User's Manual

CS+ コード生成ツール

統合開発環境

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

対象ツール

CS+ V4.00.00

ルネサス エレクトロニクス
www.renesas.com

Rev.1.01　2016.03

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して、お客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
2. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
3. 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に関し、当社は、何らの責任を負うものではありません。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. 当社製品を改造、改変、複製等しないでください。かかる改造、改変、複製等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、
   各品質水準は、以下に示す用途に製品が使用されることを意図しております。

   標準水準：　コンピュータ、OA 機器、通信機器、計測機器、AV 機器、
   家電、工作機械、パーソナル機器、産業用ロボット等

   高品質水準：　輸送機器（自動車、電車、船舶等）、交通用信号機器、
   防災・防犯装置、各種安全装置等

   当社製品は、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（原子力制御システム、軍事機器等）に使用されることを意図しておらず、使用することはできません。たとえ、意図しない用途に当社製品を使用したことによりお客様または第三者に損害が生じても、当社は一切その責任を負いません。なお、ご不明点がある場合は、当社営業にお問い合わせください。
6. 当社製品をご使用の際は、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他の保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様が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。
9. 本資料に記載されている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。また、当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途に使用しないでください。当社製品または技術を輸出する場合は、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。
10. お客様の転売等により、本ご注意書き記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は何らの責任も負わず、お客様にてご負担して頂きますのでご了承ください。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。

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その総株主の議決権の過半数を直接または間接に保有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。

(2012. 4)

このマニュアルの使い方

このマニュアルは，RH850 ファミリ用アプリケーション・システムを開発する際の統合開発環境である CS+ について説明します。
CS+ は，RH850 ファミリの統合開発環境（ソフトウエア開発における，設計，実装，デバッグなどの各開発フェーズに必要なツールをプラットフォームである IDE に統合）です。統合することで，さまざまなツールを使い分ける必要がなく，本製品のみを使用して開発のすべてを行うことができます。

対象者	このマニュアルは，CS+ を使用してアプリケーション・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，CS+ の持つソフトウエア機能をユーザに理解していただき，これらのデバイス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。 1. 概　　説 2. 出力ファイル 3.API 関数
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

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

1.1 概　　要

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

1.2 特　　長

以下に，コード生成ツールの特長を示します。

- コード生成機能
  コード生成では，GUIベースで設定した情報に応じたデバイス・ドライバ・プログラムを出力するだけでなく，main関数を含んだサンプル・プログラム，リンク・ディレクティブ・ファイルなどといったビルド環境一式を出力することもできます。

- レポート機能
  端子配置／コード生成を用いて設定した情報を各種形式のファイルで出力し，設計資料として利用することができます。

- リネーム機能
  コード生成が出力するファイル名，およびソース・コードに含まれているAPI関数の関数名については，デフォルトの名前が付与されますが，ユーザ独自の名前に変更することもできます。

- ユーザ・コード保護機能
  各API関数には，ユーザが独自にコードを追加できるように，ユーザ・コード記述用のコメントが設けられています。

  [ユーザ・コード記述用のコメント]
  /* Start user code. Do not edit comment generated here */
  /* End user code. Do not edit comment generated here */
  このコメント内にコードを記述すると，再度コード生成した場合でもユーザが記述したコードは保護されます。

1.3 RH850用コード生成ツールについて

以下に，RH850用コード生成ツールの独自の仕様を示します。

- 同期化処理
  RH850用コード生成ツールは、各周辺のCreateとStopに同期化処理を出力しています。それ以外に必要な同期化処理はユーザが記述する必要があります

- 割り込みベクタ・テーブル
  RH850用コード生成ツールは、割り込みベクタ・テーブルの書き換えを行いません。生成した割り込み関数を使用する場合は、割り込みベクタ・テーブルを書き換えてください。(対象デバイス：RH850/E1L, E1M-S)

- RH850用コード生成のサンプル・プロジェクト
  コード生成のインストール・フォルダにあるSampleProjectsフォルダに専用のサンプル・プロジェクトがあります。サンプル・プロジェクトを利用することでプロジェクトを準備する工数を削減できます。

2. 出力ファイル

本章では，コード生成ツールが出力するファイルについて説明します。

2.1 説　　明

以下に，コード生成ツールが出力するファイルの一覧を示します。

表 2.1　　出力ファイル

周辺機能	ファイル名	API 関数名
共　　通	r_cg_main.c	main R_MAIN_UserInit
	r_cg_systeminit.c	R_SystemInit
	r_cg_macrodriver.h	—
	r_cg_userdefine.h	—
	r_cg_intvector.c	—
クロックコントローラ	r_cg_cgc.c	R_CGC_Create R_CGC_CK_Output_Enable R_CGC_CK_Output_Disable
	r_cg_cgc_user.c	R_CGC_Create_UserInit
	r_cg_cgc.h	—
ポート機能	r_cg_port.c	R_PORT_Create
	r_cg_port_user.c	R_PORT_Create_UserInit
	r_cg_port.h	—
割り込み	r_cg_intc.c	R_INTC_Create R_IRQn_Start R_IRQn_Stop R_SINTn_Start R_SINTn_Stop R_SINTn_TriggerOn R_INTPn_Start R_INTPn_Stop
	r_cg_intc_user.c	R_INTC_Create_UserInit r_nmi_interrupt r_irqn_interrupt r_sintn_interrupt r_intpn_interrupt
	r_cg_intc.h	—

周辺機能	ファイル名	API 関数名
DMAC	r_cg_dmac.c	R_DMACn_Create R_DMACn_Suspend R_DMACn_Resume R_DMACnm_Create R_DMACnm_Start R_DMACnm_Stop R_DMACnm_Set_SoftwareTrigger R_DMACnm_Suspend R_DMACnm_Resume
	r_cg_dmac_user.c	R_DMACn_Create_UserInit r_dmacnm_interrupt r_dmacnm_callback_transfer_completion r_dmacnm_callback_transfer_count_match
	r_cg_dmac.h	—
DTS	r_cg_dts.c	R_DTS_Create R_DTS_Suspend R_DTS_Resume R_DTS_All_Stop R_DTSx_y_Stop_Interrupt R_DTSm_Create R_DTSm_Start R_DTSm_Stop R_DTSm_Set_SoftwareTrigger R_DTSm_Suspend R_DTSm_Resume
	r_cg_dts_user.c	R_DTS_Create_UserInit R_DTSm_Create_UserInit r_dtsx_y_transfer_match_interrupt r_dtsx_y_transfer_completion_interrupt
	r_cg_dts.h	—
クロック同期シリアルインタフェース G	r_cg_csig.c	R_CSIGm_Create R_CSIGm_Start R_CSIGm_Stop R_CSIGm_Send R_CSIGm_Receive
	r_cg_csig_user.c	R_CSIGm_Create_UserInit r_csigm_interrupt_receive r_csigm_interrupt_error r_csigm_interrupt_send r_csigm_callback_receiveend r_csigm_callback_sendend r_csigm_callback_error
	r_cg_csig.h	—

周辺機能	ファイル名	API 関数名
クロック同期シリアルインタフェース H	r_cg_csih.c	R_CSIHm_Create R_CSIHm_Start R_CSIHm_Stop R_CSIHm_Master_Send R_CSIHm_Master_Receive R_CSIHm_Slave_Send R_CSIHm_Slave_Receive R_CSIHm_Extended_Data_Master_Send R_CSIHm_Extended_Data_Master_Receive R_CSIHm_Extended_Data_Slave_Send R_CSIHm_Extended_Data_Slave_Receive
	r_cg_csih_user.c	R_CSIHm_Create_UserInit r_csihm_interrupt_receive r_csihm_interrupt_error r_csihm_interrupt_send r_csihm_interrupt_jobend r_csihm_callback_receiveend r_csihm_callback_sendend r_csihm_callback_error
	r_cg_csih.h	—
シリアルコミュニケーションインタフェース 3	r_cg_sci3.c	R_SCI3m_Create R_SCI3m_Start R_SCI3m_Stop R_SCI3m_Send R_SCI3m_Receive R_SCI3m_Multiprocessor_Send R_SCI3m_Multiprocessor_Receive
	r_cg_sci3_user.c	R_SCI3m_Create_UserInit r_sci3m_interrupt_receive r_sci3m_interrupt_error r_sci3m_interrupt_send r_sci3m_interrupt_sendend r_sci3m_callback_receiveend r_sci3m_callback_sendend r_sci3m_callback_error
	r_cg_sci3.h	—
UART インタフェース	r_cg_uart.c	R_UARTm_Create R_UARTm_Start R_UARTm_Stop R_UARTm_Send R_UARTm_Receive
	r_cg_uart_user.c	R_UARTm_Create_UserInit r_uartm_interrupt_receive r_uartm_interrupt_error r_uartm_interrupt_send r_uartm_callback_receiveend r_uartm_callback_sendend r_uartm_callback_error
	r_cg_uart.h	—

周辺機能	ファイル名	API 関数名
ウィンドウウォッチドッグタイマ	r_cg_wdt.c	R_WDTm_Create R_WDTm_Restart
	r_cg_wdt_user.c	R_WDTm_Create_UserInit r_wdtm_interrupt
	r_cg_wdt.h	—
OS タイマ	r_cg_ostm.c	R_OSTMm_Create R_OSTMm_Start R_OSTMm_Stop R_OSTMm_Set_CompareValue
	r_cg_ostm_user.c	R_OSTMm_Create_UserInit r_ostmm_interrupt
	r_cg_ostm.h	—
アドバンストタイマユニットⅣ	r_cg_atuiv.c	R_ATUIV_Common_Create R_ATUIV_Timerkn_Create R_ATUIV_Timerk_OperationOn R_ATUIV_Timerk_OperationOff R_ATUIV_Timerknm_Start R_ATUIV_Timerknm_Stop R_ATUIV_Timerknm_Get_PulseWidth R_ATUIV_Timerknm_Get_CaptureValue R_ATUIV_Timerknm_Set_Compare_Match R_ATUIV_Timerknm_Set_One_Shot_Pulse R_ATUIV_Timerknm_Forced_Compare_Match R_ATUIV_Timerknm_Forced_Output_Compare_Match R_ATUIV_Timerknm_Start_Down_Count R_ATUIV_Timerknm_Get_InputCapturex R_ATUIV_Timerknm_xpin_Output_Nomal R_ATUIV_Timerknm_xpin_Output_Low R_ATUIV_Timerknm_xpin_Output_High R_ATUIV_Timerknm_Get_Count R_ATUIV_Timerknm_Get_PWM_Measure_Value R_ATUIV_Timerknm_Get_Measure_Value R_ATUIV_Timerknm_Reset_FIFO
	r_cg_atuiv_user.c	R_ATUIV_Common_Create_UserInit R_ATUIV_Timerkn_Create_UserInit r_atuiv_timerknm_overflow_interrupt r_atuiv_timerknm_interrupt r_atuiv_timerknm_icrnx_interrupt r_atuiv_timerknm_ocrnx_interrupt r_atuiv_timerknm_tcntnx_interrupt r_atuiv_timerknm_cmfnx_interrupt r_atuiv_timerknm_callback_ocrc r_atuiv_timerknm_callback_grc r_atuiv_timerknm_underflow_interrupt r_atuiv_timerknm_comparex_interrupt r_atuiv_timerknm_callback_overflow r_atuiv_timerknm_callback_cycle r_atuiv_timerknm_callback_duty r_atuiv_timerknm_fifo_overflow_interrupt r_atuiv_timerknm_fifo_datafull_interrupt r_atuiv_timerknm_tcntk_overflow_interrupt
	r_cg_atuiv.h	—

周辺機能	ファイル名	API 関数名
タイマ・アレイ・ユニット B	r_cg_taub.c	R_TAUBn_Create R_TAUBn_Channelm_Start R_TAUBn_Channelm_Stop R_TAUBn_Channelm_Get_PulseWidth
	r_cg_taub_user.c	R_TAUBn_Create_UserInit r_taubn_channelm_interrupt
	r_cg_taub.h	—
タイマ・アレイ・ユニット D	r_cg_taud.c	R_TAUDn_Create R_TAUDn_Channelm_Start R_TAUDn_Channelm_Stop R_TAUDn_Channelm_Get_PulseWidth
	r_cg_taud_user.c	R_TAUDn_Create_UserInit r_taudn_channelm_interrupt
	r_cg_taud.h	—
タイマ・アレイ・ユニット J	r_cg_tauj.c	R_TAUJn_Create R_TAUJn_Channelm_Start R_TAUJn_Channelm_Stop R_TAUJn_Channelm_Get_PulseWidth
	r_cg_tauj_user.c	R_TAUJn_Create_UserInit r_taujn_channelm_interrupt
	r_cg_tauj.h	—
タイマオプション	r_cg_tapa.c	R_TAPAm_Create R_TAPAm_Start R_TAPAm_Stop R_TAPAm_Trigger_Start R_TAPAm_Trigger_Stop
	r_cg_tapa_user.c	R_TAPAm_Create_UserInit
	r_cg_tapa.h	—
ペリフェラルインターコネクション	r_cg_pic.c	R_PICn_Create R_PICn_Timer_SyncStart
	r_cg_pic_user.c	R_PICn_Create_UserInit
	r_cg_pic.h	—

周辺機能	ファイル名	API 関数名
AD コンバータ	r_cg_adc.c	R_ADCn_Create R_ADCn_Halt R_ADCn_SetMultiplexerCommand R_ADCn_ScanGroupm_Start R_ADCn_ScanGroupm_GetResult R_ADCn_ScanGroupm_GetFloatingPointDataResult R_ADCn_ScanGroupm_TimerStart R_ADCn_ScanGroupm_TimerStop R_ADCn_ADCSummation_Channelm_GetResult R_ADCn_ADCSummation_Start R_ADCn_ADCSummation_Stop R_ADC_SyncStart R_ADC_SyncTimerStart R_ADCn_ScanGroupm_OperationOn R_ADCn_TH_Groupx_Start R_ADCn_TH_Sampling_Start
	r_cg_adc_user.c	R_ADCn_Create_UserInit r_adcn_error_interrupt r_adcn_scan_groupm_end_interrupt r_adcn_multiplexer_request_interrupt r_adcn_adc_summation_channelm_end_interrupt
	r_cg_adc.h	—
ΔΣAD コンバータ	r_cg_dsadc.c	R_DSADC_Create R_DSADC_SyncStart R_DSADCm_Start R_DSADCm_Stop R_DSADCm_GetResult
	r_cg_dsadc_user.c	R_DSADC_Create_UserInit r_dsadc_error_interrupt
	r_cg_dsadc.h	—
デジタルフィルタ	r_cg_dfe.c	R_DFE_Create R_DFE_Set_SoftwareData R_DFE_Generate_SoftwareTrigger R_DFE_Channelm_Create R_DFE_Channelm_Enable R_DFE_Channelm_Disable R_DFE_Channelm_GetResult
	r_cg_dfe_user.c	R_DFE_Create_UserInit r_dfe_error_interrupt R_DFE_Channelm_Create_UserInit r_dfe_channelm_interrupt r_dfe_channelm_callback_output_data r_dfe_channelm_callback_condition_match
	r_cg_dfe.h	—
データ CRC	r_cg_dcra.c	R_DCRAn_Create R_DCRAn_Input32bitData R_DCRAn_Input16bitData R_DCRAn_Input8bitData R_DCRAn_GetResult_32bitData R_DCRAn_GetResult_16bitData
	r_cg_dcra_user.c	R_DCRAn_Create_UserInit
	r_cg_dcra.h	—

周辺機能	ファイル名	API 関数名
リアルタイムクロック	r_cg_rtca.c	R_RTC_Create R_RTC_Start R_RTC_Stop R_RTC_Set_HourSystem R_RTC_Set_CounterValue R_RTC_Get_CounterValue R_RTC_Set_AlarmOn R_RTC_Set_AlarmOff R_RTC_Set_AlarmValue R_RTC_Get_AlarmValue R_RTC_Set_ConstPeriodInterruptOn R_RTC_Set_ConstPeriodInterruptOff R_RTC_Set_1secondInterruptOn R_RTC_Set_1secondInterruptOff R_RTC_Set_RTC1HZOn R_RTC_Set_RTC1HZOff
	r_cg_rtca_user.c	R_RTC_Create_UserInit r_rtc_interrupt_periodic r_rtc_interrupt_alarm r_rtc_interrupt_1second
	r_cg_rtca.h	—
キーリターン	r_cg_key.c	R_KEY_Create R_KEY_Start R_KEY_Stop
	r_cg_key_user.c	R_KEY_Create_UserInit r_key_interrupt
	r_cg_key.h	—
スタンバイコントローラ	r_cg_stbc.c	R_STBC_Start_Stop_Mode R_STBC_Prepare_Stop_Mode R_STBC_Start_Deep_Stop_Mode R_STBC_Prepare_Deep_Stop_Mode R_STBC_Deep_Stop_Loop
	r_cg_stbc_user.c	R_STBC_Prepare_Stop_Mode_Set_Peripheral R_STBC_Prepare_Stop_Mode_Set_Interrupt R_STBC_Prepare_Stop_Mode_Set_Clock_Mask R_STBC_Prepare_Stop_Mode_Set_Clock_Source R_STBC_Prepare_Deep_Stop_Mode_Set_Peripheral R_STBC_Prepare_Deep_Stop_Mode_Set_Interrupt
	r_cg_stbc.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

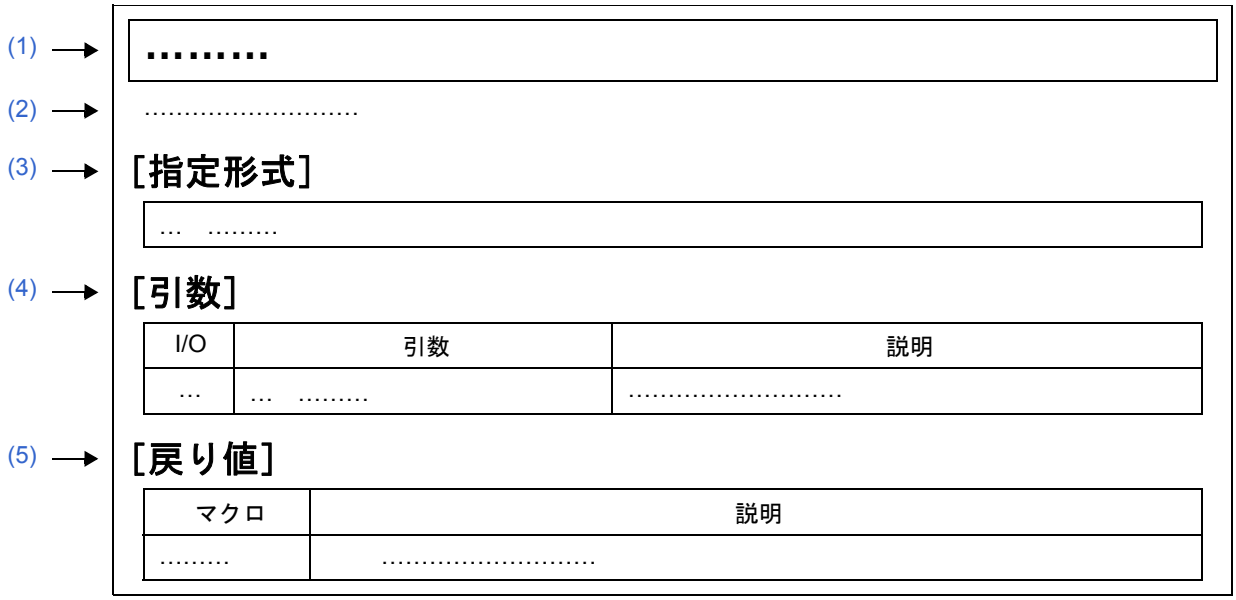
以下に，コード生成が API 関数を出力する際の命名規則を示します。

- 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。

3.2 関数リファレンス

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

図 3.1　　API 関数の記述フォーマット



(1)　名称
API 関数の名称を示しています。

(2)　機能
API 関数の機能概要を示しています。

(3)　［指定形式］
API 関数を C 言語で呼び出す際の記述形式を示しています。

(4)　［引数］
API 関数の引数を次の形式で示しています。

I/O	引数	説明
(a)	(b)	(c)

(a) I/O
引数の種類
I … 入力引数
O … 出力引数

(b) 引数
引数のデータ・タイプ

(c) 説明
引数の説明

(5) ［戻り値］
API 関数からの戻り値を次の形式で示しています。

マクロ	説明
(a)	(b)

(a) マクロ
戻り値のマクロ

(b) 説明
戻り値の説明

3.2.1 共　　通

以下に，コード生成が共通用として出力する API 関数の一覧を示します。

表 3.1　　共通用 API 関数

API 関数名	機能概要
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	各種ハードウエアを制御するうえで必要となる初期化処理を行います。

main

main 関数です。

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

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_MAIN_UserInit

ユーザ独自の初期化処理を行います。

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

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_SystemInit

各種ハードウエアを制御するうえで必要となる初期化処理を行います。

[指定形式]

```
void    R_SystemInit ( void );
```

[引数]

なし

[戻り値]

なし

3.2.2 クロックコントローラ

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

表 3.2 クロックコントローラ用 API 関数

API 関数名	機能概要
R_CGC_Create	クロック発生回路（リセット機能，オンチップ・デバッグ機能などを含む）を制御するうえで必要となる初期化処理を行います。
R_CGC_Create_UserInit	クロック発生回路（リセット機能，オンチップ・デバッグ機能などを含む）に関するユーザ独自の初期化処理を行います。
R_CGC_CK_Output_Enable	CK 端子からのクロック出力を許可します。
R_CGC_CK_Output_Disable	CK 端子からのクロック出力を禁止します。

R_CGC_Create

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

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_CGC_Create_UserInit

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

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

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_CGC_CK_Output_Enable

CK 端子からのクロック出力を許可します。

[指定形式]

```
void R_CGC_CK_Output_Enable(void);
```

[引数]

なし

[戻り値]

なし

R_CGC_CK_Output_Disable

CK 端子からのクロック出力を禁止します。

[指定形式]

```
void R_CGC_CK_Output_Disable(void);
```

[引数]

なし

[戻り値]

なし

3.2.3 ポート機能

以下に，コード生成が I/O ポート用として出力する API 関数の一覧を示します。

表 3.3 I/O ポート用 API 関数

API 関数名	機能概要
R_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_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_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。

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_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_PORT_Create のコールバック・ルーチンとして呼び出されます。

[指定形式]

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

[引数]

なし

[戻り値]

なし

3.2.4 割り込み

以下に，コード生成が割り込み機能用として出力する API 関数の一覧を示します。

表 3.4 割り込み機能用 API 関数

API 関数名	機能概要
R_INTC_Create	割り込み機能を制御するうえで必要となる初期化処理を行います。
R_INTC_Create_UserInit	割り込み機能に関するユーザ独自の初期化処理を行います。
R_IRQn_Start	IRQ*n* 割り込みを許可します。
R_IRQn_Stop	IRQ*n* 割り込み要求をマスク状態にします。
R_SINTn_Start	SINT 割り込み（SINT*n*）を許可します。
R_SINTn_Stop	SINT 割り込み（SINT*n*）要求をマスク状態にします。
R_SINTn_TriggerOn	ソフトウェア割り込みレジスタ（SINTC*n*）のカウンタ値をインクリメントします。
R_INTPn_Start	INTP*n* 割り込みを許可します。
R_INTPn_Stop	INTP*n* 割り込み要求をマスク状態にします。
r_nmi_interrupt	NMI の発生に伴う処理を行います。
r_irqn_interrupt	IRQ*n* の発生に伴う処理を行います。
r_sintn_interrupt	SINT*n* の発生に伴う処理を行います。
r_intpn_interrupt	INTP*n* の発生に伴う処理を行います。

R_INTC_Create

割り込み機能を制御するうえで必要となる初期化処理を行います。

[指定形式]

```
void    R_INTC_Create ( void );
```

[引数]

なし

[戻り値]

なし

R_INTC_Create_UserInit

割り込み機能に関するユーザ独自の初期化処理を行います。

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

[指定形式]

```
void    R_INTC_Create_UserInit ( void );
```

[引数]

なし

[戻り値]

なし

R_IRQ*n*_Start

IRQ*n* 割り込みを許可します。

[指定形式]

```
void    R_IRQn_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_IRQ*n*_Stop

IRQ*n* 割り込みを禁止します。

[指定形式]

```
void    R_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_SINT*n*_Start

ソフトウェア割り込みを許可します。

[指定形式]

```
void    R_SINTn_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_SINT*n*_Stop

ソフトウェア割り込みを禁止します。

[指定形式]

```
void    R_SINTn_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_SINT*n*_TriggerOn

ソフトウェア割り込みレジスタ（SINTC*n*）のカウンタ値をインクリメントします。

[指定形式]

```
void    r_SINTn_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_nmi_interrupt

NMI の発生に伴う処理を行います。

[指定形式]

```
void    r_nmi_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_irq*n*_interrupt

IRQ*n* の発生に伴う処理を行います。

[指定形式]

```
void    r_irqn_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_sint*n*_interrupt

SINT*n* の発生に伴う処理を行います。

[指定形式]

```
void    r_sintn_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_INTP*n*_Start

INTP*n* 割り込みを許可します。

[指定形式]

```
void    R_INTPn_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_INTP*n*_Stop

INTP*n* 割り込みを禁止します。

[指定形式]

```
void    R_INTPn_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_intp*n*_interrupt

INTP*n* の発生に伴う処理を行います。

[指定形式]

```
void    r_intpn_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

3.2.5 DMAC

以下に，コード生成が DMAC 用として出力する API 関数の一覧を示します。

表 3.5 DMAC 用 API 関数

API 関数名	機能概要
R_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_DMACn_Suspend	DMAC*n* 全チャネルの DMA 転送を一時停止します。
R_DMACn_Resume	DMAC*n* 全チャネルの DMA 転送の一時停止を解除します。
R_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_dmacnm_interrupt	DMAC*n* チャネル *m* の割り込みの発生に伴う処理を行います。
r_dmacnm_callback_transfer_completion	DMAC 転送完了割り込みの発生に伴う処理を行います。
r_dmacnm_callback_transfer_count_match	DMAC 転送回数一致割り込みの発生に伴う処理を行います。
R_DMACnm_Create	DMAC*n* 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_DMACnm_Start	DMAC*n* チャネル *m* の DMA 転送を許可します。
R_DMACnm_Stop	DMAC*n* チャネル *m* の DMA 転送を停止します。
R_DMACnm_Set_SoftwareTrigger	DMAC*n* チャネル *m* の転送要求を発生します。
R_DMACnm_Suspend	DMAC*n* チャネル *m* の DMA 転送を一時停止します。
R_DMACnm_Resume	DMAC*n* チャネル *m* の DMA 転送を一時停止を解除します。

R_DMAC*n*_Create

DMA コントローラを制御するうえで必要となる初期化処理を行います。

[指定形式]

```
void    R_DMACn_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*n*_Suspend

全チャネルの DMA 転送を一時停止します。

[指定形式]

```
void    R_DMACn_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_DMAC*n*_Resume

全チャネルの DMA 転送の一時停止を解除します。

[指定形式]

```
void    R_DMACn_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*n*_Create_UserInit

DMA コントローラに関するユーザ独自の初期化処理を行います。

備考 本 API 関数は，R_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_DMACn_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_dmac*nm*_interrupt

DMAC*n* 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_dmacnm_interrupt(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_dmac*nm*_callback_transfer_completion

DMA 転送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_dmacnm_callback_transfer_completion(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_dmac*nm*_callback_transfer_count_match

DMA 転送回数一致割り込みの発生に伴う処理を行います。

[指定形式]

```
void r_dmacnm_callback_transfer_count_match(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_DMAC*nm*_Create

DMAC*n* 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。

[指定形式]

```
void R_DMACnm_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_DMAC*nm*_Start

DMAC*n* チャネル *m* の DMA 転送を許可します。

[指定形式]

```
void    R_DMACnm_Start ( 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_DMAC*nm*_Stop

DMAC*n* チャネル *m* の DMA 転送を停止します。

[指定形式]

```
void    R_DMACnm_Stop ( 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_DMAC*nm*_Set_SoftwareTrigger

DMAC*n* チャネル *m* の転送要求を発生します。

[指定形式]

```
void    R_DMACnm_Set_SoftwareTrigger ( 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_DMAC*nm*_Suspend

DMAC*n* チャネル *m* の DMA 転送を一時停止します。

[指定形式]

```
void    R_DMACnm_Suspend ( 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_DMAC*nm*_Resume

DMAC*n* チャネル *m* の DMA 転送の一時停止を解除します。

[指定形式]

```
void    R_DMACnm_Resume ( 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

3.2.6 DTS

以下に，コード生成が DTS 用として出力する API 関数の一覧を示します。

表 3.6 DTS 用 API 関数

API 関数名	機能概要
R_DTS_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_DTS_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_DTS_Suspend	DTS 転送を一時停止します。
R_DTS_Resume	DTS 転送の一時停止を解除します。
R_DTS_All_Stop	DTS 転送をすべて停止します。
R_DTSx_y_Stop_Interrupt	DTS 転送の割り込みを停止します。
R_DTSm_Create	DTS チャネル *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_DTSm_Start	DTS チャネル *m* の DMA 転送を許可します。
R_DTSm_Stop	DTS チャネル m の DMA 転送を停止します。
R_DTSm_Set_SoftwareTrigger	DTS チャネル *m* の転送要求を発生します。
R_DTSm_Suspend	DTS チャネル *m* の DMA 転送を一時停止します。
R_DTSm_Resume	DTS チャネル *m* の DMA 転送の一時停止を解除します。
R_DTSm_Create_UserInit	DTS 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_dtsx_y_transfer_match_interrupt	DTS 転送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_dtsx_y_transfer_completion_interrupt	DTS 転送の転送完了割り込みの発生に伴う処理を行います。

R_DTS_Create

DTS を制御するうえで必要となる初期化処理を行います。

[指定形式]

```
void    R_DTS_Create ( void );
```

[引数]

なし

[戻り値]

なし

R_DTS_Create_UserInit

DTS 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_DTS_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_DTS_Create_UserInit ( void );
```

[引数]

なし

[戻り値]

なし

R_DTS_Suspend

DTS 転送を一時停止します。

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_DTS_Resume

DTS 転送を一時停止を解除します。

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_DTS_All_Stop

DTS 転送をすべて停止します。

[指定形式]

```
void    R_DTS_All_Stop ( void );
```

[引数]

なし

[戻り値]

なし

R_DTS*x*_*y*_Stop_Interrupt

DTS 転送の割り込み（チャネル *x* から *y* の範囲）を停止します。

[指定形式]

```
void    R_DTSx_y_Stop_Interrupt ( void );
```

備考　　*x*, *y* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_DTS*m*_Create

DTS 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。

[指定形式]

```
void    R_DTSm_Create ( void );
```

備考　　*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_DTS*m*_Start

DTS チャネル *m* の DMA 転送を許可します。

[指定形式]

```
void    R_DTSm_Start ( void );
```

備考　　*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_DTS*m*_Stop

DTS チャネル *m* の DMA 転送を停止します。

[指定形式]

```
void    R_DTSm_Stop ( void );
```

備考　　*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_DTS*m*_Set_SoftwareTrigger

DTS チャネル *m* の転送要求を発生します。

[指定形式]

```
void    R_DTSm_Set_SoftwareTrigger ( void );
```

備考　　*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_DTS*m*_Suspend

DTS チャネル *m* の DMA 転送を一時停止します。

[指定形式]

```
void    R_DTSm_Suspend ( void );
```

備考　　*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_DTS*m*_Resume

DTS チャネル *m* の DMA 転送の一時停止を解除します。

[指定形式]

```
void    R_DTSm_Resume ( void );
```

備考　　*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_DTS*m*_Create_UserInit

DTS 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_DTSm_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_DTSm_Create_UserInit ( void );
```

備考 *m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_dts*x*_*y*_transfer_match_interrupt

DTS 転送の転送回数一致割り込み（チャネル *x* から *y* の範囲）の発生に伴う処理を行います。

[指定形式]

```
void    r_dtsx_y_transfer_match_interrupt ( void );
```

備考　　*x*, *y* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_dts*x*_*y*_transfer_completion_interrupt

DTS 転送の転送完了割り込み（チャネル *x* から *y* の範囲）の発生に伴う処理を行います。

[指定形式]

```
void    r_dtsx_y_transfer_completion_interrupt ( void );
```

備考　　*x*, *y* 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 クロック同期シリアルインタフェース G

以下に，コード生成がクロック同期シリアルインタフェース G 用として出力する API 関数の一覧を示します。

表 3.7 クロック同期シリアルインタフェース G 用 API 関数

API 関数名	機能概要
R_CSIGm_Create	クロック同期シリアルインタフェース G を制御するうえで必要となる初期化処理を行います。
R_CSIGm_Create_UserInit	クロック同期シリアルインタフェース G に関するユーザ独自の初期化処理を行います。
r_csigm_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_csigm_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_csigm_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_csigm_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_csigm_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_csigm_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_CSIGm_Start	CSIG*m* の動作を許可します。
R_CSIGm_Stop	CSIG*m* の動作を禁止します。
R_CSIGm_Send	データ送信を行います。
R_CSIGm_Receive	データ受信を行います。

R_CSIG*m*_Create

クロック同期シリアルインタフェース G を制御するうえで必要となる初期化処理を行います。

[指定形式]

```
void    R_CSIGm_Create ( void );
```

備考　　*m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_CSIG*m*_Create_UserInit

クロック同期シリアルインタフェース G 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_CSIGm_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_CSIGm_Create_UserInit ( void );
```

備考 *m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_csig*m*_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_csigm_interrupt_receive ( void );
```

備考　　*m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_csig*m*_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_csigm_interrupt_error ( void );
```

備考　　*m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_csig*m*_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_csigm_interrupt_send ( void );
```

備考　　*m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_csig*m*_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，受信ステータス割り込みに対応した割り込み処理 r_csigm_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_csigm_callback_receiveend ( void );
```

備考 *m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_csig*m*_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，通信ステータス割り込みに対応した割り込み処理 r_csigm_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_csigm_callback_sendend ( void );
```

備考 *m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_csig*m*_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，通信エラー割り込みに対応した割り込み処理 r_csigm_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_csigm_callback_error ( uint8_t err_type );
```

備考 *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 *err_type*;	通信エラー割り込みの発生要因 0000x0*x*1B： オーバラン・エラー 0000x01*x*B： パリティ・エラー 000010*xx*B： データ整合性チェック・エラー

[戻り値]

なし

R_CSIG*m*_Start

CSIG*m* を待機状態にします。

[指定形式]

```
void    R_CSIGm_Start ( void );
```

備考　　*m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_CSIG*m*_Stop

CSIG*m* を終了します。

[指定形式]

```
void    R_CSIGm_Stop ( void );
```

備考　　*m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_CSIG*m*_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 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 関数の呼び出し以前に R_CSIGm_Start を呼び出す必要があります。

[指定形式]

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

備考 *m* は，チャネル番号を意味します。

[引数]

I/O	引数	説明
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
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_CSIG*m*_Receive

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. 本 API 関数の呼び出し後，R_CSIGm_Start を呼び出すことにより開始されます。

[指定形式]

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

備考 *m* は，チャネル番号を意味します。

[引数]

I/O	引数	説明
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
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

3.2.8 クロック同期シリアルインタフェース H

以下に，コード生成がクロック同期シリアルインタフェース H 用として出力する API 関数の一覧を示します。

表 3.8 クロック同期シリアルインタフェース H 用 API 関数

API 関数名	機能概要
R_CSIHm_Create	クロック同期シリアルインタフェース H を制御するうえで必要となる初期化処理を行います。
R_CSIHm_Create_UserInit	クロック同期シリアルインタフェース H に関するユーザ独自の初期化処理を行います。
r_csihm_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_csihm_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_csihm_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_csihm_interrupt_jobend	ジョブ完了割り込みの発生に伴う処理を行います。
r_csihm_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_csihm_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_csihm_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_CSIHm_Start	CSIH*m* の動作を許可します。
R_CSIHm_Stop	CSIH*m* の動作を禁止します。
R_CSIHm_Master_Send	マスタモードでデータ送信を行います。
R_CSIHm_Master_Receive	マスタモードでデータ受信を行います。
R_CSIHm_Slave_Send	スレーブモードでデータ送信を行います。
R_CSIHm_Slave_Receive	スレーブモードでデータ受信を行います。
R_CSIHm_Extended_Data_Master_Send	マスタモードで拡張データ長の送信を行います。
R_CSIHm_Extended_Data_Master_Receive	マスタモードで拡張データ長の受信を行います。
R_CSIHm_Extended_Data_Slave_Send	スレーブモードで拡張データ長の送信を行います。
R_CSIHm_Extended_Data_Slave_Receive	スレーブモードで拡張データ長の受信を行います。

R_CSIH*m*_Create

クロック同期シリアルインタフェース H を制御するうえで必要となる初期化処理を行います。

[指定形式]

```
void    R_CSIHm_Create ( void );
```

備考　　*m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_CSIH*m*_Create_UserInit

クロック同期シリアルインタフェース H 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_CSIHm_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_CSIHm_Create_UserInit ( void );
```

備考 *m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_csih*m*_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_csihm_interrupt_receive ( void );
```

備考　　*m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_csih*m*_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_csihm_interrupt_error ( void );
```

備考　　*m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_csih*m*_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_csihm_interrupt_send ( void );
```

備考　　*m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_csih*m*_interrupt_jobend

ジョブ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_csihm_interrupt_jobend ( void );
```

備考　　*m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_csih*m*_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，受信ステータス割り込みに対応した割り込み処理 r_csihm_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_csihm_callback_receiveend ( void );
```

備考 *m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_csih*m*_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，通信ステータス割り込みに対応した割り込み処理 r_csihm_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_csih*m*_callback_sendend ( void );
```

備考 *m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_csih*m*_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，通信エラー割り込みに対応した割り込み処理 r_csihm_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_csihm_callback_error ( uint8_t err_type );
```

備考 *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 *err_type*;	通信エラー割り込みの発生要因 0000x0*x*1B： オーバラン・エラー 0000x01*x*B： パリティ・エラー 000010*xx*B： データ整合性チェック・エラー

[戻り値]

なし

R_CSIH*m*_Start

CSIH*m* を待機状態にします。

[指定形式]

```
void    R_CSIHm_Start ( void );
```

備考　　*m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_CSIH*m*_Stop

CSIH*m* を終了します。

[指定形式]

```
void    R_CSIHm_Stop ( void );
```

備考　　*m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_CSIH*m*_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，引数 *tx_buf* で指定されたバッファから 2 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 関数の呼び出し以前に R_CSIHm_Start を呼び出す必要があります。

[指定形式]

```
#include    "r_cg_macrodriver.h"
MD_STATUS    R_CSIHm_Master_Send ( const uint16_t * tx_buf, uint16_t tx_num,
uint32_t chip_Id );
```

備考 *m* は，チャネル番号を意味します。

[引数]

I/O	引数	説明
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
I	`uint16_t tx_num;`	送信するデータの総数
I	`uint32_t chip_Id;`	チップセレクトの指定

[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_CSIH*m*_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，2 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 関数の呼び出し後，R_CSIHm_Start を呼び出すことにより開始されます。

[指定形式]

```
#include    "r_cg_macrodriver.h"
MD_STATUS R_CSIHm_Master_Receive ( uint16_t * rx_buf, uint16_t rx_num, uint32_t
chip_Id );
```

備考 *m* は，チャネル番号を意味します。

[引数]

I/O	引数	説明
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
I	uint16_t *rx_num*;	受信するデータの総数
I	uint32_t *chipId*;	チップセレクトの指定

[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_CSIH*m*_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 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 関数の呼び出し以前に，R_CSIHm_Start を呼び出す必要があります。

[指定形式]

```
MD_STATUS    r_CSIHm_Slave_Send ( const uint16_t * tx_buf, uint16_t tx_num);
```

備考 *m* は，チャネル番号を意味します。

[引数]

I/O	引数	説明
I	`const uint16_t * tx_buf;`	送信するデータを格納するバッファへのポインタ
O	`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_CSIH*m*_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，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. 本 API 関数の呼び出し後，R_CSIHm_Start を呼び出すことにより開始されます。

[指定形式]

```
MD_STATUS     R_CSIHm_Slave_Receive ( uint16_t * rx_buf, uint16_t rx_num );
```

備考 *m* は，チャネル番号を意味します。

[引数]

I/O	引数	説明
O	`uint16_t * rx_buf;`	受信する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_CSIH*m*_Extended_Data_Master_Send

マスタモードで拡張データ長の送信を行います。

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 バイト単位のマスタ送信を引数 *tx_bit_num* で指定されたビット数だけ送信を行います。

備考 2. 本 API 関数の呼び出し以前に R_CSIHm_Start を呼び出す必要があります。

[指定形式]

```
MD_STATUS    R_CSIHm_Extended_Data_Master_Send ( const uint16_t * tx_buf, uint16_t
tx_bit_num, uint32_t chip_Id );
```

備考 *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
I	`uint16_t tx_bit_num;`	送信するデータのビット数
I	`uint32_t chip_Id;`	チップセレクトの指定

[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_CSIH*m*_Extended_Data_Master_Receive

マスタモードで拡張データ長の受信を行います。

備考 1. 本 API 関数では，2 バイト単位のマスタ受信を引数 *rx_bit_num* で指定されたビット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 関数の呼び出し後，R_CSIHm_Start を呼び出すことにより開始されます。

[指定形式]

```
MD_STATUS R_CSIHm_Extended_Data_Master_Receive ( uint16_t * rx_buf, uint16_t
rx_bit_num, uint32_t chip_Id );
```

備考 *m* は，チャネル番号を意味します。

[引数]

I/O	引数	説明
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
I	uint16_t *rx_bit_num*;	受信するデータのビット数
I	uint32_t *chipId*;	チップセレクトの指定

[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_CSIH*m*_Extended_Data_Slave_Send

スレーブモードで拡張データ長の送信を行います。

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_bit_num* で指定されたビット数だけ繰り返し行います。

備考 2. 本 API 関数の呼び出し以前に R_CSIHm_Start を呼び出す必要があります。

[指定形式]

```
MD_STATUS    r_CSIHm_Extended_Data_Slave_Send ( const uint16_t * tx_buf, uint16_t tx_bit_num);
```

備考 *m* は，チャネル番号を意味します。

[引数]

I/O	引数	説明
I	`const uint16_t * tx_buf;`	送信するデータを格納するバッファへのポインタ
O	`uint16_t tx_bit_num;`	送信するデータのビット数

[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_CSIH*m*_Extended_Data_Slave_Receive

スレーブモードで拡張データ長の受信を行います。

備考 1. 本 API 関数では，2 バイト単位の受信を引数 *rx_bit_num* で指定されたビット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 関数の呼び出し後，R_CSIHm_Start を呼び出すことにより開始されます。

[指定形式]

```
MD_STATUS     R_CSIHm_Extended_Data_Slave_Receive ( uint16_t * rx_buf, uint16_t
rx_bit_num );
```

備考 *m* は，チャネル番号を意味します。

[引数]

I/O	引数	説明
O	`uint16_t * rx_buf;`	受信するデータを格納するバッファへのポインタ
I	`uint16_t rx_bit_num;`	受信するデータのビット数

[戻り値]

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

3.2.9 シリアルコミュニケーションインタフェース 3

以下に，コード生成がシリアルコミュニケーションインタフェース 3 用として出力する API 関数の一覧を示します。

表 3.9 シリアルコミュニケーションインタフェース 3 用 API 関数

API 関数名	機能概要
R_SCI3m_Create	シリアルコミュニケーションインタフェース 3 を制御するうえで必要となる初期化処理を行います。
R_SCI3m_Create_UserInit	シリアルコミュニケーションインタフェース 3 に関するユーザ独自の初期化処理を行います。
r_sci3m_interrupt_receive	割り込み要因 TXI（送信 FIFO データエンプティ（TDFE) による割り込み）の発生に伴う処理を行います。
r_sci3m_interrupt_error	割り込み要因 RXI（受信 FIFO データフル（RDF) による割り込み）の発生に伴う処理を行います。
r_sci3m_interrupt_send	割り込み要因 BRI（ブレーク（BRK) またはオーバラン（ORER）による割り込み）の発生に伴う処理を行います。
r_sci3m_interrupt_sendend	割り込み要因 DRI（受信データレディ（DR) による割り込み）の発生に伴う処理を行います。
r_sci3m_callback_receiveend	割り込み要因 TEI（トランスミットエンド（TEND) による割り込み）の発生に伴う処理を行います。
r_sci3m_callback_sendend	割り込み要因 ERI（フレーミングエラーまたはパリティエラー（ER) による割り込み）の発生に伴う処理を行います。
r_sci3m_callback_error	トランスミットエンド割り込みの発生に伴う処理を行います。
R_SCI3m_Start	SCI3 通信を待機状態にします。
R_SCI3m_Stop	SCI3 通信を終了します。
R_SCI3m_Send	調歩同期式モードで，送信を開始します。
R_SCI3m_Receive	調歩同期式モードで，受信を開始します。
R_SCI3m_Multiprocessor_Send	クロック同期式モードで，送受信を開始します。
R_SCI3m_Multiprocessor_Receive	クロック同期式モードで，送受信を開始します。

R_SCI3*m*_Create

シリアルコミュニケーションインタフェース 3 を制御するうえで必要となる初期化処理を行います。

[指定形式]

```
void    R_SCI3m_Create ( void );
```

備考　　*m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_SCI3*m*_Create_UserInit

シリアルコミュニケーションインタフェース 3 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_SCI3m_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_SCI3m_Create_UserInit ( void );
```

備考 *m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_sci3*m*_interrupt_receive

割り込み要因 RXI（受信データフルによる割り込み）の発生に伴う処理を行います。

[指定形式]

```
void    r_sci3m_interrupt_receive ( void );
```

備考　　*m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_sci3*m*_interrupt_error

割り込み要因 ERI（受信エラーによる割り込み）の発生に伴う処理を行います。

[指定形式]

```
void    r_sci3m_interrupt_error ( void );
```

備考　　*m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_sci3*m*_interrupt_send

割り込み要因 TXI（送信データエンプティによる割り込み）の発生に伴う処理を行います。

[指定形式]

```
void    r_sci3m_interrupt_send ( void );
```

備考　　*m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_sci3*m*_interrupt_sendend

割り込み要因 TEI（送信終了による割り込み）の発生に伴う処理を行います。

[指定形式]

```
void    r_sci3m_interrupt_sendend ( void );
```

備考　　*m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_sci3*m*_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，受信完了割り込みに対応した割り込み処理のコールバック・ルーチンとして呼び出されます。

[指定形式]

```
void    r_sci3m_callback_receiveend ( void );
```

備考 *m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_sci3*m*_callback_sendend

送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，送信完了割り込みに対応した割り込み処理のコールバック・ルーチンとして呼び出されます。

[指定形式]

```
void    r_sci3m_callback_sendend ( void );
```

備考 *m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_sci3*m*_callback_error

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，エラー割り込みに対応した割り込み処理のコールバック・ルーチンとして呼び出されます。

[指定形式]

```
static  void    r_SCI3m_callback_error ( void );
```

備考 *m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_SCI3*m*_Start

SCI3 通信を待機状態にします。

[指定形式]

```
void    R_SCI3m_Start ( void );
```

備考　　*m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_SCI3*m*_Stop

SCI3 通信を終了します。

[指定形式]

```
void    R_SCI3m_Stop ( void );
```

備考　　*m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_SCI3*m*_Send

送信を開始します。

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

備考 2. 本 API 関数の呼び出し以前に R_SCI3m_Start を呼び出す必要があります。

[指定形式]

```
MD_STATUS   R_SCI3m_Send ( const uint8_t * tx_buf, uint16_t tx_num );
```

備考 *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
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_SCI3*m*_Receive

受信を開始します。

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. 本 API 関数の呼び出し後，R_SCI3m_Start を呼び出すことにより開始されます。

[指定形式]

```
MD_STATUS   R_SCI3m_Receive ( uint8_t * rx_buf, uint16_t rx_num );
```

備考 *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
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_SCI3*m*_Multiprocessor_Send

マルチプロセッサモードで送信を開始します。

備考 1. 本 API 関数では，引数 *tx_buf* で指定されたバッファから 1 バイト単位の送信を引数 *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 関数の呼び出し以前に R_SCI3m_Start を呼び出す必要があります。

[指定形式]

```
MD_STATUS R_SCI3m_Multiprocessor_Send(const uint8_t * tx_buf, uint16_t tx_num,
uint8_t rx_id);
```

備考 *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
O	`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_SCI3*m*_Multiprocessor_Receive

マルチプロセッサモードで，受信を開始します。

備考 3. 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 4. 本 API 関数の呼び出し後，R_SCI3m_Start を呼び出す必要があります。

[指定形式]

```
MD_STATUS R_SCI3m_Multiprocessor_Receive( uint8_t * rx_buf, uint16_t rx_num, uint8_t rx_id);
```

備考 *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 * rx_buf;`	受信データを格納した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
O	`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

3.2.10 UART インタフェース

以下に，コード生成が UART インタフェース用として出力する API 関数の一覧を示します。

表 3.10 UART インタフェース用 API 関数

API 関数名	機能概要
R_UARTm_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_UARTm_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_uartm_interrupt_receive	受信完了割り込みの発生に伴う処理を行います。
r_uartm_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_uartm_interrupt_send	送信割り込みの発生に伴う処理を行います。
r_uartm_callback_receiveend	受信完了割り込みの発生に伴う処理を行います。
r_uartm_callback_sendend	送信割り込みの発生に伴う処理を行います。
r_uartm_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_UARTm_Start	UART*m* の動作を許可します。
R_UARTm_Stop	UART*m* の動作を禁止します。
R_UARTm_Send	データ送信を行います。
R_UARTm_Receive	データ受信を行います。

R_UART*m*_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。

[指定形式]

```
void    R_UARTm_Create ( void );
```

備考　　*m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_UART*m*_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_UARTm_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_UARTm_Create_UserInit ( void );
```

備考 *m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_uart*m*_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_uartm_interrupt_receive ( void );
```

備考　　*m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_uart*m*_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_uartm_interrupt_error ( void );
```

備考　　*m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_uart*m*_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_uartm_interrupt_send ( void );
```

備考　　*m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_uart*m*_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，受信完了割り込みに対応した割り込み処理 r_uartm_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_uartm_callback_receiveend ( void );
```

備考 *m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_uart*m*_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，送信割り込みに対応した割り込み処理 r_uartm_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_uartm_callback_sendend ( void );
```

備考 *m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_uart*m*_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，通信エラー割り込みに対応した割り込み処理 r_uartm_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_uartm_callback_error ( uint8_t err_type );
```

備考 *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 *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

R_UART*m*_Start

UART*m* を待機状態にします。

[指定形式]

```
void    R_UARTm_Start ( void );
```

備考　　*m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_UART*m*_Stop

UART*m* を終了します。

[指定形式]

```
void    R_UARTm_Stop ( void );
```

備考 *m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_UART*m*_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. 本 API 関数の呼び出し以前に R_UARTm_Start を呼び出す必要があります。

[指定形式]

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

備考 *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
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_UART*m*_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. 本 API 関数の呼び出し後，R_UARTm_Start を呼び出すことにより開始されます。

[指定形式]

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

備考 *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
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

3.2.11 ウィンドウウォッチドッグタイマ

以下に，コード生成がウィンドウウォッチドッグタイマ用として出力する API 関数の一覧を示します。

表 3.11 ウィンドウウォッチドッグタイマ用 API 関数

API 関数名	機能概要
R_WDTm_Create	ウィンドウ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_WDTm_Create_UserInit	ウィンドウ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_wdtm_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_WDTm_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_WDT*m*_Create

ウォッチドッグタイマを制御するうえで必要となる初期化処理を行います。

[指定形式]

```
void    R_WDTm_Create ( void );
```

備考　　*m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_WDT*m*_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_WDTm_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_WDTm_Create_UserInit ( void );
```

備考 *m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_wdt*m*_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_wdtm_interrupt ( void );
```

備考　　*m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_WDT*m*_Restart

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

[指定形式]

```
void    R_WDTm_Restart ( void );
```

備考　　*m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

3.2.12 OS タイマ

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

表 3.12 OS タイマ用 API 関数

API 関数名	機能概要
R_OSTMm_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_OSTMm_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_ostmm_interrupt	OSTM の割り込みの発生に伴う処理を行います。
R_OSTMm_Start	OS タイマのカウントを開始します。
R_OSTMm_Stop	OS タイマのカウントを終了します。
R_OSTMm_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_OSTM*m*_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。

[指定形式]

```
void    R_OSTMm_Create ( void );
```

備考　　*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_OSTM*m*_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_OSTMm_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_OSTMm_Create_UserInit ( void );
```

備考 *m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_ostm*m*_interrupt

OSTM の割り込みの発生に伴う処理を行います。

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

[指定形式]

```
void    r_ostmm_interrupt ( void );
```

備考 *m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_OSTM*m*_Start

OS タイマのカウントを開始します。

[指定形式]

```
void    R_OSTMm_Start ( void );
```

備考　　*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_OSTM*m*_Stop

OS タイマのカウントを終了します。

[指定形式]

```
void    R_OSTMm_Stop ( void );
```

備考　　*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_OSTM*m*_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_OSTMm_Set_CompareValue ( uint32_t value );
```

備考　　*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 *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

3.2.13 アドバンストタイマユニットⅣ

以下に，コード生成がアドバンストタイマユニットⅣ用として出力する API 関数の一覧を示します。

表 3.13 アドバンストタイマユニットⅣ用 API 関数

API 関数名	機能概要
R_ATUIV_Common_Create	アドバンストタイマユニットⅣを制御するうえで必要となる初期化処理を行います。
R_ATUIV_Common_Create_UserInit	アドバンストタイマユニットⅣに関するユーザ独自の初期化処理を行います。
R_ATUIV_Timerkn_Create	タイマ k およびタイマ *kn* を制御するうえで必要となる初期化処理を行います。
R_ATUIV_Timerkn_Create_UserInit	タイマ *n* およびタイマ *nm* に関するユーザ独自の初期化処理を行います。
r_atuiv_timerknm_overflow_interrupt	オーバフロー割り込みの発生に伴う処理を行います。
r_atuiv_timerknm_interrupt	タイマの割り込みの発生に伴う処理を行います。
r_atuiv_timerknm_icrnx_interrupt	タイマの ICR*nx* レジスタ割り込みの発生に伴う処理を行います。
r_atuiv_timerknm_ocrnx_interrupt	タイマの OCR*nx* レジスタ割り込みの発生に伴う処理を行います。
r_atuiv_timerknm_tcntnx_interrupt	タイマの TCNT*nx* レジスタ割り込みの発生に伴う処理を行います。
r_atuiv_timerknm_cmfnx_interrupt	タイマの CMF*nx* レジスタ割り込みの発生に伴う処理を行います。
r_atuiv_timerknm_callback_ocrc	タイマの OCRC*nm* レジスタ割り込みの発生に伴う処理を行います。
r_atuiv_timerknm_callback_grc	タイマの GRC*nm* レジスタ割り込みの発生に伴う処理を行います。
r_atuiv_timerknm_underflow_interrupt	ダウンカウンタアンダフロー割り込みの発生に伴う処理を行います。
r_atuiv_timerknm_comparex_interrupt	コンペアレジスタとカウンタのコンペアマッチ割り込みの発生に伴う処理を行います。
r_atuiv_timerknm_callback_overflow	オーバフロー割り込みの発生に伴う処理を行います。
r_atuiv_timerknm_callback_cycle	サイクルマッチ割り込みの発生に伴う処理を行います。
r_atuiv_timerknm_callback_duty	デューティマッチ割り込みの発生に伴う処理を行います。
r_atuiv_timerknm_fifo_overflow_interrupt	FIFO オーバフロー割り込みの発生に伴う処理を行います。
r_atuiv_timerknm_fifo_datafull_interrupt	FIFO データフル割り込みの発生に伴う処理を行います。
r_atuiv_timerknm_tcntk_overflow_interrupt	カウンタのオーバフロー割り込みの発生に伴う処理を行います。
R_ATUIV_Timerk_OperationOn	タイマの動作を許可します。
R_ATUIV_Timerk_OperationOff	タイマの動作を禁止します。
R_ATUIV_Timerknm_Start	タイマのカウントを開始します。
R_ATUIV_Timerknm_Stop	タイマのカウントを停止します。
R_ATUIV_Timerknm_Get_PulseWidth	タイマのインプットパルス幅を測定します。
R_ATUIV_Timerknm_Get_CaptureValue	インプットキャプチャレジスタの値を読み出します。
R_ATUIV_Timerknm_Set_Compare_Match	コンペアマッチレジスタの値を設定します。
R_ATUIV_Timerknm_Set_One_Shot_Pulse	ワンショットパルス用のレジスタ値を更新します。

API 関数名	機能概要
R_ATUIV_Timerknm_Forced_Compare_Match	強制コンペアマッチを実行します。
R_ATUIV_Timerknm_Forced_Output_Compare_Match	強制コンペアマッチ出力を実行します。
R_ATUIV_Timerknm_Start_Down_Count	ダウンカウントを開始します。
R_ATUIV_Timerknm_Get_InputCapturex	インプットキャプチャレジスタの値を読み出します。
R_ATUIV_Timerknm_xpin_Output_Nomal	端子の出力を通常出力に設定します。
R_ATUIV_Timerknm_xpin_Output_Low	端子の出力をロウレベルに設定します。
R_ATUIV_Timerknm_xpin_Output_High	端子の出力をハイレベルに設定します。
R_ATUIV_Timerknm_Get_Count	計測したカウントの値を読み出します。
R_ATUIV_Timerknm_Get_PWM_Measure_Value	計測した PWM 波の値を読み出します。
R_ATUIV_Timerknm_Get_Measure_Value	計測した値（入力エッジ数，エッジ入力時刻，PWM 波形のオフデューティ，PWM サイクル）を読み出します。
R_ATUIV_Timerknm_Reset_FIFO	FIFO を空の状態にリセットします。

R_ATUIV_Common_Create

アドバンストタイマユニットⅣを制御するうえで必要となる初期化処理を行います。

[指定形式]

```
void    R_ATUIV_Common_Create ( void );
```

[引数]

なし

[戻り値]

なし

R_ATUIV_Common_Create_UserInit

アドバンストタイマユニットⅣ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_ATUIV_Common_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_ATUIV_Common_Create_UserInit ( void );
```

[引数]

なし

[戻り値]

なし

R_ATUIV_Timer*kn*_Create

タイマ *k* およびタイマ *kn* を制御するうえで必要となる初期化処理を行います。

[指定形式]

```
void    R_ATUIV_Timerk_Create ( void );
```

```
void    R_ATUIV_Timerkn_Create ( void );
```

備考　　*k* はタイマ種別，*n* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_ATUIV_Timer*kn*_Create_UserInit

タイマ *k* およびタイマ *kn* 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_ATUIV_Timerkn_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_ATUIV_Timerk_Create_UserInit ( void );
```

```
void    R_ATUIV_Timerkn_Create_UserInit ( void );
```

備考 *k* はタイマ種別，*n* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

r_atuiv_timer*knm*_overflow_interrupt

オーバフロー割り込みの発生に伴う処理を行います。

[指定形式]

```
void    r_atuiv_timer knm _overflow_interrupt ( void );
```

```
void    r_atuiv_timer knm _overflow11_interrupt ( void );
void    r_atuiv_timer knm _overflow12_interrupt ( void );
```

備考　*k* はタイマ種別，*n* はユニット番号，*m* はチャネル番号を意味します。

備考　overflow1 と overflow2 はタイマにより異なります。

[引数]

なし

[戻り値]

なし

r_atuiv_timer*knm*_interrupt

タイマの割り込み（インプットキャプチャ，コンペアマッチ）の発生に伴う処理を行います。

[指定形式]

```
void    r_atuiv_timerknm_interrupt ( void );
```

```
void    r_atuiv_timerkn_channelm_interrupt ( void );
```

備考　　*k* はタイマ種別，*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_atuiv_timer*knm*_icr*nx*_interrupt

タイマの ICR*nx* レジスタ割り込みの発生に伴う処理を行います。

[指定形式]

```
void    r_atuiv_timerknm_icrnx_interrupt ( void );
```

備考　　*k* はタイマ種別，*n* はユニット番号，*m* はチャネル番号，*x* はレジスタ番号を意味します。

[引数]

なし

[戻り値]

なし

r_atuiv_timer*knm*_ocr*nx*_interrupt

タイマの OCR*nx* レジスタ割り込みの発生に伴う処理を行います。

[指定形式]

```
void    r_atuiv_timerknm_ocrnx_interrupt ( void );
```

備考　　*k* はタイマ種別，*n* はユニット番号，*m* はチャネル番号，*x* はレジスタ番号を意味します。

[引数]

なし

[戻り値]

なし

r_atuiv_timer*knm*_tcnt*nx*_interrupt

タイマの TCNT*nx* 割り込みの発生に伴う処理を行います。

[指定形式]

```
void    r_atuiv_timerknm_tcntnx_interrupt ( void );
```

備考　　*k* はタイマ種別，*n* はユニット番号，*m* はチャネル番号，*x* はレジスタ番号を意味します。

[引数]

なし

[戻り値]

なし

r_atuiv_timer*knm*_cmf*nx*_interrupt

タイマの CMF*nx* レジスタ割り込みの発生に伴う処理を行います。

[指定形式]

```
void    r_atuiv_timerknm_cmfnx_interrupt ( void );
```

備考　　*k* はタイマ種別，*n* はユニット番号，*m* はチャネル番号，*x* はレジスタ番号を意味します。

[引数]

なし

[戻り値]

なし

r_atuiv_timer*knm*_callback_ocrc

タイマの OCRCnm レジスタ割り込みの発生に伴う処理を行います。

備考 本 API 関数は，タイマの OCRC*nm* レジスタ割り込みに対応した割り込み処理 r_atuiv_timerknm_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_atuiv_timer knm _callback_ocrc ( void );
```

備考 *k* はタイマ種別，*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_atuiv_timer*knm*_callback_grc

タイマの GRC*nm* レジスタ割り込みの発生に伴う処理を行います。

備考 本 API 関数は，タイマの GRC*nm* レジスタ割り込みに対応した割り込み処理 r_atuiv_timerknm_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_atuiv_timer knm _callback_grc ( void );
```

備考 *k* はタイマ種別，*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_atuiv_timer*knm*_underflow_interrupt

ダウンカウンタアンダフロー割り込みの発生に伴う処理を行います。

[指定形式]

```
void    r_atuiv_timerknm_underflow_interrupt ( void );
```

備考　　*k* はタイマ種別，*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_atuiv_timer*knm*_compare*x*_interrupt

コンペア・レジスタとカウンタのコンペアマッチ割り込みの発生に伴う処理を行います。

[指定形式]

```
void    r_atuiv_timerknm_comparex_interrupt ( void );
```

備考　　*k* はタイマ種別，*n* はユニット番号，*m* はチャネル番号，*x* はレジスタ番号を意味します。

[引数]

なし

[戻り値]

なし

r_atuiv_timer*knm*_callback_overflow

オーバフロ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，オーバフロー割り込みに対応した割り込み処理 r_atuiv_timerknm_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_atuiv_timer knm _callback_overflow ( void );
```

備考 *k* はタイマ種別，*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_atuiv_timer*knm*_callback_cycle

サイクルマッチ割り込みの発生に伴う処理を行います。

備考 本 API 関数は，サイクルマッチ割り込みに対応した割り込み処理 r_atuiv_timerknm_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_atuiv_timerknm_callback_cycle ( void );
```

備考 *k* はタイマ種別，*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_atuiv_timer*knm*_callback_duty

デューティマッチ割り込みの発生に伴う処理を行います。

備考 本 API 関数は，デューティ割り込みに対応した割り込み処理 r_atuiv_timerknm_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_atuiv_timer knm _callback_duty ( void );
```

備考 *k* はタイマ種別，*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_atuiv_timer*knm*_fifo_overflow_interrupt

FIFO オーバフロー割り込みの発生に伴う処理を行います。

[指定形式]

```
void    r_atuiv_timerknm_fifo_overflow_interrupt ( void );
```

備考　　*k* はタイマ種別，*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_atuiv_timer*knm*_fifo_datafull_interrupt

FIFO 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_atuiv_timerknm_fifo_datafull_interrupt ( void );
```

備考　　*k* はタイマ種別，*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_atuiv_timer*knm*_tcnt*k*_overflow_interrupt

カウンタのオーバフロー割り込みの発生に伴う処理を行います。

[指定形式]

```
void    r_atuiv_timerknm_tcntk_overflow_interrupt ( void );
```

備考　　*k* はタイマ種別，*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_ATUIV_Timer*k*_OperationOn

タイマ *k* の動作を許可します。

[指定形式]

```
void    R_ATUIV_Timerk_OperationOn ( void );
```

備考　　*k* はタイマ種別を意味します。

[引数]

なし

[戻り値]

なし

R_ATUIV_Timer*k*_OperationOff

タイマ *k* の動作を禁止します。

[指定形式]

```
void    R_ATUIV_Timerk_OperationOff ( void );
```

備考　　k はタイマ種別を意味します。

[引数]

なし

[戻り値]

なし

R_ATUIV_Timer*knm*_Start

タイマのカウントを開始します。

[指定形式]

```
void    R_ATUIV_Timerknm_Start ( void );
```

備考　　*k* はタイマ種別，*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_ATUIV_Timer*knm*_Stop

タイマのカウントを停止します。

[指定形式]

```
void    R_ATUIV_Timerknm_Stop ( void );
```

備考　　*k* はタイマ種別，*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_ATUIV_Timer*knm*_Get_PulseWidth

タイマのインプットパルス幅を測定します。

[指定形式]

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

備考　　*k* はタイマ種別，*n* はユニット番号，*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 * *width*;	インプットパルス幅を格納する領域へのポインタ

[戻り値]

なし

R_ATUIV_Timer*knm*_Get_CaptureValue

インプットキャプチャレジスタの値を読み出します。

[指定形式]

```
#include    "r_cg_macrodriver.h"
void    R_ATUIV_Timerknm_Get_CaptureValue ( uint32_t * value );
```

備考　　*k* はタイマ種別，*n* はユニット番号，*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 * *value*;	インプットキャプチャレジスタの値を格納する領域へのポインタ

[戻り値]

なし

R_ATUIV_Timer*knm*_Set_Compare_Match

コンペアマッチレジスタの値を設定します。

[指定形式]

```
void    R_ATUIV_Timerknm_Set_Compare_Match ( void );
```

備考　　*k* はタイマ種別，*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_ATUIV_Timer*knm*_Set_One_Shot_Pulse

ワンショットパルス用のレジスタ値を更新します。

[指定形式]

```
void    R_ATUIV_Timerknm_Set_One_Shot_Pulse ( void );
```

備考　　*k* はタイマ種別，*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_ATUIV_Timer*knm*_Forced_Compare_Match

強制コンペアマッチを実行します。

[指定形式]

```
void    R_ATUIV_Timerknm_Forced_Compare_Match ( void );
```

備考　　*k* はタイマ種別，*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_ATUIV_Timer*knm*_Forced_Output_Compare_Match

強制コンペアマッチ出力を実行します。

[指定形式]

```
void    R_ATUIV_Timerknm_Forced_Output_Compare_Match ( void );
```

備考　　*k* はタイマ種別，*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_ATUIV_Timer*knm*_Start_Down_Count

ダウンカウントを開始します。

[指定形式]

```
void    R_ATUIV_Timerknm_Start_Down_Count ( void );
```

備考　　*k* はタイマ種別，*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_ATUIV_Timer*knm*_Get_InputCapture*x*

インプットキャプチャレジスタの値を読み出します。

[指定形式]

```
#include    "r_cg_macrodriver.h"
void    R_ATUIV_Timerknm_Get_InputCapturex ( uint32_t * value );
```

備考　　*k* はタイマ種別，*n* はユニット番号，*m* はチャネル番号，*x* はレジスタ番号を意味します。

[引数]

I/O	引数	説明
O	`uint32_t * value;`	インプットキャプチャレジスタの値を格納する領域へのポインタ

[戻り値]

なし

R_ATUIV_Timer*knm*_*x*pin_Output_Nomal

端子の出力を通常出力に設定します。

[指定形式]

```
void    R_ATUIV_Timerknm_xpin_Output_Nomal ( void );
```

備考 *k* はタイマ種別，*n* はユニット番号，*m* はチャネル番号，*x* は出力端子を意味します。

[引数]

なし

[戻り値]

なし

R_ATUIV_Timer*knm*_*x*pin_Output_Low

端子の出力をロウレベルに設定します。

[指定形式]

```
void    R_ATUIV_Timerknm_xpin_Output_Low ( void );
```

備考　*k* はタイマ種別，*n* はユニット番号，*m* はチャネル番号，*x* は出力端子を意味します。

[引数]

なし

[戻り値]

なし

R_ATUIV_Timer*knm*_*x*pin_Output_High

端子の出力をハイレベルに設定します。

[指定形式]

```
void    R_ATUIV_Timerknm_xpin_Output_High ( void );
```

備考　　*k* はタイマ種別，*n* はユニット番号，*m* はチャネル番号，*x* は出力端子を意味します。

[引数]

なし

[戻り値]

なし

R_ATUIV_Timer*knm*_Get_Count

計測したカウントの値を読み出します。

[指定形式]

```
#include    "r_cg_macrodriver.h"
void    R_ATUIV_Timerknm_Get_Count ( uint16_t * count );
```

備考　　*k* はタイマ種別，*n* はユニット番号，*m* はチャネル番号を意味します。

[引数]

I/O	引数	説明
O	`uint16_t * count;`	カウントの値を格納する領域へのポインタ

[戻り値]

なし

R_ATUIV_Timer*knm*_Get_PWM_Measure_Value

計測した PWM 波の値を読み出します。

[指定形式]

```
void    R_ATUIV_Timerknm_Get_PWM_Mesure_Value ( uint32_t * low_width, uint32_t *
edge_width );
```

備考　　*k* はタイマ種別，*n* はユニット番号，*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 * low_width;`	計測したロウレベル幅を格納する領域へのポインタ
O	`uint32_t * edge_width;`	計測したエッジ数カウント幅を格納する領域へのポインタ

[戻り値]

なし

R_ATUIV_Timer*knm*_Get_Measure_Value

計測した値（入力エッジ数，エッジ入力時刻，PWM 波形のオフデューティ，PWM サイクル）を読み出します。

[指定形式]

```
void    R_ATUIV_Timerknm_Get_Mesure_Value ( uint16_t * edge, uint32_t *
off_duty_cycle, uint32_t * pwm_cycle, uint32_t * edge_input_time );
```

備考　　*k* はタイマ種別，*n* はユニット番号，*m* はチャネル番号を意味します。

[引数]

I/O	引数	説明
O	`uint16_t * edge;`	入力エッジ数を格納する領域へのポインタ
O	`uint32_t * off_duty_cycle;`	PWM 波形のオフデューティを格納する領域へのポインタ
O	`uint32_t * pwm_cycle;`	PWM サイクルを格納する領域へのポインタ
O	`uint32_t * edge_input_time;`	エッジ入力時刻を格納する領域へのポインタ

[戻り値]

なし

R_ATUIV_Timer*knm*_Reset_FIFO

FIFO を空の状態にリセットします。

[指定形式]

```
void    R_ATUIV_Timerknm_Reset_FIFO ( void );
```

備考 *k* はタイマ種別，*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

3.2.14 タイマ・アレイ・ユニット B

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

表 3.14 タイマ・アレイ・ユニット B 用 API 関数

API 関数名	機能概要
R_TAUBn_Create	タイマ・アレイ・ユニット B*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_TAUBn_Create_UserInit	タイマ・アレイ・ユニット B*n* に関するユーザ独自の初期化処理を行います。
r_taubn_channelm_interrupt	タイマ割り込みの発生に伴う処理を行います。
R_TAUBn_Channelm_Start	タイマのカウントを開始します。
R_TAUBn_Channelm_Stop	タイマのカウントを停止します。
R_TAUBn_Channelm_Get_PulseWidth	タイマのパルス幅を測定します。

R_TAUB*n*_Create

タイマ・アレイ・ユニット B*n* を制御するうえで必要となる初期化処理を行います。

[指定形式]

```
void    R_TAUBn_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_TAUB*n*_Create_UserInit

タイマ・アレイ・ユニット B*n* に関するユーザ独自の初期化処理を行います。

備考 本 API 関数は，R_TAUBn_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_TAUBn_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_taub*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_taubn_channelm_interrupt ( 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_TAUB*n*_Channel*m*_Start

タイマのカウントを開始します。

[指定形式]

```
void    R_TAUBn_Channelm_Start ( 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_TAUB*n*_Channel*m*_Stop

タイマのカウントを停止します。

[指定形式]

```
void    R_TAUBn_Channelm_Stop ( 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_TAUB*n*_Channel*m*_Get_PulseWidth

タイマのパルス幅を測定します。

[指定形式]

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

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]

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

[戻り値]

なし

3.2.15 タイマ・アレイ・ユニット D

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

表 3.15 タイマ・アレイ・ユニット D 用 API 関数

API 関数名	機能概要
R_TAUDn_Create	タイマ・アレイ・ユニット D*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_TAUDn_Create_UserInit	タイマ・アレイ・ユニット D*n* に関するユーザ独自の初期化処理を行います。
r_taudn_channelm_interrupt	タイマ割り込みの発生に伴う処理を行います。
R_TAUDn_Channelm_Start	タイマのカウントを開始します。
R_TAUDn_Channelm_Stop	タイマのカウントを停止します。
R_TAUDn_Channelm_Get_PulseWidth	タイマのパルス幅を測定します。

R_TAUD*n*_Create

タイマ・アレイ・ユニット D*n* を制御するうえで必要となる初期化処理を行います。

[指定形式]

```
void    R_TAUDn_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_TAUD*n*_Create_UserInit

タイマ・アレイ・ユニット D*n* に関するユーザ独自の初期化処理を行います。

備考 本 API 関数は，R_TAUDn_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_TAUDn_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_taud*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_taudn_channelm_interrupt ( 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_TAUD*n*_Channel*m*_Start

タイマのカウントを開始します。

[指定形式]

```
void    R_TAUDn_Channelm_Start ( 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_TAUD*n*_Channel*m*_Stop

タイマのカウントを停止します。

[指定形式]

```
void    R_TAUDn_Channelm_Stop ( 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_TAUD*n*_Channel*m*_Get_PulseWidth

タイマのパルス幅を測定します。

[指定形式]

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

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]

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

[戻り値]

なし

3.2.16 タイマ・アレイ・ユニット J

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

表 3.16 タイマ・アレイ・ユニット J 用 API 関数

API 関数名	機能概要
R_TAUJn_Create	タイマ・アレイ・ユニット J*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_TAUJn_Create_UserInit	タイマ・アレイ・ユニット J*n* に関するユーザ独自の初期化処理を行います。
r_taujn_channelm_interrupt	タイマ割り込みの発生に伴う処理を行います。
R_TAUJn_Channelm_Start	タイマのカウントを開始します。
R_TAUJn_Channelm_Stop	タイマのカウントを停止します。
R_TAUJn_Channelm_Get_PulseWidth	タイマのパルス幅を測定します。

R_TAUJ*n*_Create

タイマ・アレイ・ユニット J*n* を制御するうえで必要となる初期化処理を行います。

[指定形式]

```
void    R_TAUJn_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_TAUJ*n*_Create_UserInit

タイマ・アレイ・ユニット J*n* に関するユーザ独自の初期化処理を行います。

備考 本 API 関数は，R_TAUJn_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_TAUJn_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_tauj*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_tauj n_channel m_interrupt ( 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_TAUJ*n*_Channel*m*_Start

タイマのカウントを開始します。

[指定形式]

```
void    R_TAUJn_Channelm_Start ( 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_TAUJ*n*_Channel*m*_Stop

タイマのカウントを停止します。

[指定形式]

```
void    R_TAUJn_Channelm_Stop ( 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_TAUJ*n*_Channel*m*_Get_PulseWidth

タイマのパルス幅を測定します。

[指定形式]

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

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]

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

[戻り値]

なし

3.2.17 タイマオプション

以下に，コード生成がタイマオプション用として出力する API 関数の一覧を示します。

表 3.17 タイマオプション用 API 関数

API 関数名	機能概要
R_TAPAm_Create	タイマオプションを制御するうえで必要となる初期化処理を行います。
R_TAPAm_Create_UserInit	タイマオプションに関するユーザ独自の初期化処理を行います。
R_TAPAm_Start	非同期 Hi-Z 制御を許可します。
R_TAPAm_Stop	非同期 Hi-Z 制御を停止します。
R_TAPAm_Trigger_Start	Hi-Z 制御信号をロウレベルに設定します。
R_TAPAm_Trigger_Stop	Hi-Z 制御信号をハイレベルに設定します。

R_TAPA*m*_Create

タイマオプションを制御するうえで必要となる初期化処理を行います。

[指定形式]

```
void    R_TAPAm_Create ( void );
```

備考　　*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_TAPA*m*_Create_UserInit

タイマオプション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_TAPAm_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_TAPAm_Create_UserInit ( void );
```

備考 *m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_TAPA*m*_Start

非同期 Hi-Z 制御を許可します。

[指定形式]

```
void    R_TAPAm_Start ( void );
```

備考　　*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_TAPA*m*_Stop

非同期 Hi-Z 制御を停止します。

[指定形式]

```
void    R_TAPAm_Stop ( void );
```

備考　　*m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_TAPA*m*_Trigger_Start

Hi-Z 制御信号をロウレベルに設定します。

[指定形式]

```
void    R_TAPAm_Trigger_Start ( void );
```

備考 *m* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_TAPA*m*_Trigger_Stop

Hi-Z 制御信号をハイレベルに設定します。

[指定形式]

```
void    R_TAPAm_Trigger_Stop ( void );
```

備考 *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 ペリフェラルインターコネクション

以下に，コード生成ツールがペリフェラルインターコネクション用として出力する API 関数の一覧を示します。

表 3.18 ペリフェラルインターコネクション用 API 関数

API 関数名	機能概要
R_PICn_Create	ペリフェラルインターコネクションを制御するうえで必要となる初期化処理を行います。
R_PICn_Create_UserInit	ペリフェラルインターコネクションに関するユーザ独自の初期化処理を行います。
R_PICn_Timer_SyncStart	同時スタート許可に設定したタイマに対して，スタートトリガを生成します。

R_PIC*n*_Create

ペリフェラルインターコネクションを制御するうえで必要となる初期化処理を行います。

[指定形式]

```
void    R_PICn_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_PIC*n*_Create_UserInit

ペリフェラルインターコネクション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_PICn_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_PICn_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_PIC*n*_Timer_SyncStart

同時スタート許可に設定したタイマに対して，スタートトリガを生成します。

[指定形式]

```
void    R_PICn_Timer_SyncStart ( 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 AD コンバータ

以下に，コード生成ツールが AD コンバータ用として出力する API 関数の一覧を示します。

表 3.19 AD コンバータ用 API 関数

API 関数名	機能概要
R_ADCn_Create	AD コンバータを制御するうえで必要となる初期化処理を行います。
R_ADCn_Create_UserInit	AD コンバータに関するユーザ独自の初期化処理を行います。
r_adcn_error_interrupt	AD エラー割り込みの発生に伴う処理を行います。
r_adcn_scan_groupm_end_interrupt	スキャングループ *m* 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_adcn_multiplexer_request_interrupt	ADC*n* の MPX 割り込みの発生に伴う処理を行います。
r_adcn_adc_summation_channelm_end_interrupt	ADC*n* の積算チャネル m の積算終了割り込みの発生に伴う処理を行います。
R_ADCn_Halt	ADC*n* を停止します。
R_ADCn_SetMultiplexerCommand	ADC*n* の MPX コマンドを設定します。
R_ADCn_ScanGroupm_Start	ADC*n* のスキャングループ *m* の AD 変換を開始します。
R_ADCn_ScanGroupm_GetResult	ADC*n* のスキャングループ *m* の AD 変換結果を取得します。
R_ADCn_ScanGroupm_GetFloatingPointDataResult	ADC*n* スキャングループ *m* の浮動小数点フォーマットに変換した結果を取得します。
R_ADCn_ScanGroupm_TimerStart	ADC*n* スキャングループ *m* のタイマを開始します。
R_ADCn_ScanGroupm_TimerStop	ADC*n* スキャングループ *m* のタイマを停止します。
R_ADCn_ADCSummation_Channelm_GetResult	ADC*n* の積算チャネル *m* から積算データを取得します。
R_ADCn_ADCSummation_Start	積算処理をスタートします。
R_ADCn_ADCSummation_Stop	積算処理を停止します。
R_ADC_SyncStart	スキャングループ同期開始許可に設定した ADC*n* のスキャングループを開始します。
R_ADC_SyncTimerStart	AD タイマ同期開始許可に設定した ADC*n* のスキャングループを開始します。
R_ADCn_ScanGroupm_OperationOn	ADC*n* のスキャングループ *m* のスキャンを開始します。
R_ADCn_TH_Groupx_Start	ADC*n* の T&H のグループ *x* のホールドを開始します。
R_ADCn_TH_Sampling_Start	ADC*n* の T&H のサンプリングを開始します。

R_ADC*n*_Create

AD コンバータを制御するうえで必要となる初期化処理を行います。

[指定形式]

```
void    R_ADCn_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_ADC*n*_Create_UserInit

AD コンバータに関するユーザ独自の初期化処理を行います。

備考 本 API 関数は，R_ADCn_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_ADCn_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_adc*n*_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_adcn_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_adc*n*_scan_group*m*_end_interrupt

スキャングループ *m* 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_adcn_scan_groupm_end_interrupt ( void );
```

備考　　*n* はユニット番号，*m* 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_adc*n*_multiplexer_request_interrupt

ADC*n* の MPX 割り込みの発生に伴う処理を行います。

[指定形式]

```
void    r_adcn_multiplexer_request_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_adc*n*_adc_summation_channel*m*_end_interrupt

ADC*n* の積算チャネル *m* の積算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_adcn_adc_summation_channelm_end_interrupt ( 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_ADC*n*_Halt

ADC*n* を停止します。

[指定形式]

```
void    R_ADCn_Halt ( 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_ADC*n*_SetMultiplexerCommand

ADC*n* の MPX コマンドを設定します。

[指定形式]

```
void    R_ADCn_SetMultiplexerCommand ( 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_ADC*n*_ScanGroup*m*_Start

ADC*n* のスキャングループ *m* の AD 変換を開始します。

[指定形式]

```
void    R_ADCn_ScanGroupm_Start ( void );
```

備考 *n* はユニット番号，*m* 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_ADC*n*_ScanGroup*m*_GetResult

ADC*n* のスキャングループ *m* の AD 変換結果を取得します。

[指定形式]

```
void    R_ADCn_ScanGroupm_GetResult ( uint16_t * buffer );
```

備考　　*n* はユニット番号，*m* 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 * 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_ADC*n*_ScanGroup*m*_GetFloatingPointDataResult

ADC*n* のスキャングループ *m* の浮動小数点フォーマットに変換した結果を取得します。

[指定形式]

```
void    R_ADCn_ScanGroupm_GetFloatingPointDataResult ( uint32_t * buffer );
```

備考　　*n* はユニット番号，*m* 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 * 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_ADC*n*_ScanGroup*m*_TimerStart

ADC*n* のスキャングループ *m* のタイマを開始します。

[指定形式]

```
void    R_ADCn_ScanGroupm_TimerStart ( void );
```

備考 *n* はユニット番号，*m* 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_ADC*n*_ScanGroup*m*_TimerStop

ADC*n* スキャングループ *m* のタイマを停止します。

[指定形式]

```
void    R_ADCn_ScanGroupm_TimerStop ( void );
```

備考 *n* はユニット番号，*m* 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_ADC*n*_ADCSummation_Cannel*m*_GetResult

ADC*n* の積算チャネル *m* から積算データを取得します。

[指定形式]

```
void    R_ADCn_ADCSummation_Channelm_GetResult ( uint32_t * buffer );
```

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]

I/O	引数	説明
O	`uint32_t * buffer;`	積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_ADC*n*_ADCSummation_Start

積算処理を開始します。

[指定形式]

```
void    R_ADCn_ADCSummation_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_ADC*n*_ADCSummation_Stop

積算処理を停止します。

[指定形式]

```
void    R_ADCn_ADCSummation_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_ADC_SyncStart

スキャングループ同期開始許可に設定したADC*n*のスキャングループを開始します。

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_ADC_SyncTimerStart

AD タイマ同期開始許可に設定した ADC*n* のスキャングループを開始します。

[指定形式]

```
void    R_ADC_SyncTimerStart ( void );
```

[引数]

なし

[戻り値]

なし

R_ADC*n*_ScanGroup*m*_OperationOn

ADC*n* のスキャングループ *m* のスキャンを開始します。

[指定形式]

```
void    R_ADCn_ScanGroupm_OperationOn ( void );
```

備考　　*n* はユニット番号，*m* 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_ADC*n*_TH_Group*x*_Start

ADC*n* の T&H グループ *x* のホールドを開始します。

[指定形式]

```
void    R_ADCn_TH_Groupx_Start ( void );
```

備考 *n* はユニット番号，*x* はグループ番号を意味します。

[引数]

なし

[戻り値]

なし

R_ADC*n*_TH_Sampling_Start

ADCn の T&H のサンプリングを開始します。

[指定形式]

```
void    R_ADCn_Sampling_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

3.2.20 ΔΣAD コンバータ

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

表 3.20 ΔΣA/D コンバータ用 API 関数

API 関数名	機能概要
R_DSADC_Create	ΔΣA/D コンバータを制御するうえで必要となる初期化処理を行います。
R_DSADC_Create_UserInit	ΔΣA/D コンバータに関するユーザ独自の初期化処理を行います。
r_dsadc_error_interrupt	ΔΣAD エラー割り込みの発生伴う処理を行います。
R_DSADC_SyncStart	同期開始許可に設定した ΔΣA/D 変換を開始します。
R_DSADCm_Start	ΔΣA/D 変換を開始します。
R_DSADCm_Stop	ΔΣA/D 変換を終了します。
R_DSADCm_GetResult	ΔΣA/D 変換結果を読み出します。

R_DSADC_Create

ΔΣA/D コンバータを制御するうえで必要となる初期化処理を行います。

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_DSADC_Create_UserInit

ΔΣA/D コンバータに関するユーザ独自の初期化処理を行います。

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

[指定形式]

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

[引数]

なし

[戻り値]

なし

r_dsadc_error_interrupt

AD エラー割り込みの発生伴う処理を行います。

[指定形式]

```
void    r_dsadc_error_interrupt ( void );
```

[引数]

なし

[戻り値]

なし

R_DSADC_SyncStart

同期開始許可に設定したΔΣA/D 変換を開始します。

[指定形式]

```
void    R_DSADC_SyncStart ( void );
```

[引数]

なし

[戻り値]

なし

R_DSADC*m*_Start

A/D 変換を開始します。

[指定形式]

```
void    R_DSADCm_Start ( void );
```

備考 *m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_DSADC*m*_Stop

A/D 変換を終了します。

[指定形式]

```
void    R_DSADCm_Stop ( void );
```

備考 *m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_DSADC*m*_GetResult

A/D 変換結果を読み出します。

[指定形式]

```
void    R_DSADCm_GetResult ( DSADC_RESULT *result );
```

備考　　*m* は，チャネル番号を意味します。

[引数]

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

[戻り値]

なし

3.2.21 デジタルフィルタ

以下に，コード生成ツールがデジタルフィルタ用として出力する API 関数の一覧を示します。

表 3.21 デジタルフィルタ用 API 関数

API 関数名	機能概要
R_DFE_Create	デジタルフィル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_DFE_Create_UserInit	デジタルフィルタに関するユーザ独自の初期化処理を行います。
r_dfe_error_interrupt	エラー割り込みの発生に伴う処理を行います。
R_DFE_Set_SoftwareData	フィルタ対象データを設定します。
R_DFE_Generate_SoftwareTrigger	ソフトウェアトリガを発生します。
R_DFE_ChannelM_Create	デジタルフィルタ チャネル *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_DFE_ChannelM_Create_UserInit	デジタルフィルタ チャネル *n* に関するユーザ独自の初期化処理を行います。
r_dfe_channelm_interrupt	DFE*n* の割り込みの発生に伴う処理を行います。
r_dfe_channelm_callback_output_data	出力データの割り込みの発生に伴う処理を行います。
r_dfe_channelm_callback_condition_match	条件一致割り込みの発生に伴う処理を行います。
R_DFE_ChannelM_Enable	チャネルを有効にします。
R_DFE_ChannelM_Disable	チャネルを無効にします。
R_DFE_ChannelM_GetResult	DFE 演算結果を読み出します。

R_DFE_Create

デジタルフィルタを制御するうえで必要となる初期化処理を行います。

[指定形式]

```
void    R_DFE_Create ( void );
```

[引数]

なし

[戻り値]

なし

R_DFE_Create_UserInit

デジタルフィル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_DFE_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_DFE_Create_UserInit ( void );
```

[引数]

なし

[戻り値]

なし

r_dfe_error_interrupt

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_dfe_error_interrupt ( void );
```

[引数]

なし

[戻り値]

なし

R_DFE_Set_SoftwareData

対象フィルタに処理データを設定します。

[指定形式]

```
void    R_DFE_Set_SoftwareData ( uint8_t tag, int16_t const data );
```

[引数]

I/O	引数	説明
I	uint8_t *tag*;	割り当てるチャネルタグ
I	int16_t const *data*;	ソフトウェア入力データ

[戻り値]

なし

R_DFE_Generate_SoftwareTrigger

ソフトウェアトリガを発生します。

[指定形式]

```
void    R_DFE_Generate_SoftwareTrigger ( void );
```

[引数]

なし

[戻り値]

なし

R_DFE_Channel*m*_Create

デジタルフィルタ 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。

[指定形式]

```
void   R_DFE_Channelm_Create (void);
```

備考　　*m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_DFE_Channel*m*_Create_UserInit

デジタルフィルタ チャネル *n* に関するユーザ独自の初期化処理を行います。

備考 本 API 関数は，R_DFE_Channelm_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_DFE_Channelm_Create_UserInit ( void );
```

備考 *m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_dfe_channel*m*_interrupt

DFE 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_dfe_channelm_interrupt ( void );
```

備考　　*m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_dfe_channel*m*_callback_output_data

出力データの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_dfe_channelm_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_dfe_channelm_callback_output_data (void);
```

備考 *m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_dfe_channel*m*_callback_condition_match

条件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_dfe_channelm_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_dfe_channelm_callback_condition_match ( void );
```

備考 *m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_DFE_Channel*m*_Enable

チャネルを有効にします。

[指定形式]

```
void    R_DFE_Channelm_Enable ( void );
```

備考 *m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_DFE_Channel*m*_Disable

チャネルを無効にします。

[指定形式]

```
void    R_DFE_Channelm_Disable ( void );
```

備考　　*m* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_DFE_Channel*m*_GetResult

DFE 演算結果を読み出します。

[指定形式]

```
void    R_DFE_Channelm_GetResult ( int32_t * buffer );
```

備考　　*m* は，チャネル番号を意味します。

[引数]

I/O	引数	説明
O	`int32_t * buffer;`	DFE 演算結果を格納する領域へのポインタ

[戻り値]

なし

3.2.22 データ CRC

以下に，コード生成ツールがデータ CRC 用として出力する API 関数の一覧を示します。

表 3.22 データ CRC 用 API 関数

API 関数名	機能概要
R_DCRAn_Create	データ CRC*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_DCRAn_Create_UserInit	データ CRC*n* に関するユーザ独自の初期化処理を行います。
R_DCRAn_Input32bitData	32bit 幅の演算用入力データを設定します。
R_DCRAn_Input16bitData	16bit 幅の演算用入力データを設定します。
R_DCRAn_Input8bitData	8bit 幅の演算用入力データを設定します。
R_DCRAn_GetResult_32bitData	32bit 幅の CRC 演算結果を読み出します。
R_DCRAn_GetResult_16bitData	16bit 幅の CRC 演算結果を読み出します。

R_DCRA*n*_Create

データ CRC*n* を制御するうえで必要となる初期化処理を行います。

[指定形式]

```
void    R_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_DCRA*n*_Create_UserInit

データ CRC*n* に関するユーザ独自の初期化処理を行います。

備考 本 API 関数は，R_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_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_DCRA*n*_Input32bitData

32bit 幅の演算用入力データを設定します。

[指定形式]

```
void    R_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_DCRA*n*_Input16bitData

16bit 幅の演算用入力データを設定します。

[指定形式]

```
void    R_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_DCRA*n*_Input8bitData

8bit 幅の演算用入力データを設定します。

[指定形式]

```
void    R_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_DCRA*n*_GetResult_32bitData

32bit 幅の CRC 演算結果を読み出します。

[指定形式]

```
void    R_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_DCRA*n*_GetResult_16bitData

16bit 幅の CRC 演算結果を読み出します。

[指定形式]

```
void    R_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

3.2.23 リアルタイムクロック

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

表 3.23 リアルタイムクロック用 API 関数

API 関数名	機能概要
R_RTC_Create	リアルタイムクロックを制御するうえで必要となる初期化処理を行います。
R_RTC_Create_UserInit	リアルタイムクロックに関するユーザ独自の初期化処理を行います。
R_RTC_Start	リアルタイムクロックのカウントを開始します。
R_RTC_Stop	リアルタイムクロックのカウントを終了します。
R_RTC_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_RTC_Set_CounterValue	リアルタイムクロックにカウント値を設定します。
R_RTC_Get_CounterValue	リアルタイムクロックにカウント値を読み出します。
R_RTC_Set_AlarmOn	アラーム割り込み機能を開始します。
R_RTC_Set_AlarmOff	アラーム割り込み機能を終了します。
R_RTC_Set_AlarmValue	アラームの発生条件（曜日，時，分）を設定します。
R_RTC_Get_AlarmValue	アラームの発生条件（曜日，時，分）を読み出します。
R_RTC_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_RTC_Set_ConstPeriodInterruptOff	定周期割り込み機能を終了します。
R_RTC_Set_1secondInterruptOn	1 秒周期割り込みを開始します。
R_RTC_Set_1secondInterruptOff	1 秒周期割り込みを終了します。
R_RTC_Set_RTC1HZOn	1Hz パルス出力機能を開始します。
R_RTC_Set_RTC1HZOff	1Hz パルス出力機能を終了します。
r_rtc_interrupt_periodic	定周期割り込みの発生に伴う処理を行います。
r_rtc_interrupt_alarm	アラーム割り込みの発生に伴う処理を行います。
r_rtc_interrupt_1second	1 秒周期割り込みの発生に伴う処理を行います。

R_RTC_Create

リアルタイムクロックを制御するうえで必要となる初期化処理を行います。

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_RTC_Create_UserInit

リアルタイムクロックに関するユーザ独自の初期化処理を行います。

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

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_RTC_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_RTC_Start ( void );
```

[引数]

なし

[戻り値]

なし

R_RTC_Stop

リアルタイムクロックのカウントを終了します。

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_RTC_Set_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]

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

[引数]

I/O	引数	説明
I	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	カウント処理を停止中（設定変更後）
MD_ARGERROR	引数の指定が不正

備考　MD_BUSY1，または MD_BUSY2 が返却さえる場合は，カウンタの動作が停止している，またはカウンタの動作開始待ち時間が短いことに起因している可能性があるため、ヘッダ・ファイル r_cg_rtc.h で定義されているマクロ _RTC_WAITTIME の値を大きくしてください。

R_RTC_Set_CounterValue

リアルタイムクロックにカウント値を設定します。

[指定形式]

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

[引数]

I/O	引数	説明
I	rtc_counter_value_t *counter_write_val*;	カウント値

備考 以下に，リアルタイム・クロックのカウント値 rtc_counter_value_t の構成を示します。

```
typedef struct {
    uint8_t sec;    /* 秒 */
    uint8_t min;    /* 分 */
    uint8_t hour;   /* 時 */
    uint8_t day;    /* 日 */
    uint8_t week;   /* 曜日（0：日曜日，6：土曜日）*/
    uint8_t month;  /* 月 */
    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	カウント処理を停止中（設定変更後）

備考 MD_BUSY1，または MD_BUSY2 が返却される場合は，カウンタの動作が停止している，またはカウンタの動作開始待ち時間が短いことに起因している可能性があるため，ヘッダ・ファイル r_cg_rtc.h で定義されているマクロ _RTC_WAITTIME の値を大きくしてください。

R_RTC_Get_CounterValue

リアルタイムクロックのカウント値を読み出します。

[指定形式]

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

[引数]

I/O	引数	説明
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 カウント値 rtc_counter_value_t についての詳細は，R_RTC_Set_CounterValue を参照してください。

[戻り値]

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

備考 MD_BUSY1，または MD_BUSY2 が返却される場合は，カウンタの動作が停止している，またはカウンタの動作開始待ち時間が短いことに起因している可能性があるため，ヘッダ・ファイル r_cg_rtc.h で定義されているマクロ _RTC_WAITTIME の値を大きくしてください。

R_RTC_Set_AlarmOn

アラーム割り込み機能を開始します。

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_RTC_Set_AlarmOff

アラーム割り込み機能を終了します。

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_RTC_Set_AlarmValue

アラームの発生条件（曜日，時，分）を設定します。

[指定形式]

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

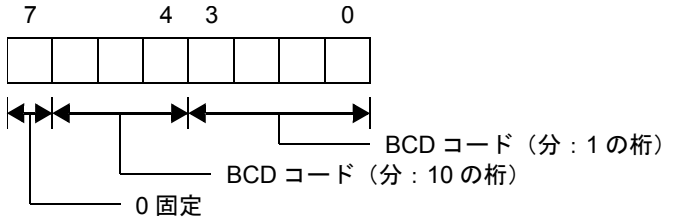
[引数]

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

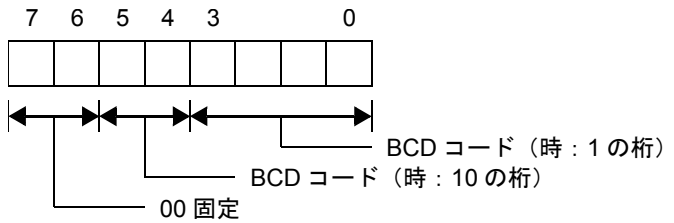
備考 以下に，アラームの発生条件 rtc_alarm_value_t の構成を示します。

```
typedef struct {
    uint8_t alarmwm;    /* 分 */
    uint8_t alarmwh;    /* 時 */
    uint8_t alarmww;    /* 曜日 */
} rtc_alarm_value_t;
```

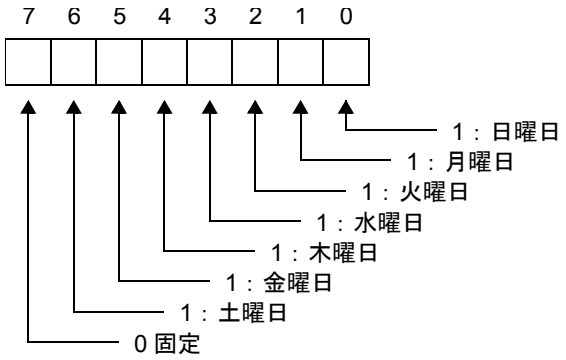
- alarmwm（分）
  以下に，構成メンバ alarmwm の各ビットに対する意味を示します。



- alarmwh（時）
  以下に，構成メンバ alarmwh の各ビットに対する意味を示します。
  なお，ビット 5 は，リアルタイム・クロックが 12 時間制の場合，以下の意味となります。
  0： 午前
  1： 午後



- alarmww（曜日）
以下に，構成メンバ alarmww の各ビットに対する意味を示します。



［戻り値］

なし

R_RTC_Get_AlarmValue

アラームの発生条件（曜日，時，分）を読み出します。

[指定形式]

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

備考　　アラームの発生条件 rtc_alarm_value_t についての詳細は，R_RTC_Set_AlarmValue を参照してください。

[引数]

I/O	引数	説明
O	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

[戻り値]

なし

備考

R_RTC_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]

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

[引数]

I/O	引数	説明
I	`rtc_int_period_t period;`	定周期割り込みの発生周期 QUARTERSEC： 0.25 秒 HALFSEC： 0.5 秒 ONESEC： 1 秒 ONEMIN： 1 分 ONEHOUR： 1 時間 ONEDAY： 1 日 ONEMONTH： 1ヵ月

[戻り値]

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

R_RTC_Set_ConstPeriodInterruptOff

定周期割り込み機能を終了します。

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_RTC_Set_1secondInterruptOn

1秒周期割り込み機能を開始します。

[指定形式]

```
void    R_RTC_Set_1secondInterruptOn ( void );
```

[引数]

なし

[戻り値]

なし

R_RTC_Set_1secondInterruptOff

1秒周期割り込み機能を終了します。

[指定形式]

```
void    R_RTC_Set_1secondInterruptOff ( void );
```

[引数]

なし

[戻り値]

なし

R_RTC_Set_RTC1HZOn

1Hz パルス出力機能を開始します。

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_RTC_Set_RTC1HZOff

1Hz パルス出力機能を終了します。

[指定形式]

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

[引数]

なし

[戻り値]

なし

r_rtc_interrupt_periodic

RTC の定周期割り込みの発生に伴う処理を行います。

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

[指定形式]

```
void    r_rtc_interrupt_periodic ( void );
```

[引数]

なし

[戻り値]

なし

r_rtc_interrupt_alarm

アラーム割り込みの発生に伴う処理を行います。

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

[指定形式]

```
void    r_rtc_interrupt_alarm ( void );
```

[引数]

なし

[戻り値]

なし

r_rtc_interrupt_1second

1 秒周期割り込みの発生に伴う処理を行います。

備考 本 API 関数は，1 秒周期割り込みに対応した割り込み処理として呼び出されます。

[指定形式]

```
void    r_rtc_interrupt_1second ( void );
```

[引数]

なし

[戻り値]

なし

3.2.24 キーリターン

以下に，コード生成がキーリターン用として出力する API 関数の一覧を示します。

表 3.24 キーリターン用 API 関数

API 関数名	機能概要
R_KEY_Create	キーリターンを制御するうえで必要となる初期化処理を行います。
R_KEY_Create_UserInit	キーリターンに関するユーザ独自の初期化処理を行います。
r_key_interrupt	キーリターンの割り込みの発生に伴う処理を行います。
R_KEY_Start	キーリターン割り込みの受け付けを許可します。
R_KEY_Stop	キーリターン割り込みの受け付けを禁止します。

R_KEY_Create

キーリターンを制御するうえで必要となる初期化処理を行います。

[指定形式]

```
void    R_KEY_Create ( void );
```

[引数]

なし

[戻り値]

なし

R_KEY_Create_UserInit

キーリターンに関するユーザ独自の初期化処理を行います。

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

[指定形式]

```
void    R_KEY_Create_UserInit ( void );
```

[引数]

なし

[戻り値]

なし

r_key_interrupt

キー割り込みの発生に伴う処理を行います。

備考　本 API 関数は，キー割り込みに対応した割り込み処理として呼び出されます。

[指定形式]

```
void    r_key_interrupt ( void );
```

[引数]

なし

[戻り値]

なし

R_KEY_Start

キーリターン割り込みの受け付けを許可します。

[指定形式]

```
void    R_KEY_Start ( void );
```

[引数]

なし

[戻り値]

なし

R_KEY_Stop

キーリターン割り込みの受け付けを禁止します。

[指定形式]

```
void    R_KEY_Stop ( void );
```

[引数]

なし

[戻り値]

なし

3.2.25 スタンバイコントローラ

以下に，コード生成がスタンバイコントローラ用として出力する API 関数の一覧を示します。

表 3.25 スタンバイコントローラ用 API 関数

API 関数名	機能概要
R_STBC_Start_Stop_Mode	スタンバイ（STOP モード）を開始します。
R_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_STBC_Start_Deep_Stop_Mode	スタンバイ（DeepSTOP モード）を開始します。
R_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_STBC_Deep_Stop_Loop	スタンバイ（DeepSTOP モード）の待ち処理を行います。
R_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_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_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_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_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_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_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_STBC_Prepare_Stop_Mode を呼び出す必要があります。

[指定形式]

```
void    R_STBC_Start_Stop_Mode ( void );
```

[引数]

なし

[戻り値]

なし

R_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_STBC_Prepare_Stop_Mode ( void );
```

[引数]

なし

[戻り値]

なし

R_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_STBC_Prepare_Deep_Stop_Mode を呼び出す必要があります。

[指定形式]

```
void    R_STBC_Start_Deep_Stop_Mode ( void );
```

[引数]

なし

[戻り値]

なし

R_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_STBC_Prepare_Deep_Stop_Mode ( void );
```

[引数]

なし

[戻り値]

なし

R_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_STBC_Deep_Stop_Loop ( void );
```

[引数]

なし

[戻り値]

なし

R_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_STBC_Prepare_Stop_Mode_Set_Peripheral ( void );
```

[引数]

なし

[戻り値]

なし

R_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_STBC_Prepare_Stop_Mode_Set_Interrupt ( void );
```

[引数]

なし

[戻り値]

なし

R_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_STBC_Prepare_Stop_Mode_Set_Clock_Mask ( void );
```

[引数]

なし

[戻り値]

なし

R_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_STBC_Prepare_Stop_Mode_Set_Clock_Source ( void );
```

[引数]

なし

[戻り値]

なし

R_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_STBC_Prepare_Deep_Stop_Mode_Set_Peripheral ( void );
```

[引数]

なし

[戻り値]

なし

R_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_STBC_Prepare_Deep_Stop_Mode_Set_Interrupt ( void );
```

[引数]

なし

[戻り値]

なし

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2015.09.01	―	初版発行
1.01	2016.03.01	38 - 40	3.2.4 割り込み API 追加
		71 - 83	3.2.7 クロック同期シリアルインタフェース G 章の追加
		120 - 132	3.2.10 UART インタフェース 章の追加
		186 - 192	3.2.14 タイマ・アレイ・ユニット B 章の追加
		193 - 199	3.2.15 タイマ・アレイ・ユニット D 章の追加
		200 - 206	3.2.16 タイマ・アレイ・ユニット J 章の追加
		237 - 239	3.2.19 AD コンバータ API 追加
		270 - 291	3.2.23 リアルタイムクロック 章の追加
		292 - 297	3.2.24 キーリターン 章の追加
		298 - 309	3.2.25 スタンバイコントローラ 章の追加

CS+ コード生成ツール　ユーザーズマニュアル
RH850 API リファレンス編

発行年月日　2016年 3月1日　　Rev.1.01

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

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口

http://www.renesas.com

※営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

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

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：http://japan.renesas.com/contact/

Colophon 3.0

CS+ コード生成ツール

ルネサス エレクトロニクス株式会社

R20UT3567JJ0101