User's Manual

Applilet3

デバイス・ドライバ・コンフィギュレータ
ユーザーズマニュアル　RL78 API リファレンス編

対象デバイス

RL78 ファミリ

ルネサス エレクトロニクス
www.renesas.com

Rev.1.00　2011.10

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。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。
3. 当社製品を改造、改変、複製等しないでください。
4. 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。
5. 輸出に際しては、「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。本資料に記載されている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の目的で使用しないでください。また、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に使用することができません。
6. 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。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されてい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生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   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

   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、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社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。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、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、社会的損害など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、機器またはシステムとしての出荷保証をお願いいたします。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、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、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。
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固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

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。

このマニュアルの使い方

対 象 者　このマニュアルは，RL78 ファミリ用デバイス・ドライバ・コンフィギュレータ Applilet3 の機能を理解し，それを用いたアプリケーション・システムを設定するユーザを対象とします。

目　　的　このマニュアルは，Applilet3 の持つソフトウエア機能をユーザに理解していただき，これを使用するシステムのハードウエア，ソフトウエア開発の参照用資料として役立つことを目的としています。

構　　成　このマニュアルは，大きく分けて次の内容で構成しています。

読 み 方　このマニュアルを読むにあたっては，電気，論理回路，マイクロコンピュータに関する一般知識が必要となります。

凡　　例

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	：	2 進数　...　XXXXB
		10 進数　...　XXXX
		16 進数　...　XXXXH，または 0xXXXX

目　　次

第 1 章 概 説

本章では，Applilet3 の概要について説明します。

1.1 概 要

Applilet3 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が提供している周辺機能（クロック発生回路の機能，ポートの機能など）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，Applilet3 の特長を示します。

- コード生成機能

  Applilet3 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。

- プロジェクト／ワークスペース・ファイル生成機能

  Applilet3 では，アプリケーション・システムの統合開発環境（IAR Embedded Workbench）で利用可能なプロジェクト／ワークスペース・ファイルを出力することができます。

- レポート機能

  Applilet3 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

  Applilet3 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ができます。

第 2 章　出力ファイル

本章では，Applilet3 が出力するファイルについて説明します。

2.1　概　　要

以下に，Applilet3 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

出力単位	ファイル名	出力内容
各周辺機能	r_cg_ *周辺機能名* .c	初期化関数，API 関数
	r_cg_ *周辺機能名* _user.c	割り込み関数，コールバック関数
	r_cg_ *周辺機能名* .h	レジスタへの代入値マクロを定義
プロジェクト	r_main.c	main 関数
	r_systeminit.c	各周辺機能の初期化関数コール R_CGC_Get_ResetSource のコール
	r_cg_macrodrivr.h	全ソース・ファイルで共通使用するマクロを定義
	r_cg_userdefine.h	空ファイル（ユーザ定義用）
	r_lk.dr	リンク・ディレクティブ
	md_lnk*xxx*_*xx*_xcl	リンク・ディレクティブ（IAR Systems 用）
	r_option_ca.txt	ユーザ・オプション・バイト値など

2.2　出力ファイル

以下に，Applilet3 が出力するファイル（各周辺機能）を示します。

表 2—2　出力ファイル（各周辺機能）

周辺機能	ファイル名	含まれる API 関数名
クロック発生回路	r_cg_cgc.c	R_CGC_Create R_CGC_Set_ClockMode R_CGC_Set_CRCOn
	r_cg_cgc_user.c	R_CGC_Create_UserInit R_CGC_Get_ResetSource
	r_cg_cgc.h	—
ポート	r_cg_port.c	R_PORT_Create
	r_cg_port_user.c	R_PORT_Create_UserInit
	r_cg_port.h	—

周辺機能	ファイル名	含まれる API 関数名
割り込み	r_cg_intc.c	R_INTC_Create R_INTCn_Start R_INTCn_Stop R_KEY_Create R_KEY_Start R_KEY_Stop
	r_cg_intc_user.c	R_INTC_Create_UserInit r_intcn_interrupt R_KEY_Create_UserInit r_key_interrupt
	r_cg_intc.h	—
シリアル	r_cg_serial.c	R_SAUm_Create R_SAUm_Set_PowerOff R_SAUm_Set_SnoozeOn R_SAUm_Set_SnoozeOff R_UARTn_Create R_UARTn_Start R_UARTn_Stop R_UARTn_Send R_UARTn_Receive R_CSImn_Create R_CSImn_Start R_CSImn_Stop R_CSImn_Send R_CSImn_Receive R_CSImn_Send_Receive R_IICmn_Create R_IICmn_StartCondition R_IICmn_StopCondition R_IICmn_Stop R_IICmn_Master_Send R_IICmn_Master_Receive R_DALIn_Create R_DALIn_Start R_DALIn_Stop R_DALIn_Send R_DALIn_Receive R_IICAn_Create R_IICAn_StopCondition R_IICAn_Stop R_IICAn_Set_PowerOff R_IICAn_Master_Send R_IICAn_Master_Receive R_IICAn_Slave_Send

周辺機能	ファイル名	含まれる API 関数名
シリアル	r_cg_serial.c	R_IICAn_Slave_Receive
	r_cg_serial_user.c	R_SAUm_Create_UserInit r_uartn_interrupt_send r_uartn_interrupt_receive r_uartn_interrupt_error r_uartn_callback_sendend r_uartn_callback_receiveend r_uartn_callback_error r_uartn_callback_softwareoverrun r_csimn_interrupt r_csimn_callback_sendend r_csimn_callback_receiveend r_csimn_callback_error r_iicmn_interrupt r_iicmn_callback_master_sendend r_iicmn_callback_master_receiveend r_iicmn_callback_master_error r_dalin_interrupt_send r_dalin_interrupt_receive r_dalin_interrupt_error r_dalin_callback_sendend r_dalin_callback_receiveend r_dalin_callback_error r_dalin_callback_softwareoverrun R_IICAn_Create_UserInit r_iican_interrupt r_iican_callback_master_sendend r_iican_callback_master_receiveend r_iican_callback_master_error r_iican_callback_slave_sendend r_iican_callback_slave_receiveend r_iican_callback_slave_error r_iican_callback_getstopcondition
	r_cg_serial.h	—
A/D コンバータ	r_cg_adc.c	R_ADC_Create R_ADC_Set_OperationOn R_ADC_Set_OperationOff R_ADC_Start R_ADC_Stop R_ADC_Set_PowerOff R_ADC_Set_ADChannel R_ADC_Set_SnoozeOn R_ADC_Set_SnoozeOff R_ADC_Set_TestChannel

周辺機能	ファイル名	含まれる API 関数名
A/D コンバータ	r_cg_adc.c	R_ADC_Get_Result R_ADC_Get_Result_8bit
	r_cg_adc_user.c	R_ADC_Create_UserInit r_adc_interrupt
	r_cg_adc.h	—
D/A コンバータ	r_cg_dac.c	R_DAC_Create R_DACn_Start R_DACn_Stop R_DAC_Set_PowerOff R_DACn_Set_ConversionValue
	r_cg_dac_user.c	R_DAC_Create_UserInit
	r_cg_dac.h	—
タイマ	r_cg_timer.c	R_TAUm_Create R_TAUm_Channeln_Start R_TAUm_Channeln_Higher8bits_Start R_TAUm_Channeln_Lower8bits_Start R_TAUm_Channeln_Stop R_TAUm_Channeln_Higher8bits_Stop R_TAUm_Channeln_Lower8bits_Stop R_TAUm_Set_PowerOff R_TAUm_Channeln_Get_PulseWidth R_TAUm_Channeln_Set_SoftwareTriggerOn R_TMR_RDn_Create R_TMR_RDn_Start R_TMR_RDn_Stop R_TMR_RDn_Set_PowerOff R_TMR_RDn_ForcedOutput_Start R_TMR_RDn_ForcedOutput_Stop R_TMR_RDn_Get_PulseWidth R_TMR_RG0_Create R_TMR_RG0_Start R_TMR_RG0_Stop R_TMR_RG0_Set_PowerOff R_TMR_RG0_Get_PulseWidth R_TMR_RJ0_Create R_TMR_RJ0_Start R_TMR_RJ0_Stop R_TMR_RJ0_Set_PowerOff R_TMR_RJ0_Get_PulseWidth R_TMR_KB_Create R_TMR_KBm_Start R_TMR_KBm_Stop R_TMR_KBm_Set_PowerOff

周辺機能	ファイル名	含まれる API 関数名
タイマ	r_cg_timer.c	R_TMR_KBm_ForcedOutput_Start R_TMR_KBm_ForcedOutput_Stop R_TMR_KC0_Create R_TMR_KC0_Start R_TMR_KC0_Stop R_TMR_KC0_Set_PowerOff
	r_cg_timer_user.c	R_TAUm_Create_UserInit r_taum_channeln_interrupt r_taum_channeln_higher8bits_interrupt R_TMR_RDn_Create_UserInit r_tmr_rdn_interrupt R_TMR_RG0_Create_UserInit r_tmr_rg0_interrupt R_TMR_RJ0_Create_UserInit r_tmr_rj0_interrupt R_TMR_KBm_Create_UserInit r_tmr_kbm_interrupt R_TMR_KC0_Create_UserInit r_tmr_kc0_interrupt
	r_cg_timer.h	—
ウォッチドッグ・タイマ	r_cg_wdt.c	R_WDT_Create R_WDT_Restart
	r_cg_wdt_user.c	R_WDT_Create_UserInit r_wdt_interrupt
	r_cg_wdt.h	—
リアルタイム・クロック	r_cg_rtc.c	R_RTC_Create R_RTC_Start R_RTC_Stop R_RTC_Set_PowerOff R_RTC_Set_HourSystem R_RTC_Set_CounterValue R_RTC_Get_CounterValue R_RTC_Set_ConstPeriodInterruptOn R_RTC_Set_ConstPeriodInterruptOff R_RTC_Set_AlarmOn R_RTC_Set_AlarmOff R_RTC_Set_AlarmValue R_RTC_Get_AlarmValue R_RTC_Set_RTC1HZOn R_RTC_Set_RTC1HZOff
	r_cg_rtc_user.c	R_RTC_Create_UserInit r_rtc_interrupt r_rtc_callback_constperiod

周辺機能	ファイル名	含まれる API 関数名
リアルタイム・クロック	r_cg_rtc_user.c	r_rtc_callback_alarm
	r_cg_rtc.h	—
インターバル・タイマ	r_cg_it.c	R_IT_Create R_IT_Start R_IT_Stop R_IT_Set_PowerOff
	r_cg_it_user.c	R_IT_Create_UserInit r_it_interrupt
	r_cg_it.h	—
コンパレータ	r_cg_comp.c	R_COMP_Create R_COMPn_Start R_COMPn_Stop R_COMP_Set_PowerOff
	r_cg_comp_user.c	R_COMP_Create_UserInit r_compn_interrupt
	r_cg_comp.h	—
クロック出力／ブザー出力	r_cg_pclbuz.c	R_PCLBUZn_Create R_PCLBUZn_Start R_PCLBUZn_Stop
	r_cg_pclbuz_user.c	R_PCLBUZn_Create_UserInit
	r_cg_pclbuz.h	—
データトランスファコントローラ	r_cg_dtc.c	R_DTC_Create R_DTCn_Start R_DTCn_Stop R_DTC_Set_PowerOff
	r_cg_dtc_user.c	R_DTC_Create_UserInit
	r_cg_dtc.h	—
イベントリンクコントローラ	r_cg_elc.c	R_ELC_Create R_ELC_Stop
	r_cg_elc_user.c	R_ELC_Create_UserInit
	r_cg_elc.h	—
DMA コントローラ	r_cg_dmac.c	R_DMACn_Create R_DMACn_Start R_DMACn_Stop R_DMACn_Set_SoftwareTriggerOn
	r_cg_dmac_user.c	R_DMACn_Create_UserInit r_dmacn_interrupt
	r_cg_dmac.h	—
電圧検出回路	r_cg_lvd.c	R_LVD_Create R_LVD_InterruptMode_Start

周辺機能	ファイル名	含まれる API 関数名
電圧検出回路	r_cg_lvd_user.c	R_LVD_Create_UserInit r_lvd_interrupt
	r_cg_lvd.h	—
プログラマブル・ゲイン・アンプ	r_cg_pga.c	R_PGA_Create R_PGA_Start R_PGA_Stop
	r_cg_pga_user.c	R_PGA_Create_UserInit
	r_cg_pga.h	—

第 3 章　API 関数

本章では，Applilet3 が出力する API 関数について説明します。

3.1　概　　要

以下に，Applilet3 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　出力関数

以下に，Applilet3 が出力する API 関数の一覧を示します。

表 3—1　API 関数一覧

周辺機能	API 関数名	機能概要
クロック発生回路	R_CGC_Create	クロック発生回路の機能，オンチップ・デバッグ機能など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，オンチップ・デバッグなどに関するユーザ独自の初期化処理を行います。
	R_CGC_Get_ResetSource	内部リセットの発生に伴う処理を行います。
	R_CGC_Set_ClockMode	CPU クロック／周辺ハードウエア・クロックを変更します。
	R_CGC_Set_CRCOn	CRC 演算機能を開始します。

周辺機能	API 関数名	機能概要
ポート	R_PORT_Create	ポートの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_PORT_Create_UserInit	ポートに関するユーザ独自の初期化処理を行います。
割り込み	R_INTC_Create	外部マスカブル割り込み INTP*n* の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	外部マスカブル割り込み INTP*n* に関するユーザ独自の初期化処理を行います。
	r_intcn_interrupt	外部マスカブル割り込み INTP*n* の発生に伴う処理を行います。
	R_INTCn_Start	外部マスカブル割り込み INTP*n* の受け付けを許可します。
	R_INTCn_Stop	外部マスカブル割り込み INTP*n* の受け付けを禁止します。
	R_KEY_Create	キー割り込み INTKR の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_KEY_Create_UserInit	キー割り込み INTKR に関するユーザ独自の初期化処理を行います。
	r_key_interrupt	キー割り込み INTKR の発生に伴う処理を行います。
	R_KEY_Start	キー割り込み INTKR の受け付けを許可します。
	R_KEY_Stop	キー割り込み INTKR の受け付けを禁止します。
シリアル	R_SAUm_Create	シリアル・アレイ・ユニット，およびシリアル・インタフェースの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_SAUm_Create_UserInit	シリアル・アレイ・ユニット，およびシリアル・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_SAUm_Set_PowerOff	シリアル・アレイ・ユニットに対するクロック供給を停止します。
	R_SAUm_Set_SnoozeOn	STOP モードから SNOOZE モードへの切り替えを許可します。
	R_SAUm_Set_SnoozeOff	STOP モードから SNOOZE モードへの切り替えを禁止します。
	R_UARTn_Create	シリアル・インタフェース（UART）用チャネルの初期化処理を行います。
	r_uartn_interrupt_send	UART 送信完了割り込み INTST*n* の発生に伴う処理を行います。
	r_uartn_interrupt_receive	UART 受信完了割り込み INTSR*n* の発生に伴う処理を行います。

周辺機能	API 関数名	機能概要
シリアル	r_uartn_interrupt_error	受信エラー割り込み INTSRE*n* の発生に伴う処理を行います。
	R_UARTn_Start	UART 通信を待機状態にします。
	R_UARTn_Stop	UART 通信を終了します。
	R_UARTn_Send	データの UART 送信を開始します。
	R_UARTn_Receive	データの UART 受信を開始します。
	r_uartn_callback_sendend	UART 送信完了割り込み INTST*n* の発生に伴う処理を行います。
	r_uartn_callback_receiveend	UART 受信完了割り込み INTSR*n* の発生に伴う処理を行います。
	r_uartn_callback_error	UART 受信エラー割り込み INTSRE*n* の発生に伴う処理を行います。
	r_uartn_callback_softwareoverrun	オーバラン・エラーの検出に伴う処理を行います。
	R_CSImn_Create	シリアル・インタフェース（CSI）用チャネルの初期化処理を行います。
	r_csimn_interrupt	CSI 通信完了割り込み INTCSI*mn* の発生に伴う処理を行います。
	R_CSImn_Start	CSI 通信を待機状態にします。
	R_CSImn_Stop	CSI 通信を終了します。
	R_CSImn_Send	データの CSI 送信を開始します。
	R_CSImn_Receive	データの CSI 受信を開始します。
	R_CSImn_Send_Receive	データの CSI 送受信を開始します。
	r_csimn_callback_sendend	CSI 送信完了割り込み INTCSI*mn* の発生に伴う処理を行います。
	r_csimn_callback_receiveend	CSI 受信完了割り込み INTCSI*mn* の発生に伴う処理を行います。
	r_csimn_callback_error	CSI 受信エラー割り込み INTSRE*n* の発生に伴う処理を行います。
	R_IICmn_Create	シリアル・インタフェース（簡易 IIC）用チャネルの初期化処理を行います。
	r_iicmn_interrupt	簡易 IIC 通信完了割り込み INTIIC*mn* の発生に伴う処理を行います。
	R_IICmn_StartCondition	スタート・コンディションを発生します。
	R_IICmn_StopCondition	ストップ・コンディションを発生します。
	R_IICmn_Stop	簡易 IIC 通信を終了します。
	R_IICmn_Master_Send	簡易 IIC マスタ送信を開始します。
	R_IICmn_Master_Receive	簡易 IIC マスタ受信を開始します。
	r_iicmn_callback_master_sendend	簡易 IIC マスタ送信完了割り込み INTIIC*mn* の発生に伴う処理を行います。

周辺機能	API 関数名	機能概要
シリアル	r_iicmn_callback_master_receiveend	簡易 IIC マスタ受信完了割り込み INTIIC*mn* の発生に伴う処理を行います。
	r_iicmn_callback_master_error	パリティ・エラー（ACK エラー）の検出に伴う処理を行います。
	R_DALIn_Create	シリアル・インタフェース（DALI）用チャネルの初期化処理を行います。
	r_dalin_interrupt_send	DALI 送信完了割り込み INTSTDL*n* の発生に伴う処理を行います。
	r_dalin_interrupt_receive	DALI 受信完了割り込み INTSRDL*n* の発生に伴う処理を行います。
	r_dalin_interrupt_error	DALI 受信エラー割り込み INTSREDL*n* の発生に伴う処理を行います。
	R_DALIn_Start	DALI 通信を待機状態にします。
	R_DALIn_Stop	DALI 通信を終了します。
	R_DALIn_Send	データの DALI 送信を開始します。
	R_DALIn_Receive	データの DALI 受信を開始します。
	r_dalin_callback_sendend	DALI 送信完了割り込み INTSTDL*n* の発生に伴う処理を行います。
	r_dalin_callback_receiveend	DALI 受信完了割り込み INTSRDL*n* の発生に伴う処理を行います。
	r_dalin_callback_error	DALI 受信エラー割り込み INTSREDL*n* の発生に伴う処理を行います。
	r_dalin_callback_softwareoverrun	オーバラン・エラーの検出に伴う処理を行います。
	R_IICAn_Create	シリアル・インタフェース（IICA）の初期化処理を行います。
	R_IICAn_Create_UserInit	シリアル・インタフェース（IICA）に関するユーザ独自の初期化処理を行います。
	r_iican_interrupt	IICA 通信完了割り込み INTIICA*n* の発生に伴う処理を行います。
	R_IICAn_StopCondition	ストップ・コンディションを発生します。
	R_IICAn_Stop	IICA 通信を終了します。
	R_IICAn_Set_PowerOff	シリアル・インタフェース（IICA）に対するクロック供給を停止します。
	R_IICAn_Master_Send	IICA マスタ送信を開始します。
	R_IICAn_Master_Receive	IICA マスタ受信を開始します。
	r_iican_callback_master_sendend	IICA マスタ送信完了割り込み INTIICA*n* の発生に伴う処理を行います。
	r_iican_callback_master_receiveend	IICA マスタ受信完了割り込み INTIICA*n* の発生に伴う処理を行います。

周辺機能	API 関数名	機能概要
シリアル	r_iican_callback_master_error	IICA マスタ通信エラーの検出に伴う処理を行います。
	R_IICAn_Slave_Send	IICA スレーブ送信を開始します。
	R_IICAn_Slave_Receive	IICA スレーブ受信を開始します。
	r_iican_callback_slave_sendend	IICA スレーブ送信完了割り込み INTIICA*n* の発生に伴う処理を行います。
	r_iican_callback_slave_receiveend	IICA スレーブ受信完了割り込み INTIICA*n* の発生に伴う処理を行います。
	r_iican_callback_slave_error	IICA スレーブ通信エラーの検出に伴う処理を行います。
	r_iican_callback_getstopcondition	ストップ・コンディションの検出に伴う処理を行います。
A/D コンバータ	R_ADC_Create	A/D コンバータの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_ADC_Create_UserInit	A/D コンバータに関するユーザ独自の初期化処理を行います。
	r_adc_interrupt	A/D 変換終了割り込み INTAD の発生に伴う処理を行います。
	R_ADC_Set_OperationOn	電圧コンパレータを動作許可状態に設定します。
	R_ADC_Set_OperationOff	電圧コンパレータを動作停止状態に設定します。
	R_ADC_Start	A/D 変換を開始します。
	R_ADC_Stop	A/D 変換を終了します。
	R_ADC_Set_PowerOff	A/D コンバータに対するクロック供給を停止します。
	R_ADC_Set_ADChannel	A/D 変換するアナログ電圧の入力端子を設定します。
	R_ADC_Set_SnoozeOn	STOP モードから SNOOZE モードへの切り替えを許可します。
	R_ADC_Set_SnoozeOff	STOP モードから SNOOZE モードへの切り替えを禁止します。
	R_ADC_Set_TestChannel	A/D コンバータの動作モードを設定します。
	R_ADC_Get_Result	A/D 変換結果（10 ビット）を読み出します。
	R_ADC_Get_Result_8bit	A/D 変換結果（8 ビット：10 ビット分解能の上位 8 ビット）を読み出します。
D/A コンバータ	R_DAC_Create	D/A コンバータの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_DAC_Create_UserInit	D/A コンバータに関するユーザ独自の初期化処理を行います。
	R_DACn_Start	D/A 変換を開始します。
	R_DACn_Stop	D/A 変換を終了します。

周辺機能	API 関数名	機能概要
D/A コンバータ	R_DAC_Set_PowerOff	D/A コンバータに対するクロック供給を停止します。
	R_DACn_Set_ConversionValue	ANO*n* 端子に出力するアナログ電圧値を設定します。
タイマ	R_TAUm_Create	タイマ・アレイ・ユニットの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_TAUm_Create_UserInit	タイマ・アレイ・ユニットに関するユーザ独自の初期化処理を行います。
	r_taum_channeln_interrupt	タイマ割り込み INTTM*mn* の発生に伴う処理を行います。
	r_taum_channeln_higher8bits_interrupt	タイマ割り込み INTTM*mn*H の発生に伴う処理を行います。
	R_TAUm_Channeln_Start	チャネル *n* のカウントを開始します。
	R_TAUm_Channeln_Higher8bits_Start	チャネル 1，またはチャンネル 3 のカウント（上位 8 ビット）を開始します。
	R_TAUm_Channeln_Lower8bits_Start	チャネル 1，またはチャンネル 3 のカウント（下位 8 ビット）を開始します。
	R_TAUm_Channeln_Stop	チャネル *n* のカウントを終了します。
	R_TAUm_Channeln_Higher8bits_Stop	チャネル 1，またはチャンネル 3 のカウント（上位 8 ビット）を終了します。
	R_TAUm_Channeln_Lower8bits_Stop	チャネル 1，またはチャンネル 3 のカウント（下位 8 ビット）を終了します。
	R_TAUm_Set_PowerOff	タイマ・アレイ・ユニットに対するクロック供給を停止します。
	R_TAUm_Channeln_Get_PulseWidth	TI*mn* 端子に対する入力信号（入力パルス）のパルス間隔，またはハイ／ロウ・レベルの測定幅を獲得します。
	R_TAUm_Channeln_Set_SoftwareTriggerOn	ワンショット・パルス出力のためのトリガ（ソフトウエア・トリガ）を発生させます。
	R_TMR_RDn_Create	16 ビット・タイマ RD*n* の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_TMR_RDn_Create_UserInit	16 ビット・タイマ RD*n* に関するユーザ独自の初期化処理を行います。
	r_tmr_rdn_interrupt	タイマ割り込みの発生に伴う処理を行います。
	R_TMR_RDn_Start	16 ビット・タイマ RD*n* のカウント処理を開始します。
	R_TMR_RDn_Stop	16 ビット・タイマ RD*n* のカウント処理を終了します。
	R_TMR_RDn_Set_PowerOff	16 ビット・タイマ RD*n* に対するクロック供給を停止します。

周辺機能	API 関数名	機能概要
タイマ	R_TMR_RDn_ForcedOutput_Start	16 ビット・タイマ RD*n* のパルス出力強制遮断処理を開始します。
	R_TMR_RDn_ForcedOutput_Stop	16 ビット・タイマ RD*n* のパルス出力強制遮断処理を終了します。
	R_TMR_RDn_Get_PulseWidth	16 ビット・タイマ RD*n* のパルス幅を読み出します。
	R_TMR_RG0_Create	16 ビット・タイマ RG0 の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_TMR_RG0_Create_UserInit	16 ビット・タイマ RG0 に関するユーザ独自の初期化処理を行います。
	r_tmr_rg0_interrupt	タイマ割り込みの発生に伴う処理を行います。
	R_TMR_RG0_Start	16 ビット・タイマ RG0 のカウント処理を開始します。
	R_TMR_RG0_Stop	16 ビット・タイマ RG0 のカウント処理を終了します。
	R_TMR_RG0_Set_PowerOff	16 ビット・タイマ RG0 に対するクロック供給を停止します。
	R_TMR_RG0_Get_PulseWidth	16 ビット・タイマ RG*0* のパルス幅を読み出します。
	R_TMR_RJ0_Create	16 ビット・タイマ RJ0 の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_TMR_RJ0_Create_UserInit	16 ビット・タイマ RJ0 に関するユーザ独自の初期化処理を行います。
	r_tmr_rj0_interrupt	タイマ割り込みの発生に伴う処理を行います。
	R_TMR_RJ0_Start	16 ビット・タイマ RJ0 のカウント処理を開始します。
	R_TMR_RJ0_Stop	16 ビット・タイマ RJ0 のカウント処理を終了します。
	R_TMR_RJ0_Set_PowerOff	16 ビット・タイマ RJ0 に対するクロック供給を停止します。
	R_TMR_RJ0_Get_PulseWidth	16 ビット・タイマ RJ0 のパルス幅を読み出します。
	R_TMR_KB_Create	16 ビット・タイマ KB*m* の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_TMR_KBm_Create_UserInit	16 ビット・タイマ KB*m* に関するユーザ独自の初期化処理を行います。
	r_tmr_kbm_interrupt	タイマ割り込みの発生に伴う処理を行います。
	R_TMR_KBm_Start	16 ビット・タイマ KB*m* のカウント処理を開始します。

周辺機能	API 関数名	機能概要
タイマ	R_TMR_KBm_Stop	16 ビット・タイマ KB*m* のカウント処理を終了します。
	R_TMR_KBm_Set_PowerOff	16 ビット・タイマ KB*m* に対するクロック供給を停止します。
	R_TMR_KBm_ForcedOutput_Start	強制出力停止機能に使用するトリガ信号の入力を許可します。
	R_TMR_KBm_ForcedOutput_Stop	強制出力停止機能に使用するトリガ信号の入力を禁止します。
	R_TMR_KC0_Create	16 ビット・タイマ KC0 の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_TMR_KC0_Create_UserInit	16 ビット・タイマ KC0 に関するユーザ独自の初期化処理を行います。
	r_tmr_kc0_interrupt	タイマ割り込みの発生に伴う処理を行います。
	R_TMR_KC0_Start	16 ビット・タイマ KC0 のカウント処理を開始します。
	R_TMR_KC0_Stop	16 ビット・タイマ KC0 のカウント処理を終了します。
	R_TMR_KC0_Set_PowerOff	16 ビット・タイマ KC0 に対するクロック供給を停止します。
ウォッチドッグ・タイマ	R_WDT_Create	ウォッチドッグ・タイマの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_WDT_Create_UserInit	ウォッチドッグ・タイマに関するユーザ独自の初期化処理を行います。
	r_wdt_interrupt	ウォッチドッグ・タイマのインターバル割り込み INTWDTI の発生に伴い処理を行います。
	R_WDT_Restart	ウォッチドッグ・タイマのカウンタをクリアしたのち，カウント処理を再開します。
リアルタイム・クロック	R_RTC_Create	リアルタイム・クロックの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_RTC_Create_UserInit	リアルタイム・クロックに関するユーザ独自の初期化処理を行います。
	r_rtc_interrupt	リアルタイム・クロック割り込み INTRTC の発生に伴う処理を行います。
	R_RTC_Start	リアルタイム・クロック（年，月，曜日，日，時，分，秒）のカウントを開始します。
	R_RTC_Stop	リアルタイム・クロック（年，月，曜日，日，時，分，秒）のカウントを終了します。
	R_RTC_Set_PowerOff	リアルタイム・クロックに対するクロック供給を停止します。
	R_RTC_Set_HourSystem	リアルタイム・クロックの時間制（12 時間制，24 時間制）を設定します。

周辺機能	API 関数名	機能概要
リアルタイム・クロック	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・クロックのカウント値を読み出します。
	R_RTC_Set_ConstPeriodInterruptOn	割り込み INTRTC の発生周期を設定したのち，定周期割り込み機能を開始します。
	R_RTC_Set_ConstPeriodInterruptOff	定周期割り込み機能を終了します。
	r_rtc_callback_constperiod	定周期割り込み INTRTC の発生に伴う処理を行います。
	R_RTC_Set_AlarmOn	アラーム割り込み機能を開始します。
	R_RTC_Set_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_callback_alarm	アラーム割り込み INTRTC の発生に伴う処理を行います。
	R_RTC_Set_RTC1HZOn	RTC1HZ 端子に対する補正クロック（1 Hz）の出力を許可します。
	R_RTC_Set_RTC1HZOff	RTC1HZ 端子に対する補正クロック（1 Hz）の出力を禁止します。
インターバル・タイマ	R_IT_Create	インターバル・タイマの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_IT_Create_UserInit	インターバル・タイマに関するユーザ独自の初期化処理を行います。
	r_it_interrupt	インターバル・タイマ割り込み INTIT の発生に伴う処理を行います。
	R_IT_Start	インターバル・タイマのカウントを開始します。
	R_IT_Stop	インターバル・タイマのカウントを終了します。
	R_IT_Set_PowerOff	インターバル・タイマに対するクロック供給を停止します。
コンパレータ	R_COMP_Create	コンパレータの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_COMP_Create_UserInit	コンパレータに関するユーザ独自の初期化処理を行います。
	r_compn_interrupt	コンパレータ割り込み INTCMP*n* の発生に伴う処理を行います。
	R_COMPn_Start	リファレンス入力電圧とアナログ入力電圧の比較動作を開始します。

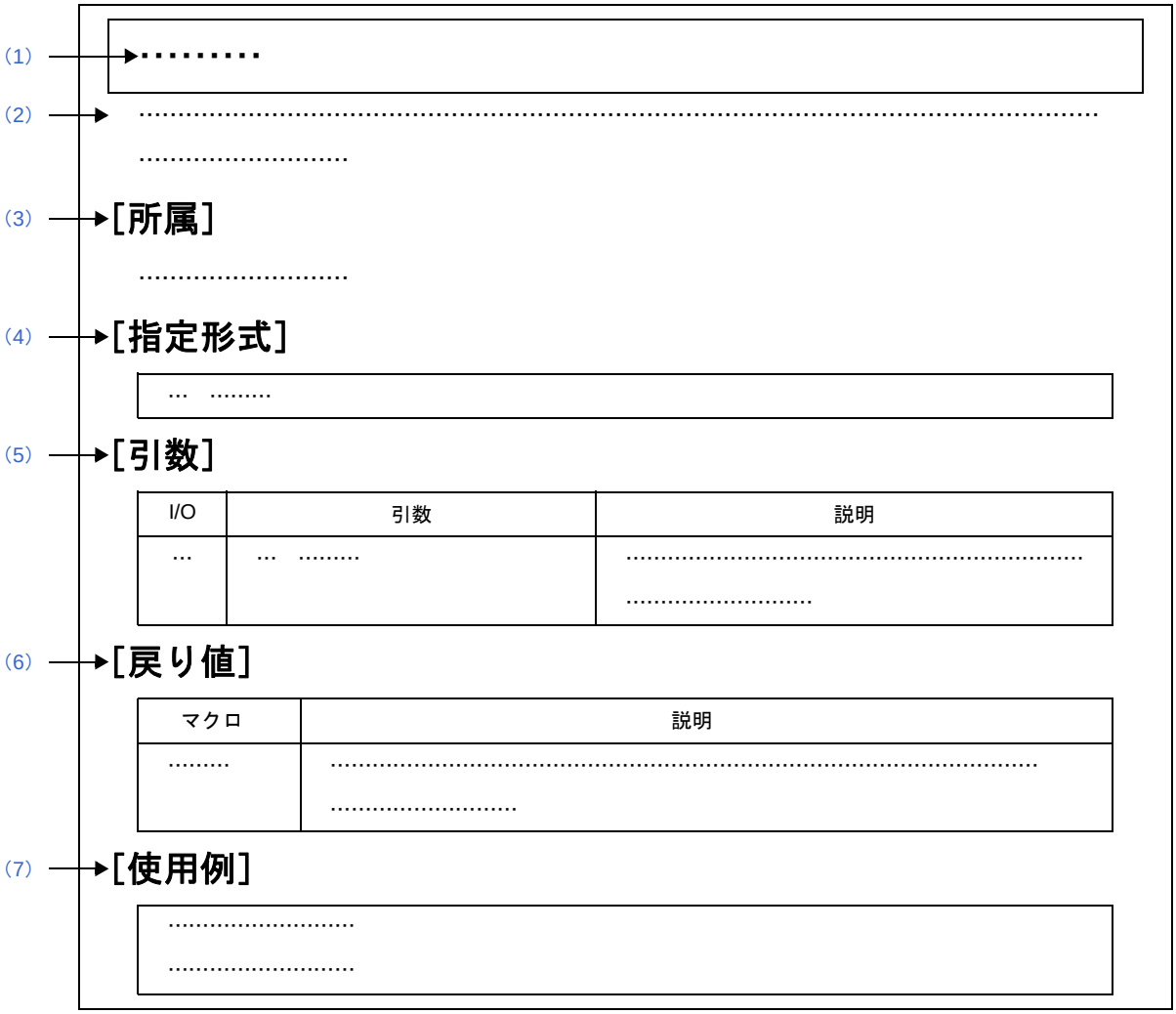
周辺機能	API 関数名	機能概要
コンパレータ	R_COMPn_Stop	リファレンス入力電圧とアナログ入力電圧の比較動作を停止します。
	R_COMP_Set_PowerOff	コンパレータに対するクロック供給を停止します。
クロック出力／ブザー出力	R_PCLBUZn_Create	クロック出力／ブザー出力制御回路の機能を制御するうえで必要となる初期化処理を行います。
	R_PCLBUZn_Create_UserInit	クロック出力／ブザー出力制御回路に関するユーザ独自の初期化処理を行います。
	R_PCLBUZn_Start	クロック出力／ブザー出力を開始します。
	R_PCLBUZn_Stop	クロック出力／ブザー出力を停止します。
データトランスファコントローラ	R_DTC_Create	データトランスファコントローラの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_DTC_Create_UserInit	データトランスファコントローラに関するユーザ独自の初期化処理を行います。
	R_DTCn_Start	データトランスファコントローラを動作可能状態にします。
	R_DTCn_Stop	データトランスファコントローラを動作停止状態にします。
	R_DTC_Set_PowerOff	データトランスファコントローラに対するクロック供給を停止します。
イベントリンクコントローラ	R_ELC_Create	イベントリンクコントローラの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_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_Stop	イベントリンクコントローラを動作停止状態にします。
DMA コントローラ	R_DMACn_Create	DMA コントローラの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_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_interrupt	DMA 転送終了割り込み INTDMA*n* の発生に伴う処理を行います。
	R_DMACn_Start	チャネル *n* を動作許可状態に設定します。
	R_DMACn_Stop	チャネル *n* を動作停止状態に設定します。
	R_DMACn_Set_SoftwareTriggerOn	DMA 動作許可状態の際，DMA 転送を開始します。
電圧検出回路	R_LVD_Create	電圧検出回路の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_LVD_Create_UserInit	電圧検出回路に関するユーザ独自の初期化処理を行います。
	r_lvd_interrupt	電圧検出割り込み INTLVI の発生に伴う処理を行います。

周辺機能	API 関数名	機能概要
電圧検出回路	R_LVD_InterruptMode_Start	電圧検出動作を開始します（割り込みモード時，および割り込み＆リセット・モード時）。
プログラマブル・ゲイン・アンプ	R_PGA_Create	プログラマブル・ゲイン・アンプの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_PGA_Create_UserInit	プログラマブル・ゲイン・アンプに関するユーザ独自の初期化処理を行います。
	R_PGA_Start	プログラマブル・ゲイン・アンプの動作を開始します。
	R_PGA_Stop	プログラマブル・ゲイン・アンプの動作を停止します。

3.3 関数リファレンス

本節では，Applilet3 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) [所属]

API 関数が出力される C ソース・ファイル名を示しています。

(4) [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。

(5) [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

(6) [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

(7) [使用例]

API 関数の使用例を示しています。

3. 3. 1 クロック発生回路

以下に，Applilet3 がクロック発生回路用として出力する 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	クロック発生回路の機能，オンチップ・デバッグ機能など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，オンチップ・デバッグなどに関するユーザ独自の初期化処理を行います。
R_CGC_Get_ResetSource	内部リセットの発生に伴う処理を行います。
R_CGC_Set_ClockMode	CPU クロック／周辺ハードウエア・クロックを変更します。
R_CGC_Set_CRCOn	CRC 演算機能を開始します。

R_CGC_Create

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

[所属]

r_cg_cgc.c

[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，オンチップ・デバッグなど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。

[所属]

r_cg_cgc_user.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_CGC_Get_ResetSource

内部リセットの発生に伴う処理を行います。

[所属]

r_cg_cgc_user.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_CGC_Set_ClockMode

CPU クロック／周辺ハードウエア・クロックを変更します。

[所属]

r_cg_cgc.c

[指定形式]

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

[引数]

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

[戻り値]

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

R_CGC_Set_CRCOn

CRC 演算機能を開始します。

[所属]

r_cg_cgc.c

[指定形式]

```
void    R_CGC_Set_CRCOn ( void );
```

[引数]

なし

[戻り値]

なし

3. 3. 2　ポート

以下に，Applilet3 が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　ポート用 API 関数

API 関数名	機能概要
R_PORT_Create	ポートの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_PORT_Create_UserInit	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

ポートの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。

[所属]

r_cg_port.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_PORT_Create_UserInit

ポート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。

[所属]

r_cg_port_user.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

3. 3. 3 割り込み

以下に，Applilet3 が割り込み用として出力する API 関数の一覧を示します。

表 3—4 割り込み用 API 関数

API 関数名	機能概要
R_INTC_Create	外部マスカブル割り込み INTP*n* の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	外部マスカブル割り込み INTP*n* に関するユーザ独自の初期化処理を行います。
r_intcn_interrupt	外部マスカブル割り込み INTP*n* の発生に伴う処理を行います。
R_INTCn_Start	外部マスカブル割り込み INTP*n* の受け付けを許可します。
R_INTCn_Stop	外部マスカブル割り込み INTP*n* の受け付けを禁止します。
R_KEY_Create	キー割り込み INTKR の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_KEY_Create_UserInit	キー割り込み INTKR に関するユーザ独自の初期化処理を行います。
r_key_interrupt	キー割り込み INTKR の発生に伴う処理を行います。
R_KEY_Start	キー割り込み INTKR の受け付けを許可します。
R_KEY_Stop	キー割り込み INTKR の受け付けを禁止します。

R_INTC_Create

外部マスカブル割り込み INTP*n* の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。

[所属]

r_cg_intc.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_INTC_Create_UserInit

外部マスカブル割り込み INTP*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_INTC_Create のコールバック・ルーチンとして呼び出されます。

[所属]

r_cg_intc_user.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

r_intc*n*_interrupt

外部マスカブル割り込み INTP*n* の発生に伴う処理を行います。

備考 本 API 関数は，外部マスカブル割り込み INTP*n* に対応した割り込み処理として呼び出されます。

[所属]

r_cg_intc_user.c

[指定形式]

```
__interrupt static  void    r_intcn_interrupt ( void );
```

備考 *n* は，割り込み要因番号を意味します。

[引数]

なし

[戻り値]

なし

R_INTC*n*_Start

外部マスカブル割り込み INTP*n* の受け付けを許可します。

[所属]

r_cg_intc.c

[指定形式]

```
void    R_INTCn_Start ( void );
```

備考 *n* は，割り込み要因番号を意味します。

[引数]

なし

[戻り値]

なし

R_INTC*n*_Stop

外部マスカブル割り込み INTP*n* の受け付けを禁止します。

[所属]

r_cg_intc.c

[指定形式]

```
void    R_INTCn_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_KEY_Create

キー割り込み INTKR の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。

[所属]

r_cg_intc.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_KEY_Create_UserInit

キー割り込み INTKR 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。

[所属]

r_cg_intc_user.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

r_key_interrupt

キー割り込み INTKR の発生に伴う処理を行います。

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

[所属]

r_cg_intc_user.c

[指定形式]

```
__interrupt static  void    r_key_interrupt ( void );
```

[引数]

なし

[戻り値]

なし

R_KEY_Start

キー割り込み INTKR の受け付けを許可します。

[所属]

r_cg_intc.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_KEY_Stop

キー割り込み INTKR の受け付けを禁止します。

[所属]

r_cg_intc.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

3. 3. 4 シリアル

以下に，Applilet3 がシリアル用として出力する API 関数の一覧を示します。

表 3—5 シリアル用 API 関数

API 関数名	機能概要
R_SAUm_Create	シリアル・アレイ・ユニット，およびシリアル・インタフェースの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_SAUm_Create_UserInit	シリアル・アレイ・ユニット，およびシリアル・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_SAUm_Set_PowerOff	シリアル・アレイ・ユニットに対するクロック供給を停止します。
R_SAUm_Set_SnoozeOn	STOP モードから SNOOZE モードへの切り替えを許可します。
R_SAUm_Set_SnoozeOff	STOP モードから SNOOZE モードへの切り替えを禁止します。
R_UARTn_Create	シリアル・インタフェース（UART）用チャネルの初期化処理を行います。
r_uartn_interrupt_send	UART 送信完了割り込み INTST*n* の発生に伴う処理を行います。
r_uartn_interrupt_receive	UART 受信完了割り込み INTSR*n* の発生に伴う処理を行います。
r_uartn_interrupt_error	受信エラー割り込み INTSRE*n* の発生に伴う処理を行います。
R_UARTn_Start	UART 通信を待機状態にします。
R_UARTn_Stop	UART 通信を終了します。
R_UARTn_Send	データの UART 送信を開始します。
R_UARTn_Receive	データの UART 受信を開始します。
r_uartn_callback_sendend	UART 送信完了割り込み INTST*n* の発生に伴う処理を行います。
r_uartn_callback_receiveend	UART 受信完了割り込み INTSR*n* の発生に伴う処理を行います。
r_uartn_callback_error	UART 受信エラー割り込み INTSRE*n* の発生に伴う処理を行います。
r_uartn_callback_softwareoverrun	オーバラン・エラーの検出に伴う処理を行います。
R_CSImn_Create	シリアル・インタフェース（CSI）用チャネルの初期化処理を行います。
r_csimn_interrupt	CSI 通信完了割り込み INTCSI*mn* の発生に伴う処理を行います。
R_CSImn_Start	CSI 通信を待機状態にします。
R_CSImn_Stop	CSI 通信を終了します。
R_CSImn_Send	データの CSI 送信を開始します。
R_CSImn_Receive	データの CSI 受信を開始します。
R_CSImn_Send_Receive	データの CSI 送受信を開始します。
r_csimn_callback_sendend	CSI 送信完了割り込み INTCSI*mn* の発生に伴う処理を行います。
r_csimn_callback_receiveend	CSI 受信完了割り込み INTCSI*mn* の発生に伴う処理を行います。
r_csimn_callback_error	CSI 受信エラー割り込み INTSRE*n* の発生に伴う処理を行います。
R_IICmn_Create	シリアル・インタフェース（簡易 IIC）用チャネルの初期化処理を行います。
r_iicmn_interrupt	簡易 IIC 通信完了割り込み INTIIC*mn* の発生に伴う処理を行います。

API 関数名	機能概要
R_IICmn_StartCondition	スタート・コンディションを発生します。
R_IICmn_StopCondition	ストップ・コンディションを発生します。
R_IICmn_Stop	簡易 IIC 通信を終了します。
R_IICmn_Master_Send	簡易 IIC マスタ送信を開始します。
R_IICmn_Master_Receive	簡易 IIC マスタ受信を開始します。
r_iicmn_callback_master_sendend	簡易 IIC マスタ送信完了割り込み INTIIC*mn* の発生に伴う処理を行います。
r_iicmn_callback_master_receiveend	簡易 IIC マスタ受信完了割り込み INTIIC*mn* の発生に伴う処理を行います。
r_iicmn_callback_master_error	パリティ・エラー（ACK エラー）の検出に伴う処理を行います。
R_DALIn_Create	シリアル・インタフェース（DALI）用チャネルの初期化処理を行います。
r_dalin_interrupt_send	DALI 送信完了割り込み INTSTDL*n* の発生に伴う処理を行います。
r_dalin_interrupt_receive	DALI 受信完了割り込み INTSRDL*n* の発生に伴う処理を行います。
r_dalin_interrupt_error	DALI 受信エラー割り込み INTSREDL*n* の発生に伴う処理を行います。
R_DALIn_Start	DALI 通信を待機状態にします。
R_DALIn_Stop	DALI 通信を終了します。
R_DALIn_Send	データの DALI 送信を開始します。
R_DALIn_Receive	データの DALI 受信を開始します。
r_dalin_callback_sendend	DALI 送信完了割り込み INTSTDL*n* の発生に伴う処理を行います。
r_dalin_callback_receiveend	DALI 受信完了割り込み INTSRDL*n* の発生に伴う処理を行います。
r_dalin_callback_error	DALI 受信エラー割り込み INTSREDL*n* の発生に伴う処理を行います。
r_dalin_callback_softwareoverrun	オーバラン・エラーの検出に伴う処理を行います。
R_IICAn_Create	シリアル・インタフェース（IICA）の初期化処理を行います。
R_IICAn_Create_UserInit	シリアル・インタフェース（IICA）に関するユーザ独自の初期化処理を行います。
r_iican_interrupt	IICA 通信完了割り込み INTIICA*n* の発生に伴う処理を行います。
R_IICAn_StopCondition	ストップ・コンディションを発生します。
R_IICAn_Stop	IICA 通信を終了します。
R_IICAn_Set_PowerOff	シリアル・インタフェース（IICA）に対するクロック供給を停止します。
R_IICAn_Master_Send	IICA マスタ送信を開始します。
R_IICAn_Master_Receive	IICA マスタ受信を開始します。
r_iican_callback_master_sendend	IICA マスタ送信完了割り込み INTIICA*n* の発生に伴う処理を行います。
r_iican_callback_master_receiveend	IICA マスタ受信完了割り込み INTIICA*n* の発生に伴う処理を行います。

API 関数名	機能概要
r_iican_callback_master_error	IICA マスタ通信エラーの検出に伴う処理を行います。
R_IICAn_Slave_Send	IICA スレーブ送信を開始します。
R_IICAn_Slave_Receive	IICA スレーブ受信を開始します。
r_iican_callback_slave_sendend	IICA スレーブ送信完了割り込み INTIICA*n* の発生に伴う処理を行います。
r_iican_callback_slave_receiveend	IICA スレーブ受信完了割り込み INTIICA*n* の発生に伴う処理を行います。
r_iican_callback_slave_error	IICA スレーブ通信エラーの検出に伴う処理を行います。
r_iican_callback_getstopcondition	ストップ・コンディションの検出に伴う処理を行います。

R_SAU*m*_Create

シリアル・アレイ・ユニット，およびシリアル・インタフェースの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。

[所属]

r_cg_serial.c

[指定形式]

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

備考 *m* は，ユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_SAU*m*_Create_UserInit

シリアル・アレイ・ユニット，およびシリアル・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_SAUm_Create のコールバック・ルーチンとして呼び出されます。

[所属]

r_cg_serial_user.c

[指定形式]

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

備考　*m* は，ユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_SAU*m*_Set_PowerOff

シリアル・アレイ・ユニットに対するクロック供給を停止します。

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

［所属］

r_cg_serial.c

［指定形式］

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

備考　*m* は，ユニット番号を意味します。

［引数］

なし

［戻り値］

なし

R_SAU*m*_Set_SnoozeOn

STOP モードから SNOOZE モードへの切り替えを許可します。

[所属]

r_cg_serial.c

[指定形式]

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

備考 *m* は，ユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_SAU*m*_Set_SnoozeOff

STOP モードから SNOOZE モードへの切り替えを禁止します。

[所属]

r_cg_serial.c

[指定形式]

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

備考 *m* は，ユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_UART*n*_Create

シリアル・インタフェース（UART）用チャネルの初期化処理を行います。

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

[所属]

r_cg_serial.c

[指定形式]

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

備考 *n* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_uart*n*_interrupt_send

UART 送信完了割り込み INTST*n* の発生に伴う処理を行います。

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

[所属]

r_cg_serial_user.c

[指定形式]

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

備考 *n* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_uart*n*_interrupt_receive

UART 受信完了割り込み INTSR*n* の発生に伴う処理を行います。

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

[所属]

r_cg_serial_user.c

[指定形式]

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

備考 *n* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_uart*n*_interrupt_error

受信エラー割り込み INTSRE*n* の発生に伴う処理を行います。

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

[所属]

r_cg_serial_user.c

[指定形式]

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

備考 *n* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_UART*n*_Start

UART 通信を待機状態にします。

[所属]

r_cg_serial.c

[指定形式]

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

備考 *n* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_UART*n*_Stop

UART 通信を終了します。

[所属]

r_cg_serial.c

[指定形式]

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

備考 *n* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_UART*n*_Send

データの UART 送信を開始します。

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

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

[所属]

r_cg_serial.c

[指定形式]

```
#include    "r_cg_macrodriver.h"
MD_STATUS   R_UARTn_Send ( 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	`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*n*_Receive

データの UART 受信を開始します。

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

2. 実際の UART 受信は，本 API 関数の呼び出し後，R_UARTn_Start を呼び出すことにより開始されます。

[所属]

r_cg_serial.c

[指定形式]

```
#include    "r_cg_macrodriver.h"
MD_STATUS   R_UARTn_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]

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

r_uart*n*_callback_sendend

UART 送信完了割り込み INTST*n* の発生に伴う処理を行います。

備考 本 API 関数は，UART 送信完了割り込み INTST*n* に対応した割り込み処理 r_uartn_interrupt_send のコールバック・ルーチン（R_UARTn_Send の引数 *tx_num* で指定された数のデータ送信が完了した際の処理）として呼び出されます。

[所属]

r_cg_serial_user.c

[指定形式]

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

備考 *n* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_uart*n*_callback_receiveend

UART 受信完了割り込み INTSR*n* の発生に伴う処理を行います。

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

[所属]

r_cg_serial_user.c

[指定形式]

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

備考 *n* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_uart*n*_callback_error

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

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

[所属]

r_cg_serial_user.c

[指定形式]

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

備考 *n* は，チャネル番号を意味します。

[引数]

I/O	引数	説明
O	uint8_t *err_type*;	UART 受信エラー割り込みの発生要因 00000*xx*1B：オーバラン・エラー 00000*x*1*x*B：パリティ・エラー 000001*xx*B：フレーミング・エラー

[戻り値]

なし

r_uart*n*_callback_softwareoverrun

オーバラン・エラーの検出に伴う処理を行います。

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

[所属]

r_cg_serial_user.c

[指定形式]

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

備考 *n* は，チャネル番号を意味します。

[引数]

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

[戻り値]

なし

R_CSI*mn*_Create

シリアル・インタフェース（CSI）用チャネルの初期化処理を行います。

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

[所属]

r_cg_serial.c

[指定形式]

```
void    R_CSImn_Create ( 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_csi*mn*_interrupt

CSI 通信完了割り込み INTCSI*mn* の発生に伴う処理を行います。

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

[所属]

r_cg_serial_user.c

[指定形式]

```
__interrupt static  void    r_csimn_interrupt ( 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_CSI*mn*_Start

CSI 通信を待機状態にします。

[所属]

r_cg_serial.c

[指定形式]

```
void    R_CSImn_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_CSI*mn*_Stop

CSI 通信を終了します。

[所属]

r_cg_serial.c

[指定形式]

```
void    R_CSImn_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_CSI*mn*_Send

データの CSI 送信を開始します。

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

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

[所属]

r_cg_serial.c

[指定形式]

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

備考 *m* はユニット番号を，*n* はチャネル番号を意味します。

[引数]

I/O	引数	説明
I	`uint8_t * 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_CSI*mn*_Receive

データの CSI 受信を開始します。

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

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

[所属]

r_cg_serial.c

[指定形式]

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

備考 *m* はユニット番号を，*n* はチャネル番号を意味します。

[引数]

I/O	引数	説明
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_CSI*mn*_Send_Receive

データの CSI 送受信を開始します。

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

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

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

[所属]

r_cg_serial.c

[指定形式]

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

備考 *m* はユニット番号を，*n* はチャネル番号を意味します。

[引数]

I/O	引数	説明
I	`uint8_t * 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	`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

[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_csi*mn*_callback_sendend

CSI 送信完了割り込み INTCSI*mn* の発生に伴う処理を行います。

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

[所属]

r_cg_serial_user.c

[指定形式]

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

備考 *m* はユニット番号を，*n* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_csi*mn*_callback_receiveend

CSI 受信完了割り込み INTCSI*mn* の発生に伴う処理を行います。

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

[所属]

r_cg_serial_user.c

[指定形式]

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

備考 *m* はユニット番号を，*n* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_csi*mn*_callback_error

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

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

[所属]

r_cg_serial_user.c

[指定形式]

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

備考 *m* はユニット番号を，*n* はチャネル番号を意味します。

[引数]

I/O	引数	説明
O	`uint8_t err_type;`	CSI 受信エラー割り込みの発生要因 00000*xx*1B：オーバラン・エラー

[戻り値]

なし

R_IIC*mn*_Create

シリアル・インタフェース（簡易 IIC）用チャネルの初期化処理を行います。

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

［所属］

r_cg_serial.c

［指定形式］

```
void    R_IIC*mn*_Create ( 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_iic*mn*_interrupt

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

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

[所属]

r_cg_serial_user.c

[指定形式]

```
__interrupt static  void    r_iicmn_interrupt ( 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_IIC*mn*_StartCondition

スタート・コンディションを発生します。

備考　本 API 関数は，R_IICmn_Master_Send，および R_IICmn_Master_Receive の内部関数として位置づけられているため，通常，ユーザの処理プログラムから呼び出す必要はありません。

[所属]

r_cg_serial.c

[指定形式]

```
void    R_IICmn_StartCondition ( 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_IIC*mn*_StopCondition

ストップ・コンディションを発生します。

[所属]

r_cg_serial.c

[指定形式]

```
void    R_IICmn_StopCondition ( 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_IIC*mn*_Stop

簡易 IIC 通信を終了します。

[所属]

r_cg_serial.c

[指定形式]

```
void    R_IICmn_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_IIC*mn*_Master_Send

簡易 IIC マスタ送信を開始します。

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

[所属]

r_cg_serial.c

[指定形式]

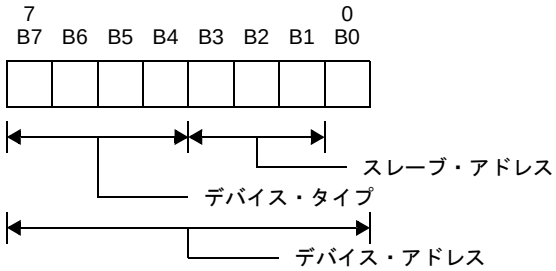
```
#include    "r_cg_macrodriver.h"
void    R_IICmn_Master_Send ( uint8_t adr, uint8_t * tx_buf, uint16_t tx_num );
```

備考 *m* はユニット番号を，*n* はチャネル番号を意味します。

[引数]

I/O	引数	説明
I	`uint8_t adr;`	デバイス・アドレス
I	`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

備考 以下に，デバイス・アドレス *adr* の指定形式を示します。



[戻り値]

なし

R_IIC*mn*_Master_Receive

簡易 IIC マスタ受信を開始します。

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

[所属]

r_cg_serial.c

[指定形式]

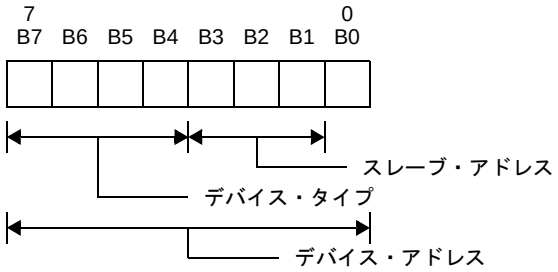
```
#include    "r_cg_macrodriver.h"
void    R_IICmn_Master_Receive ( uint8_t adr, uint8_t * rx_buf, uint16_t rx_num );
```

備考 *m* はユニット番号を，*n* はチャネル番号を意味します。

[引数]

I/O	引数	説明
I	`uint8_t adr;`	デバイス・アドレス
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

備考 以下に，デバイス・アドレス *adr* の指定形式を示します。



[戻り値]

なし

r_iic*mn*_callback_master_sendend

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

備考 本 API 関数は，簡易 IIC マスタ送信完了割り込み INTIIC*mn* に対応した割り込み処理 r_iicmn_interrupt のコールバック・ルーチン（R_IICmn_Master_Send の引数 *tx_num* で指定された数のデータ送信が完了した際の処理）として呼び出されます。

[所属]

r_cg_serial_user.c

[指定形式]

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

備考 *m* はユニット番号を，*n* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_iic*mn*_callback_master_receiveend

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

備考 本 API 関数は，簡易 IIC マスタ受信完了割り込み INTIIC*mn* に対応した割り込み処理 r_iicmn_interrupt のコールバック・ルーチン（R_IICmn_Master_Receive の引数 *rx_num* で指定された数のデータ送信が完了した際の処理）として呼び出されます。

[所属]

r_cg_serial_user.c

[指定形式]

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

備考 *m* はユニット番号を，*n* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_iic*mn*_callback_master_error

パリティ・エラー（ACK エラー）の検出に伴う処理を行います。

［所属］

r_cg_serial_user.c

［指定形式］

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

備考 *m* はユニット番号を，*n* はチャネル番号を意味します。

［引数］

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

［戻り値］

なし

R_DALI*n*_Create

シリアル・インタフェース（DALI）用チャネルの初期化処理を行います。

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

[所属]

r_cg_serial.c

[指定形式]

```
void    R_DALIn_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_dali*n*_interrupt_send

DALI 送信完了割り込み INTSTDL*n* の発生に伴う処理を行います。

備考 本 API 関数は，DALI 送信完了割り込み INTSTDL*n* に対応した割り込み処理として呼び出されます。

[所属]

r_cg_serial_user.c

[指定形式]

```
__interrupt static  void    r_dali*n*_interrupt_send ( void );
```

備考 *n* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_dali*n*_interrupt_receive

DALI 受信完了割り込み INTSRDL*n* の発生に伴う処理を行います。

備考 本 API 関数は，DALI 受信完了割り込み INTSRDL*n* に対応した割り込み処理として呼び出されます。

[所属]

r_cg_serial_user.c

[指定形式]

```
__interrupt static  void    r_dalin_interrupt_receive ( void );
```

備考 *n* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_dali*n*_interrupt_error

DALI 受信エラー割り込み INTSREDL*n* の発生に伴う処理を行います。

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

[所属]

r_cg_serial_user.c

[指定形式]

```
static  void    r_dali*n*_interrupt_error ( void );
```

備考 *n* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_DALI*n*_Start

DALI 通信を待機状態にします。

[所属]

r_cg_serial.c

[指定形式]

```
void    R_DALIn_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_DALI*n*_Stop

DALI 通信を終了します。

[所属]

r_cg_serial.c

[指定形式]

```
void    R_DALIn_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_DALI*n*_Send

データの DALI 送信を開始します。

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

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

[所属]

r_cg_serial.c

[指定形式]

```
#include    "r_cg_macrodriver.h"
MD_STATUS   R_DALIn_Send ( 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	`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_DALI*n*_Receive

データの DALI 受信を開始します。

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

2. 実際の DALI 受信は，本 API 関数の呼び出し後，R_DALIn_Start を呼び出すことにより開始されます。

[所属]

r_cg_serial.c

[指定形式]

```
#include    "r_cg_macrodriver.h"
MD_STATUS   R_DALIn_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]

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

r_dali*n*_callback_sendend

DALI 送信完了割り込み INTSTDL*n* の発生に伴う処理を行います。

備考　本 API 関数は，DALI 送信完了割り込み INTSTDL*n* に対応した割り込み処理 r_dalin_interrupt_send のコールバック・ルーチン（R_DALIn_Send の引数 *tx_num* で指定された数のデータ送信が完了した際の処理）として呼び出されます。

[所属]

r_cg_serial_user.c

[指定形式]

```
static void    r_dalin_callback_sendend ( void );
```

備考　*n* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_dali*n*_callback_receiveend

DALI 受信完了割り込み INTSRDL*n* の発生に伴う処理を行います。

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

[所属]

r_cg_serial_user.c

[指定形式]

```
static void    r_dalin_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_dali*n*_callback_error

DALI 受信エラー割り込み INTSREDL*n* の発生に伴う処理を行います。

備考 本 API 関数は，DALI 受信エラー割り込み INTSREDL*n* に対応した割り込み処理 r_dalin_interrupt_error のコールバック・ルーチンとして呼び出されます。

[所属]

r_cg_serial_user.c

[指定形式]

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

備考 *n* はチャネル番号を意味します。

[引数]

I/O	引数	説明
O	uint8_t *err_type*;	DALI 受信エラー割り込みの発生要因 00000*xx*1B：オーバラン・エラー 00000*x*1*x*B：パリティ・エラー 000001*xx*B：フレーミング・エラー

[戻り値]

なし

r_dali*n*_callback_softwareoverrun

オーバラン・エラーの検出に伴う処理を行います。

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

[所属]

r_cg_serial_user.c

[指定形式]

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

備考 *n* はチャネル番号を意味します。

[引数]

I/O	引数	説明
O	`uint16_t  rx_data;`	受信したデータ（R_DALIn_Receive の引数 *rx_num* で指定された数以上に受信したデータ）

[戻り値]

なし

R_IICA*n*_Create

シリアル・インタフェース（IICA）の初期化処理を行います。

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

［所属］

r_cg_serial.c

［指定形式］

```
void    R_IICAn_Create ( void );
```

備考　*n* はチャネル番号を意味します。

［引数］

なし

［戻り値］

なし

R_IICA*n*_Create_UserInit

シリアル・インタフェース（IICA）に関するユーザ独自の初期化処理を行います。

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

[所属]

r_cg_serial_user.c

[指定形式]

```
void    R_IICAn_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_iica*n*_interrupt

IICA 通信完了割り込み INTIICA*n* の発生に伴う処理を行います。

備考 本 API 関数は，IICA 通信完了割り込み INTIICA*n* に対応した割り込み処理として呼び出されます。

[所属]

r_cg_serial_user.c

[指定形式]

```
__interrupt static  void    r_iican_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_IICA*n*_StopCondition

ストップ・コンディションを発生します。

[所属]

r_cg_serial.c

[指定形式]

```
void    R_IICAn_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_IICA*n*_Stop

IICA 通信を終了します。

[所属]

r_cg_serial.c

[指定形式]

```
void    R_IICAn_Stop ( void );
```

備考　*n* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_IICA*n*_Set_PowerOff

シリアル・インタフェース（IICA）に対するクロック供給を停止します。

備考 本 API 関数の呼び出しにより，シリアル・インタフェース（IICA）はリセット状態へと移行します。
このため，本 API 関数の呼び出し後，制御レジスタ（IICA コントロール・レジスタ *n*：IICCTL*n* など）への書き込みは無視されます。

[所属]

r_cg_serial.c

[指定形式]

```
void    R_IICAn_Set_PowerOff ( 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_IICA*n*_Master_Send

IICA マスタ送信を開始します。

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

[所属]

r_cg_serial.c

[指定形式]

```
#include    "r_cg_macrodriver.h"
MD_STATUS   R_IICAn_Master_Send ( uint8_t adr, uint8_t * tx_buf, uint16_t tx_num, uint8_t wait
);
```

備考 *n* はチャネル番号を意味します。

[引数]

I/O	引数	説明
I	uint8_t *adr*;	スレーブ・アドレス
I	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
I	uint8_t *wait*;	スタート・コンディションのセットアップ時間

[戻り値]

マクロ	説明
MD_OK	正常終了
MD_ERROR1	バス通信状態
MD_ERROR2	バス未解放状態

R_IICA*n*_Master_Receive

IICA マスタ受信を開始します。

備考 本 API 関数では，1 バイト単位の IICA マスタ受信を引数 *rx_num* で指定された回数だけ繰り返し行い，引数 *rx_buf* で指定されたバッファに格納します。

[所属]

r_cg_serial.c

[指定形式]

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

備考 *n* はチャネル番号を意味します。

[引数]

I/O	引数	説明
I	uint8_t *adr*;	スレーブ・アドレス
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
I	uint8_t *wait*;	スタート・コンディションのセットアップ時間

[戻り値]

マクロ	説明
MD_OK	正常終了
MD_ERROR1	バス通信状態
MD_ERROR2	バス未解放状態

r_iica*n*_callback_master_sendend

IICA マスタ送信完了割り込み INTIICA*n* の発生に伴う処理を行います。

備考 本 API 関数は，IICA マスタ送信完了割り込み INTIICA*n* に対応した割り込み処理 r_iican_interrupt のコールバック・ルーチンとして呼び出されます。

[所属]

r_cg_serial_user.c

[指定形式]

```
static  void    r_iica*n*_callback_master_sendend ( void );
```

備考 *n* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_iica*n*_callback_master_receiveend

IICA マスタ受信完了割り込み INTIICA*n* の発生に伴う処理を行います。

備考 本 API 関数は，IICA マスタ受信完了割り込み INTIICA*n* に対応した割り込み処理 r_iican_interrupt のコールバック・ルーチンとして呼び出されます。

[所属]

r_cg_serial_user.c

[指定形式]

```
static  void    r_iica n_callback_master_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_iica*n*_callback_master_error

IICA マスタ通信エラーの検出に伴う処理を行います。

[所属]

r_cg_serial_user.c

[指定形式]

```
#include    "r_cg_macrodriver.h"
static  void    r_iica*n*_callback_master_error ( MD_STATUS *flag*  );
```

備考 *n* はチャネル番号を意味します。

[引数]

I/O	引数	説明
I	MD_STATUS *flag*;	通信エラーの発生要因 MD_SPT : ストップ・コンディションの検出 MD_NACK : アクノリッジの未検出

[戻り値]

なし

R_IICA*n*_Slave_Send

IICA スレーブ送信を開始します。

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

[所属]

r_cg_serial.c

[指定形式]

```
#include    "r_cg_macrodriver.h"
void    R_IICAn_Slave_Send ( 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	`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_IICA*n*_Slave_Receive

IICA スレーブ受信を開始します。

備考 本 API 関数では，1 バイト単位の IICA スレーブ受信を引数 *rx_num* で指定された回数だけ繰り返し行い，引数 *rx_buf* で指定されたバッファに格納します。

[所属]

r_cg_serial.c

[指定形式]

```
#include    "r_cg_macrodriver.h"
void    R_IICAn_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_iica*n*_callback_slave_sendend

IICA スレーブ送信完了割り込み INTIICA*n* の発生に伴う処理を行います。

備考 本 API 関数は，IICA スレーブ送信完了割り込み INTIICA*n* に対応した割り込み処理 r_iican_interrupt のコールバック・ルーチンとして呼び出されます。

[所属]

r_cg_serial_user.c

[指定形式]

```
static  void    r_iican_callback_slave_sendend ( void );
```

備考 *n* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

r_iica*n*_callback_slave_receiveend

IICA スレーブ受信完了割り込み INTIICA*n* の発生に伴う処理を行います。

備考 本 API 関数は，IICA スレーブ受信完了割り込み INTIICA*n* に対応した割り込み処理 r_iican_interrupt のコールバック・ルーチンとして呼び出されます。

[所属]

r_cg_serial_user.c

[指定形式]

```
static  void    r_iica*n*_callback_slave_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_iica*n*_callback_slave_error

IICA スレーブ通信エラーの検出に伴う処理を行います。

[所属]

r_cg_serial_user.c

[指定形式]

```
#include    "r_cg_macrodriver.h"
static  void    r_iica*n*_callback_slave_error ( MD_STATUS *flag*  );
```

備考 *n* はチャネル番号を意味します。

[引数]

I/O	引数	説明
I	MD_STATUS *flag*;	通信エラーの発生要因 MD_ERROR： アドレス不一致の検出 MD_NACK： アクノリッジの未検出

[戻り値]

なし

r_iica*n*_callback_getstopcondition

ストップ・コンディションの検出に伴う処理を行います。

[所属]

r_cg_serial_user.c

[指定形式]

```
static  void    r_iica*n*_callback_getstopcondition ( void );
```

備考 *n* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

3. 3. 5 A/D コンバータ

以下に，Applilet3 が A/D コンバータ用として出力する API 関数の一覧を示します。

表 3—6 A/D コンバータ用 API 関数

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

R_ADC_Create

A/D コンバータの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。

[所属]

r_cg_adc.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_ADC_Create_UserInit

A/D コンバータに関するユーザ独自の初期化処理を行います。

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

[所属]

r_cg_adc_user.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

r_adc_interrupt

A/D 変換終了割り込み INTAD の発生伴う処理を行います。

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

[所属]

r_cg_adc_user.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_ADC_Set_OperationOn

電圧コンパレータを動作許可状態に設定します。

備考 1. 電圧コンパレータが動作停止状態から動作許可状態へと移行した際，約 1μ 秒の安定時間を必要とします。したがって，本 API 関数と R_ADC_Start の間には，約 1μ 秒の時間を空ける必要があります。

2. ［A/D コンバータ］の［コンパレータ動作設定］エリアで“許可”を選択した場合，電圧コンパレータは“常時 ON”となるため，本 API 関数の呼び出しは不要となります。

［所属］

r_cg_adc.c

［指定形式］

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

［引数］

なし

［戻り値］

なし

R_ADC_Set_OperationOff

電圧コンパレータを動作停止状態に設定します。

[所属]

r_cg_adc.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_ADC_Start

A/D 変換を開始します。

備考 電圧コンパレータが動作停止状態から動作許可状態へと移行した際，約 1μ 秒の安定時間を必要とします。
したがって，R_ADC_Set_OperationOn と本 API 関数の間には，約 1μ 秒の時間を空ける必要があります。

[所属]

r_cg_adc.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_ADC_Stop

A/D 変換を終了します。

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

[所属]

r_cg_adc.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_ADC_Set_PowerOff

A/D コンバータに対するクロック供給を停止します。

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

[所属]

r_cg_adc.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_ADC_Set_ADChannel

A/D 変換するアナログ電圧の入力端子を設定します。

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

[所属]

r_cg_adc.c

[指定形式]

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

[引数]

I/O	引数	説明
I	ad_channel_t *channel*;	アナログ電圧の入力端子 ADCHANNEL*n*： 入力端子

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

[戻り値]

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

R_ADC_Set_SnoozeOn

STOP モードから SNOOZE モードへの切り替えを許可します。

[所属]

r_cg_adc.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_ADC_Set_SnoozeOff

STOP モードから SNOOZE モードへの切り替えを禁止します。

[所属]

r_cg_adc.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_ADC_Set_TestChannel

A/D コンバータの動作モードを設定します。

[所属]

r_cg_adc.c

[指定形式]

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

[引数]

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

[戻り値]

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

R_ADC_Get_Result

A/D 変換結果（10 ビット）を読み出します。

[所属]

r_cg_adc.c

[指定形式]

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

[引数]

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

[戻り値]

なし

R_ADC_Get_Result_8bit

A/D 変換結果（8 ビット：10 ビット分解能の上位 8 ビット）を読み出します。

[所属]

r_cg_adc.c

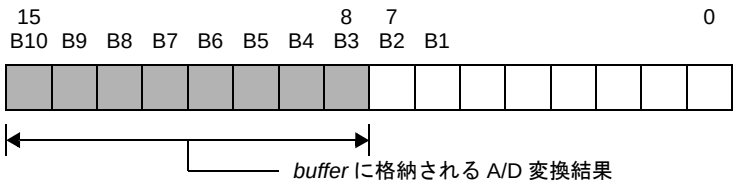
[指定形式]

```
#include    "r_cg_macrodriver.h"
void    R_ADC_Get_Result_8bit ( uint8_t * buffer );
```

[引数]

I/O	引数	説明
O	uint8_t * *buffer*;	読み出した A/D 変換結果（8 ビット：10 ビット分解能の上位 8 ビット）を格納する領域へのポインタ

備考 以下に，*buffer* に格納される A/D 変換結果を示します。



[戻り値]

なし

3. 3. 6 D/A コンバータ

以下に，Applilet3 が D/A コンバータ用として出力する API 関数の一覧を示します。

表 3—7 D/A コンバータ用 API 関数

API 関数名	機能概要
R_DAC_Create	D/A コンバータの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_DAC_Create_UserInit	D/A コンバータに関するユーザ独自の初期化処理を行います。
R_DACn_Start	D/A 変換を開始します。
R_DACn_Stop	D/A 変換を終了します。
R_DAC_Set_PowerOff	D/A コンバータに対するクロック供給を停止します。
R_DACn_Set_ConversionValue	ANO*n* 端子に出力するアナログ電圧値を設定します。

R_DAC_Create

D/A コンバータの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。

[所属]

r_cg_dac.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_DAC_Create_UserInit

D/A コンバータに関するユーザ独自の初期化処理を行います。

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

[所属]

r_cg_dac_user.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_DAC*n*_Start

D/A 変換を開始します。

[所属]

r_cg_dac.c

[指定形式]

```
void    R_DACn_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_DAC*n*_Stop

D/A 変換を終了します。

[所属]

r_cg_dac.c

[指定形式]

```
void    R_DACn_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_DAC_Set_PowerOff

D/A コンバータに対するクロック供給を停止します。

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

[所属]

r_cg_dac.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_DAC*n*_Set_ConversionValue

ANO*n* 端子に出力するアナログ電圧値を設定します。

[所属]

r_cg_dac.c

[指定形式]

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

備考 *n* は，チャネル番号を意味します。

[引数]

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

[戻り値]

なし

3. 3. 7 タ イ マ

以下に，Applilet3 がタイマ用として出力する API 関数の一覧を示します。

表 3—8 タイマ用 API 関数

API 関数名	機能概要
R_TAUm_Create	タイマ・アレイ・ユニットの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_TAUm_Create_UserInit	タイマ・アレイ・ユニットに関するユーザ独自の初期化処理を行います。
r_taum_channeln_interrupt	タイマ割り込み INTTM*mn* の発生に伴う処理を行います。
r_taum_channeln_higher8bits_interrupt	タイマ割り込み INTTM*mn*H の発生に伴う処理を行います。
R_TAUm_Channeln_Start	チャネル *n* のカウントを開始します。
R_TAUm_Channeln_Higher8bits_Start	チャネル 1，またはチャンネル 3 のカウント（上位 8 ビット）を開始します。
R_TAUm_Channeln_Lower8bits_Start	チャネル 1，またはチャンネル 3 のカウント（下位 8 ビット）を開始します。
R_TAUm_Channeln_Stop	チャネル *n* のカウントを終了します。
R_TAUm_Channeln_Higher8bits_Stop	チャネル 1，またはチャンネル 3 のカウント（上位 8 ビット）を終了します。
R_TAUm_Channeln_Lower8bits_Stop	チャネル 1，またはチャンネル 3 のカウント（下位 8 ビット）を終了します。
R_TAUm_Set_PowerOff	タイマ・アレイ・ユニットに対するクロック供給を停止します。
R_TAUm_Channeln_Get_PulseWidth	TI*mn* 端子に対する入力信号（入力パルス）のパルス間隔，またはハイ／ロウ・レベルの測定幅を獲得します。
R_TAUm_Channeln_Set_SoftwareTriggerOn	ワンショット・パルス出力のためのトリガ（ソフトウエア・トリガ）を発生させます。
R_TMR_RDn_Create	16 ビット・タイマ RD*n* の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_TMR_RDn_Create_UserInit	16 ビット・タイマ RD*n* に関するユーザ独自の初期化処理を行います。
r_tmr_rdn_interrupt	タイマ割り込みの発生に伴う処理を行います。
R_TMR_RDn_Start	16 ビット・タイマ RD*n* のカウント処理を開始します。
R_TMR_RDn_Stop	16 ビット・タイマ RD*n* のカウント処理を終了します。
R_TMR_RDn_Set_PowerOff	16 ビット・タイマ RD*n* に対するクロック供給を停止します。
R_TMR_RDn_ForcedOutput_Start	16 ビット・タイマ RD*n* のパルス出力強制遮断処理を開始します。
R_TMR_RDn_ForcedOutput_Stop	16 ビット・タイマ RD*n* のパルス出力強制遮断処理を終了します。
R_TMR_RDn_Get_PulseWidth	16 ビット・タイマ RD*n* のパルス幅を読み出します。
R_TMR_RG0_Create	16 ビット・タイマ RG0 の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。

API 関数名	機能概要
R_TMR_RG0_Create_UserInit	16 ビット・タイマ RG0 に関するユーザ独自の初期化処理を行います。
r_tmr_rg0_interrupt	タイマ割り込みの発生に伴う処理を行います。
R_TMR_RG0_Start	16 ビット・タイマ RG0 のカウント処理を開始します。
R_TMR_RG0_Stop	16 ビット・タイマ RG0 のカウント処理を終了します。
R_TMR_RG0_Set_PowerOff	16 ビット・タイマ RG0 に対するクロック供給を停止します。
R_TMR_RG0_Get_PulseWidth	16 ビット・タイマ RG0 のパルス幅を読み出します。
R_TMR_RJ0_Create	16 ビット・タイマ RJ0 の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_TMR_RJ0_Create_UserInit	16 ビット・タイマ RJ0 に関するユーザ独自の初期化処理を行います。
r_tmr_rj0_interrupt	タイマ割り込みの発生に伴う処理を行います。
R_TMR_RJ0_Start	16 ビット・タイマ RJ0 のカウント処理を開始します。
R_TMR_RJ0_Stop	16 ビット・タイマ RJ0 のカウント処理を終了します。
R_TMR_RJ0_Set_PowerOff	16 ビット・タイマ RJ0 に対するクロック供給を停止します。
R_TMR_RJ0_Get_PulseWidth	16 ビット・タイマ RJ0 のパルス幅を読み出します。
R_TMR_KB_Create	16 ビット・タイマ KB*m* の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_TMR_KBm_Create_UserInit	16 ビット・タイマ KB*m* に関するユーザ独自の初期化処理を行います。
r_tmr_kbm_interrupt	タイマ割り込みの発生に伴う処理を行います。
R_TMR_KBm_Start	16 ビット・タイマ KB*m* のカウント処理を開始します。
R_TMR_KBm_Stop	16 ビット・タイマ KB*m* のカウント処理を終了します。
R_TMR_KBm_Set_PowerOff	16 ビット・タイマ KB*m* に対するクロック供給を停止します。
R_TMR_KBm_ForcedOutput_Start	強制出力停止機能に使用するトリガ信号の入力を許可します。
R_TMR_KBm_ForcedOutput_Stop	強制出力停止機能に使用するトリガ信号の入力を禁止します。
R_TMR_KC0_Create	16 ビット・タイマ KC0 の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_TMR_KC0_Create_UserInit	16 ビット・タイマ KC0 に関するユーザ独自の初期化処理を行います。
r_tmr_kc0_interrupt	タイマ割り込みの発生に伴う処理を行います。
R_TMR_KC0_Start	16 ビット・タイマ KC0 のカウント処理を開始します。
R_TMR_KC0_Stop	16 ビット・タイマ KC0 のカウント処理を終了します。
R_TMR_KC0_Set_PowerOff	16 ビット・タイマ KC0 に対するクロック供給を停止します。

R_TAU*m*_Create

タイマ・アレイ・ユニットの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。

[所属]

r_cg_timer.c

[指定形式]

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

備考　*m* は，ユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_TAU*m*_Create_UserInit

タイマ・アレイ・ユニットに関するユーザ独自の初期化処理を行います。

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

[所属]

r_cg_timer_user.c

[指定形式]

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

備考　*m* は，ユニット番号を意味します。

[引数]

なし

[戻り値]

なし

r_tau*m*_channel*n*_interrupt

タイマ割り込み INTTM*mn* の発生に伴う処理を行います。

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

[所属]

r_cg_timer_user.c

[指定形式]

```
__interrupt static  void    r_taum_channeln_interrupt ( 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_tau*m*_channel*n*_higher8bits_interrupt

タイマ割り込み INTTM*mn*H の発生に伴う処理を行います。

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

[所属]

r_cg_timer_user.c

[指定形式]

```
__interrupt static void    r_tau*m*_channel*n*_higher8bits_interrupt ( 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_TAU*m*_Channel*n*_Start

チャネル *n* のカウントを開始します。

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

[所属]

r_cg_timer.c

[指定形式]

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

備考 *m* はユニット番号を，*n* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_TAU*m*_Channel*n*_Higher8bits_Start

チャネル 1，またはチャンネル 3 のカウント（上位 8 ビット）を開始します。

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

[所属]

r_cg_timer.c

[指定形式]

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

備考 *m* はユニット番号を，*n* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_TAU*m*_Channel*n*_Lower8bits_Start

チャネル 1，またはチャンネル 3 のカウント（下位 8 ビット）を開始します。

備考 1. 本 API 関数の呼び出しは，タイマ・アレイ・ユニットを 8 ビット・タイマとして使用している場合に限られます。

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

[所属]

r_cg_timer.c

[指定形式]

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

備考 *m* はユニット番号を，*n* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_TAU*m*_Channel*n*_Stop

チャネル *n* のカウントを終了します。

[所属]

r_cg_timer.c

[指定形式]

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

備考　*m* はユニット番号を，*n* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_TAU*m*_Channel*n*_Higher8bits_Stop

チャネル 1，またはチャンネル 3 のカウント（上位 8 ビット）を終了します。

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

[所属]

r_cg_timer.c

[指定形式]

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

備考 *m* はユニット番号を，*n* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_TAU*m*_Channel*n*_Lower8bits_Stop

チャネル 1，またはチャンネル 3 のカウント（下位 8 ビット）を終了します。

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

[所属]

r_cg_timer.c

[指定形式]

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

備考 *m* はユニット番号を，*n* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_TAU*m*_Set_PowerOff

タイマ・アレイ・ユニットに対するクロック供給を停止します。

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

[所属]

r_cg_timer.c

[指定形式]

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

備考 *m* は，ユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_TAU*m*_Channel*n*_Get_PulseWidth

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

［所属］

r_cg_timer.c

［指定形式］

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

備考 *m* はユニット番号を，*n* はチャネル番号を意味します。

［引数］

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

［戻り値］

なし

R_TAU*m*_Channel*n*_Set_SoftwareTriggerOn

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

[所属]

r_cg_timer.c

[指定形式]

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

備考 *m* はユニット番号を，*n* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_TMR_RD*n*_Create

16 ビット・タイマ RD*n* の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。

［所属］

r_cg_timer.c

［指定形式］

```
void    R_TMR_RDn_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_TMR_RD*n*_Create_UserInit

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

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

[所属]

r_cg_timer_user.c

[指定形式]

```
void    R_TMR_RDn_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_tmr_rd*n*_interrupt

タイマ割り込みの発生に伴う処理を行います。

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

[所属]

r_cg_timer_user.c

[指定形式]

```
__interrupt static  void    r_tmr_rdn_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_TMR_RD*n*_Start

16 ビット・タイマ RD*n* のカウント処理を開始します。

[所属]

r_cg_timer.c

[指定形式]

```
void    R_TMR_RDn_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_TMR_RD*n*_Stop

16 ビット・タイマ RD*n* のカウント処理を終了します。

[所属]

r_cg_timer.c

[指定形式]

```
void    R_TMR_RDn_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_TMR_RD*n*_Set_PowerOff

16 ビット・タイマ RD*n* に対するクロック供給を停止します。

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

[所属]

r_cg_timer.c

[指定形式]

```
void    R_TMR_RDn_Set_PowerOff ( 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_TMR_RD*n*_ForcedOutput_Start

16 ビット・タイマ RD*n* のパルス出力強制遮断処理を開始します。

[所属]

r_cg_timer.c

[指定形式]

```
void    R_TMR_RDn_ForcedOutput_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_TMR_RD*n*_ForcedOutput_Stop

16 ビット・タイマ RD*n* のパルス出力強制遮断処理を終了します。

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

[所属]

r_cg_timer.c

[指定形式]

```
void    R_TMR_RDn_ForcedOutput_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_TMR_RD*n*_Get_PulseWidth

16 ビット・タイマ RD*n* のパルス幅を読み出します。

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

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

[所属]

r_cg_timer.c

[指定形式]

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

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

[戻り値]

マクロ	説明
MD_OK	正常終了

R_TMR_RG0_Create

16 ビット・タイマ RG0 の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。

[所属]

r_cg_timer.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_TMR_RG0_Create_UserInit

16 ビット・タイマ RG0 に関するユーザ独自の初期化処理を行います。

備考　本 API 関数は，R_TMR_RG0_Create のコールバック・ルーチンとして呼び出されます。

[所属]

r_cg_timer_user.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

r_tmr_rg0_interrupt

タイマ割り込みの発生に伴う処理を行います。

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

[所属]

r_cg_timer_user.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_TMR_RG0_Start

16 ビット・タイマ RG0 のカウント処理を開始します。

[所属]

r_cg_timer.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_TMR_RG0_Stop

16 ビット・タイマ RG0 のカウント処理を終了します。

[所属]

r_cg_timer.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_TMR_RG0_Set_PowerOff

16 ビット・タイマ RG0 に対するクロック供給を停止します。

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

[所属]

r_cg_timer.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_TMR_RG0_Get_PulseWidth

16 ビット・タイマ RG0 のパルス幅を読み出します。

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

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

[所属]

r_cg_timer.c

[指定形式]

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

[引数]

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

[戻り値]

マクロ	説明
MD_OK	正常終了

R_TMR_RJ0_Create

16 ビット・タイマ RJ0 の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。

[所属]

r_cg_timer.c

[指定形式]

```
void    R_TMR_RJ0_Create ( void );
```

[引数]

なし

[戻り値]

なし

R_TMR_RJ0_Create_UserInit

16 ビット・タイマ RJ0 に関するユーザ独自の初期化処理を行います。

備考　本 API 関数は，R_TMR_RJ0_Create のコールバック・ルーチンとして呼び出されます。

[所属]

r_cg_timer_user.c

[指定形式]

```
void    R_TMR_RJ0_Create_UserInit ( void );
```

[引数]

なし

[戻り値]

なし

r_tmr_rj0_interrupt

タイマ割り込みの発生に伴う処理を行います。

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

[所属]

r_cg_timer_user.c

[指定形式]

```
__interrupt static  void    r_tmr_rj0_interrupt ( void );
```

[引数]

なし

[戻り値]

なし

R_TMR_RJ0_Start

16 ビット・タイマ RJ0 のカウント処理を開始します。

[所属]

r_cg_timer.c

[指定形式]

```
void    R_TMR_RJ0_Start ( void );
```

[引数]

なし

[戻り値]

なし

R_TMR_RJ0_Stop

16 ビット・タイマ RJ0 のカウント処理を終了します。

[所属]

r_cg_timer.c

[指定形式]

```
void    R_TMR_RJ0_Stop ( void );
```

[引数]

なし

[戻り値]

なし

R_TMR_RJ0_Set_PowerOff

16 ビット・タイマ RJ0 に対するクロック供給を停止します。

備考 本 API 関数の呼び出しにより，16 ビット・タイマ RJ0 はリセット状態へと移行します。
このため，本 API 関数の呼び出し後，制御レジスタへの書き込みは無視されます。

[所属]

r_cg_timer.c

[指定形式]

```
void    R_TMR_RJ0_Set_PowerOff ( void );
```

[引数]

なし

[戻り値]

なし

R_TMR_RJ0_Get_PulseWidth

16 ビット・タイマ RJ0 のパルス幅を読み出します。

備考 1. 本 API 関数の呼び出しは，16 ビット・タイマ RJ0 をパルス幅測定モード／パルス周期測定モードで使用している場合に限られます。

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

[所属]

r_cg_timer.c

[指定形式]

```
#include    "r_cg_macrodriver.h"
void    R_TMR_RJ0_Get_PulseWidth ( uint32_t * active_width );
```

[引数]

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

[戻り値]

なし

R_TMR_KB_Create

16 ビット・タイマ KB*m* の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。

[所属]

r_cg_timer.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_TMR_KB*m*_Create_UserInit

16 ビット・タイマ KB*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_TMR_KB_Create のコールバック・ルーチンとして呼び出されます。

[所属]

r_cg_timer_user.c

[指定形式]

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

備考　*m* は，ユニット番号を意味します。

[引数]

なし

[戻り値]

なし

r_tmr_kb*m*_interrupt

タイマ割り込みの発生に伴う処理を行います。

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

[所属]

r_cg_timer_user.c

[指定形式]

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

備考 *m* は，ユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_TMR_KB*m*_Start

16 ビット・タイマ KB*m* のカウント処理を開始します。

[所属]

r_cg_timer.c

[指定形式]

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

備考　*m* は，ユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_TMR_KB*m*_Stop

16 ビット・タイマ KB*m* のカウント処理を終了します。

[所属]

r_cg_timer.c

[指定形式]

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

備考 *m* は，ユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_TMR_KB*m*_Set_PowerOff

16 ビット・タイマ KB*m* に対するクロック供給を停止します。

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

[所属]

r_cg_timer.c

[指定形式]

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

備考　*m* は，ユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_TMR_KB*m*_ForcedOutput_Start

強制出力停止機能に使用するトリガ信号の入力を許可します。

[所属]

r_cg_timer.c

[指定形式]

```
void    R_TMR_KBm_ForcedOutput_Start ( void );
```

備考　*m* は，ユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_TMR_KB*m*_ForcedOutput_Stop

強制出力停止機能に使用するトリガ信号の入力を禁止します。

[所属]

r_cg_timer.c

[指定形式]

```
void    R_TMR_KBm_ForcedOutput_Stop ( void );
```

備考　*m* は，ユニット番号を意味します。

[引数]

なし

[戻り値]

なし

R_TMR_KC0_Create

16 ビット・タイマ KC0 の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。

[所属]

r_cg_timer.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_TMR_KC0_Create_UserInit

16 ビット・タイマ KC0 に関するユーザ独自の初期化処理を行います。

備考　本 API 関数は，R_TMR_KC0_Create のコールバック・ルーチンとして呼び出されます。

[所属]

r_cg_timer_user.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

r_tmr_kc0_interrupt

タイマ割り込みの発生に伴う処理を行います。

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

[所属]

r_cg_timer_user.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_TMR_KC0_Start

16 ビット・タイマ KC0 のカウント処理を開始します。

[所属]

r_cg_timer.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_TMR_KC0_Stop

16 ビット・タイマ KC0 のカウント処理を終了します。

[所属]

r_cg_timer.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_TMR_KC0_Set_PowerOff

16 ビット・タイマ KC0 に対するクロック供給を停止します。

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

[所属]

r_cg_timer.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

3. 3. 8　ウォッチドッグ・タイマ

以下に，Applilet3 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—9　ウォッチドッグ・タイマ用 API 関数

API 関数名	機能概要
R_WDT_Create	ウォッチドッグ・タイマの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_WDT_Create_UserInit	ウォッチドッグ・タイマに関するユーザ独自の初期化処理を行います。
r_wdt_interrupt	ウォッチドッグ・タイマのインターバル割り込み INTWDTI の発生に伴い処理を行います。
R_WDT_Restart	ウォッチドッグ・タイマのカウンタをクリアしたのち，カウント処理を再開します。

R_WDT_Create

ウォッチドッグ・タイマの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。

[所属]

r_cg_wdt.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_WDT_Create_UserInit

ウォッチドッグ・タイマに関するユーザ独自の初期化処理を行います。

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

[所属]

r_cg_wdt_user.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

r_wdt_interrupt

ウォッチドッグ・タイマのインターバル割り込み INTWDTI の発生に伴う処理を行います。

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

[所属]

r_cg_wdt_user.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_WDT_Restart

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

［所属］

r_cg_wdt.c

［指定形式］

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

［引数］

なし

［戻り値］

なし

3. 3. 9 リアルタイム・クロック

以下に，Applilet3 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—10 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・クロックの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_RTC_Create_UserInit	リアルタイム・クロックに関するユーザ独自の初期化処理を行います。
r_rtc_interrupt	リアルタイム・クロック割り込み INTRTC の発生に伴う処理を行います。
R_RTC_Start	リアルタイム・クロック（年，月，曜日，日，時，分，秒）のカウントを開始します。
R_RTC_Stop	リアルタイム・クロック（年，月，曜日，日，時，分，秒）のカウントを終了します。
R_RTC_Set_PowerOff	リアルタイム・クロックに対するクロック供給を停止します。
R_RTC_Set_HourSystem	リアルタイム・クロックの時間制（12 時間制，24 時間制）を設定します。
R_RTC_Set_CounterValue	リアルタイム・クロックにカウント値を設定します。
R_RTC_Get_CounterValue	リアルタイム・クロックのカウント値を読み出します。
R_RTC_Set_ConstPeriodInterruptOn	割り込み INTRTC の発生周期を設定したのち，定周期割り込み機能を開始します。
R_RTC_Set_ConstPeriodInterruptOff	定周期割り込み機能を終了します。
r_rtc_callback_constperiod	定周期割り込み INTRTC の発生に伴う処理を行います。
R_RTC_Set_AlarmOn	アラーム割り込み機能を開始します。
R_RTC_Set_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_callback_alarm	アラーム割り込み INTRTC の発生に伴う処理を行います。
R_RTC_Set_RTC1HZOn	RTC1HZ 端子に対する補正クロック（1 Hz）の出力を許可します。
R_RTC_Set_RTC1HZOff	RTC1HZ 端子に対する補正クロック（1 Hz）の出力を禁止します。

R_RTC_Create

リアルタイム・クロックの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。

[所属]

r_cg_rtc.c

[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。

[所属]

r_cg_rtc_user.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

r_rtc_interrupt

リアルタイム・クロック割り込み INTRTC の発生に伴う処理を行います。

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

[所属]

r_cg_rtc_user.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_RTC_Start

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

[所属]

r_cg_rtc.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_RTC_Stop

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

[所属]

r_cg_rtc.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_RTC_Set_PowerOff

リアルタイム・クロックに対するクロック供給を停止します。

備考 本 API 関数の呼び出しにより，リアルタイム・クロックはリセット状態へと移行します。
このため，本 API 関数の呼び出し後，制御レジスタへの書き込みは無視されます。

[所属]

r_cg_rtc.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_RTC_Set_HourSystem

リアルタイム・クロックの時間制（12 時間制，24 時間制）を設定します。

［所属］

r_cg_rtc.c

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

R_RTC_Set_CounterValue

リアルタイム・クロックにカウント値を設定します。

[所属]

r_cg_rtc.c

[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。

[所属]

r_cg_rtc.c

[指定形式]

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

[引数]

I/O	引数	説明
O	rtc_counter_value_t * 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_ConstPeriodInterruptOn

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

[所属]

r_cg_rtc.c

[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*;	割り込み INTRTC の発生周期 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。

[所属]

r_cg_rtc.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

r_rtc_callback_constperiod

定周期割り込み INTRTC の発生に伴う処理を行います。

備考 本 API 関数は，定周期割り込み INTRTC に対応した割り込み処理 r_rtc_interrupt のコールバック・ルーチンとして呼び出されます。

[所属]

r_cg_rtc_user.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_RTC_Set_AlarmOn

アラーム割り込み機能を開始します。

[所属]

r_cg_rtc.c

[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。

[所属]

r_cg_rtc.c

[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。

［所属］

r_cg_rtc.c

［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］

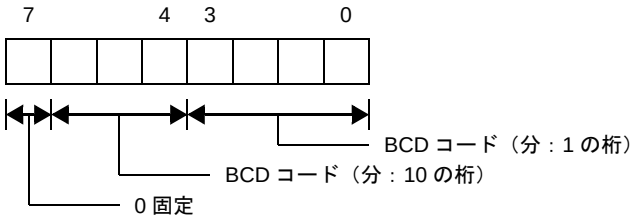
I/O	引数	説明
I	rtc_alarm_value_t *alarm_val*;	アラームの発生条件（曜日，時，分）

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。



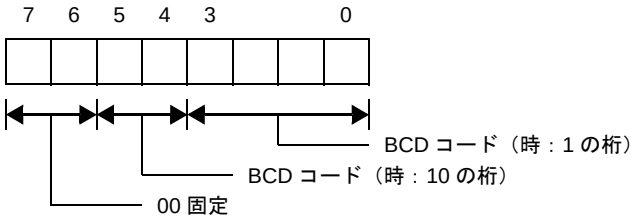
- Alarmwh（時）

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。

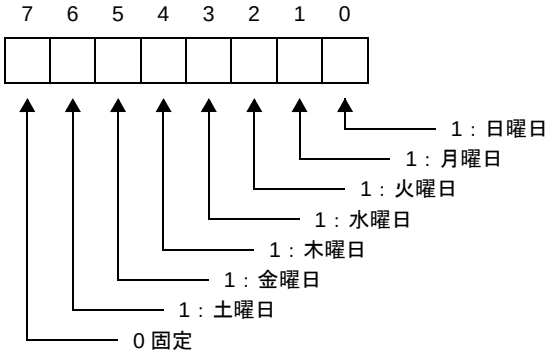
0： 午前

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。

[所属]

r_cg_rtc.c

[指定形式]

```
#include    "r_cg_macrodriver.h"
#include    "r_cg_rtc.h"
void    R_RTC_Get_AlarmValue ( rtc_alarm_value_t * 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 * 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

r_rtc_callback_alarm

アラーム割り込み INTRTC の発生に伴う処理を行います。

備考　本 API 関数は，アラーム割り込み INTRTC に対応した割り込み処理 r_rtc_interrupt のコールバック・ルーチンとして呼び出されます。

[所属]

r_cg_rtc_user.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_RTC_Set_RTC1HZOn

RTC1HZ 端子に対する補正クロック（1 Hz）の出力を許可します。

[所属]

r_cg_rtc.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_RTC_Set_RTC1HZOff

RTC1HZ 端子に対する補正クロック（1 Hz）の出力を禁止します。

[所属]

r_cg_rtc.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

3. 3. 10 インターバル・タイマ

以下に，Applilet3 がインターバル・タイマ用として出力する API 関数の一覧を示します。

表 3—11 インターバル・タイマ用 API 関数

API 関数名	機能概要
R_IT_Create	インターバル・タイマの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_IT_Create_UserInit	インターバル・タイマに関するユーザ独自の初期化処理を行います。
r_it_interrupt	インターバル・タイマ割り込み INTIT の発生に伴う処理を行います。
R_IT_Start	インターバル・タイマのカウントを開始します。
R_IT_Stop	インターバル・タイマのカウントを終了します。
R_IT_Set_PowerOff	インターバル・タイマに対するクロック供給を停止します。

R_IT_Create

インターバル・タイマの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。

[所属]

r_cg_it.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_IT_Create_UserInit

インターバル・タイマに関するユーザ独自の初期化処理を行います。

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

［所属］

r_cg_it_user.c

［指定形式］

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

［引数］

なし

［戻り値］

なし

r_it_interrupt

インターバル・タイマ割り込み INTIT の発生に伴う処理を行います。

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

[所属]

r_cg_it_user.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_IT_Start

インターバル・タイマのカウントを開始します。

[所属]

r_cg_it.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_IT_Stop

インターバル・タイマのカウントを終了します。

[所属]

r_cg_it.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_IT_Set_PowerOff

インターバル・タイマに対するクロック供給を停止します。

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

[所属]

r_cg_it.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

3. 3. 11 コンパレータ

以下に，Applilet3 がコンパレータ用として出力する API 関数の一覧を示します。

表 3—12 コンパレータ用 API 関数

API 関数名	機能概要
R_COMP_Create	コンパレータの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_COMP_Create_UserInit	コンパレータに関するユーザ独自の初期化処理を行います。
r_compn_interrupt	コンパレータ割り込み INTCMP*n* の発生に伴う処理を行います。
R_COMPn_Start	リファレンス入力電圧とアナログ入力電圧の比較動作を開始します。
R_COMPn_Stop	リファレンス入力電圧とアナログ入力電圧の比較動作を停止します。
R_COMP_Set_PowerOff	コンパレータに対するクロック供給を停止します。

R_COMP_Create

コンパレータの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。

[所属]

r_cg_comp.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_COMP_Create_UserInit

コンパレータに関するユーザ独自の初期化処理を行います。

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

[所属]

r_cg_comp_user.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

r_comp*n*_interrupt

コンパレータ割り込み INTCMP*n* の発生に伴う処理を行います。

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

[所属]

r_cg_comp_user.c

[指定形式]

```
__interrupt static  void    r_compn_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_COMP*n*_Start

リファレンス入力電圧とアナログ入力電圧の比較動作を開始します。

[所属]

r_cg_comp.c

[指定形式]

```
void    R_COMPn_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_COMP*n*_Stop

リファレンス入力電圧とアナログ入力電圧の比較動作を停止します。

[所属]

r_cg_comp.c

[指定形式]

```
void    R_COMPn_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_COMP_Set_PowerOff

コンパレータに対するクロック供給を停止します。

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

[所属]

r_cg_comp.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

3. 3. 12 クロック出力／ブザー出力

以下に，Applilet3 がクロック出力／ブザー出力用として出力する API 関数の一覧を示します。

表 3—13 クロック出力／ブザー出力用 API 関数

API 関数名	機能概要
R_PCLBUZn_Create	クロック出力／ブザー出力制御回路の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_PCLBUZn_Create_UserInit	クロック出力／ブザー出力制御回路に関するユーザ独自の初期化処理を行います。
R_PCLBUZn_Start	クロック出力／ブザー出力を開始します。
R_PCLBUZn_Stop	クロック出力／ブザー出力を停止します。

R_PCLBUZ*n*_Create

クロック出力／ブザー出力制御回路の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。

[所属]

r_cg_pclbuz.c

[指定形式]

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

備考　*n* は，出力端子を意味します。

[引数]

なし

[戻り値]

なし

R_PCLBUZ*n*_Create_UserInit

クロック出力／ブザー出力制御回路に関するユーザ独自の初期化処理を行います。

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

[所属]

r_cg_pclbuz_user.c

[指定形式]

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

備考　*n* は，出力端子を意味します。

[引数]

なし

[戻り値]

なし

R_PCLBUZ*n*_Start

クロック出力／ブザー出力を開始します。

［所属］

r_cg_pclbuz.c

［指定形式］

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

備考　*n* は，出力端子を意味します。

［引数］

なし

［戻り値］

なし

R_PCLBUZ*n*_Stop

クロック出力／ブザー出力を停止します。

［所属］

r_cg_pclbuz.c

［指定形式］

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

備考　*n* は，出力端子を意味します。

［引数］

なし

［戻り値］

なし

3. 3. 13 データトランスファコントローラ

以下に，Applilet3 がデータトランスファコントローラ用として出力する API 関数の一覧を示します。

表 3—14 データトランスファコントローラ用 API 関数

API 関数名	機能概要
R_DTC_Create	データトランスファコントローラの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_DTC_Create_UserInit	データトランスファコントローラに関するユーザ独自の初期化処理を行います。
R_DTCn_Start	データトランスファコントローラを動作可能状態にします。
R_DTCn_Stop	データトランスファコントローラを動作停止状態にします。
R_DTC_Set_PowerOff	データトランスファコントローラに対するクロック供給を停止します。

R_DTC_Create

データトランスファコントローラの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。

[所属]

r_cg_dtc.c

[指定形式]

```
void    R_DTC_Create ( void );
```

[引数]

なし

[戻り値]

なし

R_DTC_Create_UserInit

データトランスファコントローラに関するユーザ独自の初期化処理を行います。

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

[所属]

r_cg_dtc_user.c

[指定形式]

```
void    R_DTC_Create_UserInit ( void );
```

[引数]

なし

[戻り値]

なし

R_DTC*n*_Start

データトランスファコントローラを動作可能状態にします。

[所属]

r_cg_dtc.c

[指定形式]

```
void    R_DTCn_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_DTC*n*_Stop

データトランスファコントローラを動作停止状態にします。

[所属]

r_cg_dtc.c

[指定形式]

```
void    R_DTCn_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_DTC_Set_PowerOff

データトランスファコントローラに対するクロック供給を停止します。

備考 本 API 関数の呼び出しにより，データトランスファコントローラはリセット状態へと移行します。
このため，本 API 関数の呼び出し後，制御レジスタへの書き込みは無視されます。

[所属]

r_cg_dtc.c

[指定形式]

```
void    R_DTC_Set_PowerOff ( void );
```

[引数]

なし

[戻り値]

なし

3. 3. 14 イベントリンクコントローラ

以下に，Applilet3 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—15 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の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_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_Stop	イベントリンクコントローラを動作停止状態にします。

R_ELC_Create

イベントリンクコントローラの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。

[所属]

r_cg_elc.c

[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。

[所属]

r_cg_elc_user.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_ELC_Stop

イベントリンクコントローラを動作停止状態にします。

[所属]

r_cg_elc.c

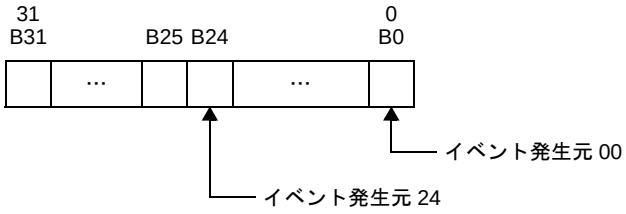
[指定形式]

```
void    R_ELC_Stop ( uint32_t event );
```

[引数]

I/O	引数	説明
I	uint32_t *event*;	停止するイベント発生元

備考 以下に，停止するイベント発生元 *event* の指定形式を示します。
なお，*event* に 0x01010101 を設定した場合，イベント発生元 00，08，16，24 のイベントリンク動作が禁止されます。



[戻り値]

なし

3. 3. 15 DMA コントローラ

以下に，Applilet3 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—16 DMA コントローラ用 API 関数

API 関数名	機能概要
R_DMACn_Create	DMA コントローラの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_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_interrupt	DMA 転送終了割り込み INTDMA*n* の発生に伴う処理を行います。
R_DMACn_Start	チャネル *n* を動作許可状態に設定します。
R_DMACn_Stop	チャネル *n* を動作停止状態に設定します。
R_DMACn_Set_SoftwareTriggerOn	DMA 動作許可状態の際，DMA 転送を開始します。

R_DMAC*n*_Create

DMA コントローラの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。

[所属]

r_cg_dmac.c

[指定形式]

```
void    R_DMACn_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_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。

[所属]

r_cg_dmac_user.c

[指定形式]

```
void    R_DMACn_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_dmac*n*_interrupt

DMA 転送終了割り込み INTDMA*n* の発生に伴う処理を行います。

備考 本 API 関数は，DMA 転送終了割り込み INTDMA*n* に対応した割り込み処理として呼び出されます。

[所属]

r_cg_dmac_user.c

[指定形式]

```
__interrupt static void    r_dmacn_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_DMAC*n*_Start

チャネル *n* を動作許可状態に設定します。

[所属]

r_cg_dmac.c

[指定形式]

```
void    R_DMACn_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_DMAC*n*_Stop

チャネル *n* を動作停止状態に設定します。

備考 1. 本 API 関数は，DMA 転送を強制終了させるものではありません。

2. 本 API 関数は，“転送終了”の確認後に呼び出す必要があります。

[所属]

r_cg_dmac.c

[指定形式]

```
void    R_DMACn_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_DMAC*n*_Set_SoftwareTriggerOn

DMA 動作許可状態の際，DMA 転送を開始します。

［所属］

r_cg_dmac.c

［指定形式］

```
void    R_DMACn_Set_SoftwareTriggerOn ( 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. 3. 16 電圧検出回路

以下に，Applilet3 が電圧検出回路用として出力する API 関数の一覧を示します。

表 3—17 電圧検出回路用 API 関数

API 関数名	機能概要
R_LVD_Create	電圧検出回路の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_LVD_Create_UserInit	電圧検出回路に関するユーザ独自の初期化処理を行います。
r_lvd_interrupt	電圧検出割り込み INTLVI の発生に伴う処理を行います。
R_LVD_InterruptMode_Start	電圧検出動作を開始します（割り込みモード時，および割り込み & リセット・モード時）。

R_LVD_Create

電圧検出回路の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。

[所属]

r_cg_lvd.c

[指定形式]

```
void    R_LVD_Create ( void );
```

[引数]

なし

[戻り値]

なし

R_LVD_Create_UserInit

電圧検出回路に関するユーザ独自の初期化処理を行います。

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

[所属]

r_cg_lvd_user.c

[指定形式]

```
void    R_LVD_Create_UserInit ( void );
```

[引数]

なし

[戻り値]

なし

r_lvd_interrupt

電圧検出割り込み INTLVI の発生に伴う処理を行います。

備考 本 API 関数は，電圧検出割り込み INTLVI に対応した割り込み処理として呼び出されます。

[所属]

r_cg_lvd_user.c

[指定形式]

```
__interrupt static  void    r_lvd_interrupt ( void );
```

[引数]

なし

[戻り値]

なし

R_LVD_InterruptMode_Start

電圧検出動作を開始します（割り込みモード時，および割り込み＆リセット・モード時）。

[所属]

r_cg_lvd.c

[指定形式]

```
void    R_LVD_InterruptMode_Start ( void );
```

[引数]

なし

[戻り値]

なし

3. 3. 17 プログラマブル・ゲイン・アンプ

以下に，Applilet3 がプログラマブル・ゲイン・アンプ用として出力する API 関数の一覧を示します。

表 3—18 プログラマブル・ゲイン・アンプ用 API 関数

API 関数名	機能概要
R_PGA_Create	プログラマブル・ゲイン・アンプの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_PGA_Create_UserInit	プログラマブル・ゲイン・アンプに関するユーザ独自の初期化処理を行います。
R_PGA_Start	プログラマブル・ゲイン・アンプの動作を開始します。
R_PGA_Stop	プログラマブル・ゲイン・アンプの動作を停止します。

R_PGA_Create

プログラマブル・ゲイン・アンプの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。

[所属]

r_cg_pga.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_PGA_Create_UserInit

プログラマブル・ゲイン・アンプに関するユーザ独自の初期化処理を行います。

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

[所属]

r_cg_pga_user.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_PGA_Start

プログラマブル・ゲイン・アンプの動作を開始します。

[所属]

r_cg_pga.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

R_PGA_Stop

プログラマブル・ゲイン・アンプの動作を停止します。

[所属]

r_cg_pga.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

付録 A 索 引

【た行】

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2011.10.01	―	初版発行

Applilet3　ユーザーズマニュアル
RL78 APIリファレンス編

発行年月日　2011年10月1日　Rev.1.00

発行　　ルネサス エレクトロニクス株式会社
　　〒211-8668　神奈川県川崎市中原区下沼部1753

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・電話番号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 エレクトロニクス販売株式会社　〒100-0004　千代田区大手町2-6-2（日本ビル）

(03)5201-5307

■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/inquiry

Colophon 1.0

Applilet3

ルネサス エレクトロニクス株式会社

R20UT0758JJ0100