC
15	BUS_Bridge_Compare_Error	16	Element_Bus_Routing_Error
17	I_Bus_Routing_Error	18	P_Bus_Routing_Error
19	CRAM_Bus_Routing_Error	20	System_Bus_Routing_Error
21	Global_Flash_Bus_Routing_Error	22	Local_Flash_Bus_Routing_Error
23	Clock_Monitor_Error_MOSC	24	Clock_Monitor_Error_HSintOSC
25	Clock_Monitor_Error_LSintOSC	26	Clock_Monitor_Error_CLK_LSB
27	Clock_Monitor_Error_CLK_UHSB	28	Clock_Monitor_Error_CLK_HBUS
29	OSTM*n*_Interrupt_Error	30	ADCJ*n*_Parity_Error
31	Temperature_Sensor_Error	32	Code_Flash_Address_Parity_Error
33	Code_Flash_ECC_Uncorrectable_Error	34	Code_Flash_ECC_Correctable_Error
35	Code_Flash_ECC_Overflow_Error	36	Data_Flash_ECC_Uncorrectable_Error
37	Data_Flash_ECC_Correctable_Error	38	Data_Flash_ECC_Overflow_Error
39	Local_RAM_ECC_Uncorrectable_Error	40	Local_RAM_ECC_Correctable_Error
41	Local_RAM_ECC_Overflow_Error	42	Cluster_RAM_ECC_Uncorrectable_Error
43	Cluster_RAM_ECC_Correctable_Error	44	Cluster_RAM_ECC_Overflow_Error
45	DTSRAM_ECC_Uncorrectable_Error	46	DTSRAM_ECC_Correctable_Error
47	DTSRAM_ECC_Overflow_Error	48	SDMACnRAM_ECC_Uncorrectable_Erro r
49	SDMACnRAM_ECC_Correctable_Error	50	FlexRayRAM_ECC_Uncorrectable_Error
51	FlexRayRAM_ECC_Correctable_Error	52	RS_CANFDRAM_ECC_Uncorrectable_E rror
53	RS_CANFDRAM_ECC_Correctable_Error	54	MSPI_RAM_ECC_Uncorrectable_Error
55	MSPI_RAM _ECC_Correctable_Error	56	GTM_RAM_ECC_Uncorrectable_Error
57	GTM_RAM _ECC_Correctable_Error	58	Fast_Ethernet_RAM_ECC_Uncorrectable _Error
59	Fast_Ethernet_RAM_ECC_Correctable_Er ror	60	Gigabit_Ethernet_RAM_ECC_Uncorrecta ble_Error

61	Gigabit_Ethernet_RAM_ECC_Correctable_Error	62	MMCA_RAM_ECC_Uncorrectable_Error
63	MMCA_RAM_ECC_Correctable_Error	64	Peripheral_ECC_Overflow_Error
65	Data_Transfer_Path_EDC_Error	66	Data_Transfer_Path_ECC_Uncorrectable_Error
67	Data_Transfer_Path_ECC_Correctable_Error	68	CRAM_Guard_Error
69	I_Bus_Guard_Error	70	P_Bus_Guard_Error
71	H_Bus_Guard_Error	72	DTS_sDMAC_Transfer_Error
73	H_Bus_Master_Transfer_Error	74	External_Error_Input_*n*_Error
75	Software_Alarm_*n*_Error	76	DCLS_Compare_Error_PE*k*
77	Unintended_Debug_Enable_Detection_PE*k*	78	Watchdog_Timer_Error_PE*k*
79	Clock_Monitor_Error_PE*k*	80	PE_Guard_Error_PE*k*
81	Data_Access_Error_PE*k*	82	PE_Guard_Error_PE*k*_Access
83	Local_RAM_PE*k*_ECC_Uncorrectable_Error	84	Local_RAM_PE*k*_ECC_Correctable_Error
85	Local_RAM_PE*k*_ECC_Overflow_Error	86	Instruction_Cache_RAM_PE*k*_ECC_Uncorrectable_Error

[引数]
なし

[戻り値]
なし

R_<Config_ECM>_Pseudo_*XXXX*_Stop

エラーコントロールモジュールの XXX pseudo デバッグを禁止します。

[指定形式]

void	R_<Config_ECM>_Pseud_*XXXX*_Stop (void);

備考 *XXXX* は表 3-40 のエラー名を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ECM>_Create_UserInit

エラーコントロールモジュール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_<Config_ECM>_Create から呼び出されます。

[指定形式]

```
void　R_<Config_ECM>_Create_UserInit ( void );
```

[引数]

なし

[戻り値]

なし

r_<Config_ECM>_ecmmi_interrupt_pe*k*

ECM マスカブル割り込みに伴う処理を行います。

[指定形式]

void	r_<Config_ECM>_ecmmi_interrupt_pe*k* (void);

備考 *k* は CPU コア番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_ECM>_mi_interrupt_pe*k*

ECM マスカブル割り込みに伴う処理を行います。

[指定形式]

void	r_<Config_ECM>_mi_interrupt_pe*k* (void);

備考 *k* は CPU コア番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_ECM>_dclsmi_interrupt_pe*k*

DCLS エラー割り込みに伴う処理を行います。

[指定形式]

void	R_<Config_ECM>_dclsmi_interrupt_pe*k* (void);

備考 *k* は CPU コア番号を意味します。

[引数]

なし

[戻り値]

なし

3.2.40 GTM Clock

以下に、コード生成ツールが GTM Clock 用に出力する API 関数の一覧を示します。
Generic Timer Module 用の API 関数の説明は英語だけとなります。

表 3-41 GTM Clock 用 API 関数

API 関数名	機能概要
R_<Config_GTM>_Clock_Create	Performs initialization necessary to control the GTM clock.
R_<Config_GTM>_CCMn_Create	Performs initialization necessary to control the cluster configuration module.
R_<Config_GTM>_Clock_Create_UserInit	Performs user-defined initialization relating to the GTM clock.

R_<Config_GTM>_Clock_Create

Performs initialization necessary to control the GTM clock.

備考　　本 API 関数は、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

```
void    R_<Config_GTM>_Clock_Create ( void );
```

[引数]

なし

[戻り値]

なし

R_<Config_GTM>_CCM*n*_Create

Performs initialization necessary to control the cluster configuration module.

[指定形式]

void	R_<Config_GTM>_CCM*n*_Create (void);

備考 *n* は CCM チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_GTM>_Clock_Create_UserInit

Performs user-defined initialization relating to the GTM clock.

備考　　本 API 関数は、R_<Config_GTM>_Clock_Create から呼び出されます。

[指定形式]

```
void    R_<Config_GTM>_Clock_Create_UserInit ( void );
```

[引数]

なし

[戻り値]

なし

3.2.41 Time Base Unit

以下に、コード生成ツールが Time Base Unit 用に出力する API 関数の一覧を示します。
Generic Timer Module 用の API 関数の説明は英語だけとなります。

表 3-42 Time Base Unit 用 API 関数

API 関数名	機能概要
R_<Config_TBU>_Create	Performs initialization necessary to control the time base unit functions.
R_<Config_TBU>_Start	Starts the count for time base unit.
R_<Config_TBU>_Stop	Ends the count for time base unit.
R_<Config_TBU>_Create_UserInit	Performs user-defined initialization relating to the time base unit functions.

R_<Config_TBU>_Create

Performs initialization necessary to control the time base unit functions.

備考　　本 API 関数は、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

```
void    R_<Config_TBU>_Create ( void );
```

[引数]

なし

[戻り値]

なし

R_<Config_TBU>_Start

Starts the count for time base unit.

[指定形式]

```
void        R_<Config_TBU>_Start ( void );
```

[引数]

なし

[戻り値]

なし

R_<Config_TBU>_Stop

Ends the count for time base unit.

[指定形式]

```
void        R_<Config_TBU>_Stop ( void );
```

[引数]

なし

[戻り値]

なし

R_<Config_TBU>_Create_UserInit

Performs user-defined initialization relating to the time base unit functions.

備考　　本 API 関数は、R_<Config_TBU>_Create から呼び出されます。

[指定形式]

```
void    R_<Config_TBU>_Create_UserInit ( void );
```

[引数]

なし

[戻り値]

なし

3.2.42 TIM Bit Compression Mode

以下に、コード生成ツールが TIM Bit Compression Mode 用に出力する API 関数の一覧を示します。Generic Timer Module 用の API 関数の説明は英語だけとなります。

表 3-43 TIM Bit Compression Mode 用 API 関数

API 関数名	機能概要
R_<Config_TIMn>_Create	Performs initialization necessary to control the TIM bit compression mode.
R_<Config_TIMn>_Start	Starts the count for channel m in unit n.
R_<Config_TIMn>_Stop	Stops the count for channel m in unit n.
R_<Config_TIMn>_SetDetectedEdge	Sets detected edge number.
R_<Config_TIMn>_Software_Reset	Stops channel operation and reset channel registers.
R_<Config_TIMn>_NEWVAL_TriggerOn	Trigger NEWVAL bit in TIMn_CHm_IRQ_NOTIFY register by software.
R_<Config_TIMn>_ECNTOFL_TriggerOn	Trigger ECNTOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.
R_<Config_TIMn>_CNTOFL_TriggerOn	Trigger CNTOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.
R_<Config_TIMn>_GPROFL_TriggerOn	Trigger GPROFL bit in TIMn_CHm_IRQ_NOTIFY register by software.
R_<Config_TIMn>_TODETOFL_TriggerOn	Trigger TODETOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.
R_<Config_TIMn>_Channelm_GLITCHDETOFL_TriggerOn	Trigger GLITCHDET bit in TIMn_CHm_IRQ_NOTIFY register by software.
R_<Config_TIMn>_Create_UserInit	Performs user-defined initialization relating to the TIM bit compression mode.
r_<Config_TIMn>_Callback_GTM_Error	Processing is performed according to the occurrence of a count error interrupt.
r_<Config_TIMn>_share_interrupt	Performs processing in response to the TIMn channelm interrupt.

R_<Config_TIMn>_Create

Performs initialization necessary to control the TIM bit compression mode.

備考　本 API 関数は、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

void	R_<Config_TIMn>_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_<Config_TIMn>_Start

Starts the count for channel m in unit n.

[指定形式]

void	R_<Config_TIMn>_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_<Config_TIMn>_Stop

Stops the count for channel m in unit n.

[指定形式]

void	R_<Config_TIMn>_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_<Config_TIMn>_SetDetectedEdge

Sets detected edge number.

[指定形式]

void	R_<Config_TIMn>_SetDetectedEdge (uint16_t number);

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	uint16_t *number*;	Detected edge number.

[戻り値]

なし

R_<Config_TIMn>_Software_Reset

Stops channel operation and reset channel registers.

[指定形式]

void	R_<Config_TIMn>_Software_Reset (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_<Config_TIMn>_NEWVAL_TriggerOn

Trigger NEWVAL bit in TIMn_CH0_IRQ_NOTIFY register by software.

[指定形式]

void	R_<Config_TIMn>_NEWVAL_TriggerOn (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_<Config_TIMn>_ECNTOFL_TriggerOn

Trigger ECNTOFL bit in TIMn_CH0_IRQ_NOTIFY register by software.

[指定形式]

void	R_<Config_TIMn>_ECNTOFL_TriggerOn (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_<Config_TIMn>_CNTOFL_TriggerOn

Trigger CNTOFL bit in TIMn_CH0_IRQ_NOTIFY register by software.

[指定形式]

void	R _Config_TIM*n*_CNTOFL_TriggerOn (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_<Config_TIMn>_GPROFL_TriggerOn

Trigger GPROFL bit in TIMn_CH0_IRQ_NOTIFY register by software.

[指定形式]

void	R_<Config_TIMn>_GPROFL_TriggerOn (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_<Config_TIMn>_TODETOFL_TriggerOn

Trigger TODETOFL bit in TIMn_CH0_IRQ_NOTIFY register by software.

[指定形式]

void	R_<Config_TIMn>_TODETOFL_TriggerOn (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_<Config_TIMn>_Channel*m*_GLITCHDETOFL_TriggerOn

Trigger GLITCHDETOFL bit in TIMn_CH0_IRQ_NOTIFY register by software.

[指定形式]

void	R_<Config_TIMn>_Channel*m*_GLITCHDETOFL_TriggerOn (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn>_Create_UserInit

Performs user-defined initialization relating to the TIM input prescaler mode.

備考　本 API 関数は、R_<Config_TIMn>_Create から呼び出されます。

[指定形式]

void	R_<Config_TIMn>_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_<Config_TIM*n*>_Callback_GTM_Error

Processing is performed according to the occurrence of a count error interrupt.

備考　本 API 関数は、r_gtm_error_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]

```
void    r_<Config_TIMn>_Callback_GTM_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_<Config_TIMn>_share_interrupt

Performs processing in response to the TIMn channelm interrupt.

[指定形式]

void	r_<Config_TIMn>_share_interrupt (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_<Config_TIMn>_ch*m*_share_interrupt

Performs processing in response to the TIMn channelm interrupt.

[指定形式]

void	r_<Config_TIMn>_ch*m*_share_interrupt (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

3.2.43 TIM Gated Periodic Sampling Mode

以下に、コード生成ツールが TIM Gated Periodic Sampling Mode 用に出力する API 関数の一覧を示します。

Generic Timer Module 用の API 関数の説明は英語だけとなります。

表 3-44 TIM Gated Periodic Sampling Mode 用 API 関数

API 関数名	機能概要
R_<Config_TIMn_m>_Create	Performs initialization necessary to control the TIM gated periodic sampling mode.
R_<Config_TIMn_m>_Start	Starts the count for channel m in unit n.
R_<Config_TIMn_m>_Stop	Stops the count for channel m in unit n.
R_<Config_TIMn_m>_UpdateElapsedClock	Updates elapsed clock number.
R_<Config_TIMn_m>_Software_Reset	Stops channel operation and reset channel registers.
R_<Config_TIMn_m>_NEWVAL_TriggerOn	Trigger NEWVAL bit in TIMn_CHm_IRQ_NOTIFY register by software.
R_<Config_TIMn_m>_ECNTOFL_TriggerOn	Trigger ECNTOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.
R_<Config_TIMn_m>_CNTOFL_TriggerOn	Trigger CNTOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.
R_<Config_TIMn_m>_GPROFL_TriggerOn	Trigger GPROFL bit in TIMn_CHm_IRQ_NOTIFY register by software.
R_<Config_TIMn_m>_TODETOFL_TriggerOn	Trigger TODETOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.
R_<Config_TIMn_m>_GLITCHDETOFL_TriggerOn	Trigger GLITCHDET bit in TIMn_CHm_IRQ_NOTIFY register by software.
R_<Config_TIMn_m>_Create_UserInit	Performs user-defined initialization relating to the TIM gated periodic sampling mode.
r_<Config_TIMn_m>_ Callback_GTM_Error	Processing is performed according to the occurrence of a count error interrupt.
r_<Config_TIMn_m>_share_interrupt	Performs processing in response to the TIMn channelm interrupt.

R_<Config_TIMn_m>_Create

Performs initialization necessary to control the TIM input prescaler mode.

備考　本 API 関数は、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

void	R_<Config_TIMn_m>_Create (void);

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_Start

Starts the count for channel m in unit n.

[指定形式]

void	R_<Config_TIMn_m>_Start (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_Stop

Stops the count for channel m in unit n.

[指定形式]

void	R_<Config_TIMn_m>_Stop (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_UpdateElapsedClock

Updates elapsed clock number.

[指定形式]

void	R_<Config_TIMn_m>_UpdateElapsedClock (uint32_t number);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

I/O	引数	説明
I	uint32_t *number*;	Elapsed clock number.

[戻り値]

なし

R_<Config_TIMn_m>_Software_Reset

Stops channel operation and reset channel registers.

[指定形式]

void	R_<Config_TIMn_m>_Software_Reset (void);

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_NEWVAL_TriggerOn

Trigger NEWVAL bit in TIMn_CHm_IRQ_NOTIFY register by software.

[指定形式]

void	R_<Config_TIMn_m>_NEWVAL_TriggerOn (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_ECNTOFL_TriggerOn

Trigger ECNTOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.

[指定形式]

void	R_<Config_TIMn_m>_ECNTOFL_TriggerOn (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_CNTOFL_TriggerOn

Trigger CNTOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.

[指定形式]

void	R _Config_TIM*n*_*m*_CNTOFL_TriggerOn (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_GPROFL_TriggerOn

Trigger GPROFL bit in TIMn_CHm_IRQ_NOTIFY register by software.

[指定形式]

void	R_<Config_TIMn_m>_GPROFL_TriggerOn (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_TODETOFL_TriggerOn

Trigger TODETOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.

[指定形式]

void	R_<Config_TIMn_m>_TODETOFL_TriggerOn (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_GLITCHDETOFL_TriggerOn

Trigger GLITCHDETOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.

[指定形式]

void	R_<Config_TIMn_m>_GLITCHDETOFL_TriggerOn (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_Create_UserInit

Performs user-defined initialization relating to the TIM gated periodic sampling mode.

備考　本 API 関数は、R_<Config_TIMn_m>_Create から呼び出されます。

[指定形式]

```
void    R_<Config_TIMn_m>_Create_UserInit ( void );
```

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TIMn_m>_ Callback_GTM_Error

Processing is performed according to the occurrence of a count error interrupt.

備考　本 API 関数は、r_gtm_error_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]

```
void    r_<Config_TIMn_m>_ Callback_GTM_Error(void);
```

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TIMn_m>_share_interrupt

Performs processing in response to the TIMn channelm interrupt.

[指定形式]

void	r_<Config_TIMn_m>_share_interrupt (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

3.2.44 TIM Input Prescaler Mode

以下に、コード生成ツールが TIM Input Prescaler Mode 用に出力する API 関数の一覧を示します。
Generic Timer Module 用の API 関数の説明は英語だけとなります。

表 3-45 TIM Input Prescaler Mode 用 API 関数

API 関数名	機能概要
R_<Config_TIMn_m>_Create	Performs initialization necessary to control the TIM input prescaler mode.
R_<Config_TIMn_m>_Start	Starts the count for channel m in unit n.
R_<Config_TIMn_m>_Stop	Stops the count for channel m in unit n.
R_<Config_TIMn_m>_SetDetectedEdge	Sets detected edge number.
R_<Config_TIMn_m>_Software_Reset	Stops channel operation and reset channel registers.
R_<Config_TIMn_m>_NEWVAL_TriggerOn	Trigger NEWVAL bit in TIMn_CHm_IRQ_NOTIFY register by software.
R_<Config_TIMn_m>_ECNTOFL_TriggerOn	Trigger ECNTOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.
R_<Config_TIMn_m>_CNTOFL_TriggerOn	Trigger CNTOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.
R_<Config_TIMn_m>_GPROFL_TriggerOn	Trigger GPROFL bit in TIMn_CHm_IRQ_NOTIFY register by software.
R_<Config_TIMn_m>_TODETOFL_TriggerOn	Trigger TODETOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.
R_<Config_TIMn_m>_GLITCHDETOFL_TriggerOn	Trigger GLITCHDET bit in TIMn_CHm_IRQ_NOTIFY register by software.
R_<Config_TIMn_m>_Create_UserInit	Performs user-defined initialization relating to the TIM input prescaler mode.
r_<Config_TIMn_*m*>_Callback_GTM_Error	Processing is performed according to the occurrence of a count error interrupt.
r_<Config_TIMn_m>_share_interrupt	Performs processing in response to the TIMn channelm interrupt.

R_<Config_TIMn_m>_Create

Performs initialization necessary to control the TIM input prescaler mode.

備考　本 API 関数は、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

void	R_<Config_TIMn_m>_Create (void);

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_Start

Starts the count for channel m in unit n.

[指定形式]

void	R_<Config_TIMn_m>_Start (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_Stop

Stops the count for channel m in unit n.

[指定形式]

void	R_<Config_TIMn_m>_Stop (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_SetDetectedEdge

Sets detected edge number.

[指定形式]

void	R_<Config_TIMn_m>_SetDetectedEdge (uint32_t number);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

I/O	引数	説明
I	uint32_t *number*;	Detected edge number.

[戻り値]

なし

R_<Config_TIMn_m>_Software_Reset

Stops channel operation and reset channel registers.

[指定形式]

void	R_<Config_TIMn_m>_Software_Reset (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_NEWVAL_TriggerOn

Trigger NEWVAL bit in TIMn_CHm_IRQ_NOTIFY register by software.

[指定形式]

void	R_<Config_TIMn_m>_NEWVAL_TriggerOn (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_ECNTOFL_TriggerOn

Trigger ECNTOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.

[指定形式]

void	R_<Config_TIMn_m>_ECNTOFL_TriggerOn (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_CNTOFL_TriggerOn

Trigger CNTOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.

[指定形式]

void	R _Config_TIM*n*_*m*_CNTOFL_TriggerOn (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_GPROFL_TriggerOn

Trigger GPROFL bit in TIMn_CHm_IRQ_NOTIFY register by software.

[指定形式]

void	R_<Config_TIMn_m>_GPROFL_TriggerOn (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_TODETOFL_TriggerOn

Trigger TODETOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.

[指定形式]

void	R_<Config_TIMn_m>_TODETOFL_TriggerOn (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_GLITCHDETOFL_TriggerOn

Trigger GLITCHDETOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.

[指定形式]

void	R_<Config_TIMn_m>_GLITCHDETOFL_TriggerOn (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_Create_UserInit

Performs user-defined initialization relating to the TIM input prescaler mode.

備考　本 API 関数は、R_<Config_TIMn_m>_Create から呼び出されます。

[指定形式]

```
void    R_<Config_TIMn_m>_Create_UserInit ( void );
```

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TIM*n*_*m*>_Callback_GTM_Error

Processing is performed according to the occurrence of a count error interrupt.

備考　本 API 関数は、r_gtm_error_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]

```
void    r_<Config_TIMn_m>_Callback_GTM_Error(void);
```

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TIMn_m>_share_interrupt

Performs processing in response to the TIMn channelm interrupt.

[指定形式]

```
void        r_<Config_TIMn_m>_share_interrupt ( void );
```

備考　　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

3.2.45 TIM Input Event Mode

以下に、コード生成ツールが TIM Input Event Mode 用に出力する API 関数の一覧を示します。
Generic Timer Module 用の API 関数の説明は英語だけとなります。

表 3-46 TIM Input Event Mode 用 API 関数

API 関数名	機能概要
R_<Config_TIMn_m>_Create	Performs initialization necessary to control the TIM input event mode.
R_<Config_TIMn_m>_Start	Starts the count for channel m in unit n.
R_<Config_TIMn_m>_Stop	Stops the count for channel m in unit n.
R_<Config_TIMn_m>_Software_Reset	Stops channel operation and reset channel registers.
R_<Config_TIMn_m>_NEWVAL_TriggerOn	Trigger NEWVAL bit in TIMn_CHm_IRQ_NOTIFY register by software.
R_<Config_TIMn_m>_ECNTOFL_TriggerOn	Trigger ECNTOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.
R_<Config_TIMn_m>_CNTOFL_TriggerOn	Trigger CNTOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.
R_<Config_TIMn_m>_GPROFL_TriggerOn	Trigger GPROFL bit in TIMn_CHm_IRQ_NOTIFY register by software.
R_<Config_TIMn_m>_TODETOFL_TriggerOn	Trigger TODETOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.
R_<Config_TIMn_m>_GLITCHDETOFL_TriggerOn	Trigger GLITCHDET bit in TIMn_CHm_IRQ_NOTIFY register by software.
R_<Config_TIMn_m>_Create_UserInit	Performs user-defined initialization relating to the TIM input event mode.
r_<Config_TIMn_*m*>_Callback_GTM_Error	Processing is performed according to the occurrence of a count error interrupt.
r_<Config_TIMn_m>_share_interrupt	Performs processing in response to the TIMn channelm interrupt.

R_<Config_TIMn_m>_Create

Performs initialization necessary to control the TIM input event mode.

備考　本 API 関数は、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

```
void    R_<Config_TIMn_m>_Create ( void );
```

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_Start

Starts the count for channel m in unit n.

[指定形式]

void	R_<Config_TIMn_m>_Start (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_Stop

Stops the count for channel m in unit n.

[指定形式]

void	R_<Config_TIMn_m>_Stop (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_Software_Reset

Stops channel operation and reset channel registers.

[指定形式]

void	R_<Config_TIMn_m>_Software_Reset (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_NEWVAL_TriggerOn

Trigger NEWVAL bit in TIMn_CHm_IRQ_NOTIFY register by software.

[指定形式]

void	R_<Config_TIMn_m>_NEWVAL_TriggerOn (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_ECNTOFL_TriggerOn

Trigger ECNTOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.

[指定形式]

void	R_<Config_TIMn_m>_ECNTOFL_TriggerOn (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_CNTOFL_TriggerOn

Trigger CNTOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.

[指定形式]

void	R _Config_TIM*n*_*m*_CNTOFL_TriggerOn (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_GPROFL_TriggerOn

Trigger GPROFL bit in TIMn_CHm_IRQ_NOTIFY register by software.

[指定形式]

void	R_<Config_TIMn_m>_GPROFL_TriggerOn (void);

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_TODETOFL_TriggerOn

Trigger TODETOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.

[指定形式]

void	R_<Config_TIMn_m>_TODETOFL_TriggerOn (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_GLITCHDETOFL_TriggerOn

Trigger GLITCHDETOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.

[指定形式]

void	R_<Config_TIMn_m>_GLITCHDETOFL_TriggerOn (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_Create_UserInit

Performs user-defined initialization relating to the TIM input event mode.

備考　本 API 関数は、R_<Config_TIMn_m>_Create から呼び出されます。

[指定形式]

```
void    R_<Config_TIMn_m>_Create_UserInit ( void );
```

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TIM*n*_*m*>_Callback_GTM_Error

Processing is performed according to the occurrence of a count error interrupt.

備考　本 API 関数は、r_gtm_error_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]

```
void    r_<Config_TIMn_m>_Callback_GTM_Error(void);
```

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TIMn_m>_share_interrupt

Performs processing in response to the TIMn channelm interrupt.

[指定形式]

```
void        r_<Config_TIMn_m>_share_interrupt ( void );
```

備考　　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

3.2.46 TIM Pulse Integration Mode

以下に、コード生成ツールが TIM Pulse Integration Mode 用に出力する API 関数の一覧を示します。Generic Timer Module 用の API 関数の説明は英語だけとなります。

表 3-47 TIM Pulse Integration Mode 用 API 関数

API 関数名	機能概要
R_<Config_TIMn_m>_Create	Performs initialization necessary to control the TIM pulse integration mode.
R_<Config_TIMn_m>_Start	Starts the count for channel m in unit n.
R_<Config_TIMn_m>_Stop	Stops the count for channel m in unit n.
R_<Config_TIMn_m>_Software_Reset	Stops channel operation and reset channel registers.
R_<Config_TIMn_m>_NEWVAL_TriggerOn	Trigger NEWVAL bit in TIMn_CHm_IRQ_NOTIFY register by software.
R_<Config_TIMn_m>_ECNTOFL_TriggerOn	Trigger ECNTOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.
R_<Config_TIMn_m>_CNTOFL_TriggerOn	Trigger CNTOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.
R_<Config_TIMn_m>_GPROFL_TriggerOn	Trigger GPROFL bit in TIMn_CHm_IRQ_NOTIFY register by software.
R_<Config_TIMn_m>_TODETOFL_TriggerOn	Trigger TODETOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.
R_<Config_TIMn_m>_GLITCHDETOFL_TriggerOn	Trigger GLITCHDET bit in TIMn_CHm_IRQ_NOTIFY register by software.
R_<Config_TIMn_m>_Create_UserInit	Performs user-defined initialization relating to the TIM pulse integration mode.
r_<Config_TIMn_m>_Callback_GTM_Error	Processing is performed according to the occurrence of a count error interrupt.
r_<Config_TIMn_m>_share_interrupt	Performs processing in response to the TIMn channelm interrupt.

R_<Config_TIMn_m>_Create

Performs initialization necessary to control the TIM pulse integration mode.

備考　本 API 関数は、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

```
void    R_<Config_TIMn_m>_Create ( void );
```

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_Start

Starts the count for channel m in unit n.

[指定形式]

void	R_<Config_TIMn_m>_Start (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_Stop

Stops the count for channel m in unit n.

[指定形式]

void	R_<Config_TIMn_m>_Stop (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_Software_Reset

Stops channel operation and reset channel registers.

[指定形式]

```
void        R_<Config_TIMn_m>_Software_Reset ( void );
```

備考　　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_NEWVAL_TriggerOn

Trigger NEWVAL bit in TIMn_CHm_IRQ_NOTIFY register by software.

[指定形式]

void	R_<Config_TIMn_m>_NEWVAL_TriggerOn (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_ECNTOFL_TriggerOn

Trigger ECNTOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.

[指定形式]

void	R_<Config_TIMn_m>_ECNTOFL_TriggerOn (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_CNTOFL_TriggerOn

Trigger CNTOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.

[指定形式]

void	R _Config_TIM*n*_*m*_CNTOFL_TriggerOn (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_GPROFL_TriggerOn

Trigger GPROFL bit in TIMn_CHm_IRQ_NOTIFY register by software.

[指定形式]

void	R_<Config_TIMn_m>_GPROFL_TriggerOn (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_TODETOFL_TriggerOn

Trigger TODETOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.

[指定形式]

void	R_<Config_TIMn_m>_TODETOFL_TriggerOn (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_GLITCHDETOFL_TriggerOn

Trigger GLITCHDETOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.

[指定形式]

void	R_<Config_TIMn_m>_GLITCHDETOFL_TriggerOn (void);

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_Create_UserInit

Performs user-defined initialization relating to the TIM pulse integration mode.

備考　本 API 関数は、R_<Config_TIMn_m>_Create から呼び出されます。

[指定形式]

```
void    R_<Config_TIMn_m>_Create_UserInit ( void );
```

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TIMn_m>_Callback_GTM_Error

Processing is performed according to the occurrence of a count error interrupt.

備考　本 API 関数は、r_gtm_error_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]

```
void    r_<Config_TIMn_m>_Callback_GTM_Error(void);
```

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TIMn_m>_share_interrupt

Performs processing in response to the TIMn channelm interrupt.

[指定形式]

void	r_<Config_TIMn_m>_share_interrupt (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

3.2.47 TIM PWM Measurement Mode

以下に、コード生成ツールが TIM PWM Measurement Mode 用に出力する API 関数の一覧を示します。

Generic Timer Module 用の API 関数の説明は英語だけとなります。

表 3-48 TIM PWM Measurement Mode 用 API 関数

API 関数名	機能概要
R_<Config_TIMn_m>_Create	Performs initialization necessary to control the TIM pwm measurement mode.
R_<Config_TIMn_m>_Start	Starts the count for channel m in unit n.
R_<Config_TIMn_m>_Stop	Stops the count for channel m in unit n.
R_<Config_TIMn_m>_Software_Reset	Stops channel operation and reset channel registers.
R_<Config_TIMn_m>_NEWVAL_TriggerOn	Trigger NEWVAL bit in TIMn_CHm_IRQ_NOTIFY register by software.
R_<Config_TIMn_m>_ECNTOFL_TriggerOn	Trigger ECNTOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.
R_<Config_TIMn_m>_CNTOFL_TriggerOn	Trigger CNTOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.
R_<Config_TIMn_m>_GPROFL_TriggerOn	Trigger GPROFL bit in TIMn_CHm_IRQ_NOTIFY register by software.
R_<Config_TIMn_m>_TODETOFL_TriggerOn	Trigger TODETOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.
R_<Config_TIMn_m>_GLITCHDETOFL_TriggerOn	Trigger GLITCHDET bit in TIMn_CHm_IRQ_NOTIFY register by software.
R_<Config_TIMn_m>_Create_UserInit	Performs user-defined initialization relating to the TIM pwm measurement mode.
r_<Config_TIMn_m>_Callback_GTM_Error	Processing is performed according to the occurrence of a count error interrupt.
r_<Config_TIMn_m>_share_interrupt	Performs processing in response to the TIMn channelm interrupt.

R_<Config_TIMn_m>_Create

Performs initialization necessary to control the TIM pwm measurement mode.

備考　本 API 関数は、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

```
void    R_<Config_TIMn_m>_Create ( void );
```

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_Start

Starts the count for channel m in unit n.

[指定形式]

void	R_<Config_TIMn_m>_Start (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_Stop

Stops the count for channel m in unit n.

[指定形式]

void	R_<Config_TIMn_m>_Stop (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_Software_Reset

Stops channel operation and reset channel registers.

[指定形式]

void	R_<Config_TIMn_m>_Software_Reset (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_NEWVAL_TriggerOn

Trigger NEWVAL bit in TIMn_CHm_IRQ_NOTIFY register by software.

[指定形式]

void	R_<Config_TIMn_m>_NEWVAL_TriggerOn (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_ECNTOFL_TriggerOn

Trigger ECNTOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.

[指定形式]

void	R_<Config_TIMn_m>_ECNTOFL_TriggerOn (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_CNTOFL_TriggerOn

Trigger CNTOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.

[指定形式]

void	R _Config_TIM*n*_*m*_CNTOFL_TriggerOn (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_GPROFL_TriggerOn

Trigger GPROFL bit in TIMn_CHm_IRQ_NOTIFY register by software.

[指定形式]

void	R_<Config_TIMn_m>_GPROFL_TriggerOn (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_TODETOFL_TriggerOn

Trigger TODETOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.

[指定形式]

void	R_<Config_TIMn_m>_TODETOFL_TriggerOn (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_GLITCHDETOFL_TriggerOn

Trigger GLITCHDETOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.

[指定形式]

void	R_<Config_TIMn_m>_GLITCHDETOFL_TriggerOn (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_Create_UserInit

Performs user-defined initialization relating to the TIM pwm measurement mode.

備考　本 API 関数は、R_<Config_TIMn_m>_Create から呼び出されます。

[指定形式]

```
void    R_<Config_TIMn_m>_Create_UserInit ( void );
```

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TIMn_m>_Callback_GTM_Error

Processing is performed according to the occurrence of a count error interrupt.

備考　本 API 関数は、r_gtm_error_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]

```
void    r_<Config_TIMn_m>_Callback_GTM_Error(void);
```

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TIMn_m>_share_interrupt

Performs processing in response to the TIMn channelm interrupt.

[指定形式]

void	r_<Config_TIMn_m>_share_interrupt (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

3.2.48 TIM Serial Shift Mode

以下に、コード生成ツールが TIM Serial Shift Mode 用に出力する API 関数の一覧を示します。
Generic Timer Module 用の API 関数の説明は英語だけとなります。

表 3-49 TIM Serial Shift Mode 用 API 関数

API 関数名	機能概要
R_<Config_TIMn_m>_Create	Performs initialization necessary to control the TIM serial shift mode.
R_<Config_TIMn_m>_Start	Starts serial shift for channel m in unit n.
R_<Config_TIMn_m>_Stop	Stops serial shift for channel m in unit n.
R_<Config_TIMn_m>_UpdateElapsedClock	Updates elapsed clock number.
R_<Config_TIMn_m>_UpdateShiftClock	Updates shift clock number.
R_<Config_TIMn_m>_UpdateCaptrueSource	Updates capture source.
R_<Config_TIMn_m>_UpdateTssmOut	Updates tssm out signal.
R_<Config_TIMn_m>_SetGPR0AsShadowRegister	Sets TIM[i]_CH[x]_GPR0 as a shadow register for TIM[i]_CH[x]_CNT
R_<Config_TIMn_m>_Software_Reset	Stops channel operation and reset channel registers.
R_<Config_TIMn_m>_NEWVAL_TriggerOn	Trigger NEWVAL bit in TIMn_CHm_IRQ_NOTIFY register by software.
R_<Config_TIMn_m>_ECNTOFL_TriggerOn	Trigger ECNTOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.
R_<Config_TIMn_m>_CNTOFL_TriggerOn	Trigger CNTOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.
R_<Config_TIMn_m>_GPROFL_TriggerOn	Trigger GPROFL bit in TIMn_CHm_IRQ_NOTIFY register by software.
R_<Config_TIMn_m>_TODETOFL_TriggerOn	Trigger TODETOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.
R_<Config_TIMn_m>_GLITCHDETOFL_TriggerOn	Trigger GLITCHDET bit in TIMn_CHm_IRQ_NOTIFY register by software.
R_<Config_TIMn_m>_Create_UserInit	Performs user-defined initialization relating to the TIM serial shift mode.
r_<Config_TIMn_m>_Callback_GTM_Error	Processing is performed according to the occurrence of a count error interrupt.
r_<Config_TIMn_m>_share_interrupt	Performs processing in response to the TIMn channelm interrupt.

R_<Config_TIMn_m>_Create

Performs initialization necessary to control the TIM serial shift mode.

備考　本 API 関数は、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

void	R_<Config_TIMn_m>_Create (void);

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_Start

Starts the count for channel m in unit n.

[指定形式]

void	R_<Config_TIMn_m>_Start (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_Stop

Stops the count for channel m in unit n.

[指定形式]

void	R_<Config_TIMn_m>_Stop (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_UpdateElapsedClock

Updates elapsed clock number.

[指定形式]

void	R_<Config_TIMn_m>_UpdateElapsedClock (uint8_t *number*);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

I/O	引数	説明
I	uint8_t *number*;	Elapsed clock number.

[戻り値]

なし

R_<Config_TIMn_m>_UpdateShiftClock

Updates shift clock number.

[指定形式]

void	R_<Config_TIMn_m>_UpdateShiftClock (uint8_t *number*);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

I/O	引数	説明
I	uint8_t *number*;	Shift clock number.

[戻り値]

なし

R_<Config_TIMn_m>_UpdateCaptrueSource

Updates capture source.

[指定形式]

void	R_<Config_TIMn_m>_UpdateCaptrueSource (uint8_t *number*);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

I/O	引数	説明
I	uint8_t *number*;	Capture source.

[戻り値]

なし

R_<Config_TIMn_m>_UpdateTssmOut

Updates tssm out.

[指定形式]

void	R_<Config_TIMn_m>_UpdateTssmOut (uint8_t *number*);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

I/O	引数	説明
I	uint8_t *number*;	Tssm out signal.

[戻り値]

なし

R_<Config_TIMn_m>_SetGPR0AsShadowRegister

Sets TIM[i]_CH[x]_GPR0 as a shadow register for TIM[i]_CH[x]_CNT.

[指定形式]

void R_<Config_TIMn_m>_SetGPR0AsShadowRegister (char *number*);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

I/O	引数	説明
I	char *number*;	Sets whether to use GPR0 as the shadow register of the CNT register.

[戻り値]

なし

R_<Config_TIMn_m>_Software_Reset

Stops channel operation and reset channel registers.

[指定形式]

void	R_<Config_TIMn_m>_Software_Reset (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_NEWVAL_TriggerOn

Trigger NEWVAL bit in TIMn_CHm_IRQ_NOTIFY register by software.

[指定形式]

void	R_<Config_TIMn_m>_NEWVAL_TriggerOn (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_ECNTOFL_TriggerOn

Trigger ECNTOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.

[指定形式]

void	R_<Config_TIMn_m>_ECNTOFL_TriggerOn (void);

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_CNTOFL_TriggerOn

Trigger CNTOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.

[指定形式]

void	R _Config_TIM*n*_*m*_CNTOFL_TriggerOn (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_GPROFL_TriggerOn

Trigger GPROFL bit in TIMn_CHm_IRQ_NOTIFY register by software.

[指定形式]

void	R_<Config_TIMn_m>_GPROFL_TriggerOn (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_TODETOFL_TriggerOn

Trigger TODETOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.

[指定形式]

void	R_<Config_TIMn_m>_TODETOFL_TriggerOn (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_GLITCHDETOFL_TriggerOn

Trigger GLITCHDETOFL bit in TIMn_CHm_IRQ_NOTIFY register by software.

[指定形式]

void	R_<Config_TIMn_m>_GLITCHDETOFL_TriggerOn (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_TIMn_m>_Create_UserInit

Performs user-defined initialization relating to the TIM serial shift mode.

備考　本 API 関数は、R_<Config_TIMn_m>_Create から呼び出されます。

[指定形式]

```
void    R_<Config_TIMn_m>_Create_UserInit ( void );
```

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TIMn_m>_Callback_GTM_Error

Processing is performed according to the occurrence of a count error interrupt.

備考　本 API 関数は、r_gtm_error_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]

```
void    r_<Config_TIMn_m>_Callback_GTM_Error(void);
```

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_TIMn_m>_share_interrupt

Performs processing in response to the TIMn channelm interrupt.

[指定形式]

```
void        r_<Config_TIMn_m>_share_interrupt ( void );
```

備考　　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

3.2.49 ATOM Signal Output Mode Compare

以下に、コード生成ツールが ATOM Signal Output Mode Compare 用に出力する API 関数の一覧を示します。

Generic Timer Module 用の API 関数の説明は英語だけとなります。

表 3-50 ATOM Signal Output Mode Compare 用 API 関数

API 関数名	機能概要
R_<Config_ATOMn_m>_Create	Performs initialization necessary to control the ATOM signal output mode compare.
R_<Config_ATOMn_m>_Start	Starts ATOMnm signal output mode compare.
R_<Config_ATOMn_m>_Stop	Stop ATOMnm signal output mode compare.
R_<Config_ATOMn_*m*>_Output_Start	Enable ATOM*nm* output function of signal output mode compare
R_<Config_ATOMn_*m*>_Output_Stop	Disable ATOM*nm* output function of signal output mode compare
R_<Config_ATOMn_*m*>_Channel_Reset	Software reset of ATOM*nm* channel
R_<Config_ATOMn_*m*>_LateUpdate_Request	CPU Write request for ATOM*nm* late compare register update
R_<Config_ATOMn_m>_Create_UserInit	Performs user-defined initialization relating to the ATOM signal output mode compare functions.
R_<Config_ATOMn_m>_Callback_Shared_IRQk	Performs processing in response to the ATOMnm shared interrupt.

R_<Config_ATOMn_m>_Create

Performs initialization necessary to control the ATOM signal output mode compare.

備考　本 API 関数は、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

```
void    R_<Config_ATOMn_m>_Create ( void );
```

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ATOMn_m>_Start

Starts ATOMnm signal output mode compare.

[指定形式]

void	R_<Config_ATOMn_m>_Start (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ATOMn_m>_Stop

Stop ATOMnm signal output mode compare.

[指定形式]

void	R_<Config_ATOMn_m>_Stop (void);

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ATOM*n*_*m*>_Output_Start

Enable ATOMnm output function of signal output mode compare.

[指定形式]

void	R_<Config_ATOM*n*_*m*>_Output_Start (void);

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ATOM*n*_*m*>_Output_Stop

Disable ATOMnm output function of signal output mode compare.

[指定形式]

void	R_<Config_ATOM*n*_*m*>_Output_Stop (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ATOM*n*_*m*>_Channel_Reset

Software reset of ATOMnm channel.

[指定形式]

void	R_<Config_ATOM*n*_*m*>_Channel_Reset (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ATOM*n*_*m*>_LateUpdate_Request

Software reset of ATOMnm channel.

[指定形式]

void	R_<Config_ATOM*n*_*m*>_LateUpdate_Request (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ATOMn_m>_Create_UserInit

Performs user-defined initialization relating to the ATOM signal output mode compare functions.

備考　本 API 関数は、R_<Config_ATOMn_m>_Create から呼び出されます。

[指定形式]

```
void    R_<Config_ATOMn_m>_Create_UserInit ( void );
```

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ATOMn_m>_Callback_Shared_IRQ*k*

Performs processing in response to the ATOMnm shared interrupt.

備考　本 API 関数は、r_atomn_irqk_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]

void	R_<Config_ATOMn_m>_Callback_Shared_IRQ*k* (void);

備考　*n* はユニット番号を、*m* はチャネル番号、*k* は割り込みチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

3.2.50 ATOM Signal Output Mode Immediate

以下に、コード生成ツールが ATOM Signal Output Mode Immediate 用に出力する API 関数の一覧を示します。

Generic Timer Module 用の API 関数の説明は英語だけとなります。

表 3-51 ATOM Signal Output Mode Immediate 用 API 関数

API 関数名	機能概要
R_<Config_ATOMn_m>_Create	Performs initialization necessary to control the ATOM signal output mode immediate.
R_<Config_ATOMn_m>_Start	Starts ATOMnm signal output mode immediate.
R_<Config_ATOMn_m>_Stop	Stop ATOMnm signal output mode immediate.
R_<Config_ATOMn_*m*>_Output_Start	Enable ATOM*nm* output function of signal output mode immediate
R_<Config_ATOMn_m>_Output_Stop	Disable ATOM*nm* output function of signal output mode immediate
R_<Config_ATOMn_m>_Channel_Reset	Software reset of ATOM*nm* channel
R_<Config_ATOMn_m>_Create_UserInit	Performs user-defined initialization relating to the ATOM signal output mode immediate functions.

R_<Config_ATOMn_m>_Create

Performs initialization necessary to control the ATOM signal output mode immediate.

備考　本 API 関数は、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

```
void    R_<Config_ATOMn_m>_Create ( void );
```

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ATOMn_m>_Start

Starts ATOMnm signal output mode immediate.

[指定形式]

void	R_<Config_ATOMn_m>_Start (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ATOMn_m>_Stop

Stop ATOMnm signal output mode immediate.

[指定形式]

void	R_<Config_ATOMn_m>_Stop (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ATOM*n*_*m*>_Output_Start

Enable ATOMnm output function of signal output mode compare.

[指定形式]

void	R_<Config_ATOM*n*_*m*>_Output_Start (void);

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ATOM*n*_*m*>_Output_Stop

Disable ATOMnm output function of signal output mode compare.

[指定形式]

void	R_<Config_ATOM*n*_*m*>_Output_Stop (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ATOM*n*_*m*>_Channel_Reset

Software reset of ATOMnm channel.

[指定形式]

void	R_<Config_ATOM*n*_*m*>_Channel_Reset (void);

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ATOMn_m>_Create_UserInit

Performs user-defined initialization relating to the ATOM signal output mode immediate functions.

備考　本 API 関数は、R_<Config_ATOMn_m>_Create から呼び出されます。

[指定形式]

```
void    R_<Config_ATOMn_m>_Create_UserInit ( void );
```

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

3.2.51 ATOM Signal Output Mode PWM

以下に、コード生成ツールが ATOM Signal Output Mode PWM 用に出力する API 関数の一覧を示します。

Generic Timer Module 用の API 関数の説明は英語だけとなります。

表 3-52 ATOM Signal Output Mode PWM 用 API 関数

API 関数名	機能概要
R_<Config_ATOMn_m>_Create	Performs initialization necessary to control the ATOM signal output mode PWM.
R_<Config_ATOMn_m>_Start	Starts ATOMnm signal output mode PWM.
R_<Config_ATOMn_m>_Stop	Stop ATOMnm signal output mode PWM.
R_<Config_ATOMn_*m*>_Output_Start	Enable ATOM*nm* output function of signal output mode PWM
R_<Config_ATOMn_m>_Output_Stop	Disable ATOM*nm* output function of signal output mode PWM
R_<Config_ATOMn_m>_Channel_Reset	Software reset of ATOM*nm* channel
R_<Config_ATOMn_m>_Create_UserInit	Performs user-defined initialization relating to the ATOM signal output mode PWM functions.
R_<Config_ATOMn_m>_Callback_Shared_IRQk	Performs processing in response to the ATOMnm shared interrupt.

R_<Config_ATOMn_m>_Create

Performs initialization necessary to control the ATOM signal output mode PWM.

備考　本 API 関数は、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

void	R_<Config_ATOMn_m>_Create (void);

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ATOMn_m>_Start

Starts ATOMnm signal output mode PWM.

[指定形式]

void	R_<Config_ATOMn_m>_Start (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ATOMn_m>_Stop

Stop ATOMnm signal output mode PWM.

[指定形式]

void	R_<Config_ATOMn_m>_Stop (void);

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ATOM*n*_*m*>_Output_Start

Enable ATOMnm output function of signal output mode compare.

[指定形式]

void	R_<Config_ATOM*n*_*m*>_Output_Start (void);

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ATOM*n*_*m*>_Output_Stop

Disable ATOMnm output function of signal output mode compare.

[指定形式]

void	R_<Config_ATOM*n*_*m*>_Output_Stop (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ATOM*n*_*m*>_Channel_Reset

Software reset of ATOMnm channel.

[指定形式]

void	R_<Config_ATOM*n*_*m*>_Channel_Reset (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ATOMn_m>_Create_UserInit

Performs user-defined initialization relating to the ATOM signal output mode PWM functions.

備考　本 API 関数は、R_<Config_ATOMn_m>_Create から呼び出されます。

[指定形式]

```
void    R_<Config_ATOMn_m>_Create_UserInit ( void );
```

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ATOMn_m>_Callback_Shared_IRQ*k*

Performs processing in response to the ATOMnm shared interrupt.

備考　本 API 関数は、r_atomn_irqk_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]

void	R_<Config_ATOM*n*_*m*>_Callback_Shared_IRQ*k* (void);

備考　*n* はユニット番号を、*m* はチャネル番号を、*k* は割り込みチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

3.2.52 ATOM Signal Output Mode Serial

以下に、コード生成ツールが ATOM Signal Output Mode Serial 用に出力する API 関数の一覧を示します。

Generic Timer Module 用の API 関数の説明は英語だけとなります。

表 3-53 ATOM Signal Output Mode Serial 用 API 関数

API 関数名	機能概要
R_<Config_ATOMn_m>_Create	Performs initialization necessary to control the ATOM signal output mode serial.
R_<Config_ATOMn_m>_Start	Starts ATOMnm signal output mode serial.
R_<Config_ATOMn_m>_Stop	Stop ATOMnm signal output mode serial.
R_<Config_ATOMn_*m*>_Output_Start	Enable ATOM*nm* output function of signal output mode PWM
R_<Config_ATOMn_m>_Output_Stop	Disable ATOM*nm* output function of signal output mode PWM
R_<Config_ATOMn_m>_Channel_Reset	Software reset of ATOM*nm* channel
R_<Config_ATOMn_m>_Create_UserInit	Performs user-defined initialization relating to the ATOM signal output mode serial functions.
R_<Config_ATOMn_m>_Callback_Shared_IRQk	Performs processing in response to the ATOMnm shared interrupt.

R_<Config_ATOMn_m>_Create

Performs initialization necessary to control the ATOM signal output mode serial.

備考　本 API 関数は、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

```
void    R_<Config_ATOMn_m>_Create ( void );
```

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ATOMn_m>_Start

Starts ATOMnm signal output mode serial.

[指定形式]

void	R_<Config_ATOMn_m>_Start (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ATOMn_m>_Stop

Stop ATOMnm signal output mode serial.

[指定形式]

void	R_<Config_ATOMn_m>_Stop (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ATOM*n*_*m*>_Output_Start

Enable ATOMnm output function of signal output mode compare.

[指定形式]

void	R_<Config_ATOM*n*_*m*>_Output_Start (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ATOM*n*_*m*>_Output_Stop

Disable ATOMnm output function of signal output mode compare.

[指定形式]

void	R_<Config_ATOM*n*_*m*>_Output_Stop (void);

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ATOM*n*_*m*>_Channel_Reset

Software reset of ATOMnm channel.

[指定形式]

void	R_<Config_ATOM*n*_*m*>_Channel_Reset (void);

備考 *n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ATOMn_m>_Create_UserInit

Performs user-defined initialization relating to the ATOM signal output mode serial functions.

備考　本 API 関数は、R_<Config_ATOMn_m>_Create から呼び出されます。

[指定形式]

```
void    R_<Config_ATOMn_m>_Create_UserInit ( void );
```

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_ATOMn_m>_Callback_Shared_IRQ*k*

Performs processing in response to the ATOMnm shared interrupt.

備考　本 API 関数は、r_atomn_irqk_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]

void	R_<Config_ATOM*n*_*m*>_Callback_Shared_IRQ*k* (void);

備考　*n* はユニット番号を、*m* はチャネル番号を、*k* は割り込みチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

3.2.53 Dead Time Module

以下に、コード生成ツールが Dead Time Module 用に出力する API 関数の一覧を示します。
Generic Timer Module 用の API 関数の説明は英語だけとなります。

表 3-54 Dead Time Module 用 API 関数

API 関数名	機能概要
R_<Config_CDTMn_m>_Create	Performs initialization necessary to control the dead time module.
R_<Config_CDTMn_m>_Create_UserInit	Performs user-defined initialization relating to the dead time module.

R_<Config_CDTMn_m>_Create

Performs initialization necessary to control the ATOM signal output mode serial.

備考　本 API 関数は、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

```
void    R_<Config_CDTMn_m>_Create ( void );
```

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_CDTMn_m>_Create_UserInit

Performs user-defined initialization relating to the ATOM signal output mode serial functions.

備考　本 API 関数は、R_<Config_CDTMn_m>_Create から呼び出されます。

[指定形式]

```
void    R_<Config_CDTMn_m>_Create_UserInit ( void );
```

備考　*n* はユニット番号を、*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

3.2.54 DTS コントローラ

以下に、コード生成ツールが DTS コントローラ用に出力する API 関数の一覧を示します。

表 3-55 DTS コントローラ用 API 関数

API 関数名	機能概要
R_<Config_DTSn>_Create	DTS 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。
R_<Config_DTSn>_Start	DTS コントローラの制御を許可します。
R_<Config_DTSn>_Stop	DTS コントローラの制御を停止します。
R_<Config_DTSn>_Set_SoftwareTrigger	チャネルの転送要求(ソフトウェアトリガ)を発生します。
R_<Config_DTSn>_Create_UserInit	DTS 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。
r_<Config_DTSn>_Callback_Dtsg_Transfer_End	チャネルの転送完了割り込みに伴う処理を行います。
r_<Config_DTSn>_Callback_Dtsg_Transfer_Count_Match	チャネルの転送回数一致割り込みに伴う処理を行います。
r_<Config_DTSn>_Callback_PEk_Transfer_Error	チャネルの転送エラー割り込みに伴う処理を行います。
r_<Config_DTSn>_Callback_Transfer_Error	チャネルの転送エラー割り込みに伴う処理を行います。

R_<Config_DTSn>_Create

DTS 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。

備考　本 API 関数は、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

void	R_<Config_DTSn>_Create (void);

備考　*n* はチェーンヘッダのチャンネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_DTSn>_Start

DTS コントローラの制御を許可します。

[指定形式]

void	R_<Config_DTSn>_Start (void);

備考 *n* はチェーンヘッダのチャンネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_DTSn>_Stop

DTS コントローラの制御を停止します。

[指定形式]

void	R_<Config_DTSn>_Stop (void);

備考 *n* はチェーンヘッダのチャンネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_DTSn>_Set_SoftwareTrigger

チャネルの転送要求(ソフトウェアトリガ)を発生します。

[指定形式]

void	R_<Config_DTSn>_Set_SoftwareTrigger (void);

備考 *n* はチェーンヘッダのチャンネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_<Config_DTSn>_Create_UserInit

DTS 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_<Config_DTSn>_Create から呼び出されます。

[指定形式]

void	R_<Config_DTSn>_Create_UserInit (void);

備考　*n* はチェーンヘッダのチャンネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_DTSn>_Callback_Dts*g*_Transfer_End

チャネルの転送完了割り込みに伴う処理を行います。

備考　本 API 関数は、r_dtsg_transfer_end_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]

void	r_<Config_DTS*n*>_Callback_Dts*g*_Transfer_End(void);

備考　*n* はチェーンヘッダのチャンネル番号を、*g* は”31_0”、“63_32”、“95_64”、“127_96”の割り込みグループを意味します。

[引数]

なし

[戻り値]

なし

r_<Config_DTSn>_Callback_Dtsg_Transfer_Count_Match

チャネルの転送回数一致割り込みに伴う処理を行います。

備考　本 API 関数は、r_dtsg_transfer_count_match_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]

```
void    r_<Config_DTSn>_Callback_Dtsg_Transfer_Count_Match(void);
```

備考　*n* はチェーンヘッダのチャンネル番号を、*g* は”31_0”、“63_32”、“95_64”、“127_96”の割り込みグループ番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_DTSn>_Callback_PE*k*_Transfer_Error

チャネルの転送エラー割り込みに伴う処理を行います。

備考　本 API 関数は、r_dts_transfer_error_interrupt_pek 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_<Config_DTSn>_Callback_PEk_Transfer_Error(void)

備考　*n* はチェーンヘッダのチャンネル番号を、*k* は CPU コア番号を意味します。

[引数]

なし

[戻り値]

なし

r_<Config_DTS*n*>_Callback_Transfer_Error

チャネルの転送エラー割り込みに伴う処理を行います。

備考　本 API 関数は、r_dts_transfer_error_interrupt_pek 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_<Config_DTS*n*>_Callback_Transfer_Error(void)

備考　*n* はチェーンヘッダのチャンネル番号を意味します。

[引数]

なし

[戻り値]

なし

3.2.55 ADC Boundary Flag Generator

以下に、コード生成ツールが 3.2.55 ADC Boundary Flag Generator 用に出力する API 関数の一覧を示します。

表 3-56 DTS コントローラ用 API 関数

API 関数名	機能概要
R_<Config_DTSn>_Create	DTS 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。
R_<Config_DTSn>_Start	DTS コントローラの制御を許可します。
R_<Config_DTSn>_Stop	DTS コントローラの制御を停止します。
R_<Config_DTSn>_Set_SoftwareTrigger	チャネルの転送要求(ソフトウェアトリガ)を発生します。
R_<Config_DTSn>_Create_UserInit	DTS 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。
r_<Config_DTSn>_Callback_Dtsg_Transfer_End	チャネルの転送完了割り込みに伴う処理を行います。
r_<Config_DTSn>_Callback_Dtsg_Transfer_Count_Match	チャネルの転送回数一致割り込みに伴う処理を行います。
r_<Config_DTSn>_Callback_PEk_Transfer_Error	チャネルの転送エラー割り込みに伴う処理を行います。
r_<Config_DTSn>_Callback_Transfer_Error	チャネルの転送エラー割り込みに伴う処理を行います。

R_<Config_ABFG>_Create

ADC Boundary Flag Generator 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、main()関数内にて、R_Systeminit から呼び出されます。

[指定形式]

```
void    R_<Config_ABFG>_Create(void);
```

[引数]

なし

[戻り値]

なし

R_<Config_ABFG>_Start

ABFG の動作を開始します。

[指定形式]

```
void      R_<Config_ABFG>_Start(void);
```

[引数]

なし

[戻り値]

なし

R_<Config_ABFG>_Stop

ABFG の動作を停止します。

[指定形式]

```
void      R_<Config_ABFG>_Stop(void);
```

[引数]

なし

[戻り値]

なし

R_<Config_ABFG>_Create_UserInit

ABFG 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_<Config_ABFG>_Create から呼び出されます。

[指定形式]

```
void    R_<Config_ABFG>_Create_UserInit(void);
```

[引数]

なし

[戻り値]

なし

r_<Config_ABFG>_boundary_flag_pulse_*w*_interrupt

チャネルの転送エラー割り込みに伴う処理を行います。

[指定形式]

void r_<Config_ABFG>_boundary_flag_pulse_*w*_interrupt(void);

備考 *w* は boundary flag 番号を意味します。

[引数]

なし

[戻り値]

なし

改訂記録

Rev.	章	改訂内容
1.00	ー	初版発行
1.10	ー	欠番
1.20		API 名に"<>"を追加
		クロック分周の API 'start' 'stop' の名称を変更
		DMA コントローラの重複 API を削除
		ATOM Signal Output Mode PWM と ATOM Signal Output Mode Serial に "r_<Config_ATOMn_m>_callback_shared_interrupt"を追加
		3.2.1 共　　通 に API を追加
		周辺機能を追加
1.30		CSI スレーブと CSI マスタに"R_<Config_CSIHn>_Send_Receive"を追加
		引数"tx_buf"に const を追加
		引数"rx_buf"に const を追加
		備考 3 を追加
1.40	表 2-1	下記のファイルを追加 r_cg_ad_air.h r_cg_abfg.h r_cg_ad_common.c
	3.2.1	下記の関数を追加 Following functions are added: R_ADC_SyncStart, R_ADC_TimerSyncStart, R_DMAC_Create, R_DMA_Suspend, R_DMA_Resume, R_PDMAn_Suspend, R_PDMAn_Resume, R_DTS_Create, R_DMA_Start_Transfer_Error_Interrupt, R_DMA_Stop_Transfer_Error_Interrupt r_dmac_error_interrupt_pe1, R_MSPI_Master_Create 下記の関数の「備考」を更新 R_GTM_Start_Interrupt_Error R_GTM_Stop_Interrupt_Error R_ATOM*n*_Start_Interrupt_IRQ*k* R_ATOM*n*_Stop_Interrupt_IRQ*k*

	3.2.2	下記の関数を追加 R_<Config_ADCXn>_ScanGroupx_OperationOff, R_<Config_ADCXn>_TH_Sampling_Stop R_<Config_ADCXn>_TH_Stop R_<Config_ADCXn>_VoltageDivider_Start R_<Config_ADCXn>_VoltageDivider_Stop R_<Config_ADCXn>_StrongPullDown_Start R_<Config_ADCXn>_StrongPullDown_Stop R_<Config_ADCXn>_SGDiag_Start R_<Config_ADCXn>_SGDiag_OperationOn R_<Config_ADCXn>_SGDiag_OperationOff R_<Config_ADCXn>_SGDiag_GetResult R_<Config_ADCXn>_SGDiag_GetSubtractionResult R_<Config_ADCXn>_ASF_Start R_<Config_ADCXn>_ASF_Stop R_<Config_ADCXn>_ASF_GetResult R_<Config_ADCXn>_ASF_GetRealTimeResult r_<Config_ADCXn>_sg_diag_end_interrupt r_<Config_ADCXn>_mpx_request_interrupt r_<Config_ADCXn>_asf_channelk_end_interrupt 下記の関数を削除 R_<Config_ADCXn>_SyncStart R_<Config_ADCXn>_SyncTimerStart 下記の関数名を変更 [変更前]　R_<Config_ADCXn>_ScanGroupx_GetFloatingPointDataResult [変更後]　R_<Config_ADCXn>_ScanGroupx_GetFloatingPointResult
	3.2.3	下記の関数を追加。 R_<Config_CSIGn>_Send_Receive
	3.2.4	下記の関数を追加 R_<Config_CSIGn>_Send_Receive
	3.2.5	下記の関数を追加 R_<Config_INTC>_IRQn_Start R_<Config_INTC>_IRQn_Stop r_<Config_INTC>_intpn_interrupt_pek r_<Config_INTC>_irqn_interrupt
	3.2.14	下記の関数を追加 R_<Config_STBC>_Prepare_Stop_Mode_Set_CPUCLK R_<Config_STBC>_Prepare_Deep_Stop_Mode_Set_Clock_Mask R_<Config_STBC>_Prepare_Deep_Stop_Mode_Set_Clock_Source
	3.2.16	下記の関数を追加 R_<Config_WDTAn>_Create R_<Config_WDTAn>_Restart R_<Config_WDTAn>_Create_UserInit R_<Config_WDTAn>_Inerrupt

	3.2.18	下記の関数を追加 R_<Config_DCRAn>_InitializeCRCData R_<Config_KCRCn>_InitializeCRCData
	3.2.20	下記の関数を追加 R_<Config_SDMACnm>_Reset r_<Config_SDMACnm>_Callback_PEx_Address_Error 下記の関数名を変更 [変更前]　r_<Config_DMACnm>_interrupt [変更後]　r_<Config_DMACnm>_dmacnm_interrupt
	3.2.26	下記の関数名を変更 [変更前]　r_<Config_MSPInm>_callback_send [変更後]　r_<Config_MSPInm>_Callback_Interrupt_Send [変更前]　r_<Config_MSPInm>_callback_receive [変更後]　r_<Config_MSPInm>_Callback_Interrupt_Receive [変更前]　r_<Config_MSPInm>_interrupt_error [変更後]　r_<Config_MSPInm>_Callback_Interrupt_Error [変更前]　r_<Config_MSPInm>_callback_frameend [変更後]　r_<Config_MSPInm>_Callback_Interrupt_Frameend
	3.2.39	下記の関数を追加 R_<Config_ECM>_Set_Master_Error R_<Config_ECM>_Set_Checker_Error R_<Config_ECM>_Clear_Master_Error R_<Config_ECM>_Clear_Checker_Error R_<Config_ECM>_Delay_Timer_Start R_<Config_ECM>_Delay_Timer_Stop r_<Config_ECM>_mi_interrupt_pek
	3.2.42 3.2.43 3.2.44 3.2.45 3.2.46 3.2.47 3.2.48	下記の関数名を変更 [変更前]　r_<Config_TIMn>_callback_gtm_error [変更後]　r_<Config_TIMn>_Callback_GTM_Error
	3.2.49 3.2.51 3.2.52	下記の関数名を変更 [変更前]　r_<Config_ATOMn_m>_callback_shared_interrupt [変更後]　r_<Config_ATOMn_m>_Callback_Shared_IRQk

	3.2.54	下記の関数を追加 r_<Config_DTSn>_Callback_Transfer_Error 下記の関数を削除 R_<Config_DTSn>_Enable_PEk_Interrupt R_<Config_DTSn>_Disable_PEk_Interrupt 下記の関数名を変更 [変更前]　r_<Config_DTSn>_callback_dtsg_transfer_end_interrupt [変更後]　r_<Config_DTSn>_Callback_Dtsg_Transfer_End [変更前]　r_<Config_DTSn>_callback_dtsg_transfer_count_match_interrupt [変更後]　r_<Config_DTSn>_Callback_Dtsg_Transfer_Count_Match [変更前]　r_<Config_DTSn>_callback_pek_transfer_error_interrupt [変更後]　r_<Config_DTSn>_Callback_PEk_Transfer_Error
	3,2,55	「ADC Boundary Flag Generator」用に下記の関数を追加 R_<Config_ABFG>_Create R_<Config_ABFG>_Start R_<Config_ABFG>_Stop R_<Config_ABFG>_Create_UserInit r_<Config_ABFG>_boundary_flag_pulse_*w*_interrupt
	3.2.3 3.2.4 3.2.14 3.2.15 3.2.32 3.2.33 3.2.34 3.2.35 3.2.36 3.2.37	「使用例」を追加
1.50	2	下記の関数を削除。 R_<Config_ADCXn>_TH_Sampling_Stop 下記の関数を追加。 R_<Config_ADCXn>_SelfDiagnostic_VoltageCircuitOn R_<Config_ADCXn>_SelfDiagnostic_VoltageCircuitOff R_<Config_ADCXn>_ScanGroupx_GetPWMDiagResult R_<Config_TAUXn>_SoftwareTriggerOn 下記のファイルを追加。 r_smc_entry.h

	3	下記の関数を削除。 R_<Config_ADCXn>_TH_Sampling_Stop 下記の関数を追加。 R_<Config_ADCXn>_SelfDiagnostic_VoltageCircuitOn R_<Config_ADCXn>_SelfDiagnostic_VoltageCircuitOff R_<Config_ADCXn>_ScanGroupx_GetPWMDiagResult R_<Config_TAUXn>_SoftwareTriggerOn
1.60	3	下記の関数を追加。 R_<Config_ATOMn_m>_Output_Start R_<Config_ATOMn_m>_Output_Stop R_<Config_ATOMn_m>_Channel_Reset R_<Config_ATOMn_m>_LateUpdate_Request 下記の関数の誤記を修正。 [誤]　r_<Config_ATOM*n_m*>_Callback_Shared_IRQ*k* [正]　R_<Config_ATOM*n_m*>_Callback_Shared_IRQ*k*

スマート・コンフィグレータ ユーザーズマニュアル
RH850 APIリファレンス編

発行年月日　2018年7月13日　Rev.1.00
　　2024年1月20日　Rev.1.60

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）

スマート・コンフィグレータ

ルネサス エレクトロニクス株式会社

R20UT4361JJ0160