User's Manual

e² studio コード生成

ユーザーズマニュアル　RZ API リファレンス編

対象デバイス

RZ ファミリ

ルネサス エレクトロニクス
www.renesas.com

Rev.1.02　2020.03.13

このマニュアルの使い方

対象者

このマニュアルは，コード生成ツールの機能を理解し，それを用いたアプリケーション・システムを開発するユーザを対象としています。

目的

このマニュアルは，コード生成ツールの持つソフトウ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。

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

関連資料

関連資料は暫定版の場合がありますが，この資料では「暫定」の表示をしておりません。あらかじめご了承ください。

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 */

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_mpc.c	R_MPC_Create R_MPC_Create_UserInit
	r_cg_systeminit.c	R_Systeminit
	r_cg_intprg.c	r_set_exception_handler r_fiq_handler
	r_cg_macrodriver.h	—
	r_cg_userdefine.h	—
	r_cg_interrupthandlers.h	—
	r_cg_mpc.h	—
	r_cg_nestintr_wrap.asm	—
クロック発生回路	r_cg_cgc.c	R_CGC_Create
	r_cg_cgc_user.c	R_CGC_Create_UserInit r_cgc_ostde_interrupt
	r_cg_cgc.h	—
割り込みコントローラ	r_cg_icu.c	R_ICU_Create R_ICU_IRQn_Start R_ICU_IRQn_Stop R_ICU_ETHPHYIn_Start R_ICU_ETHPHYIn_Stop
	r_cg_icu_user.c	R_ICU_Create_UserInit r_icu_nmi_interrupt r_icu_irqn_interrupt r_icu_ethphyin_interrupt
	r_cg_icu.h	—
バスステートコントローラ	r_cg_bsc.c	R_BSC_Create R_BSC_InitializeSDRAM R_BSC_SDRAMPowerDown_Start R_BSC_SDRAMPowerDown_Stop R_BSC_SDRAMDeepPowerDown_Start R_BSC_SDRAMDeepPowerDown_Stop
	r_cg_bsc_user.c	R_BSC_Create_UserInit r_bsc_bsccmi_interrupt r_bsc_tostf_interrupt
	r_cg_bsc.h	—

周辺機能	ファイル名	API 関数名
DMA コントローラ	r_cg_dmac.c	R_DMACn_Create R_DMACn_Set_SoftwareTrigger R_DMACm_Cn_Start R_DMACm_Cn_Stop R_DMACm_Cn_Suspend R_DMACm_Cn_SuspendClear
	r_cg_dmac_user.c	R_DMACn_Create_UserInit r_dmac0_dmaintn_interrupt r_dmac1_dmaint2_interrupt r_dmac0_dmasrq0_interrupt r_dmac1_dmasrq1_interrupt r_dmac0_dmaerr0_interrupt r_dmac1_dmaerr1_interrupt r_callback_dmac0_dmaintn_interrupt r_callback_dmac1_dmaint2_interrupt
	r_cg_dmac.h	—
イベントリンクコントローラ	r_cg_elc.c	R_ELC_Create R_ELC_Start R_ELC_Stop R_ELC_GenerateSoftwareEvent R_ELC_Get_PortBuffern R_ELC_Set_PortBuffern
	r_cg_elc_user.c	R_ELC_Create_UserInit r_elc_elcirqn_interrupt
	r_cg_elc.h	—
I/O ポート	r_cg_port.c	R_PORT_Create
	r_cg_port_user.c	R_PORT_Create_UserInit
	r_cg_port.h	—
マルチファンクションタイマパルスユニット 3	r_cg_mtu3.c	R_MTU3_Create R_MTU3_Cm_Start R_MTU3_Cm_Stop
	r_cg_mtu3_user.c	R_MTU3_Create_UserInit r_mtu3_tgiam_interrupt r_mtu3_tgibm_interrupt r_mtu3_tgicm_interrupt r_mtu3_tgidm_interrupt r_mtu3_tgie0_interrupt r_mtu3_tgif0_interrupt r_mtu3_tcivm_interrupt r_mtu3_tcium_interrupt r_mtu3_tgiu5_interrupt r_mtu3_tgiv5_interrupt r_mtu3_tgiw5_interrupt r_mtu3_c4_tgia4_interrupt r_mtu3_c4_tgib4_interrupt r_mtu3_c4_tciv4_interrupt r_mtu3_c7_tgia7_interrupt r_mtu3_c7_tgib7_interrupt r_mtu3_c7_tciv7_interrupt
	r_cg_mtu3.h	—

周辺機能	ファイル名	API 関数名
ポートアウトプットイネーブル 3	r_cg_poe3.c	R_POE3_Create R_POE3_Start R_POE3_Stop R_POE3_Stop R_POE3_Set_HiZ_MTUm R_POE3_Clear_HiZ_MTUm R_POE3_Set_HiZ_GPT3 R_POE3_Clear_HiZ_GPT3
	r_cg_poe3_user.c	R_POE3_Create_UserInit r_poe3_oein_interrupt
	r_cg_poe3.h	—
汎用 PWM タイマ	r_cg_gpt.c	R_GPT_Create R_GPTn_Start R_GPTn_Stop R_GPTn_HardwareStart R_GPTn_HardwareStop
	r_cg_gpt_user.c	R_GPT_Create_UserInit r_gpt_etgin_interrupt r_gpt_etgip_interrupt r_gpt_gtcian_interrupt r_gpt_gtcibn_interrupt r_gpt_gtcicn_interrupt r_gpt_gtcidn_interrupt r_gpt_gtcien_interrupt r_gpt_gtcifn_interrupt r_gpt_gdten_interrupt r_gpt_gtcivn_interrupt r_gpt_gtciun_interrupt
	r_cg_gpt.h	—
e2 studio コード生成 16 ビットタイマパルスユニット	r_cg_tpu.c	R_TPU_Create R_TPUn_Start R_TPUn_Stop
	r_cg_tpu_user.c	R_TPU_Create_UserInit r_tpu_tgina_interrupt r_tpu_tginb_interrupt r_tpu_tginc_interrupt r_tpu_tgind_interrupt r_tpu_tcinv_interrupt r_tpu_tcinu_interrupt
	r_cg_tpu.h	—
プログラマブルパルスジェネレータ	r_cg_ppg.c	R_PPG_Create
	r_cg_ppg_user.c	R_PPG_Create_UserInit
	r_cg_ppg.h	—
コンペアマッチタイマ	r_cg_cmt.c	R_CMTn_Create R_CMTn_Start R_CMTn_Stop
	r_cg_cmt_user.c	R_CMTn_Create_UserInit r_cmt_cmin_interrupt
	r_cg_cmt.h	—

周辺機能	ファイル名	API 関数名
コンペアマッチタイマ W	r_cg_cmtw.c	R_CMTWm_Create R_CMTWm_Start R_CMTWm_Stop
	r_cg_cmtw_user.c	R_CMTWm_Create_UserInit r_cmtw_cmwim_interrupt r_cmtw_icnim_interrupt r_cmtw_ocnim_interrupt
	r_cg_cmtw.h	—
ウォッチドッグタイマ	r_cg_wdt.c	R_WDTn_Create R_WDTn_Restart
	r_cg_wdt_user.c	R_WDTn_Create_UserInit r_wdtn_undff_refef_interrupt
	r_cg_wdt.h	—
独立ウォッチドッグタイマ	r_cg_iwdt.c	R_IWDT_Create R_IWDT_Restart
	r_cg_iwdt_user.c	R_IWDT_Create_UserInit r_iwdt_undff_refef_interrupt
	r_cg_iwdt.h	—
FIFO 内蔵シリアルコミュニケーションインタフェース	r_cg_scifa.c	R_SCIFAn_Create R_SCIFAn_Start R_SCIFAn_Stop R_SCIFAn_Serial_Send R_SCIFAn_Serial_Receive R_SCIFAn_Serial_Send_Receive
	r_cg_scifa_user.c	R_SCIFAn_Create_UserInit r_scifan_txifn_interrupt r_scifan_rxifn_interrupt r_scifan_brifn_interrupt r_scifan_drifn_interrupt r_scifan_callback_transmitend r_scifan_callback_receiveend r_scifan_callback_error
	r_cg_scifa.h	—

周辺機能	ファイル名	API 関数名
I2C バスインタフェース	r_cg_riic.c	R_RIICn_Create R_RIICn_Start R_RIICn_Stop R_RIICn_Master_Send R_RIICn_Master_Send_Without_Stop R_RIICn_Master_Receive R_RIICn_Slave_Send R_RIICn_Slave_Receive R_RIICn_StartCondition R_RIICn_StopCondition
	r_cg_riic_user.c	R_RIICn_Create_UserInit r_riicn_error_interrupt r_riicn_receive_interrupt r_riicn_transmit_interrupt r_riicn_transmitend_interrupt r_riicn_callback_receiveerror r_riicn_callback_transmitend r_riicn_callback_receiveend
	r_cg_riic.h	–
シリアルペリフェラルインタフェース	r_cg_rspi.c	R_RSPIn_Create R_RSPIn_Start R_RSPIn_Stop R_RSPIn_Send R_RSPIn_Send_Receive
	r_cg_rspi_user.c	R_RSPIn_Create_UserInit r_rspin_receive_interrupt r_rspin_transmit_interrupt r_rspin_error_interrupt r_rspin_idle_interrupt r_rspin_callback_receiveend r_rspin_callback_error r_rspin_callback_transmitend
	r_cg_rspi.h	–
SPI マルチ I/O バスコントローラ	r_cg_spibsc.c	R_SPIBSC_Create R_SPIBSC_EAVUpperAddressChange R_SPIBSC_SPIRead R_SPIBSC_SPIWrite R_SPIBSC_SPIRead_Write
	r_cg_spibsc_user.c	R_SPIBSC_Create_UserInit
	r_cg_spibsc.h	–
CRC 演算器	r_cg_crc.c	R_CRC_SetCRC8_2F R_CRC_SetCRC8_SAE R_CRC_SetCRC16_CCITT R_CRC_SetCRC32_ETHER R_CRC_Input_Data R_CRC_Get_Result
	r_cg_crc.h	–

周辺機能	ファイル名	API 関数名
ΔΣ インタフェース	r_cg_dsmif.c	R_DSMIF_Create R_DSMIF_UVW_Start R_DSMIF_UVW_Stop R_DSMIF_X_Start R_DSMIF_X_Stop
	r_cg_dsmif_user.c	R_DSMIF_Create_UserInit
	r_cg_dsmif.h	—

周辺機能	ファイル名	API 関数名
エラーコントロールモジュール	r_cg_ecm.c	R_ECM_Create R_ECM_Pseudo_WDT0_Error_Start R_ECM_Pseudo_WDT0_Error_Stop R_ECM_Pseudo_IWDTa_Error_Start R_ECM_Pseudo_IWDTa_Error_Stop R_ECM_Pseudo_CGC_Error_Start R_ECM_Pseudo_CGC_Error_Stop R_ECM_Pseudo_ADC_Unit0_Error_Start R_ECM_Pseudo_ADC_Unit0_Error_Stop R_ECM_Pseudo_ADC_Unit1_Error_Start R_ECM_Pseudo_ADC_Unit1_Error_Stop R_ECM_Pseudo_DSMIF_UVWovercurrent_Error_Start R_ECM_Pseudo_DSMIF_UVWovercurrent_Error_Stop R_ECM_Pseudo_DSMIF_UVWtotalcurrent_Error_Start R_ECM_Pseudo_DSMIF_UVWtotalcurrent_Error_Stop R_ECM_Pseudo_DSMIF_UVWshortcircuit_Error_Start R_ECM_Pseudo_DSMIF_UVWshortcircuit_Error_Stop R_ECM_Pseudo_DSMIF_Xovercurrent_Error_Start R_ECM_Pseudo_DSMIF_Xovercurrent_Error_Stop R_ECM_Pseudo_DSMIF_Xshortcircuit_Error_Start R_ECM_Pseudo_DSMIF_Xshortcircuit_Error_Stop R_ECM_Pseudo_DOC_Error_Start R_ECM_Pseudo_DOC_Error_Stop R_ECM_Pseudo_BSC_Error_Start R_ECM_Pseudo_BSC_Error_Stop R_ECM_Pseudo_Error35_Error_Start R_ECM_Pseudo_Error35_Error_Stop R_ECM_Pseudo_Error36_Error_Start R_ECM_Pseudo_Error36_Error_Stop R_ECM_Pseudo_Error37_Error_Start R_ECM_Pseudo_Error37_Error_Stop R_ECM_Pseudo_Error38_Error_Start R_ECM_Pseudo_Error38_Error_Stop R_ECM_Pseudo_Error39_Error_Start R_ECM_Pseudo_Error39_Error_Stop R_ECM_Pseudo_Error40_Error_Start R_ECM_Pseudo_Error40_Error_Stop R_ECM_Pseudo_Error41_Error_Start R_ECM_Pseudo_Error41_Error_Stop R_ECM_Pseudo_ECM_CompareError_Error_Start R_ECM_Pseudo_ECM_CompareError_Error_Stop R_ECM_Pseudo_ECM_DelayTimerOverflow_Error_Start R_ECM_Pseudo_ECM_DelayTimerOverflow_Error_Stop
	r_cg_ecm_user.c	R_ECM_Create_UserInit r_ecm_nmi_interrupt r_ecm_errd_interrupt r_ecm_compareerror_interrupt
	r_cg_ecm.h	—

周辺機能	ファイル名	API 関数名
12 ビット A/D コンバータ	r_cg_s12ad.c	R_S12ADn_Create R_S12ADn_Start R_S12ADn_Stop R_S12ADn_Get_ValueResult R_S12ADn_Set_CompareValue
	r_cg_s12ad_user.c	R_S12ADn_Create_UserInit r_s12ad_s12adn_interrupt r_s12ad_s12gbadin_interrupt r_s12ad_s12cmpn_interrupt r_s12ad_s12aden_interrupt
	r_cg_s12ad.h	—
データ演算回路	r_cg_doc.c	R_DOC_Create R_DOC_SetMode R_DOC_WriteData R_DOC_GetResult R_DOC_ClearFlag
	r_cg_doc_user.c	R_DOC_Create_UserInit r_doc_dopcf_interrupt
	r_cg_doc.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

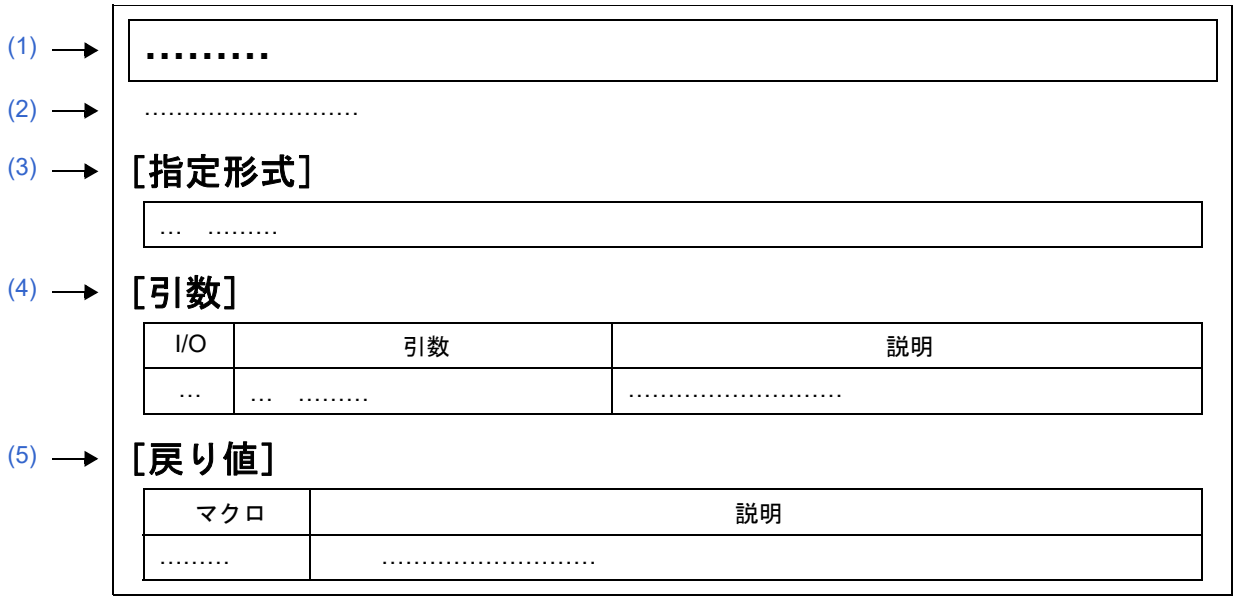
以下に，コード生成が API 関数を出力する際の命名規則を示します。

- 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
R_MPC_Create	マルチピンファンクション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_MPC_Create_UserInit	マルチピンファンクション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。
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。
r_set_exception_handler	FIQ 例外ハンドラを登録します。
r_fiq_handler	FIQ 例外処理を行います。
NESTED_INTERRUPT_PUSH	多重割り込み時のレジスタ push 処理を行います。
NESTED_INTERRUPT_POP	多重割り込み時のレジスタ pop 処理を行います。

R_MPC_Create

マルチピンファンクション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_MPC_Create ( void );
```

[引数]

なし

[戻り値]

なし

R_MPC_Create_UserInit

マルチピンファンクション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_MPC_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_MPC_Create_UserInit ( void );
```

[引数]

なし

[戻り値]

なし

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，R_MPC_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_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

r_set_exception_handler

FIQ 例外ハンドラを設定します。

[指定形式]

```
void    r_set_exception_handler ( void );
```

[引数]

なし

[戻り値]

なし

r_fiq_handler

FIQ 例外処理を行います。

[指定形式]

```
void    r_fiq_handler ( void );
```

[引数]

なし

[戻り値]

なし

NESTED_INTERRUPT_PUSH

多重割り込み時のレジスタ push 処理を行います。

備考　　本 API 関数は，多重割り込みを行う割り込み関数から呼び出されます。

[指定形式]

```
void    NESTED_INTERRUPT_PUSH ( void );
```

[引数]

なし

[戻り値]

なし

NESTED_INTERRUPT_POP

多重割り込み時のレジスタ pop 処理を行います。

備考 本 API 関数は，多重割り込みを行う割り込み関数から呼び出されます。

[指定形式]

```
void    NESTED_INTERRUPT_POP ( void );
```

[引数]

なし

[戻り値]

なし

3.2.2 クロック発生回路

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

表 3.2 クロック発生回路用 API 関数

API 関数名	機能概要
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_ostde_interrupt	メインクロック発振停止検出エラー割り込みの発生に伴う処理を行います。

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_ostde_interrupt

メインクロック発振停止検出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_cgc_ostde_interrupt(void);
```

[引数]

なし

[戻り値]

なし

3.2.3 割り込みコントローラ

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 3.3 割り込み機能用 API 関数

API 関数名	機能概要
R_ICU_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_ICU_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_ICU_IRQn_Start	IRQ*n* 割り込みを許可します。
R_ICU_IRQn_Stop	IRQ*n* 割り込み要求をマスク状態にします。
R_ICU_ETHPHYIn_Start	Ether PHY 割り込み（EHTPHY*n*）を許可します。
R_ICU_ETHPHYIn_Stop	Ether PHY 割り込み（EHTPHYI*n*）要求をマスク状態にします。
r_icu_nmi_interrupt	NMI ノンマスカブル割り込み要求をクリアします。
r_icu_irqn_interrupt	IRQ*n* の発生に伴う処理を行います。
r_icu_ethphyin_interrupt	EHTPHYI*n* の発生に伴う処理を行います。

R_ICU_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_ICU_Create ( void );
```

[引数]

なし

[戻り値]

なし

R_ICU_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_ICU_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_ICU_Create_UserInit ( void );
```

[引数]

なし

[戻り値]

なし

R_ICU_IRQ*n*_Start

IRQ*n* 割り込みを許可します。

[指定形式]

```
void    R_ICU_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_ICU_IRQ*n*_Stop

IRQ*n* 割り込み要求をマスク状態にします。

[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_ICU_ETHPHYI*n*_Start

Ether PHY 割り込み（EHTPHY*n*）を許可します。

[指定形式]

```
void    R_ICU_ETHPHYIn_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_ICU_ETHPHYI*n*_Stop

Ether PHY 割り込み（EHTPHYI*n*）要求をマスク状態にします。

[指定形式]

```
void    R_ICU_ETHPHYIn_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_icu_nmi_interrupt

NMI ノンマスカブル割り込み要求をクリアします。

[指定形式]

```
void    r_icu_nmi_interrupt ( void );
```

[引数]

なし

[戻り値]

なし

r_icu_irq*n*_interrupt

IRQ*n* の発生に伴う処理を行います。

[指定形式]

```
void    r_icu_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_icu_ethphyi*n*_interrupt

EHTPHYI*n* の発生に伴う処理を行います。

[指定形式]

```
void    r_icu_ethphyin_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

3.2.4 バスステートコントローラ

以下に，コード生成がバスステート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 3.4 バスステートコントローラ用 API 関数

API 関数名	機能概要
R_BSC_Create	バスステート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_BSC_Create_UserInit	バスステート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_bsc_bsccmi_interrupt	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_bsc_tostf_interrupt	外部 WAIT 端子による長期アクセスウェイト検出エラー割り込みの発生に伴う処理を行います。
R_BSC_InitializeSDRAM	SDRAM の初期化処理を行います。
R_BSC_SDRAMPowerDown_Start	SDRAM に対するアクセス終了後に、SDRAM をパワーダウンモードに遷移させる設定を行います。
R_BSC_SDRAMPowerDown_Stop	SDRAM に対するアクセス終了後に、SDRAM をパワーダウンモードに遷移させる設定を解除します。
R_BSC_SDRAMDeepPowerDown_Start	ローパワー SDRAM をディープパワーダウンモードに遷移させます。
R_BSC_SDRAMDeepPowerDown_Stop	ローパワー SDRAM をセルフリフレッシュモードに遷移させます。

R_BSC_Create

バスステート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_BSC_Create ( void );
```

[引数]

なし

[戻り値]

なし

R_BSC_Create_UserInit

バスステート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_BSC_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_BSC_Create_UserInit ( void );
```

[引数]

なし

[戻り値]

なし

r_bsc_bsccmi_interrupt

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_bsc_bsccmi_interrupt ( void );
```

[引数]

なし

[戻り値]

なし

r_bsc_tostf_interrupt

外部 WAIT 端子による長期アクセスウェイト検出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_bsc_tostf_interrupt ( void );
```

[引数]

なし

[戻り値]

なし

R_BSC_InitializeSDRAM

SDRAM の初期化処理を行います。

[指定形式]

```
void    R_BSC_InitializeSDRAM ( void );
```

[引数]

なし

[戻り値]

なし

R_BSC_SDRAMPowerDown_Start

SDRAM に対するアクセス終了後に、SDRAM をパワーダウンモードに遷移させる設定を行います。

[指定形式]

```
void    R_BSC_SDRAMPowerDown_Start ( void );
```

[引数]

なし

[戻り値]

なし

R_BSC_SDRAMPowerDown_Stop

SDRAM に対するアクセス終了後に、SDRAM をパワーダウンモードに遷移させる設定を解除します。

[指定形式]

```
void    R_BSC_SDRAMPowerDown_Stop ( void );
```

[引数]

なし

[戻り値]

なし

R_BSC_SDRAMDeepPowerDown_Start

ローパワー SDRAM をディープパワーダウンモードに遷移させます。

[指定形式]

```
void    R_BSC_SDRAMDeepPowerDown_Start ( void );
```

[引数]

なし

[戻り値]

なし

R_BSC_SDRAMDeepPowerDown_Stop

ローパワー SDRAM をセルフリフレッシュモードに遷移させます。

[指定形式]

```
void    R_BSC_SDRAMDeepPowerDown_Start ( void );
```

[引数]

なし

[戻り値]

なし

3.2.5 DMA コントローラ

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

表 3.5 DMA コントローラ用 API 関数

API 関数名	機能概要
R_DMACn_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。
R_DMACn_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。
R_DMACn_Set_SoftwareTrigger	DMA 転送要求を発生します。
r_dmac0_dmaintn_interrupt	DMAC0 の外部 DMA リクエスト n 割り込み の発生に伴う処理を行います。
r_dmac1_dmaint2_interrupt	DMAC1 の外部 DMA リクエスト 2 割り込み の発生に伴う処理を行います。
r_dmac0_dmasrq0_interrupt	DMAC0 の DMA ソフトウェア起動リクエスト 0 割り込みの発生に伴う処理を行います。
r_dmac1_dmasrq1_interrupt	DMAC1 の DMA ソフトウェア起動リクエスト 1 割り込みの発生に伴う処理を行います。
r_dmac0_dmaerr0_interrupt	DMAC0 の DMA 転送エラー 0 割り込みの発生に伴う処理を行います。
r_dmac1_dmaerr1_interrupt	DMAC1 の DMA 転送エラー 1 割り込みの発生に伴う処理を行います。
r_callback_dmac0_dmaintn_interrupt	DMAC0 の外部 DMA リクエスト n 割り込み (n=0~1: リクエスト信号番号) の発生に伴うコールバック処理を行います。
r_callback_dmac1_dmaint2_interrupt	DMAC1 の外部 DMA リクエスト 2 割り込みの発生に伴うコールバック処理を行います。
R_DMACm_Cn_Start	DMA チャネル *n* の DMA 転送を許可します。
R_DMACm_Cn_Stop	DMA チャネル *n* の DMA 転送を停止します。
R_DMACm_Cn_Suspend	DMA チャネル *n* の DMA 転送を一時停止します。
R_DMACm_Cn_SuspendClear	DMA チャネル *n* 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*_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_dmac0_dmaint*n*_interrupt

DMAC0 の外部 DMA リクエスト n 割り込み (n=0~2: リクエスト信号番号) の発生に伴う処理を行います。

[指定形式]

```
void r_dmac0_dmaintn_interrupt(void);
```

備考　　*n* は，リクエスト信号番号を意味します。

[引数]

なし

[戻り値]

なし

r_dmac1_dmaint2_interrupt

DMAC1 の外部 DMA リクエスト 2 割り込みの発生に伴う処理を行います。

[指定形式]

```
void r_dmac1_dmaint2_interrupt(void);
```

[引数]

なし

[戻り値]

なし

r_dmac0_dmasrq0_interrupt

DMAC0 の DMA ソフトウェア起動リクエスト 0 割り込みの発生に伴う処理を行います。

[指定形式]

```
void r_dmac0_dmasrq0_interrupt(void);
```

[引数]

なし

[戻り値]

なし

r_dmac1_dmasrq1_interrupt

DMAC1 の DMA ソフトウェア起動リクエスト 1 割り込みの発生に伴う処理を行います。

[指定形式]

```
void r_dmac1_dmasrq1_interrupt(void);
```

[引数]

なし

[戻り値]

なし

r_dmac0_dmaerr0_interrupt

DMAC0 の DMA 転送エラー 0 割り込みの発生に伴う処理を行います。

[指定形式]

```
void r_dmac0_dmaerr0_interrupt(void);
```

[引数]

なし

[戻り値]

なし

r_dmac1_dmaerr1_interrupt

DMAC1 の DMA 転送エラー 1 割り込みの発生に伴う処理を行います。

[指定形式]

```
void r_dmac1_dmaerr1_interrupt(void);
```

[引数]

なし

[戻り値]

なし

r_callback_dmac0_dmaint*n*_interrupt

DMAC0 の外部 DMA リクエスト n 割り込み (n=0~1: リクエスト信号番号) の発生に伴うコールバック処理を行います。

[指定形式]

```
void r_callback_dmac0_dmaintn_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_callback_dmac1_dmaint2_interrupt

DMAC1 の外部 DMA リクエスト 2 割り込みの発生に伴うコールバック処理を行います。

[指定形式]

```
void r_callback_dmac1_dmaint2_interrupt(void);
```

[引数]

なし

[戻り値]

なし

R_DMAC*n*_Set_SoftwareTrigger

DMA 転送要求を発生します。

[指定形式]

```
void    R_DMACn_Set_SoftwareTrigger ( 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*m*_C*n*_Start

DMA チャネル *n* の DMA 転送を許可します。

[指定形式]

```
void    R_DMACm_Cn_Start ( void );
```

備考　　*m* はユニット番号を，*n* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_DMAC*m*_C*n*_Stop

DMA チャネル *n* の DMA 転送を停止します。

[指定形式]

```
void    R_DMACm_Cn_Stop ( void );
```

備考　　*m* はユニット番号を，*n* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_DMAC*m*_C*n*_Suspend

DMA チャネル *n* の DMA 転送を一時停止します。

[指定形式]

```
void    R_DMAC*m*_C*n*_Suspend ( void );
```

備考　　*m* はユニット番号を，*n* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_DMAC*m*_C*n*_SuspendClear

DMA チャネル *n* の DMA 転送を一時停止を解除します。

[指定形式]

```
void    R_DMAC*m*_C*n*_SuspendClear ( void );
```

備考　　*m* はユニット番号を，*n* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

3.2.6 イベントリンクコントローラ

以下に，コード生成がイベントリンクコントローラ用として出力する API 関数の一覧を示します。

表 3.6 イベントリンクコントローラ用 API 関数

API 関数名	機能概要
R_ELC_Create	イベントリンクコントローラを制御するうえで必要となる初期化処理を行います。
R_ELC_Create_UserInit	イベントリンクコントローラに関するユーザ独自の初期化処理を行います。
r_elc_elcirqn_interrupt	イベントリンクコントローラを動作停止状態にします。
R_ELC_Start	ELC 機能を有効にします。
R_ELC_Stop	ELC 機能を無効にします。
R_ELC_GenerateSoftwareEvent	ソフトウエアイベント無効が発生します。
R_ELC_Get_PortBuffern	ポートバッファレジスタ *n*（PDBF*n*）に値を読み出します。
R_ELC_Set_PortBuffern	ポートバッファレジスタ *n*（PDBF*n*）に値を書き込みます。

R_ELC_Create

イベントリンクコントローラを制御するうえで必要となる初期化処理を行います。

[指定形式]

```
void    R_ELC_Create ( void );
```

[引数]

なし

[戻り値]

なし

R_ELC_Create_UserInit

イベントリンクコントローラに関するユーザ独自の初期化処理を行います。

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

[指定形式]

```
void    R_ELC_Create_UserInit ( void );
```

[引数]

なし

[戻り値]

なし

r_elc_elcirq*n*_interrupt

イベントリンクコントローラ割り込みの発生に伴う処理を行います。

[指定形式]

```
void    r_elc_elcirqn_interrupt ( void );
```

備考　　*n* はポートグループ番号を意味します。

[引数]

なし

[戻り値]

なし

R_ELC_Start

ELC 機能を有効にします。

[指定形式]

```
void    R_ELC_Start ( void );
```

[引数]

なし

[戻り値]

なし

R_ELC_Stop

ELC 機能を無効にします。

[指定形式]

```
void    R_ELC_Stop ( void );
```

[引数]

なし

[戻り値]

なし

R_ELC_GenerateSoftwareEvent

ソフトウエアイベント無効が発生します。

[指定形式]

```
void    R_ELC_GenerateSoftwareEvent ( void );
```

[引数]

なし

[戻り値]

なし

R_ELC_Get_PortBuffer*n*

ポートバッファレジスタ *n*（PDBF*n*）に値を読み出します。

[指定形式]

```
void    R_ELC_Get_PortBuffern ( uint8_t * const value );
```

[引数]

I/O	引数	説明
I	`uint8_t * const value;`	バッファレジスタの値を格納するポインタ

備考　　*n* はポートグループ番号を意味します。

[戻り値]

なし

R_ELC_Set_PortBuffer*n*

ポートバッファレジスタ *n*（PDBF*n*）に値を書き込みます。

[指定形式]

```
void    R_ELC_Set_PortBuffern ( uint8_t  value );
```

[引数]

I/O	引数	説明
I	uint8_t value;	バッファレジスタに書き込む値

備考　　*n* はポートグループ番号を意味します。

[戻り値]

なし

3.2.7 I/O ポート

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 3.7 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

3.2.8 マルチファンクションタイマパルスユニット 3

以下に，コード生成がマルチファンクションタイマパルスユニット 3 用として出力する API 関数の一覧を示します。

表 3.8 マルチファンクションタイマパルスユニット 3 用 API 関数

API 関数名	機能概要
R_MTU3_Create	マルチファンクションタイマパルスユニット 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_MTU3_Create_UserInit	マルチファンクションタイマパルスユニット 3 に関するユーザ独自の初期化処理を行います。
r_mtu3_tgiam_interrupt	チャネル *m* インプットキャプチャ / コンペアマッチ A 割り込みの発生に伴う処理を行います。
r_mtu3_tgibm_interrupt	チャネル *m* インプットキャプチャ / コンペアマッチ B 割り込みの発生に伴う処理を行います。
r_mtu3_tgicm_interrupt	チャネル *m* インプットキャプチャ / コンペアマッチ C 割り込みの発生に伴う処理を行います。
r_mtu3_tgidm_interrupt	チャネル *m* インプットキャプチャ / コンペアマッチ D 割り込みの発生に伴う処理を行います。
r_mtu3_tgie0_interrupt	チャネル 0 インプットキャプチャ / コンペアマッチ E 割り込みの発生に伴う処理を行います。
r_mtu3_tgif0_interrupt	チャネル 0 インプットキャプチャ / コンペアマッチ F 割り込みの発生に伴う処理を行います。
r_mtu3_tcivm_interrupt	チャネル *m* オーバフロー割り込みの発生に伴う処理を行います。
r_mtu3_tcium_interrupt	チャネル *m* アンダーフロー割り込みの発生に伴う処理を行います。
r_mtu3_tgiu5_interrupt	チャネル 5 インプットキャプチャ / コンペアマッチ U 割り込みの発生に伴う処理を行います。
r_mtu3_tgiv5_interrupt	チャネル 5 インプットキャプチャ / コンペアマッチ V 割り込みの発生に伴う処理を行います。
r_mtu3_tgiw5_interrupt	チャネル 5 インプットキャプチャ / コンペアマッチ W 割り込みの発生に伴う処理を行います。
r_mtu3_c4_tgia4_interrupt	TGIA4 割り込みの発生に伴う処理を行います。
r_mtu3_c4_tgib4_interrupt	TGIB 割り込みの発生に伴う処理を行います。
r_mtu3_c4_tciv4_interrupt	TCIV4 割り込みの発生に伴う処理を行います。
r_mtu3_c7_tgia7_interrupt	TGIA7 割り込みの発生に伴う処理を行います。
r_mtu3_c7_tgib7_interrupt	TGIB7 割り込みの発生に伴う処理を行います。
r_mtu3_c7_tciv7_interrupt	TCIV7 割り込みの発生に伴う処理を行います。
R_MTU3_Cm_Start	チャネル *m* のカウントを開始します。
R_MTU3_Cm_Stop	チャネル *m* のカウントを終了します。

R_MTU3_Create

マルチファンクションタイマパルスユニット 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_MTU3_Create ( void );
```

[引数]

なし

[戻り値]

なし

R_MTU3_Create_UserInit

マルチファンクションタイマパルスユニット 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_MTU3_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_MTU3_Create_UserInit ( void );
```

[引数]

なし

[戻り値]

なし

r_mtu3_tgia*m*_interrupt

チャネル *m* インプットキャプチャ / コンペアマッチ A 割り込みの発生に伴う処理を行います。

備考　本 API 関数は，TGIA*m* 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_mtu3_tgiam_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_mtu3_tgib*m*_interrupt

チャネル *m* インプットキャプチャ / コンペアマッチ B 割り込みの発生に伴う処理を行います。

備考　本 API 関数は，TGIB*m* 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_mtu3_tgibm_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_mtu3_tgic*m*_interrupt

チャネル *m* インプットキャプチャ / コンペアマッチ C 割り込みの発生に伴う処理を行います。

備考　本 API 関数は，TGIC*m* 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_mtu3_tgicm_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_mtu3_tgid*m*_interrupt

チャネル *m* インプットキャプチャ / コンペアマッチ D 割り込みの発生に伴う処理を行います。

備考　本 API 関数は，TGID*m* 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_mtu3_tgidm_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_mtu3_tgie0_interrupt

チャネル 0 コンペアマッチ E 割り込みの発生に伴う処理を行います。

備考　本 API 関数は，TGIE0 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_mtu3_tgie0_interrupt ( void );
```

[引数]

なし

[戻り値]

なし

r_mtu3_tgif0_interrupt

チャネル０コンペアマッチＦ割り込みの発生に伴う処理を行います。

備考　本 API 関数は，TGIF0 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_mtu3_tgif0_interrupt ( void );
```

[引数]

なし

[戻り値]

なし

r_mtu3_tciv*m*_interrupt

チャネル *m* 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，TCIV*m* 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_mtu3_tcivm_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_mtu3_tciu*m*_interrupt

チャネル *m* アンダー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，TCIU*m* 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_mtu3_tcium_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_mtu3_tgiu5_interrupt

チャネル 5 インプットキャプチャ / コンペアマッチ U 割り込みの発生に伴う処理を行います。

備考　本 API 関数は，TGIU5 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_mtu3_tgiu5_interrupt ( void );
```

[引数]

なし

[戻り値]

なし

r_mtu3_tgiv5_interrupt

チャネル 5 インプットキャプチャ / コンペアマッチ V 割り込みの発生に伴う処理を行います。

備考　本 API 関数は，TGIV5 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_mtu3_tgiv5_interrupt ( void );
```

[引数]

なし

[戻り値]

なし

r_mtu3_tgiw5_interrupt

チャネル 5 インプットキャプチャ / コンペアマッチ W 割り込みの発生に伴う処理を行います。

備考　本 API 関数は，TGIW5 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_mtu3_tgiw5_interrupt ( void );
```

[引数]

なし

[戻り値]

なし

r_mtu3_c4_tgia4_interrupt

TGIA4 割り込みの発生に伴う処理を行います。

備考　本 API 関数は，TGIA4 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_mtu3_c4_tgia4_interrupt ( void );
```

[引数]

なし

[戻り値]

なし

r_mtu3_c4_tgib4_interrupt

TGIB4 割り込みの発生に伴う処理を行います。

備考 本 API 関数は，TGIB4 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_mtu3_c4_tgib4_interrupt ( void );
```

[引数]

なし

[戻り値]

なし

r_mtu3_c4_tciv4_interrupt

TCIV4 割り込みの発生に伴う処理を行います。

備考 本 API 関数は，TCIV4 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_mtu3_c4_tciv4_interrupt ( void );
```

[引数]

なし

[戻り値]

なし

r_mtu3_c7_tgia7_interrupt

TGIA7 割り込みの発生に伴う処理を行います。

備考 本 API 関数は，TGIA7 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_mtu3_c7_tgia7_interrupt ( void );
```

[引数]

なし

[戻り値]

なし

r_mtu3_c7_tgib7_interrupt

TGIB7 割り込みの発生に伴う処理を行います。

備考 本 API 関数は，TGIB7 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_mtu3_c7_tgib7_interrupt ( void );
```

[引数]

なし

[戻り値]

なし

r_mtu3_c7_tciv7_interrupt

TCIV7 割り込みの発生に伴う処理を行います。

備考　本 API 関数は，TCIV7 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_mtu3_c7_tciv7_interrupt ( void );
```

[引数]

なし

[戻り値]

なし

R_MTU3_C*m*_Start

チャネル *m* のカウントを開始します。

[指定形式]

```
void    R_MTU3_Cm_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_MTU3_C*m*_Stop

チャネル *m* のカウントを終了します。

[指定形式]

```
void    R_MTU3_Cm_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

3.2.9 ポートアウトプットイネーブル 3

以下に，コード生成がポートアウトプットイネーブル 3 用として出力する API 関数の一覧を示します。

表 3.9 ポートアウトプットイネーブル 3 用 API 関数

API 関数名	機能概要
R_POE3_Create	ポートアウトプットイネーブル 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_POE3_Create_UserInit	ポートアウトプットイネーブル 3 に関するユーザ独自の初期化処理を行います。
r_poe3_oein_interrupt	アウトプットイネーブル割り込みの発生に伴う処理を行います。
R_POE3_Start	端子をハイインピーダンスにします。
R_POE3_Stop	端子のハイインピーダンス状態を解除します。
R_POE3_Set_HiZ_MTUm	MTU*m* 端子をハイインピーダンスにします。*m* の値により MTU*m* 端子または GPT*m* 端子をハイインピーダンスにする場合があります。
R_POE3_Clear_HiZ_MTUm	MTU*m* 端子のハイインピーダンス状態を解除します。*m* の値により MTU*m* 端子または GPT*m* 端子のハイインピーダンス状態を解除する場合があります。
R_POE3_Set_HiZ_GPT3	GPT3 端子をハイインピーダンス状態にします。
R_POE3_Clear_HiZ_GPT3	GPT3 端子のハイインピーダンス状態を解除します。

R_POE3_Create

ポートアウトプットイネーブル 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_POE3_Create ( void );
```

[引数]

なし

[戻り値]

なし

R_POE3_Create_UserInit

ポートアウトプットイネーブル 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_POE3_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_POE3_Create_UserInit ( void );
```

[引数]

なし

[戻り値]

なし

r_poe3_oei*n*_interrupt

アウトプットイネーブル割り込みの発生に伴う処理を行います。

[指定形式]

```
void    r_poe3_oein_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_POE3_Start

端子をハイインピーダンスにします。

[指定形式]

```
void    R_POE3_Start ( void );
```

[引数]

なし

[戻り値]

なし

R_POE3_Stop

端子のハイインピーダンス状態を解除します。

[指定形式]

```
void    R_POE3_Stop ( void );
```

[引数]

なし

[戻り値]

なし

R_POE3_Set_HiZ_MTU*m*

MTU*m* 端子をハイインピーダンスにします。*m* の値により MTU*m* 端子または GPT*m* 端子をハイインピーダンスにする場合があります。

[指定形式]

```
void    R_POE3_Set_HiZ_MTUm( void );
```

備考　*m* はチャネル番号を意味します。値域として、*m*=0, 3_4, 6_7 を取ります。*m*=3_4 の場合は、MTU3~MTU4 端子または GPT0~GPT2 端子、*m*=6_7 の場合は、MTU6~MTU7 端子を指定します。

[引数]

なし

[戻り値]

なし

R_POE3_Clear_HiZ_MTU*m*

MTU*m* 端子のハイインピーダンス状態を解除します。*m* の値により MTU*m* 端子または GPT*m* 端子のハイインピーダンス状態を解除する場合があります。

[指定形式]

```
void    R_POE3_Clear_HiZ_MTUm( void );
```

備考　*m* はチャネル番号を意味します。値域として、*m*=0, 3_4, 6_7 を取ります。*m*=3_4 の場合は、MTU3~MTU4 端子または GPT0~GPT2 端子を、*m*=6_7 の場合は、MTU6~MTU7 端子を指定します。

[引数]

なし

[戻り値]

なし

R_POE3_Set_HiZ_GPT3

GPT3 端子をハイインピーダンス状態にします。

[指定形式]

```
void R_POE3_Set_HiZ_GPT3(void);
```

[引数]

なし

[戻り値]

なし

R_POE3_Clear_HiZ_GPT3

GPT3 端子のハイインピーダンス状態を解除します。

[指定形式]

```
void    R_POE3_Clear_HiZ_GPT3( void );
```

[引数]

なし

[戻り値]

なし

3.2.10 汎用 PWM タイマ

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

表 3.10 汎用 PWM タイマ用 API 関数

API 関数名	機能概要
R_GPT_Create	汎用 PWM 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_GPT_Create_UserInit	汎用 PWM タイマに関するユーザ独自の初期化処理を行います。
r_gpt_etgin_interrupt	GPT の割り込み要因 ETGIN（外部トリガ立ち下り入力）の発生に伴う処理を行います。
r_gpt_etgip_interrupt	GPT の割り込み要因 ETGIP（外部トリガ立ち上り入力）の発生に伴う処理を行います。
r_gpt_gtcian_interrupt	GPT の割り込み要因 GTCIA*n*（GPT*n*.GTCCRA の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。
r_gpt_gtcibn_interrupt	GPT の割り込み要因 GTCIB*n*（GPT*n*.GTCCRB の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。
r_gpt_gtcicn_interrupt	GPT の割り込み要因 GTCIC*n*（GPT*n*.GTCCRC のコンペアマッチ）の発生に伴う処理を行います。
r_gpt_gtcidn_interrupt	GPT の割り込み要因 GTCID*n*（GPT*n*.GTCCRD のコンペアマッチ）の発生に伴う処理を行います。
r_gpt_gtcien_interrupt	GPT の割り込み要因 GTCIE*n*（GPT*n*.GTCCRE のコンペアマッチ）の発生に伴う処理を行います。
r_gpt_gtcifn_interrupt	GPT の割り込み要因 GTCIF*n*（GPT*n*.GTCCRF のコンペアマッチ）の発生に伴う処理を行います。
r_gpt_gdten_interrupt	GPT の割り込み要因 GDTE*n*（デッドタイムエラー）の発生に伴う処理を行います。
r_gpt_gtcivn_interrupt	GPT の割り込み要因 GTCIV*n*（GPT*n*.GTCNT のオーバフロー または GPT*n*.GTPR のコンペアマッチ）の発生に伴う処理を行います。
r_gpt_gtciun_interrupt	GPT の割り込み要因 GTCIU*n*（GPT*n*.GTCNT のアンダフロー）の発生に伴う処理を行います。
R_GPTn_Start	GPT*n*.GTCNT カウンタのカウント処理を開始します。
R_GPTn_Stop	GPT*n*.GTCNT カウンタのカウント処理を停止します。
R_GPTn_HardwareStart	GPT*n* 割り込み (GPT の割り込み要因 GDTE*n*（デッドタイムエラー）を除く) を許可します。
R_GPTn_HardwareStop	GPT*n* 割り込み (GPT の割り込み要因 GDTE*n*（デッドタイムエラー）を除く) を禁止します。

R_GPT_Create

汎用 PWM 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_GPT_Create ( void );
```

[引数]

なし

[戻り値]

なし

R_GPT_Create_UserInit

汎用 PWM 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_GPT_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_GPT_Create_UserInit ( void );
```

[引数]

なし

[戻り値]

なし

r_gpt_etgin_interrupt

GPT の割り込み要因 ETGIN（外部トリガ立ち下り入力）の発生に伴う処理を行います。

[指定形式]

```
void    r_gpt_etgin_interrupt ( void );
```

[引数]

なし

[戻り値]

なし

r_gpt_etgip_interrupt

GPT の割り込み要因 ETGIP（外部トリガ立ち上り入力）の発生に伴う処理を行います。

[指定形式]

```
void    r_gpt_etgip_interrupt ( void );
```

[引数]

なし

[戻り値]

なし

r_gpt_gtcia*n*_interrupt

GPT の割り込み要因 GTCIA*n*（GPT*n*.GTCCRA の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_gpt_gtcian_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_gpt_gtcib*n*_interrupt

GPT の割り込み要因 GTCIB*n*（GPT*n*.GTCCRB の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_gpt_gtcibn_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_gpt_gtcic*n*_interrupt

GPT の割り込み要因 GTCIC*n*（GPT*n*.GTCCRC のコンペアマッチ）の発生に伴う処理を行います。

[指定形式]

```
void    r_gpt_gtcicn_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_gpt_gtcid*n*_interrupt

GPT の割り込み要因 GTCID*n*（GPT*n*.GTCCRD のコンペアマッチ）の発生に伴う処理を行います。

[指定形式]

```
void    r_gpt_gtcidn_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_gpt_gtcie*n*_interrupt

GPT の割り込み要因 GTCIE*n*（GPT*n*.GTCCRE のコンペアマッチ）の発生に伴う処理を行います。

[指定形式]

```
void    r_gpt_gtcien_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_gpt_gtcif*n*_interrupt

GPT の割り込み要因 GTCIF*n*（GPT*n*.GTCCRF のコンペアマッチ）の発生に伴う処理を行います。

[指定形式]

```
void    r_gpt_gtcifn_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_gpt_gdte*n*_interrupt

GPT の割り込み要因 GDTE*n*（デッドタイムエラー）の発生に伴う処理を行います。

[指定形式]

```
void    r_gpt_gdten_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_gpt_gtciv*n*_interrupt

GPT の割り込み要因 GTCIV*n*（GPT*n*.GTCNT のオーバフロー または GPT*n*.GTPR のコンペアマッチ）の発生に伴う処理を行います。

[指定形式]

```
void    r_gpt_gtciv*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

r_gpt_gtciu*n*_interrupt

GPT の割り込み要因 GTCIU*n*（GPT*n*.GTCNT のアンダフロー）の発生に伴う処理を行います。

[指定形式]

```
void    r_gpt_gtciun_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_GPT*n*_Start

GPT*n*.GTCNT カウンタのカウント処理を開始します。

[指定形式]

```
void    R_GPTn_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_GPT*n*_Stop

GPT*n*.GTCNT カウンタのカウント処理を停止します。

[指定形式]

```
void    R_GPTn_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_GPT*n*_HardwareStart

GPT*n* 割り込み (GPT の割り込み要因 GDTE*n*（デッドタイムエラー）を除く) を許可します。

以下の GPT の割り込み要因の GPT*n* 割り込みを許可します。

GPT の割り込み要因		GPT*n* 割り込み
GTCIA*n*	GPT*n*.GTCCRA のインプットキャプチャ / コンペアマッチ	チャネル *n* インプットキャプチャ / コンペアマッチ A 割り込み
GTCIB*n*	GPT*n*.GTCCRB のインプットキャプチャ / コンペアマッチ	チャネル *n* インプットキャプチャ / コンペアマッチ B 割り込み
GTCIC*n*	GPT*n*.GTCCRC のコンペアマッチ	チャネル *n* コンペアマッチ C 割り込み
GTCID*n*	GPT*n*.GTCCRD のコンペアマッチ	チャネル *n* コンペアマッチ D 割り込み
GTCIE*n*	GPT*n*.GTCCRE のコンペアマッチ	チャネル *n* コンペアマッチ E 割り込み
GTCIF*n*	GPT*n*.GTCCRF のコンペアマッチ	チャネル *n* コンペアマッチ F 割り込み
GTCIV*n*	GPT*n*.GTCNT のオーバフロー	チャネル *n* オーバフロー割り込み
GTCIU*n*	GPT*n*.GTCNT のアンダフロー	チャネル *n* アンダーフロー割り込み

[指定形式]

```
void    R_GPTn_HardwareStart ( 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_GPT*n*_HardwareStop

GPT*n* 割り込み (GPT の割り込み要因 GDTE*n*（デッドタイムエラー）を除く) を禁止します。

以下の GPT の割り込み要因の GPT*n* 割り込みを禁止します。

GPT の割り込み要因		GPT*n* 割り込み
GTCIA*n*	GPT*n*.GTCCRA のインプットキャプチャ / コンペアマッチ	チャネル *n* インプットキャプチャ / コンペアマッチ A 割り込み
GTCIB*n*	GPT*n*.GTCCRB のインプットキャプチャ / コンペアマッチ	チャネル *n* インプットキャプチャ / コンペアマッチ B 割り込み
GTCIC*n*	GPT*n*.GTCCRC のコンペアマッチ	チャネル *n* コンペアマッチ C 割り込み
GTCID*n*	GPT*n*.GTCCRD のコンペアマッチ	チャネル *n* コンペアマッチ D 割り込み
GTCIE*n*	GPT*n*.GTCCRE のコンペアマッチ	チャネル *n* コンペアマッチ E 割り込み
GTCIF*n*	GPT*n*.GTCCRF のコンペアマッチ	チャネル *n* コンペアマッチ F 割り込み
GTCIV*n*	GPT*n*.GTCNT のオーバフロー	チャネル *n* オーバフロー割り込み
GTCIU*n*	GPT*n*.GTCNT のアンダフロー	チャネル *n* アンダーフロー割り込み

[指定形式]

```
void    R_GPTn_HardwareStop ( 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

3.2.11 e² studio コード生成 16 ビットタイマパルスユニット

以下に，コード生成が 16 ビットタイマパルスユニット用として出力する API 関数の一覧を示します。

表 3.11 16 ビットタイマパルスユニット用 API 関数

API 関数名	機能概要
R_TPU_Create	16 ビットタイマパルスユニットを制御するうえで必要となる初期化処理を行います。
R_TPU_Create_UserInit	16 ビットタイマパルスユニットに関するユーザ独自の初期化処理を行います。
r_tpu_tgina_interrupt	割り込み要因 TGI*n*A（TPU*n*.TGRA のインプットキャプチャ / 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_tpu_tginb_interrupt	割り込み要因 TGI*n*B（TPU*n*.TGRB のインプットキャプチャ / 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_tpu_tginc_interrupt	割り込み要因 TGI*n*C（TPU*n*.TGRC のインプットキャプチャ / 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_tpu_tgind_interrupt	割り込み要因 TGI*n*D（TPU*n*.TGRD のインプットキャプチャ / 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_tpu_tcinv_interrupt	割り込み要因 TCI*n*V（TPU*n*.TCNT 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_tpu_tcinu_interrupt	割り込み要因 TCI*n*U（TPU*n*.TCNT のアンダフロー）割り込みの発生に伴う処理を行います。
R_TPUn_Start	TPU*n* の TCNT カウンタのカウント処理を開始します。
R_TPUn_Stop	TPU*n* の TCNT カウンタのカウント処理を停止します。

R_TPU_Create

16 ビットタイマパルスユニットを制御するうえで必要となる初期化処理を行います。

[指定形式]

```
void    R_TPU_Create ( void );
```

[引数]

なし

[戻り値]

なし

R_TPU_Create_UserInit

16 ビットタイマパルスユニットに関するユーザ独自の初期化処理を行います。

備考 本 API 関数は，R_TPU_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_TPU_Create_UserInit ( void );
```

[引数]

なし

[戻り値]

なし

r_tpu_tgi*n*a_interrupt

割り込み要因 TGI*n*A（TPU*n*.TGRA のインプットキャプチャ / コンペアマッチ）割り込みの発生に伴う処理を行います。

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

[指定形式]

```
void    r_tpu_tgina_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_tpu_tgi*n*b_interrupt

割り込み要因 TGI*n*B（TPU*n*.TGRB のインプットキャプチャ / コンペアマッチ）割り込みの発生に伴う処理を行います。

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

[指定形式]

```
void    r_tpu_tgi*n*b_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_tpu_tgi*n*c_interrupt

割り込み要因 TGI*n*C（TPU*n*.TGRC のインプットキャプチャ / コンペアマッチ）割り込みの発生に伴う処理を行います。

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

[指定形式]

```
void    r_tpu_tgincinterrupt ( 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_tpu_tgi*n*d_interrupt

割り込み要因 TGI*n*D（TPU*n*.TGRD のインプットキャプチャ / コンペアマッチ）割り込みの発生に伴う処理を行います。

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

[指定形式]

```
void    r_tpu_tgind_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_tpu_tci*n*v_interrupt

割り込み要因 TCI*n*V（TPU*n*.TCNT のオーバフロー）割り込みの発生に伴う処理を行います。

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

[指定形式]

```
void    r_tpu_tci*n*v_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_tpu_tci*n*u_interrupt

割り込み要因 TCI*n*U（TPU*n*.TCNT のアンダフロー）割り込みの発生に伴う処理を行います。

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

[指定形式]

```
void    r_tpu_tci*n*u_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_TPU*n*_Start

TPU*n* の TCNT カウンタのカウント処理を開始します。

[指定形式]

```
void    R_TPUn_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_TPU*n*_Stop

TPU*n* の TCNT カウンタのカウント処理を停止します。

[指定形式]

```
void    R_TPUn_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

3.2.12 プログラマブルパルスジェネレータ

以下に，コード生成がプログラマブルパルスジェネレータ用として出力する API 関数の一覧を示します。

表 3.12 プログラマブルパルスジェネレータ用 API 関数

API 関数名	機能概要
R_PPG_Create	プログラマブルパルスジェネレータを制御するうえで必要となる初期化処理を行います。
R_PPG_Create_UserInit	プログラマブルパルスジェネレータに関するユーザ独自の初期化処理を行います。

R_PPG_Create

プログラマブルパルスジェネレータを制御するうえで必要となる初期化処理を行います。

[指定形式]

```
void    R_PPG_Create ( void );
```

[引数]

なし

[戻り値]

なし

R_PPG_Create_UserInit

プログラマブルパルスジェネレータに関するユーザ独自の初期化処理を行います。

備考 本 API 関数は，R_PPG_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_PPG_Create_UserInit ( void );
```

[引数]

なし

[戻り値]

なし

3.2.13 コンペアマッチタイマ

以下に，コード生成がコンペアマッチタイマ用として出力する API 関数の一覧を示します。

表 3.13 コンペアマッチタイマ用 API 関数

API 関数名	機能概要
R_CMTn_Create	コンペアマッチ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_CMTn_Create_UserInit	コンペアマッチタイマに関するユーザ独自の初期化処理を行います。
r_cmt_cmin_interrupt	割り込み要因 CMI*n*（CMCNT*n* カウンタと CMCOR*n* レジスタのコンペアマッチ）の発生に伴う処理を行います。
R_CMTn_Start	CMCNT*n* カウンタのカウント処理を開始します。
R_CMTn_Stop	CMCNT*n* カウンタのカウント処理を停止します。

R_CMT*n*_Create

コンペアマッチ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_CMTn_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_CMT*n*_Create_UserInit

コンペアマッチ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_CMTn_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_CMTn_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_cmt_cmi*n*_interrupt

割り込み要因 CMI*n*（CMCNT*n* カウンタと CMCOR*n* レジスタのコンペアマッチ）の発生に伴う処理を行います。

[指定形式]

```
void    r_cmt_cmin_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_CMT*n*_Start

CMCNT*n* カウンタのカウント処理を開始します。

[指定形式]

```
void    R_CMTn_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_CMT*n*_Stop

CMCNT*n* カウンタのカウント処理を停止します。

[指定形式]

```
void    R_CMTn_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

3.2.14 コンペアマッチタイマ W

以下に，コード生成がコンペアマッチタイマ W 用として出力する API 関数の一覧を示します。

表 3.14 コンペアマッチタイマ W 用 API 関数

API 関数名	機能概要
R_CMTWm_Create	コンペアマッチタイマ W を制御するうえで必要となる初期化処理を行います。
R_CMTWm_Create_UserInit	コンペアマッチタイマ W に関するユーザ独自の初期化処理を行います。
r_cmtw_cmwim_interrupt	割り込み要因 CMWI（コンペアマッチによる割り込み）の発生に伴う処理を行います。
r_cmtw_icnim_interrupt	割り込み要因 IC*n*I*m*（インプットキャプチャ *n* による割り込み）の発生に伴う処理を行います。
r_cmtw_ocnim_interrupt	割り込み要因 OC*n*I*m*（アウトプットコンペア *n* による割り込み）の発生に伴う処理を行います。
R_CMTWm_Start	CMWCNT*m* カウンタのカウント処理を開始します。
R_CMTWm_Stop	CMWCNT*m* カウンタのカウント処理を停止します。

R_CMTW*m*_Create

コンペアマッチタイマ W 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_CMTWm_Create ( void );
```

備考　　*m* は，ユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_CMTW*m*_Create_UserInit

コンペアマッチタイマ W 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_CMTWm_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_CMTWm_Create_UserInit ( void );
```

備考 *m* は，ユニット番号を意味します。

[引数]

なし

[戻り値]

なし

r_cmtw_cmwi*m*_interrupt

割り込み要因 CMWI（コンペアマッチによる割り込み）の発生に伴う処理を行います。

[指定形式]

```
void    r_cmtw_cmwim_interrupt ( void );
```

備考　　*m* は，ユニット番号を意味します。

[引数]

なし

[戻り値]

なし

r_cmtw_ic*nim*_interrupt

割り込み要因 IC*nlm*（インプットキャプチャ *n* 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_cmtw_ic*nim*_interrupt ( void );
```

備考　　*n* は割り込み番号，*m* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

r_cmtw_oc*nim*_interrupt

割り込み要因 OC*nlm*（アウトプットコンペア *n* 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_cmtw_oc*nim*_interrupt ( void );
```

備考　　*n* は割り込み番号，*m* はユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_CMTW*m*_Start

CMWCNT*m* カウンタのカウント処理を開始します。

[指定形式]

```
void    R_CMTWn_Start ( void );
```

備考　　*m* は，ユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_CMTW*m*_Stop

CMWCNT*m* カウンタのカウント処理を停止します。

[指定形式]

```
void    R_CMTWm_Stop ( void );
```

備考　　*m* は，ユニット番号を意味します。

[引数]

なし

[戻り値]

なし

3.2.15 ウォッチドッグタイマ

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

表 3.15 ウォッチドッグタイマ用 API 関数

API 関数名	機能概要
R_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_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_wdtn_undff_refef_interrupt	WDT オーバフロー／リフレッシュエラーの発生に伴う処理を行います。
R_WDTn_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。

R_WDT*n*_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_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_WDT*n*_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_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_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_wdt*n*_undff_refef_interrupt

WDT オーバフロー／リフレッシュエラーの発生に伴う処理を行います。

[指定形式]

```
void    r_wdtn_undff_refef_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_WDT*n*_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_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

3.2.16 独立ウォッチドッグタイマ

以下に，コード生成が独立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 3.16 独立ウォッチドッグタイマ用 API 関数

API 関数名	機能概要
R_IWDT_Create	独立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_IWDT_Create_UserInit	独立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_iwdt_undff_refef_interrupt	IWDT オーバフロー／リフレッシュエラーの発生に伴う処理を行います。
R_IWDT_Restart	独立ウォッチドッグタイマのカウンタをクリアしたのち，カウント処理を再開します。

R_IWDT_Create

独立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_IWDT_Create ( void );
```

[引数]

なし

[戻り値]

なし

R_IWDT_Create_UserInit

独立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_IWDT_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_IWDT_Create_UserInit ( void );
```

[引数]

なし

[戻り値]

なし

r_iwdt_undff_refef_interrupt

IWDT オーバフロー／リフレッシュエラーの発生に伴う処理を行います。

[指定形式]

```
void    r_iwdt_undff_refef_interrupt ( void );
```

[引数]

なし

[戻り値]

なし

R_IWDT_Restart

独立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_IWDT_Restart ( void );
```

[引数]

なし

[戻り値]

なし

3.2.17 FIFO 内蔵シリアルコミュニケーションインタフェース

以下に，コード生成が FIFO 内蔵シリアルコミュニケーション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 3.17 FIFO 内蔵シリアルコミュニケーションインタフェース用 API 関数

API 関数名	機能概要
R_SCIFAn_Create	シリアル・アレイ・ユニットを制御するうえで必要となる初期化処理を行います。
R_SCIFAn_Create_UserInit	シリアル・アレイ・ユニットに関するユーザ独自の初期化処理を行います。
r_scifan_txifn_interrupt	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_scifan_rxifn_interrupt	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_scifan_brifn_interrupt	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。 また、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_scifan_drifn_interrupt	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。 また、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_scifan_callback_transmitend	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_scifan_callback_receiveend	受信 FIFO データフル割り込みの発生に伴う処理を行います。
r_scifan_callback_error	エラー割り込みの発生に伴う処理を行います。
R_SCIFAn_Start	SCIFA 通信を待機状態にします。
R_SCIFAn_Stop	SCIFA 通信を終了します。
R_SCIFAn_Serial_Send	調歩同期式モードで，送信を開始します。
R_SCIFAn_Serial_Receive	調歩同期式モードで，受信を開始します。
R_SCIFAn_Serial_Send_Receive	クロック同期式モードで，送受信を開始します。

R_SCIFA*n*_Create

FIFO 内蔵シリアルコミュニケーション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_SCIFAn_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_SCIFA*n*_Create_UserInit

FIFO 内蔵シリアルコミュニケーション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_SCIFAn_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_SCIFAn_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_scifa*n*_txif*n*_interrupt

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。

備考　本 API 関数は，送信 FIFO データエンプティ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_scifan_txifn_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_scifa*n*_rxif*n*_interrupt

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。

備考　本 API 関数は，受信 FIFO データフル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_scifan_rxifn_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_scifa*n*_brif*n*_interrupt

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。また、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。

備考　本 API 関数は，ブレーク（BRK) ／オーバラン（ORER）／フレーミングエラー／パリティエラー（ER) 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_scifan_brifn_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_scifa*n*_drif*n*_interrupt

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。
また、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。

備考　本 API 関数は，受信データレディ／トランスミットエンド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_scifan_drifn_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_scifa*n*_callback_transmitend

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。

備考 本 API 関数は，トランスミットエンド割り込みに対応した割り込み処理 r_scifan_teifn_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_scifan_callback_transmitend ( 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_scifa*n*_callback_receiveend

受信 FIFO データフル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 FIFO データフル割り込みに対応した割り込み処理 r_scifan_rxifn_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_scifan_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_scifa*n*_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 r_scifan_erifn_interrupt または r_scifan_brifn_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_scifan_callback_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_SCIFA*n*_Start

SCIFA 通信を待機状態にします。

[指定形式]

```
void    R_SCIFAn_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_SCIFA*n*_Stop

SCIFA 通信を終了します。

[指定形式]

```
void    R_SCIFAn_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_SCIFA*n*_Serial_Send

調歩同期式モードで，送信を開始します。

備考 1. 本 API 関数では，引数 *txbuf* 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 *txnum* 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_SCIFAn_Start を呼び出す必要があります。

[指定形式]

```
MD_STATUS   R_SCIFAn_Serial_Send ( const uint8_t * txbuf, uint16_t txnum );
```

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	`const uint8_t * txbuf;`	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 txnum;`	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_SCIFA*n*_Serial_Receive

調歩同期式モードで，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 *rxnum* で指定された回数だけ繰り返し行い，引数 *rxbuf* 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_SCIFAn_Start を呼び出すことにより開始されます。

[指定形式]

```
MD_STATUS   R_SCIFAn_Serial_Receive ( uint8_t * rxbuf, uint16_t rxnum );
```

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 * rxbuf;`	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 rxnum;`	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_SCIFA*n*_Serial_Send_Receive

クロック同期式モードで，送受信を開始します。

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

備考 2. 本 API 関数では，1 バイト単位の受信を引数 *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. 本 API 関数の呼び出し後，R_SCIFAn_Start を呼び出す必要があります。

[指定形式]

```
MD_STATUS R_SCIFAn_Serial_Send_Receive(const uint8_t * tx_buf, uint16_t tx_num,
uint8_t * 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
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_buf;	受信する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

[戻り値]

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

3.2.18 I2C バスインタフェース

以下に，コード生成ツールが I2C バス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 3.18 I2C バスインタフェース用 API 関数

API 関数名	機能概要
R_RIICn_Create	I2C バス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_RIICn_Create_UserInit	I2C バス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_riicn_error_interrupt	通信エラー／イベント発生割り込み（EEI）の発生に伴う処理を行います。
r_riicn_receive_interrupt	受信データフル割り込み（RXI）の発生に伴う処理を行います。
r_riicn_transmit_interrupt	送信データエンプティ割り込み（TXI）の発生に伴う処理を行います。
r_riicn_transmitend_interrupt	送信終了割り込み（TEI）の発生に伴う処理を行います。
R_RIICn_Start	RIIC 通信を開始します。
R_RIICn_Stop	RIIC 通信を終了します。
R_RIICn_Master_Send	RIIC マスタ送信を開始します。 また、送信完了時に Stop コンディションを生成します。
R_RIICn_Master_Send_Without_Stop	RIIC マスタ送信を開始します。 送信完了時に Stop コンディションを生成しません。
R_RIICn_Master_Receive	RIIC マスタ受信を開始します。
R_RIICn_Slave_Send	RIIC スレーブ送信を開始します。
R_RIICn_Slave_Receive	RIIC スレーブ受信を開始します。
R_RIICn_StartCondition	スタートコンディションを発行し，通信エラー／イベント発生割り込み（EEI）を発生させます
R_RIICn_StopCondition	ストップコンディションを発行し，通信エラー／イベント発生割り込み（EEI）を発生させます。
r_riicn_callback_receiveerror	通信エラー／イベント発生割り込み（EEI）に対応した割り込み処理のうち，アビトレーションロストの検出，NACK の検出，タイムアウトの検出に特化した処理を行います。
r_riicn_callback_transmitend	通信エラー／イベント発生割り込み（EEI）に対応した割り込み処理のうち，R_RIICn_Master_Send の呼び出しに伴うスタートコンディションの検出に特化した処理を行います。
r_riicn_callback_receiveend	通信エラー／イベント発生割り込み（EEI）に対応した割り込み処理のうち，R_RIICn_Master_Receive の呼び出しに伴うスタートコンディションの検出に特化した処理を行います。

R_RIIC*n*_Create

I2C バス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_RIICn_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_RIIC*n*_Create_UserInit

I2C バス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_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_RIICn_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_riic*n*_error_interrupt

通信エラー／イベント発生割り込み（EEI）の発生に伴う処理を行います。

備考 本 API 関数は，I2C バスインタフェースが通信エラー／イベント発生（アービトレーションロスト，NACK，タイムアウト，スタートコンディション，ストップコンディション）を検出した場合に発生する通信エラー／イベント発生割り込み（EEI）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_riicn_error_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_riic*n*_receive_interrupt

受信データフル割り込み（RXI）の発生に伴う処理を行います。

備考　本 API 関数は，受信データフル割り込み（RXI）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_riicn_receive_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_riic*n*_transmit_interrupt

送信データエンプティ割り込み（TXI）の発生に伴う処理を行います。

備考　本 API 関数は，送信データエンプティ割り込み（TXI）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_riicn_transmit_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_riic*n*_transmitend_interrupt

送信終了割り込み（TEI）の発生に伴う処理を行います。

備考　本 API 関数は，送信終了割り込み（TEI）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_riicn_transmitend_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_RIIC*n*_Start

RIIC 通信を開始します。

[指定形式]

```
void    R_RIICn_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_RIIC*n*_Stop

RIIC 通信を終了します。

[指定形式]

```
void    R_RIICn_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_RIIC*n*_Master_Send

RIIC マスタ送信を開始します。
また、送信完了時に Stop コンディションを生成します。

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 RIIC 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 RIIC 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，RIIC マスタ送信の開始処理として，内部的に R_RIICn_StartCondition の呼び出しを行っています。

備考 3. RIIC マスタ送信を行う際には，本 API 関数の呼び出し以前に R_RIICn_Start を呼び出す必要があります。

[指定形式]

```
MD_STATUS R_RIICn_Master_Send(uint16_t adr, const uint8_t * 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	`uint16_t adr;`	スレーブ・アドレス
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_ERROR1	バス・ビジー
MD_ERROR2	引数 *adr* の指定が不正

R_RIIC*n*_Master_Send_Without_Stop

RIIC マスタ送信を開始します。
送信完了時に Stop コンディションを生成しません。

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 RIIC 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 RIIC 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，RIIC マスタ送信の開始処理として，内部的に R_RIICn_StartCondition の呼び出しを行っています。

備考 3. RIIC マスタ送信を行う際には，本 API 関数の呼び出し以前に R_RIICn_Start を呼び出す必要があります。

[指定形式]

```
MD_STATUS R_RIICn_Master_Send_Without_Stop(uint16_t adr, const uint8_t * 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	`uint16_t adr;`	スレーブ・アドレス
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_ERROR1	バス・ビジー
MD_ERROR2	引数 *adr* の指定が不正

R_RIIC*n*_Master_Receive

RIIC マスタ受信を開始します。

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 RIIC マスタ送信したのち，1 バイト単位の RIIC 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，RIIC マスタ受信の開始処理として，内部的に R_RIICn_StartCondition の呼び出しを行っています。

備考 3. RIIC マスタ受信を行う際には，本 API 関数の呼び出し以前に R_RIICn_Start を呼び出す必要があります。

[指定形式]

```
MD_STATUS R_RIICn_Master_Receive(uint16_t adr, 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
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_RIIC*n*_Slave_Send

RIIC スレーブ送信を開始します。

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 RIIC 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. RIIC 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_RIICn_Start を呼び出す必要があります。

[指定形式]

```
void    R_RIICn_Slave_Send ( const uint8_t * 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	`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]

なし

R_RIIC*n*_Slave_Receive

RIIC スレーブ受信を開始します。

備考 1. 本 API 関数では，1 バイト単位の RIIC 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. RIIC 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_RIICn_Start を呼び出す必要があります。

[指定形式]

```
void    R_RIICn_Slave_Receive ( uint8_t * 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 * 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]

なし

R_RIIC*n*_StartCondition

スタートコンディションを発行し，通信エラー／イベント発生割り込み（EEI）を発生させます。

備考 1. 本 API 関数は，R_RIICn_Master_Send，および R_RIICn_Master_Receive の内部関数として呼び出されます。

備考 2. 本 API 関数の呼び出しに伴い，r_riicn_error_interrupt が呼び出されます。

[指定形式]

```
void    R_RIICn_StartCondition ( 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_RIIC*n*_StopCondition

ストップコンディションを発行し，通信エラー／イベント発生割り込み（EEI）を発生させます。

備考　本 API 関数の呼び出しに伴い，r_riicn_error_interrupt が呼び出されます。

[指定形式]

```
void    R_RIICn_StopCondition ( 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_riic*n*_callback_receiveerror

通信エラー／イベント発生割り込み（EEI）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の検出，NACK の検出，タイムアウトの検出に特化した処理を行います。

備考 本 API 関数は，r_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_riicn_callback_receiveerror ( MD_STATUS status );
```

備考 *n* は，チャネル番号を意味します。

[引数]

I/O	引数	説明
I	MD_STATUS *status*;	通信エラー／イベント発生割り込みの発生要因 MD_ERROR1 ： アビトレーションロストの検出 MD_ERROR2 ： タイムアウトの検出 MD_ERROR3 ： NACK の検出

[戻り値]

なし

r_riic*n*_callback_transmitend

通信エラー／イベント発生割り込み（EEI）に対応した割り込み処理のうち，R_RIICn_Master_Send の呼び出しに伴うスタートコンディションの検出に特化した処理を行います。

備考 本 API 関数は，r_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_riicn_callback_transmitend ( 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_riic*n*_callback_receiveend

通信エラー／イベント発生割り込み（EEI）に対応した割り込み処理のうち，R_RIICn_Master_Receive の呼び出しに伴うスタートコンディションの検出に特化した処理を行います。

備考 本 API 関数は，r_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_riicn_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

3.2.19 シリアルペリフェラルインタフェース

以下に，コード生成ツールがシリアルペリフェラル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 3.19 シリアルペリフェラルインタフェース用 API 関数

API 関数名	機能概要
R_RSPIn_Create	シリアルペリフェラル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_RSPIn_Create_UserInit	シリアルペリフェラル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_rspin_receive_interrupt	受信バッファフル割り込みの発生に伴う処理を行います。
r_rspin_transmit_interrupt	送信バッファエンプティ割り込みの発生に伴う処理を行います。
r_rspin_error_interrupt	RSPI エラー割り込みの発生に伴う処理を行います。
r_rspin_idle_interrupt	RSPI アイドル割り込みの発生に伴う処理を行います。
R_RSPIn_Start	RSPI 通信を開始します。
R_RSPIn_Stop	RSPI 通信を終了します。
R_RSPIn_Send	RSPI 送信を開始します。
R_RSPIn_Send_Receive	RSPI 送受信を開始します。
r_rspin_callback_receiveend	受信バッファフル割り込みの発生に伴う処理を行います。
r_rspin_callback_error	RSPI エラー割り込みの発生に伴う処理を行います。
r_rspin_callback_transmitend	RSPI アイドル割り込みの発生に伴う処理を行います。

R_RSPI*n*_Create

シリアルペリフェラル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_RSPIn_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_RSPI*n*_Create_UserInit

シリアルペリフェラル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_RSPIn_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_RSPIn_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_rspi*n*_receive_interrupt

受信バッファフル割り込みの発生に伴う処理を行います。

備考　本 API 関数は，受信バッファ・フル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_rspin_receive_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_rspi*n*_transmit_interrupt

送信バッファエンプティ割り込みの発生に伴う処理を行います。

備考　本 API 関数は，送信バッファエンプティ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_rspin_transmit_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_rspi*n*_error_interrupt

RSPI 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，RSPI エラ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_rspin_error_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_rspi*n*_idle_interrupt

RSPI アイドル割り込みの発生に伴う処理を行います。

備考　本 API 関数は，RSPI アイドル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_rspin_idle_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_RSPI*n*_Start

RSPI 通信を開始します。

[指定形式]

```
void    R_RSPIn_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_RSPI*n*_Stop

RSPI 通信を終了します。

[指定形式]

```
void    R_RSPIn_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_RSPI*n*_Send

RSPI 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 RSPI 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. RSPI 送信を行う際には，本 API 関数の呼び出し以前に R_RSPIn_Start を呼び出す必要があります。

[指定形式]

```
MD_STATUS   R_RSPIn_Send (const uint32_t * 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	`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
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_RSPI*n*_Send_Receive

RSPI 送受信を開始します。

備考 1. 本 API 関数では，RSPI 送信処理として，引数 *tx_buf* で指定されたバッファから 1 バイト単位の RSPI 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，RSPI 受信処理として，1 バイト単位の RSPI 受信を引数 *tx_num* で指定された回数だけ繰り返し行い，引数 *rx_buf* で指定されたバッファに格納します。

備考 3. RSPI 送受信を行う際には，本 API 関数の呼び出し以前に R_RSPIn_Start を呼び出す必要があります。

[指定形式]

```
MD_STATUS   R_RSPIn_Send_Receive (const uint32_t * tx_buf, uint16_t tx_num, uint32_t *
rx_buf);
```

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	`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
I	`uint16_t tx_num;`	送受信するデータの総数
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

[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_rspi*n*_callback_receiveend

受信バッファフル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_rspin_receive_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_rspin_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_rspi*n*_callback_error

RSPI 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_rspin_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_rspin_callback_error ( uint8_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
O	uint8_t *err_type*;	RSPI エラー割り込みの発生要因（*x* 部は不定） *xxxx*00*x*1B　：オーバランエラーの検出 *xxxx*01*x*0B　：モードフォルトエラーの検出 *xxxx*10*x*0B　：パリティエラーの検出

[戻り値]

なし

r_rspi*n*_callback_transmitend

RSPI アイドル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_rspin_idle_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_rspin_callback_transmitend ( 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

3.2.20 SPI マルチ I/O バスコントローラ

以下に，コード生成が SPI マルチ I/O バス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 3.20 SPI マルチ I/O バスコントローラ用 API 関数

API 関数名	機能概要
R_SPIBSC_Create	SPI マルチ I/O バス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_SPIBSC_Create_UserInit	SPI マルチ I/O バス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_SPIBSC_EAVUpperAddressChange	アドレス 32 ビット出力時の上位 8 ビットのアドレスを変更します。
R_SPIBSC_SPIRead	データ受信を行います。
R_SPIBSC_SPIWrite	データ送信を行います。
R_SPIBSC_SPIRead_Write	データ送受信を行います。

R_SPIBSC_Create

SPI マルチ I/O バス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_SPIBSC_Create ( void );
```

[引数]

なし

[戻り値]

なし

R_SPIBSC_Create_UserInit

SPI マルチ I/O バス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_SPIBSC_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_SPIBSC_Create_UserInit ( void );
```

[引数]

なし

[戻り値]

なし

R_SPIBSC_EAVUpperAddressChange

アドレス 32 ビット出力時の上位 8 ビットのアドレスを変更します。

[指定形式]

```
void R_SPIBSC_EAVUpperAddressChange(uint8_t const up_adr);
```

[引数]

I/O	引数	説明
I	uint8_t const up_adr;	アドレス 32 ビット出力時の上位 8 ビット

[戻り値]

なし

R_SPIBSC_SPIRead

SPI 動作モードによる，データの受信を行います。

[指定形式]

```
MD_STATUS R_SPIBSC_SPIRead(st_spibsc_spiparam const param, uint32_t rx_data);
```

[引数]

I/O	引数	説明
I	st_spibsc_spiparam const param;	SPI 動作モードのパラメータ（リード / ライト）
O	uint32_t rx_data;	受信データ

[戻り値]

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

R_SPIBSC_SPIWrite

SPI 動作モードによる，データの送信を行います。

[指定形式]

```
MD_STATUS R_SPIBSC_SPIWrite(st_spibsc_spiparam const param, uint32_t const tx_data);
```

[引数]

I/O	引数	説明
I	st_spibsc_spiparam const param;	SPI 動作モードのパラメータ（リード / ライト）
I	uint32_t const tx_data;	送信データ

[戻り値]

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

R_SPIBSC_SPIRead_Write

SPI 動作モードによる，データの送受信を行います。

[指定形式]

```
MD_STATUS R_SPIBSC_SPIRead_Write(st_spibsc_spiparam const param, uint32_t rx_data,
uint32_t const tx_data);
```

[引数]

I/O	引数	説明
I	st_spibsc_spiparam const param;	SPI 動作モードのパラメータ（リード / ライト）
O	uint32_t const rx_data;	受信データ
I	uint32_t const tx_data;	送信データ

[戻り値]

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

3.2.21 CRC 演算器

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

表 3.21 CRC 演算器用 API 関数

API 関数名	機能概要
R_CRC_SetCRC8_2F	8 ビット CRC 演算（CRC8-0x2F）を実施するうえで必要となる CRC 演算器の初期化処理を行います。
R_CRC_SetCRC8_SAE	8 ビット CRC 演算（CRC8-SAE-J1850）を実施するうえで必要となる CRC 演算器の初期化処理を行います。
R_CRC_SetCRC16_CCITT	16 ビット CRC 演算（CRC16-CCITT）を実施するうえで必要となる CRC 演算器の初期化処理を行います。
R_CRC_SetCRC32_ETHER	32 ビット CRC 演算（CRC32_ETHERNET）を実施するうえで必要となる CRC 演算器の初期化処理を行います。
R_CRC_Input_Data	CRC 演算を行うデータの初期値を設定します。
R_CRC_Get_Result	演算結果を獲得します。

R_CRC_SetCRC8_2F

8 ビット CRC 演算（CRC8-0x2F）を実施するうえで必要となる CRC 演算器の初期化処理を行います。

[指定形式]

```
void    R_CRC_SetCRC8_2F ( void );
```

[引数]

なし

[戻り値]

なし

R_CRC_SetCRC8_SAE

8 ビット CRC 演算（CRC8-SAE-J1850）を実施するうえで必要となる CRC 演算器の初期化処理を行います。

[指定形式]

```
void    R_CRC_SetCRC8_SAE ( void );
```

[引数]

なし

[戻り値]

なし

R_CRC_SetCRC16_CCITT

16 ビット CRC 演算（CRC16-CCITT）を実施するうえで必要となる CRC 演算器の初期化処理を行います。

[指定形式]

```
void    R_CRC_SetCRC16_CCITT ( void );
```

[引数]

なし

[戻り値]

なし

R_CRC_SetCRC32_ETHER

32 ビット CRC 演算（CRC32_ETHERNET）を実施するうえで必要となる CRC 演算器の初期化処理を行います。

[指定形式]

```
void    R_CRC_SetCRC32_ETHER ( void );
```

[引数]

なし

[戻り値]

なし

R_CRC_Input_Data

CRC 演算を行うデータの初期値を設定します。

[指定形式]

```
void    R_CRC_Input_Data ( uint32_t data );
```

[引数]

I/O	引数	説明
I	uint32_t *data*;	CRC 演算を行うデータの初期値

[戻り値]

なし

R_CRC_Get_Result

演算結果を獲得します。

[指定形式]

```
void    R_CRC_Get_Result ( uint32_t * const result );
```

[引数]

I/O	引数	説明
O	uint32_t * const *result*;	獲得した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 ΔΣ インタフェース

以下に，コード生成が ΔΣ 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 3.22 ΔΣ インタフェース用 API 関数

API 関数名	機能概要
R_DSMIF_Create	ΔΣ 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_DSMIF_Create_UserInit	ΔΣ 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_DSMIF_UVW_Start	ΔΣ インタフェース（ユニット 0（U，V，W））のフィルタ動作を開始します。
R_DSMIF_UVW_Stop	ΔΣ インタフェース（ユニット 0（U，V，W））のフィルタ動作を停止します。
R_DSMIF_X_Start	ΔΣ インタフェース（ユニット 1（X））のフィルタ動作を開始します。
R_DSMIF_X_Stop	ΔΣ インタフェース（ユニット 1（X））のフィルタ動作を停止します。
r_dsmif_uvw_overcurrent_interrupt	ΔΣ インタフェース（ユニット 0（U，V，W））の過電流異常検出エラー割り込みの発生に伴う処理を行います。
r_dsmif_uvw_totalcurrent_interrupt	ΔΣ インタフェース（ユニット 0（U，V，W））の合計電流異常検出エラー割り込みの発生に伴う処理を行います。
r_dsmif_uvw_shortcircuit_interrupt	ΔΣ インタフェース（ユニット 0（U，V，W））の短絡異常検出エラー割り込みの発生に伴う処理を行います。
r_dsmif_x_overcurrent_interrupt	ΔΣ インタフェース（ユニット 1（X））の過電流異常検出エラー割り込みの発生に伴う処理を行います。
r_dsmif_x_shortcircuit_interrupt	ΔΣ インタフェース（ユニット 1（X））の短絡異常検出エラー割り込みの発生に伴う処理を行います。

R_DSMIF_Create

ΔΣ 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_DSMIF_Create ( void );
```

[引数]

なし

[戻り値]

なし

R_DSMIF_Create_UserInit

ΔΣ 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_DSMIF_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_DSMIF_Create_UserInit ( void );
```

[引数]

なし

[戻り値]

なし

R_DSMIF_UVW_Start

ΔΣ インタフェース（ユニット 0（U，V，W））のフィルタ動作を開始します。

[指定形式]

```
void    R_DSMIF_UVW_Start ( void );
```

[引数]

なし

[戻り値]

なし

R_DSMIF_UVW_Stop

ΔΣ インタフェース（ユニット 0（U，V，W））のフィルタ動作を停止します。

[指定形式]

```
void    R_DSMIF_UVW_Stop ( void );
```

[引数]

なし

[戻り値]

なし

R_DSMIF_X_Start

ΔΣ インタフェース（ユニット 1（X））のフィルタ動作を開始します。

[指定形式]

```
void    R_DSMIF_X_Start ( void );
```

[引数]

なし

[戻り値]

なし

R_DSMIF_X_Stop

ΔΣ インタフェース（ユニット 1（X））のフィルタ動作を停止します。

[指定形式]

```
void    R_DSMIF_X_Stop ( void );
```

[引数]

なし

[戻り値]

なし

r_dsmif_uvw_overcurrent_interrupt

ΔΣ インタフェース（ユニット 0（U，V，W））の過電流異常検出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_dsmif_uvw_overcurrent_interrupt ( void );
```

[引数]

なし

[戻り値]

なし

r_dsmif_uvw_totalcurrent_interrupt

ΔΣ インタフェース（ユニット 0（U，V，W））の合計電流異常検出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_dsmif_uvw_totalcurrent_interrupt ( void );
```

[引数]

なし

[戻り値]

なし

r_dsmif_uvw_shortcircuit_interrupt

ΔΣ インタフェース（ユニット 0（U，V，W））の短絡異常検出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_dsmif_uvw_shortcircuit_interrupt( void );
```

[引数]

なし

[戻り値]

なし

r_dsmif_x_overcurrent_interrupt

ΔΣ インタフェース（ユニット 1（X））の過電流異常検出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_dsmif_x_overcurrent_interrupt ( void );
```

[引数]

なし

[戻り値]

なし

r_dsmif_x_shortcircuit_interrupt

ΔΣ インタフェース（ユニット 1（X））の短絡異常検出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_dsmif_x_shortcircuit_interrupt ( void );
```

[引数]

なし

[戻り値]

なし

3.2.23 メモリプロテクションユニット

以下に，コード生成がメモリプロテクションユニット用に出力する API 関数の一覧を示します。

表 3.23 メモリプロテクションユニット用 API 関数

API 関数名	機能概要
R_MPU_Create	メモリプロテクションユニットを制御するうえで必要となる初期化処理を行います。

R_MPU_Create

メモリプロテクションユニットを制御するうえで必要となる初期化処理を行います。

[指定形式]

```
void    R_MPU_Create ( void );
```

[引数]

なし

[戻り値]

なし

3.2.24 エラーコントロールモジュール

以下に，コード生成がエラーコントロールモジュール用として出力する API 関数の一覧を示します。

表 3.24 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_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_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_ecm_nmi_interrupt	ECM ノンマスカブル割り込みの発生に伴う処理を行います。
r_ecm_errd_interrupt	エラー検出（ERRD) の発生に伴う処理を行います。
r_ecm_compareerror_interrupt	コンペアエラー発生に伴う処理を行います。
R_ECM_Pseudo_WDT0_Error_Start	エラー要因 1　WDT オーバフロー / リフレッシュエラー（Cortex-R4F）の疑似エラーを発行します。
R_ECM_Pseudo_WDT0_Error_Stop	エラー要因 1　WDT オーバフロー / リフレッシュエラー（Cortex-R4F）の疑似エラーを発行しません。
R_ECM_Pseudo_IWDTa_Error_Start	エラー要因 3　IWDTa オーバフロー / リフレッシュエラーの疑似エラーを発行します。
R_ECM_Pseudo_IWDTa_Error_Stop	エラー要因 3　IWDTa オーバフロー / リフレッシュエラーの疑似エラーを発行しません。
R_ECM_Pseudo_CGC_Error_Start	エラー要因 20　メインクロック発振停止検出の疑似エラーを発行します。
R_ECM_Pseudo_CGC_Error_Stop	エラー要因 20　メインクロック発振停止検出の疑似エラーを発行しません。
R_ECM_Pseudo_ADC_Unit0_Error_Start	エラー要因 24　12 ビット A/D コンバータ・ユニット 0 オーバライト割り込みの疑似エラーを発行します。
R_ECM_Pseudo_ADC_Unit0_Error_Stop	エラー要因 24　12 ビット A/D コンバータ・ユニット 0 オーバライト割り込みの疑似エラーを発行しません。
R_ECM_Pseudo_ADC_Unit1_Error_Start	エラー要因 25　12 ビット A/D コンバータ・ユニット 1 オーバライト割り込みの疑似エラーを発行します。
R_ECM_Pseudo_ADC_Unit1_Error_Stop	エラー要因 25　12 ビット A/D コンバータ・ユニット 1 オーバライト割り込みの疑似エラーを発行しません。
R_ECM_Pseudo_DSMIF_UVWovercurrent_Error_Start	エラー要因 26　ΔΣ インタフェース UVW 過電流異常検出エラーの疑似エラーを発行します。
R_ECM_Pseudo_DSMIF_UVWovercurrent_Error_Stop	エラー要因 26　ΔΣ インタフェース UVW 過電流異常検出エラーの疑似エラーを発行しません。
R_ECM_Pseudo_DSMIF_UVWtotalcurrent_Error_Start	エラー要因 27　ΔΣ インタフェース UVW 合計電流異常検出エラーの疑似エラーを発行します。
R_ECM_Pseudo_DSMIF_UVWtotalcurrent_Error_Stop	エラー要因 27　ΔΣ インタフェース UVW 合計電流異常検出エラーの疑似エラーを発行しません。
R_ECM_Pseudo_DSMIF_UVWshortcircuit_Error_Start	エラー要因 28　ΔΣ インタフェース UVW 短絡異常検出エラーの疑似エラーを発行します。
R_ECM_Pseudo_DSMIF_UVWshortcircuit_Error_Stop	エラー要因 28　ΔΣ インタフェース UVW 短絡異常検出エラーの疑似エラーを発行しません。

API 関数名	機能概要
R_ECM_Pseudo_DSMIF_Xovercurrent_Error_Start	エラー要因 29　ΔΣ インタフェース X 過電流異常検出エラーの疑似エラーを発行します。
R_ECM_Pseudo_DSMIF_Xovercurrent_Error_Stop	エラー要因 29　ΔΣ インタフェース X 過電流異常検出エラーの疑似エラーを発行しません。
R_ECM_Pseudo_DSMIF_Xshortcircuit_Error_Start	エラー要因 31　ΔΣ インタフェース X 短絡異常検出エラーの疑似エラーを発行します。
R_ECM_Pseudo_DSMIF_Xshortcircuit_Error_Stop	エラー要因 31　ΔΣ インタフェース X 短絡異常検出エラーの疑似エラーを発行しません。
R_ECM_Pseudo_DOC_Error_Start	エラー要因 32　データ演算回路 DOC 演算エラーの疑似エラーを発行します。
R_ECM_Pseudo_DOC_Error_Stop	エラー要因 32　データ演算回路 DOC 演算エラーの疑似エラーを発行しません。
R_ECM_Pseudo_BSC_Error_Start	エラー要因 34　バスステートコントローラ外部 WAIT 端子による長期アクセスウェイト検出の疑似エラーを発行します。
R_ECM_Pseudo_BSC_Error_Stop	エラー要因 34　バスステートコントローラ外部 WAIT 端子による長期アクセスウェイト検出の疑似エラーを発行しません。
R_ECM_Pseudo_Error35_Error_Start	エラー要因 35　拡張疑似エラー 35 の疑似エラーを発行します。
R_ECM_Pseudo_Error35_Error_Stop	エラー要因 35　拡張疑似エラー 35 の疑似エラーを発行しません。
R_ECM_Pseudo_Error36_Error_Start	エラー要因 36　拡張疑似エラー 36 の疑似エラーを発行します。
R_ECM_Pseudo_Error36_Error_Stop	エラー要因 36　拡張疑似エラー 36 の疑似エラーを発行しません。
R_ECM_Pseudo_Error37_Error_Start	エラー要因 37　拡張疑似エラー 37 の疑似エラーを発行します。
R_ECM_Pseudo_Error37_Error_Stop	エラー要因 37　拡張疑似エラー 37 の疑似エラーを発行しません。
R_ECM_Pseudo_Error38_Error_Start	エラー要因 38　拡張疑似エラー 38 の疑似エラーを発行します。
R_ECM_Pseudo_Error38_Error_Stop	エラー要因 38　拡張疑似エラー 38 の疑似エラーを発行しません。
R_ECM_Pseudo_Error39_Error_Start	エラー要因 39　拡張疑似エラー 39 の疑似エラーを発行します。
R_ECM_Pseudo_Error39_Error_Stop	エラー要因 39　拡張疑似エラー 39 の疑似エラーを発行しません。
R_ECM_Pseudo_Error40_Error_Start	エラー要因 40　拡張疑似エラー 40 の疑似エラーを発行します。
R_ECM_Pseudo_Error40_Error_Stop	エラー要因 40　拡張疑似エラー 40 の疑似エラーを発行しません。
R_ECM_Pseudo_Error41_Error_Start	エラー要因 41　拡張疑似エラー 41 の疑似エラーを発行します。
R_ECM_Pseudo_Error41_Error_Stop	エラー要因 41　拡張疑似エラー 41 の疑似エラーを発行しません。
R_ECM_Pseudo_ECM_CompareError_Error_Start	エラー要因 93　エラーコントロールモジュール コンペアエラーの疑似エラーを発行します。
R_ECM_Pseudo_ECM_CompareError_Error_Stop	エラー要因 93　エラーコントロールモジュール コンペアエラーの疑似エラーを発行しません。
R_ECM_Pseudo_ECM_DelayTimerOverflow_Error_Start	エラー要因 94　エラーコントロールモジュール ディレイタイマオーバーフローエラーの疑似エラーを発行します。
R_ECM_Pseudo_ECM_DelayTimerOverflow_Error_Stop	エラー要因 94　エラーコントロールモジュール ディレイタイマオーバーフローエラーの疑似エラーを発行しません。

R_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。

[指定形式]

```
void    R_ECM_Create ( void );
```

[引数]

なし

[戻り値]

なし

R_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_ECM_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_ECM_Create_UserInit ( void );
```

[引数]

なし

[戻り値]

なし

r_ecm_nmi_interrupt

ECM ノンマスカブル割り込みの発生に伴う処理を行います。

[指定形式]

```
void    r_ecm_nmi_interrupt ( void );
```

[引数]

なし

[戻り値]

なし

r_ecm_errd_interrupt

エラー検出（ERRD) の発生に伴う処理を行います。

[指定形式]

```
void    r_ecm_errd_interrupt ( void );
```

[引数]

なし

[戻り値]

なし

r_ecm_compareerror_interrupt

コンペアエラー発生に伴う処理を行います。

[指定形式]

```
void    r_ecm_compareerror_interrupt ( void );
```

[引数]

なし

[戻り値]

なし

R_ECM_Pseudo_WDT0_Error_Start

エラー要因 1　WDT オーバフロー / リフレッシュエラー（Cortex-R4F）の疑似エラーを発行します。

[指定形式]

```
void    R_ECM_Pseudo_WDT0_Error_Start ( void );
```

[引数]

なし

[戻り値]

なし

R_ECM_Pseudo_WDT0_Error_Stop

エラー要因 1　WDT オーバフロー / リフレッシュエラー（Cortex-R4F）の疑似エラーを発行しません。

[指定形式]

```
void    R_ECM_Pseudo_WDT0_Error_Stop ( void );
```

[引数]

なし

[戻り値]

なし

R_ECM_Pseudo_IWDTa_Error_Start

エラー要因 3　IWDTa オーバフロー / リフレッシュエラーの疑似エラーを発行します。

[指定形式]

```
void    R_ECM_Pseudo_IWDTa_Error_Start ( void );
```

[引数]

なし

[戻り値]

なし

R_ECM_Pseudo_IWDTa_Error_Stop

エラー要因 3　IWDTa オーバフロー / リフレッシュエラーの疑似エラーを発行しません。

[指定形式]

```
void    R_ECM_Pseudo_IWDTa_Error_Stop ( void );
```

[引数]

なし

[戻り値]

なし

R_ECM_Pseudo_CGC_Error_Start

エラー要因 20　メインクロック発振停止検出の疑似エラーを発行します。

[指定形式]

```
void    R_ECM_Pseudo_CGC_Error_Start ( void );
```

[引数]

なし

[戻り値]

なし

R_ECM_Pseudo_CGC_Error_Stop

エラー要因 20　メインクロック発振停止検出の疑似エラーを発行しません。

[指定形式]

```
void    R_ECM_Pseudo_CGC_Error_Stop ( void );
```

[引数]

なし

[戻り値]

なし

R_ECM_Pseudo_ADC_Unit0_Error_Start

エラー要因 24　12 ビット A/D コンバータ・ユニット 0 オーバライト割り込みの疑似エラーを発行します。

[指定形式]

```
void    R_ECM_Pseudo_ADC_Unit0_Error_Start ( void );
```

[引数]

なし

[戻り値]

なし

R_ECM_Pseudo_ADC_Unit0_Error_Stop

エラー要因 24　12 ビット A/D コンバータ・ユニット 0 オーバライト割り込みの疑似エラーを発行しません。

[指定形式]

```
void    R_ECM_Pseudo_ADC_Unit0_Error_Stop ( void );
```

[引数]

なし

[戻り値]

なし

R_ECM_Pseudo_ADC_Unit1_Error_Start

エラー要因 25　12 ビット A/D コンバータ・ユニット 1 オーバライト割り込みの疑似エラーを発行します。

[指定形式]

```
void    R_ECM_Pseudo_ADC_Unit1_Error_Start ( void );
```

[引数]

なし

[戻り値]

なし

R_ECM_Pseudo_ADC_Unit1_Error_Stop

エラー要因 25　12 ビット A/D コンバータ・ユニット 1 オーバライト割り込みの疑似エラーを発行しません。

[指定形式]

```
void    R_ECM_Pseudo_ADC_Unit1_Error_Stop ( void );
```

[引数]

なし

[戻り値]

なし

R_ECM_Pseudo_DSMIF_UVWovercurrent_Error_Start

エラー要因 26　ΔΣ インタフェース UVW 過電流異常検出エラーの疑似エラーを発行します。

[指定形式]

```
void    R_ECM_Pseudo_DSMIF_UVWovercurrent_Error_Start ( void );
```

[引数]

なし

[戻り値]

なし

R_ECM_Pseudo_DSMIF_UVWovercurrent_Error_Stop

エラー要因 26　ΔΣ インタフェース UVW 過電流異常検出エラーの疑似エラーを発行しません。

[指定形式]

```
void    R_ECM_Pseudo_DSMIF_UVWovercurrent_Error_Stop ( void );
```

[引数]

なし

[戻り値]

なし

R_ECM_Pseudo_DSMIF_UVWtotalcurrent_Error_Start

エラー要因 27　ΔΣ インタフェース UVW 合計電流異常検出エラーの疑似エラーを発行します。

[指定形式]

```
void    R_ECM_Pseudo_DSMIF_UVWtotalcurrent_Error_Start ( void );
```

[引数]

なし

[戻り値]

なし

R_ECM_Pseudo_DSMIF_UVWtotalcurrent_Error_Stop

エラー要因 27　ΔΣ インタフェース UVW 合計電流異常検出エラーの疑似エラーを発行しません。

[指定形式]

```
void    R_ECM_Pseudo_DSMIF_UVWtotalcurrent_Error_Stop ( void );
```

[引数]

なし

[戻り値]

なし

R_ECM_Pseudo_DSMIF_UVWshortcircuit_Error_Start

エラー要因 28　ΔΣ インタフェース UVW 短絡異常検出エラーの疑似エラーを発行します。

[指定形式]

```
void    R_ECM_Pseudo_DSMIF_UVWshortcircuit_Error_Start ( void );
```

[引数]

なし

[戻り値]

なし

R_ECM_Pseudo_DSMIF_UVWshortcircuit_Error_Stop

エラー要因 28　ΔΣ インタフェース UVW 短絡異常検出エラーの疑似エラーを発行しません。

[指定形式]

```
void    R_ECM_Pseudo_DSMIF_UVWshortcircuit_Error_Stop ( void );
```

[引数]

なし

[戻り値]

なし

R_ECM_Pseudo_DSMIF_Xovercurrent_Error_Start

エラー要因 29　ΔΣ インタフェース X 過電流異常検出エラーの疑似エラーを発行します。

[指定形式]

```
void    R_ECM_Pseudo_DSMIF_Xovercurrent_Error_Start ( void );
```

[引数]

なし

[戻り値]

なし

R_ECM_Pseudo_DSMIF_Xovercurrent_Error_Stop

エラー要因 29　ΔΣ インタフェース X 過電流異常検出エラーの疑似エラーを発行しません。

[指定形式]

```
void    R_ECM_Pseudo_DSMIF_Xovercurrent_Error_Stop ( void );
```

[引数]

なし

[戻り値]

なし

R_ECM_Pseudo_DSMIF_Xshortcircuit_Error_Start

エラー要因 31　ΔΣ インタフェース X 短絡異常検出エラーの疑似エラーを発行します。

[指定形式]

```
void    R_ECM_Pseudo_DSMIF_Xshortcircuit_Error_Start ( void );
```

[引数]

なし

[戻り値]

なし

R_ECM_Pseudo_DSMIF_Xshortcircuit_Error_Stop

エラー要因 31　ΔΣ インタフェース X 短絡異常検出エラーの疑似エラーを発行しません。

[指定形式]

```
void    R_ECM_Pseudo_DSMIF_Xshortcircuit_Error_Stop ( void );
```

[引数]

なし

[戻り値]

なし

R_ECM_Pseudo_DOC_Error_Start

エラー要因 32　データ演算回路 DOC 演算エラーの疑似エラーを発行します。

[指定形式]

```
void    R_ECM_Pseudo_DOC_Error_Start ( void );
```

[引数]

なし

[戻り値]

なし

R_ECM_Pseudo_DOC_Error_Stop

エラー要因 32　データ演算回路 DOC 演算エラーの疑似エラーを発行しません。

[指定形式]

```
void    R_ECM_Pseudo_DOC_Error_Stop ( void );
```

[引数]

なし

[戻り値]

なし

R_ECM_Pseudo_BSC_Error_Start

エラー要因 34 バスステートコントローラ外部 WAIT 端子による長期アクセスウェイト検出の疑似エラーを発行します。

[指定形式]

```
void    R_ECM_Pseudo_BSC_Error_Start ( void );
```

[引数]

なし

[戻り値]

なし

R_ECM_Pseudo_BSC_Error_Stop

エラー要因 34 バスステートコントローラ外部 WAIT 端子による長期アクセスウェイト検出の疑似エラーを発行しません。

[指定形式]

```
void    R_ECM_Pseudo_BSC_Error_Stop ( void );
```

[引数]

なし

[戻り値]

なし

R_ECM_Pseudo_Error35_Error_Start

エラー要因 35　拡張疑似エラー 35 の疑似エラーを発行します。

[指定形式]

```
void    R_ECM_Pseudo_Error35_Error_Start ( void );
```

[引数]

なし

[戻り値]

なし

R_ECM_Pseudo_Error35_Error_Stop

エラー要因 35　拡張疑似エラー 35 の疑似エラーを発行しません。

[指定形式]

```
void    R_ECM_Pseudo_Error35_Error_Stop ( void );
```

[引数]

なし

[戻り値]

なし

R_ECM_Pseudo_Error36_Error_Start

エラー要因 36　拡張疑似エラー 36 の疑似エラーを発行します。

[指定形式]

```
void    R_ECM_Pseudo_Error36_Error_Start ( void );
```

[引数]

なし

[戻り値]

なし

R_ECM_Pseudo_Error36_Error_Stop

エラー要因 36　拡張疑似エラー 36 の疑似エラーを発行しません。

[指定形式]

```
void    R_ECM_Pseudo_Error36_Error_Stop ( void );
```

[引数]

なし

[戻り値]

なし

R_ECM_Pseudo_Error37_Error_Start

エラー要因 37　拡張疑似エラー 37 の疑似エラーを発行します。

[指定形式]

```
void    R_ECM_Pseudo_Error37_Error_Start ( void );
```

[引数]

なし

[戻り値]

なし

R_ECM_Pseudo_Error37_Error_Stop

エラー要因 37　拡張疑似エラー 37 の疑似エラーを発行しません。

[指定形式]

```
void    R_ECM_Pseudo_Error37_Error_Stop ( void );
```

[引数]

なし

[戻り値]

なし

R_ECM_Pseudo_Error38_Error_Start

エラー要因 38　拡張疑似エラー 38 の疑似エラーを発行します。

[指定形式]

```
void    R_ECM_Pseudo_Error38_Error_Start ( void );
```

[引数]

なし

[戻り値]

なし

R_ECM_Pseudo_Error38_Error_Stop

エラー要因 38　拡張疑似エラー 38 の疑似エラーを発行しません。

[指定形式]

```
void    R_ECM_Pseudo_Error38_Error_Stop ( void );
```

[引数]

なし

[戻り値]

なし

R_ECM_Pseudo_Error39_Error_Start

エラー要因 39　拡張疑似エラー 39 の疑似エラーを発行します。

[指定形式]

```
void    R_ECM_Pseudo_Error39_Error_Start ( void );
```

[引数]

なし

[戻り値]

なし

R_ECM_Pseudo_Error39_Error_Stop

エラー要因 39　拡張疑似エラー 39 の疑似エラーを発行しません。

[指定形式]

```
void    R_ECM_Pseudo_Error39_Error_Stop ( void );
```

[引数]

なし

[戻り値]

なし

R_ECM_Pseudo_Error40_Error_Start

エラー要因 40　拡張疑似エラー 40 の疑似エラーを発行します。

[指定形式]

```
void    R_ECM_Pseudo_Error40_Error_Start ( void );
```

[引数]

なし

[戻り値]

なし

R_ECM_Pseudo_Error40_Error_Stop

エラー要因 40　拡張疑似エラー 40 の疑似エラーを発行しません。

[指定形式]

```
void    R_ECM_Pseudo_Error40_Error_Stop ( void );
```

[引数]

なし

[戻り値]

なし

R_ECM_Pseudo_Error41_Error_Start

エラー要因 41　拡張疑似エラー 41 の疑似エラーを発行します。

[指定形式]

```
void    R_ECM_Pseudo_Error41_Error_Start ( void );
```

[引数]

なし

[戻り値]

なし

R_ECM_Pseudo_Error41_Error_Stop

エラー要因 41　拡張疑似エラー 41 の疑似エラーを発行しません。

[指定形式]

```
void    R_ECM_Pseudo_Error41_Error_Stop ( void );
```

[引数]

なし

[戻り値]

なし

R_ECM_Pseudo_ECM_CompareError_Error_Start

エラー要因 93　エラーコントロールモジュール コンペアエラーの疑似エラーを発行します。

[指定形式]

```
void    R_ECM_Pseudo_ECM_CompareError_Error_Start ( void );
```

[引数]

なし

[戻り値]

なし

R_ECM_Pseudo_ECM_CompareError_Error_Stop

エラー要因 93　エラーコントロールモジュール コンペアエラーの疑似エラーを発行しません。

[指定形式]

```
void    R_ECM_Pseudo_ECM_CompareError_Error_Stop ( void );
```

[引数]

なし

[戻り値]

なし

R_ECM_Pseudo_ECM_DelayTimerOverflow_Error_Start

エラー要因 94　エラーコントロールモジュール ディレイタイマオーバーフローエラーの疑似エラーを発行します。

[指定形式]

```
void    R_ECM_Pseudo_ECM_DelayTimerOverflow_Error_Start ( void );
```

[引数]

なし

[戻り値]

なし

R_ECM_Pseudo_ECM_DelayTimerOverflow_Error_Stop

エラー要因 94　エラーコントロールモジュール ディレイタイマオーバーフローエラーの疑似エラーを発行しません。

[指定形式]

```
void    R_ECM_Pseudo_ECM_DelayTimerOverflow_Error_Stop ( void );
```

[引数]

なし

[戻り値]

なし

3.2.25 12 ビット A/D コンバータ

以下に，コード生成ツールが 12 ビット A/D コンバータ用として出力する API 関数の一覧を示します。

表 3.25 12 ビット A/D コンバータ用 API 関数

API 関数名	機能概要
R_S12ADn_Create	12 ビット A/D コンバータを制御するうえで必要となる初期化処理を行います。
R_S12ADn_Create_UserInit	12 ビット A/D コンバータに関するユーザ独自の初期化処理を行います。
r_s12ad_s12adn_interrupt	AD 変換終了割り込みの発生に伴う処理を行います。
r_s12ad_s12gbadin_interrupt	グループ B AD 変換終了割り込みの発生に伴う処理を行います。
r_s12ad_s12aden_interrupt	AD エラー割り込みの発生に伴う処理を行います。
R_S12ADn_Start	A/D 変換を開始します。
R_S12ADn_Stop	A/D 変換を終了します。
R_S12ADn_Get_ValueResult	変換結果を獲得します。
R_S12ADn_Set_CompareValue	コンペアレベルの設定を行います。
r_s12ad_s12cmpn_interrupt	コンペア条件成立割り込みの発生に伴う処理を行います。

R_S12AD*n*_Create

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

[指定形式]

```
void    R_S12ADn_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_S12AD*n*_Create_UserInit

12 ビット A/D コンバータに関するユーザ独自の初期化処理を行います。

備考　本 API 関数は，R_S12ADn_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_S12ADn_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_s12ad_s12ad*n*_interrupt

AD 変換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_s12ad_s12ad*n*_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_s12ad_s12gbadi*n*_interrupt

グループ B AD 変換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_s12ad_s12gbadin_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_s12ad_s12ade*n*_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_s12ad_s12aden_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_S12AD*n*_Start

A/D 変換を開始します。

[指定形式]

```
void    R_S12ADn_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_S12AD*n*_Stop

A/D 変換を終了します。

[指定形式]

```
void    R_S12ADn_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_S12AD*n*_Get_ValueResult

変換結果を獲得します。

[指定形式]

```
void    R_S12ADn_Get_ValueResult ( ad_channel_t channel, uint16_t * const buffer );
```

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	`ad_channel_t channel;`	チャネル番号 ADCHANNEL0 : 入力チャネル AN000 ADCHANNEL1 : 入力チャネル AN001 ADCHANNEL2 : 入力チャネル AN002 ADCHANNEL3 : 入力チャネル AN003 ADCHANNEL4 : 入力チャネル AN004 ADCHANNEL6 : 入力チャネル AN006 ADCHANNEL8 : 入力チャネル AN008 ADCHANNEL9 : 入力チャネル AN009 ADCHANNEL10 : 入力チャネル AN010 ADCHANNEL11 : 入力チャネル AN011 ADCHANNEL12 : 入力チャネル AN012 ADCHANNEL13 : 入力チャネル AN013 ADCHANNEL14 : 入力チャネル AN014 ADCHANNEL15 : 入力チャネル AN015 ADSELFDIAGNOSIS : 自己診断レジスタ ADTEMPSENSOR : 温度センサ出力 ADDATADUPLICATION : データ二重化レジスタ ADDATADUPLICATIONA : データ二重化レジスタ A ADDATADUPLICATIONB : データ二重化レジスタ B
O	`uint16_t * const buffer;`	獲得した変換結果を格納する領域へのポインタ

[戻り値]

なし

R_S12AD*n*_Set_CompareValue

コンペアレベルの設定を行います。

[指定形式]

```
void    R_S12ADn_Set_CompareValue ( uint16_t  reg_value0,uint16_t  reg_value1 );
```

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 reg_value0*	コンペアレジスタ 0 に設定する値
I	*uint16_t reg_value1*	コンペアレジスタ 1 に設定する値

[戻り値]

なし

r_s12ad_s12cmp*n*_interrupt

コンペア条件成立割り込みの発生に伴う処理を行います。

[指定形式]

```
void    r_s12ad_s12cmpn_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

3.2.26 データ演算回路

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

表 3.26 データ演算回路用 API 関数

API 関数名	機能概要
R_DOC_Create	データ演算回路を制御するうえで必要となる初期化処理を行います。
R_DOC_Create_UserInit	データ演算回路に関するユーザ独自の初期化処理を行います。
r_doc_dopcf_interrupt	DOC 演算エラーの発生に伴う処理を行います。
R_DOC_SetMode	動作モード，およびデータ演算を行うデータの初期値を設定します。
R_DOC_WriteData	データ演算を行う値（比較／加算／減算する値）を設定します。
R_DOC_GetResult	演算結果を獲得します。
R_DOC_ClearFlag	データ演算回路フラグをクリアします。

R_DOC_Create

データ演算回路を制御するうえで必要となる初期化処理を行います。

[指定形式]

```
void    R_DOC_Create ( void );
```

[引数]

なし

[戻り値]

なし

R_DOC_Create_UserInit

データ演算回路に関するユーザ独自の初期化処理を行います。

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

[指定形式]

```
void    R_DOC_Create_UserInit ( void );
```

[引数]

なし

[戻り値]

なし

r_doc_dopcf_interrupt

DOC 演算エラーの発生に伴う処理を行います。

[指定形式]

```
void    r_doc_dopcf_interrupt ( void );
```

[引数]

なし

[戻り値]

なし

R_DOC_SetMode

動作モード，およびデータ演算を行うデータの初期値を設定します。

備考 1. 動作モード *mode* にデータ比較モード COMPARE_MISMATCH，または COMPARE_MATCH が指定された場合，基準となる 16 ビットのデータ *value* が DOC データ・セッティング・レジスタ（DODSR）に格納されます。

備考 2. 動作モード *mode* にデータ加算モード ADDITION，またはデータ減算モード SUBTRACTION が指定された場合，初期値として 16 ビットのデータ *value* が DOC データ・セッティング・レジスタ（DODSR）に格納されます。

[指定形式]

```
void    R_DOC_SetMode ( doc_mode_t mode, uint16_t value );
```

[引数]

I/O	引数	説明
I	doc_mode_t *mode;*	動作モード（検出条件を含む）の種類 COMPARE_MISMATCH : データ比較モード（不一致） COMPARE_MATCH : データ比較モード（一致） ADDITION : データ加算モード SUBTRACTION : データ減算モード
I	uint16_t *value;*	データ演算を行うデータの初期値

[戻り値]

なし

R_DOC_WriteData

データ演算を行う値（比較／加算／減算する値）を設定します。

[指定形式]

```
void    R_DOC_WriteData ( uint16_t data );
```

[引数]

I/O	引数	説明
I	uint16_t *data*;	データ演算を行う値

[戻り値]

なし

R_DOC_GetResult

演算結果を獲得します。

[指定形式]

```
void    R_DOC_GetResult ( uint16_t * const data );
```

[引数]

I/O	引数	説明
O	`uint16_t * const data;`	獲得した演算結果を格納する領域へのポインタ

[戻り値]

なし

R_DOC_ClearFlag

データ演算回路フラグをクリアします。

[指定形式]

```
void    R_DOC_ClearFlag ( void );
```

[引数]

なし

[戻り値]

なし

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。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、本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。

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

   高品質水準：輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等

   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
6. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
9. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
10. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
12. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

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

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

(Rev.4.0-1　2017.11)

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。

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2015.04.01	―	初版発行
1.01	2017.11.20	―	誤記修正、記載抜け追記
1.02	2020.03.13	24 ― 25	NESTED_INTERRUPT_PUSH NESTED_INTERRUPT_POP を追加
		172, 178 ― 179	r_scifan_teifn_interrupt r_scifan_erifn_interrupt を削除
		188, 198	R_RIICn_Master_Send_Without_Stop を追加
		238, 245 ― 249	r_dsmif_uvw_overcurrent_interrupt r_dsmif_uvw_totalcurrent_interrupt r_dsmif_uvw_shortcircuit_interrupt r_dsmif_x_overcurrent_interrupt r_dsmif_x_shortcircuit_interrupt を追加
		250 ― 251	R_MPU_Create を追加

e^2 studio コード生成　ユーザーズマニュアル
RZ API リファレンス編

発行年月日　2015年4月　1日　Rev.1.00
2020年3月13日　Rev.1.02

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

e² studio コード生成

ルネサス エレクトロニクス株式会社

R20UT3302JJ0102