User's Manual

CubeSuite+ V1.00.01

統合開発環境

ユーザーズマニュアル　RL78 設計編

対象デバイス

RL78 ファミリ

本資料に記載の全ての情報は本資料発行時点のものであり、ルネサス エレクトロニクスは、
予告なしに、本資料に記載した製品または仕様を変更することがあります。
ルネサス エレクトロニクスのホームページなどにより公開される最新情報をご確認ください。

ルネサス エレクトロニクス
www.renesas.com

Rev.1.00　2011.06

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

このマニュアルは，RL78 **ファミリ用アプリケーション・システムを開発する際の統合開発環境である** CubeSuite+ **について説明します。**

CubeSuite+**は，**RL78 **ファミリの統合開発環境（ソフトウエア開発における，設計，実装，デバッグなどの各開発フェーズに必要なツールをプラットフォームである** IDE **に統合）です。統合することで，さまざまなツールを使い分ける必要がなく，本製品のみを使用して開発のすべてを行うことができます。**

対 象 者　**このマニュアルは，**CubeSuite+**を使用してアプリケーション・システムを開発するユーザを対象としています。**

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

構　　成　**このマニュアルは，大きく分けて次の内容で構成しています。**

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

凡　　例

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

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

資　料　名		資料番号	
		和　文	英　文
CubeSuite+ 統合開発環境 ユーザーズ・マニュアル	起動編	R20UT0545J	R20UT0545E
	78K0 設計編	R20UT0546J	R20UT0546E
	78K0R 設計編	R20UT0547J	R20UT0547E
	RL78 設計編	このマニュアル	R20UT0548E
	V850 設計編	R20UT0549J	R20UT0549E
	R8C 設計編	R20UT0550J	R20UT0550E
	78K0 コーディング編	R20UT0551J	R20UT0551E
	RL78,78K0R コーディング編	R20UT0552J	R20UT0552E
	V850 コーディング編	R20UT0553J	R20UT0553E
	コーディング編（CX コンパイラ）	R20UT0554J	R20UT0554E
	R8C コーディング編	R20UT0576J	R20UT0576E
	78K0 ビルド編	R20UT0555J	R20UT0555E
	RL78,78K0R ビルド編	R20UT0556J	R20UT0556E
	V850 ビルド編	R20UT0557J	R20UT0557E
	ビルド編（CX コンパイラ）	R20UT0558J	R20UT0558E
	R8C ビルド編	R20UT0575J	R20UT0575E
	78K0 デバッグ編	R20UT0559J	R20UT0559E
	78K0R デバッグ編	R20UT0560J	R20UT0560E
	RL78 デバッグ編	R20UT0561J	R20UT0561E
	V850 デバッグ編	R20UT0562J	R20UT0562E
	R8C デバッグ編	R20UT0574J	R20UT0574E
	解析編	R20UT0563J	R20UT0563E
	メッセージ編	R20UT0407J	R20UT0407E

注意　上記関連資料は，予告なしに内容を変更することがあります。設計などには，必ず最新の資料を使用してください。

〔メ　モ〕

〔メ　モ〕

〔メ　モ〕

目　　次

第 1 章　概　　説

CubeSuite+ は，アプリケーション・システムを開発する際の統合開発環境であり，設計／コーディング／ビルド／デバッグなどといった一連の作業を実施することができます。

本章では，設計ツール（コード生成）の概要について説明します。

1. 1　概　　要

設計ツールは，CubeSuite+ が提供しているコンポーネントの 1 種であり，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，設計ツール（コード生成）の特長を示します。

- コード生成機能

  コード生成では，GUI ベースで設定した情報に応じたデバイス・ドライバ・プログラムを出力するだけでなく，main 関数を含んだサンプル・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。

  なお，コード生成から出力されるソース・コードは，自動車向け組み込み C 言語用ガイドライン MISRA-C のコーディング規約に対応したものとなっています。

- レポート機能

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

- リネーム機能

  コード生成が出力するファイル名，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　機能（コード生成）

本章では，設計ツール（コード生成）が提供している主な機能を操作手順とともに説明します。

2.1　概　　要

コード生成は，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）を制御する際に必要な情報を CubeSuite+ のパネル上で選択／入力することにより，対応するソース・コード（デバイス・ドライバ・プログラム）を出力します。

なお，コード生成の操作手順は，以下のとおりです。

（1）CubeSuite+ の起動

Windows の［スタート］メニューから CubeSuite+ を起動します。

備考　“CubeSuite+ の起動”についての詳細は，「CubeSuite+　起動編」を参照してください。

（2）プロジェクトの作成／読み込み

プロジェクトの新規作成（プロジェクトの種類，使用するマイクロコントローラ，使用するビルド・ツールなどの定義），または既存のプロジェクトの読み込みを行います。

備考　“プロジェクトの作成／読み込み”についての詳細は，「CubeSuite+　起動編」を参照してください。

（3）コード生成 パネルのオープン

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情報を設定するための コード生成 パネルをオープンします。

（4）情報の設定

コード生成 パネルで周辺機能を制御するうえで必要な情報を設定します。

（5）ソース・コードの確認

コード生成 パネルで設定した情報に応じたソース・コード（デバイス・ドライバ・プログラム）を確認します。

（6）ソース・コードの出力

ソース・コード（デバイス・ドライバ・プログラム）を指定されたフォルダに出力します。

（7）レポート・ファイルの出力

レポート・ファイル（コード生成を用いて設定した情報を保持したファイル，ソース・コードに関する情報を保持したファイル）を指定されたフォルダに出力します。

（8）プロジェクトの保存

プロジェクトの保存を行います。

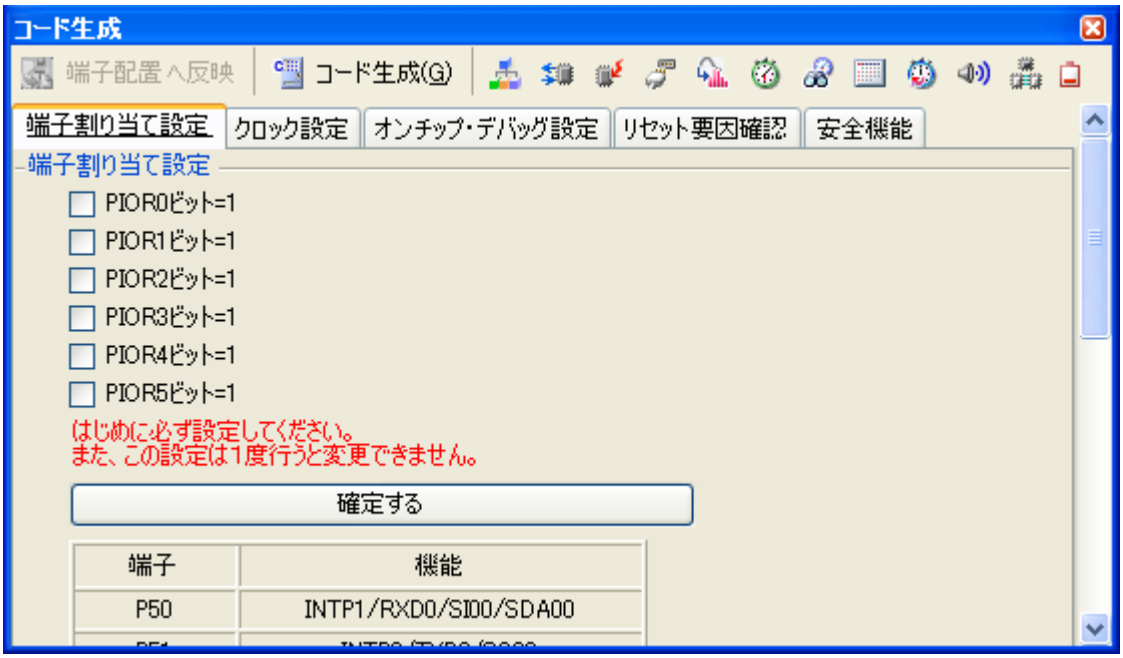
備考　“プロジェクトの保存”についての詳細は，「CubeSuite+　起動編」を参照してください。

2.2　コード生成 パネルのオープン

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情報を設定するためのコード生成 パネルをオープンします。

なお，コード生成 パネルのオープンは，プロジェクト・ツリー パネルの［*Project name*（プロジェクト）］→［コード生成（設計ツール）］→周辺機能ノード（［クロック発生回路］，［ポート］など）を選択することにより行います。

図 2—1　コード生成 パネルのオープン



備考　コード生成が未対応のマイクロコントローラがプロジェクトで定義された場合，プロジェクト・ツリー パネルの［*Project name*（プロジェクト）］に“［コード生成（設計ツール）］ノード”は表示されません。

2.3 情報の設定

「2.2 コード生成 パネルのオープン」でオープンしたコード生成 パネルの情報設定エリアで周辺機能を制御するうえで必要な情報を設定します。

備考 複数の周辺機能を制御する場合は，「2.2 コード生成 パネルのオープン」から「2.3 情報の設定」の操作を繰り返し行うことになります。

2.3.1 入力規約

以下に，コード生成 パネルに各種情報を設定する際の入力規約を示します。

（1）文字セット

以下に，コード生成が入力を許可している文字セットを示します。

表 2—1 文字セットの一覧

文字セット	概要
ASCII	半角のアルファベット（英字），半角の数字，半角の記号
Shift-JIS	全角のアルファベット（英字），全角の数字，全角の記号，全角のひらがな，全角のカタカナ，全角の漢字，および半角のカタカナ
EUC-JP	全角のアルファベット（英字），全角の数字，全角の記号，全角のひらがな，全角のカタカナ，全角の漢字，および半角のカタカナ
UTF-8	全角のアルファベット（英字），全角の数字，全角の記号，全角のひらがな，全角のカタカナ，全角の漢字（中国語を含む），および半角のカタカナ

（2）数値

以下に，コード生成が入力を許可している進数を示します。

表 2—2 進数の一覧

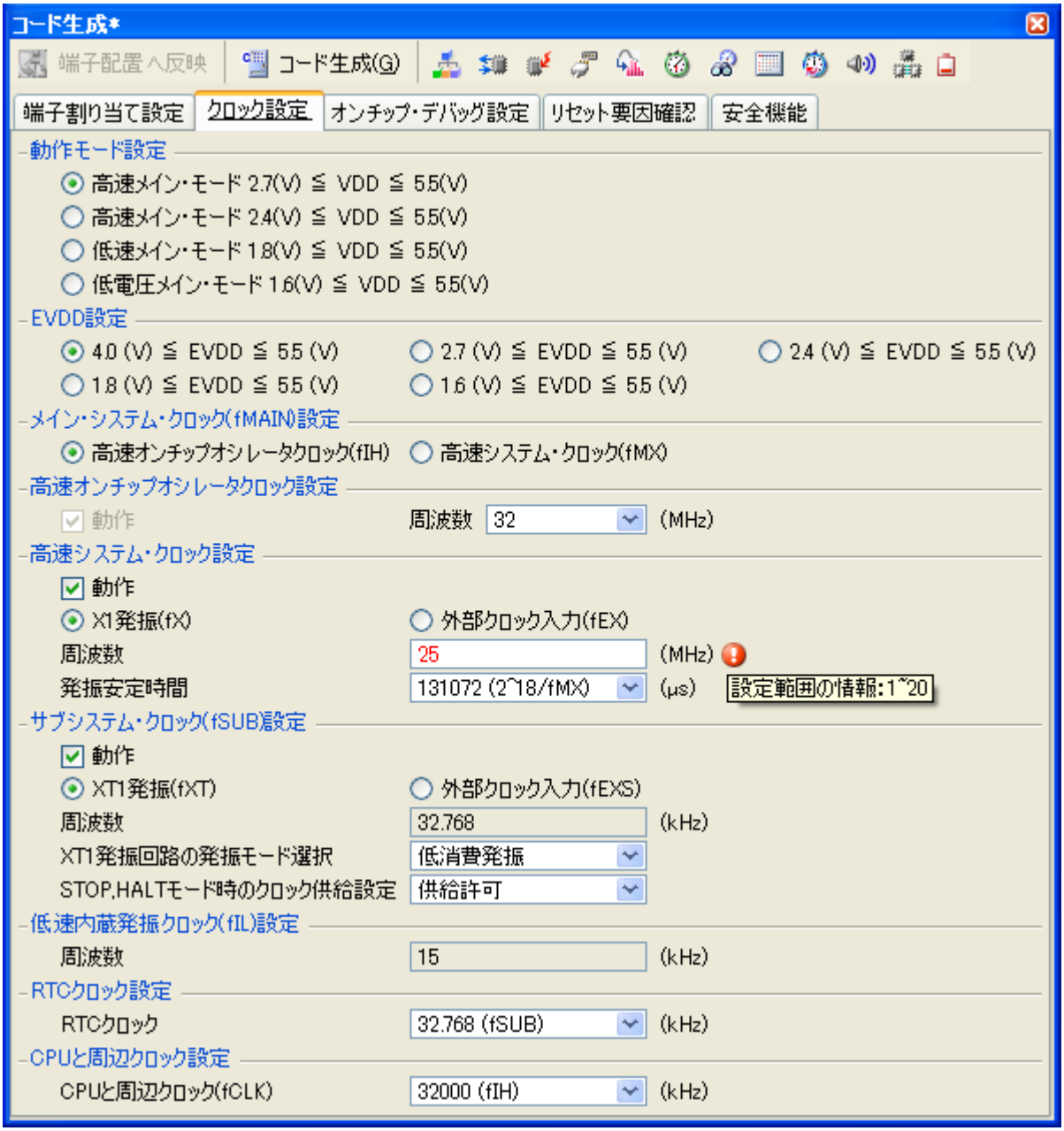
進数表記	概要
10 進数	1 ～ 9 の数字で始まり 0 ～ 9 の数字が続く数値，および 0
16 進数	0x で始まり 0 ～ 9 の数字，および a ～ f の英字が続く数値 （英字の大文字／小文字については，不問）

2. 3. 2　入力不備箇所に対するアイコン表示

コード生成では，コード生成 パネルで不正な文字列が入力された際，および入力が必須な箇所に値が未入力の際，設定すべき情報として誤っていることを示すアイコンを該当箇所に表示するとともに，文字列を赤色表示し，入力の不備を警告します。

備考　アイコン上にマウス・カーソルを移動した際には，入力すべき文字列に関する情報（入力の不備を解決するためのヒント）がポップアップ表示されます。

図 2—2　入力不備箇所に対するアイコン表示

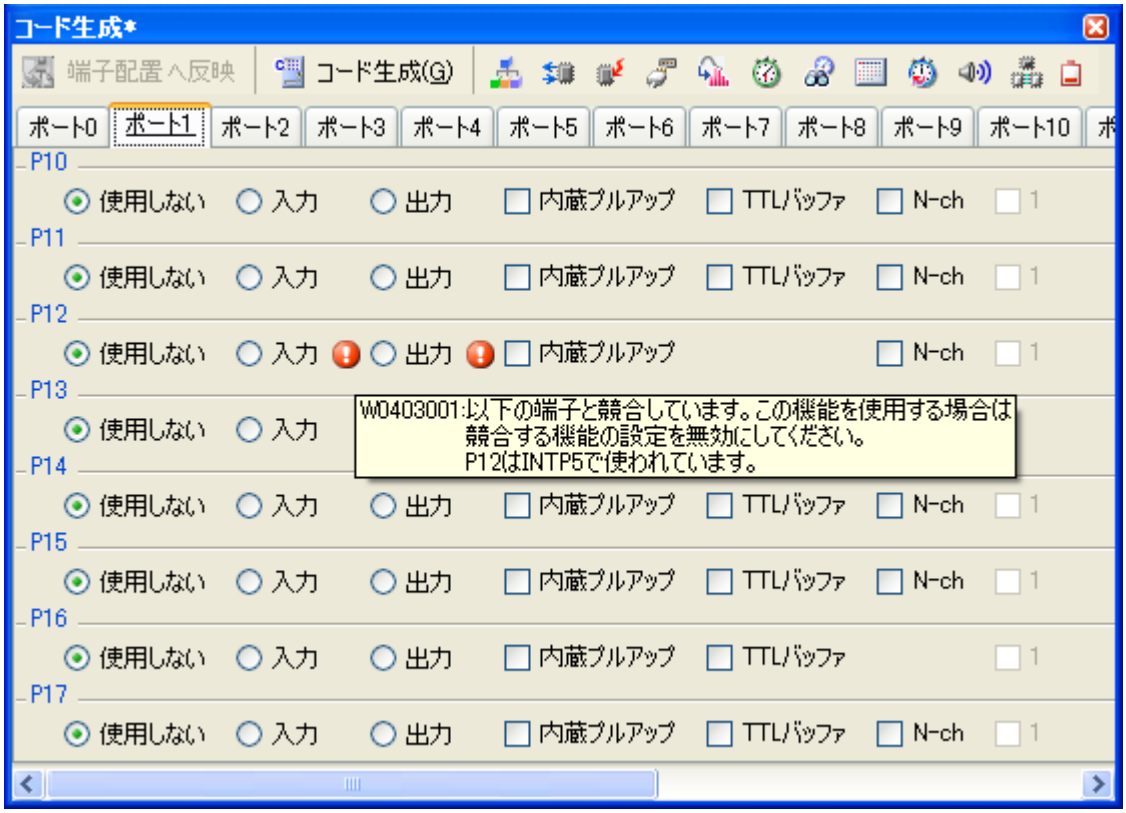


2. 3. 3　端子の競合に対するアイコン表示

コード生成では，コード生成 パネルにおける各種周辺機能の設定に伴い，端子の競合が発生する項目に対しては，競合が発生することを示すアイコンを該当箇所に表示し，端子の競合を警告します。

備考　アイコン上にマウス・カーソルを移動した際には，端子の競合に関する情報（競合を回避するためのヒント）がポップアップ表示されます。

図 2—3　端子の競合に対するアイコン表示

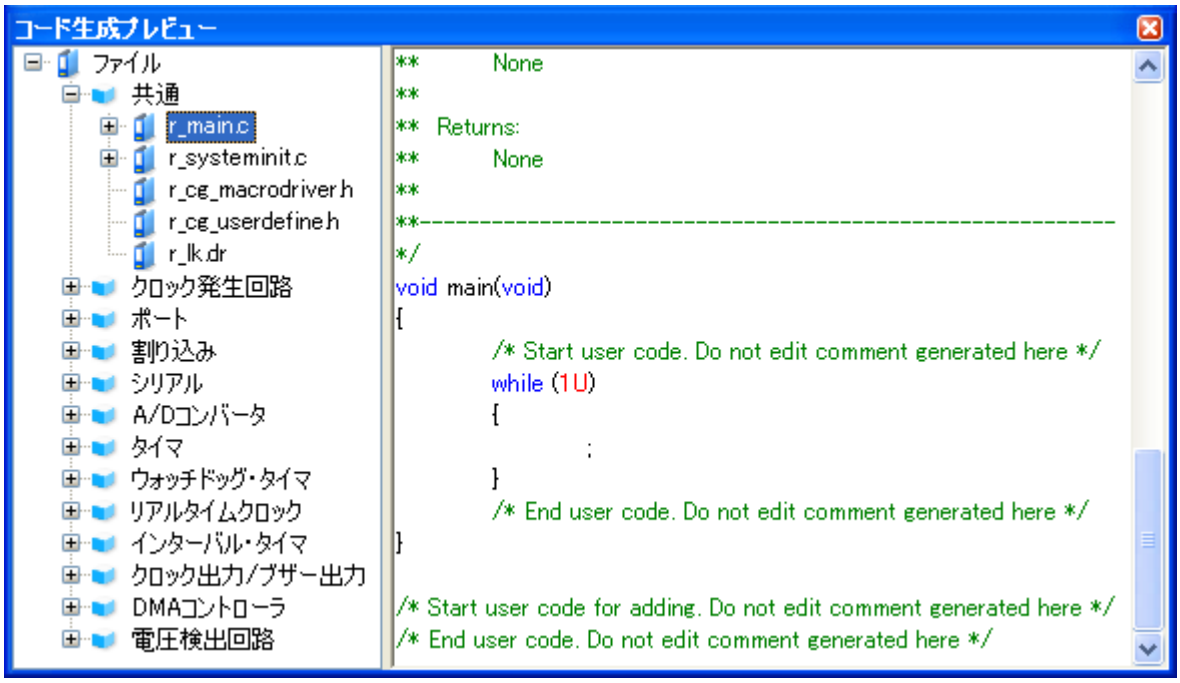


2.4　ソース・コードの確認

「2.3　情報の設定」で設定した情報に応じたソース・コード（デバイス・ドライバ・プログラム）を確認します。

なお，ソース・コードの確認は，［表示］メニュー→［コード生成プレビュー］を選択することによりオープンするコード生成プレビュー パネルで行います。

図 2—4　ソース・コードの確認



備考 1. コード生成プレビュー パネルのソース・ファイル名，または API 関数名を選択することにより，ソース・コードの表示を切り替えることができます。

2. コード生成プレビュー パネルに表示されるソース・コードの文字色は，以下の意味を持ちます。

表 2—3　ソース・コードの文字色

表示色	概要
緑	コメント文
青	C コンパイラの予約語
赤	数値
黒	コード部
グレー	ファイル名

3. コード生成プレビュー パネル内でソース・コードを編集することはできません。

4. 一部の API 関数（シリアル・アレイ・ユニット用 API 関数など）については，ソース・コードの出力時（コード生成 パネルの コード生成(G) ボタンをクリックした際）にレジスタ値 SFR などが計算され確

定するものがあります。このため，コード生成プレビュー パネルに表示されるソース・コードは，実際に出力されるソース・コードと一致しない場合があります。

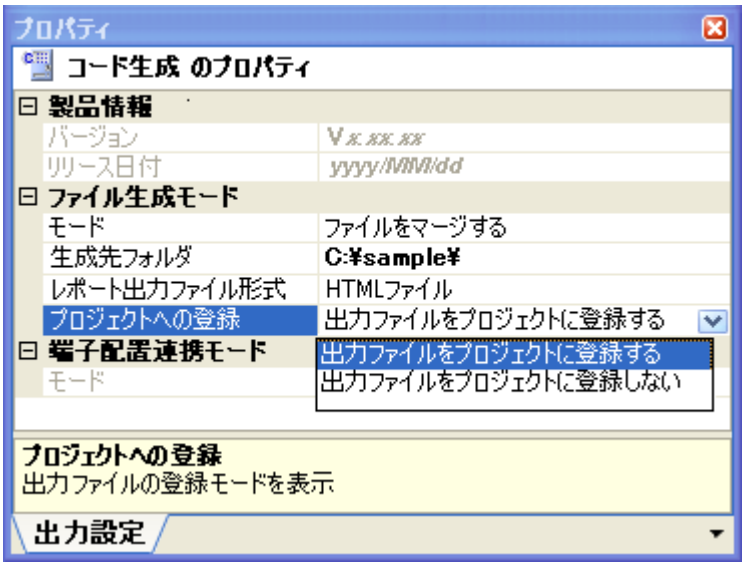
2.5　ソース・コードの出力

コード生成 パネルの コード生成(G) ボタンをクリックし，ソース・コード（デバイス・ドライバ・プログラム）を出力します。なお，ソース・コードの出力先は，プロパティ パネルの［出力設定］タブ→［生成先フォルダ］で指定されたフォルダとなります。

図 2—5　ソース・コードの出力

備考　コード生成(G) ボタンをクリックした際，ソース・コードを出力するとともに，該当ファイル群をプロジェクトに登録（プロジェクト・ツリー パネルに対する該当ソース・ファイル名の表示）する場合は，プロパティ パネルの［出力設定］タブ→［プロジェクトへの登録］で“出力ファイルをプロジェクトに登録する”を指定する必要があります。

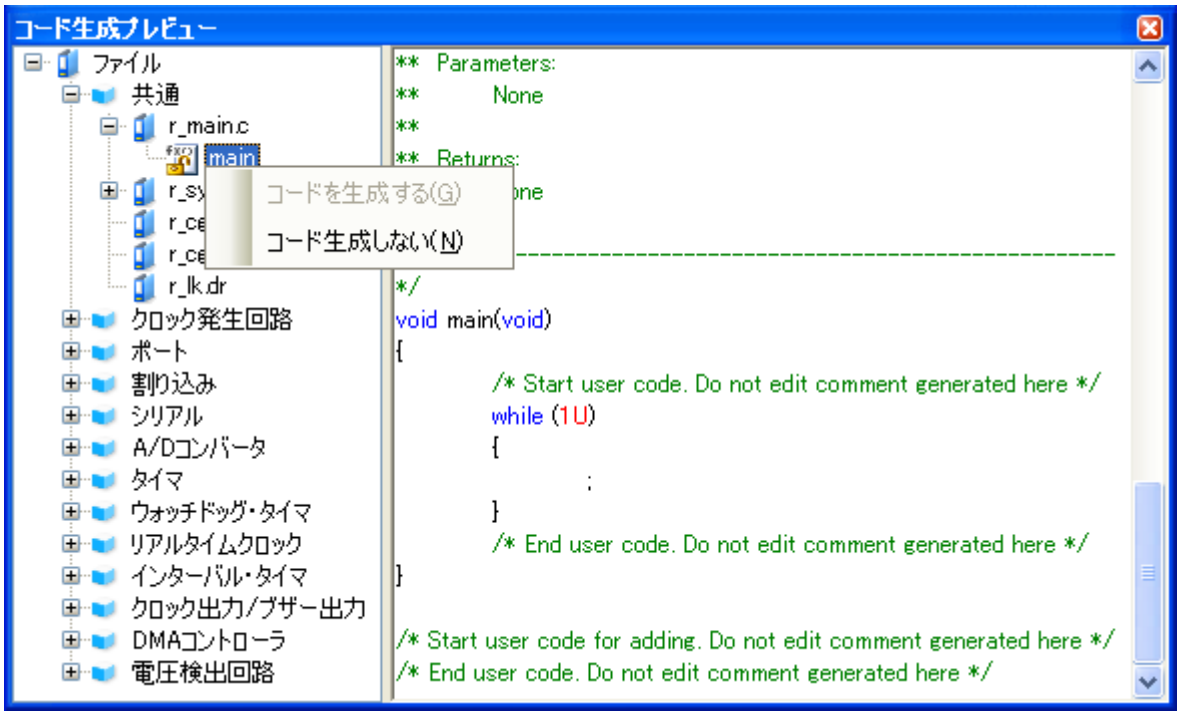
図 2—6 登録有無の設定



2. 5. 1 出力有無の設定

コード生成では，コード生成プレビュー パネルの API 関数名上でマウスを右クリックすることにより表示されるコンテキスト・メニューから“コードを生成する／コードを生成しない”を選択することにより，API 関数単位での“該当ソース・コードの出力有無”を設定することができます。

図 2—7 出力有無の設定



備考　出力有無の設定状況については，コード生成プレビュー パネルのアイコン種別により確認することができます。

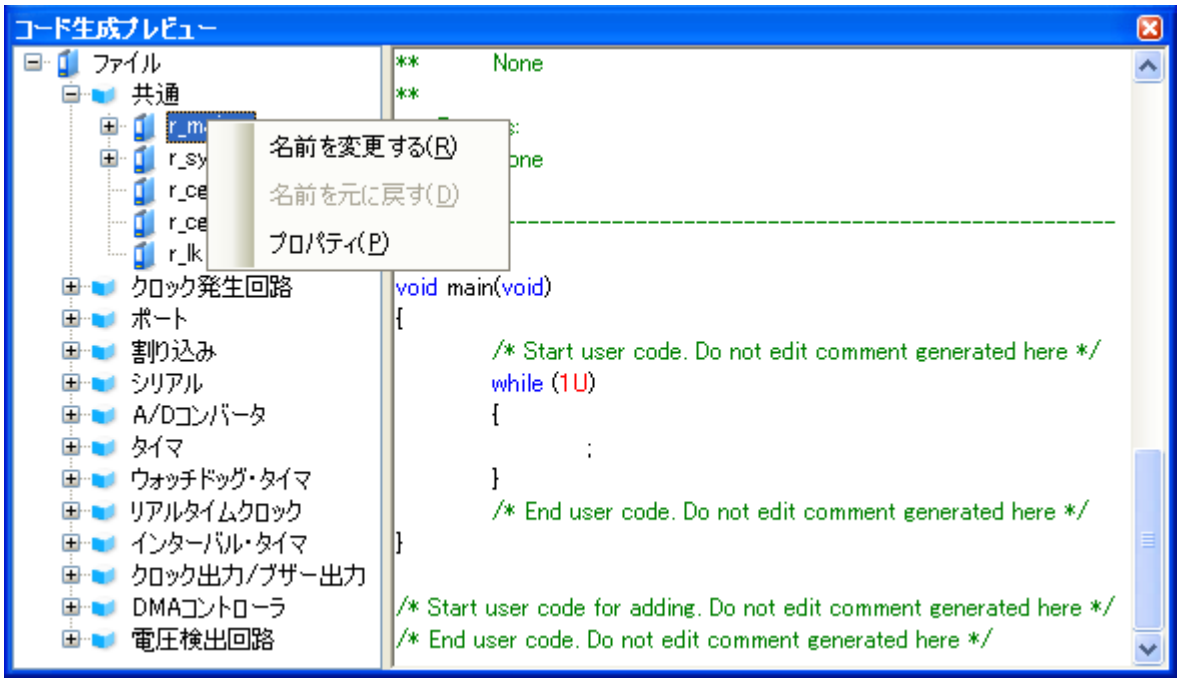
表 2—4　ソース・コードの出力有無

アイコン種別	概要
	該当 API 関数のソース・コードは，出力されます。 なお，本アイコンが表示されている API 関数は，ソース・コードの出力が必須（  への変更不可）となります。
	該当 API 関数のソース・コードは，出力されます。
	該当 API 関数のソース・コードは，出力されません。

2. 5. 2　ファイル名の変更

コード生成では，コード生成プレビュー パネルのファイル名上でマウスを右クリックすることにより表示されるコンテキスト・メニューから“名前を変更する”を選択することにより，ファイル名を変更することができます。

図 2―8　ファイル名の変更

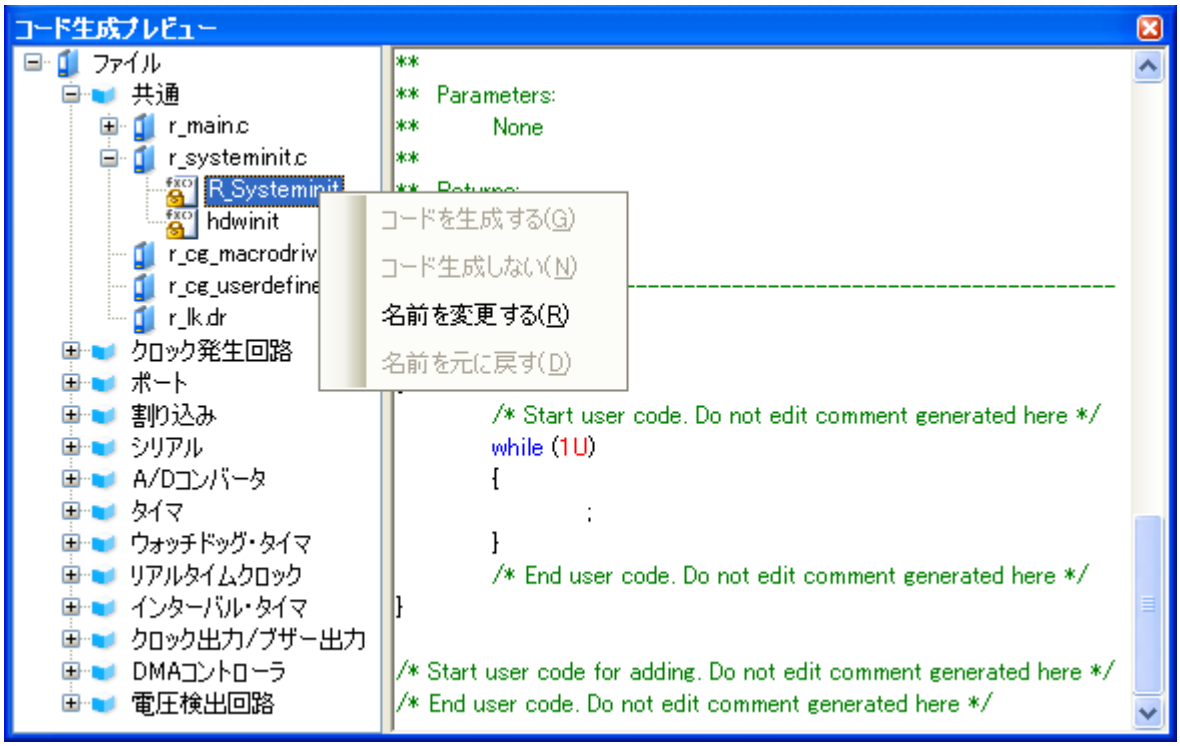


備考　コード生成が規定しているデフォルト・ファイル名に戻す際には，コンテキスト・メニューから“名前を元に戻す”を選択します。

2. 5. 3　API 関数名の変更

コード生成では，コード生成プレビュー パネルの API 関数名上でマウスを右クリックすることにより表示されるコンテキスト・メニューから“名前を変更する”を選択することにより，API 関数名を変更することができます。

図 2—9　API 関数名の変更

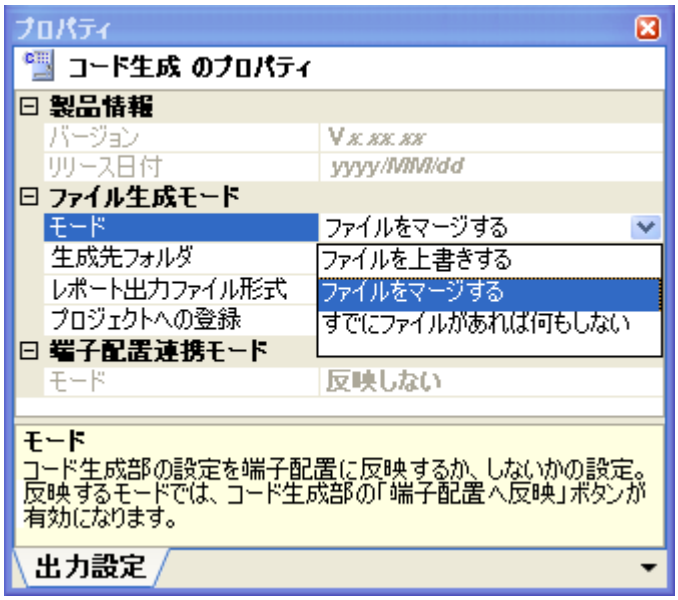


備考　コード生成が規定しているデフォルト API 関数名に戻す際には，コンテキスト・メニューから“名前を元に戻す”を選択します。

2. 5. 4　出力モードの変更

コード生成では，プロパティ パネルの［出力設定］タブ→［モード］でソース・コードの出力モード（ファイルを上書きする，ファイルをマージする，すでにファイルがあれば何もしない）を変更することができます。

図 2—10　出力モードの変更



備考　出力モードの選択は，以下の 3 種類から行います。

表 2—5　ソース・コードの出力モード

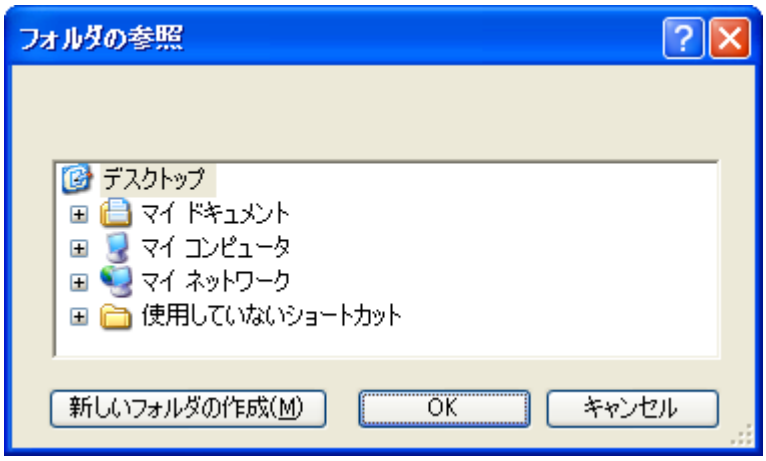
出力モード	概要
ファイルを上書きする	同一のファイル名を有するファイルが既存していた場合，該当ファイルを上書きします。
ファイルをマージする	同一のファイル名を有するファイルが既存していた場合，該当ファイルをマージします。 なお，マージする部位については， /* Start user code …. Do not edit comment generated here */ から /* End user code. Do not edit comment generated here */ で囲まれた部位に限られます。
すでにファイルがあれば何もしない	同一のファイル名を有するファイルが既存していた場合，該当ファイルの出力を行いません。

2. 5. 5　出力先の変更

コード生成では，プロパティ パネルの［出力設定］タブ→［生成先フォルダ］でソース・コードの出力先を変更することができます。

なお，出力先の変更は，［生成先フォルダ］の［...］ボタンをクリックすることによりオープンするフォルダの参照 ダイアログで行います。

図 2—11　出力先の変更



2. 6　レポート・ファイルの出力

コード生成 パネル，またはコード生成プレビュー パネルをアクティブな状態にしたのち，［ファイル］メニュー→［コード生成レポート を保存］を選択し，レポート・ファイル（コード生成を用いて設定した情報を保持したファイル，ソース・コードに関する情報を保持したファイル）を出力します。

なお，レポート・ファイルの出力先は，プロパティ パネルの［出力設定］タブ→［生成先フォルダ］で指定されたフォルダとなります。

備考 1.　レポート・ファイルのファイル名は，“macro”，および“function”に規定されています。

表 2—6　レポート・ファイルの出力

ファイル名	概要
macro	コード生成を用いて設定した情報を保持したファイル
function	ソース・コードに関する情報を保持したファイル

2.　レポート・ファイルの出力モードは，“ファイルを上書きする”となります。

図 2—12　レポート・ファイル macro の出力例

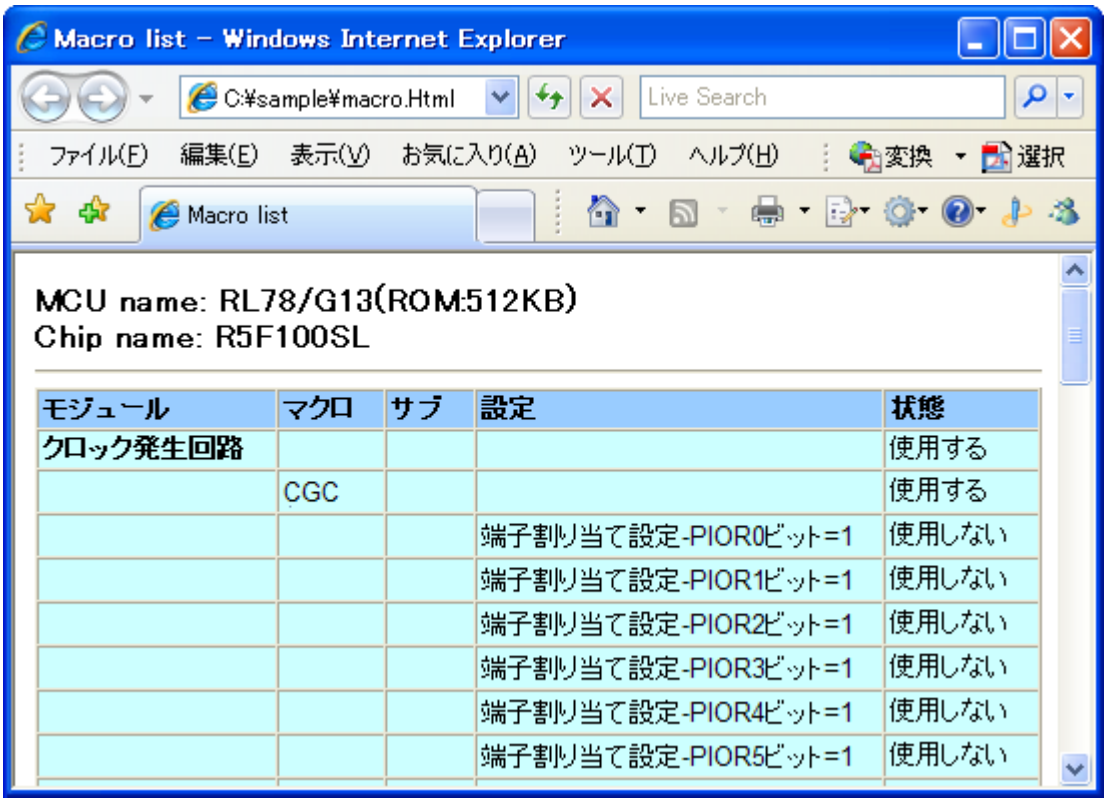
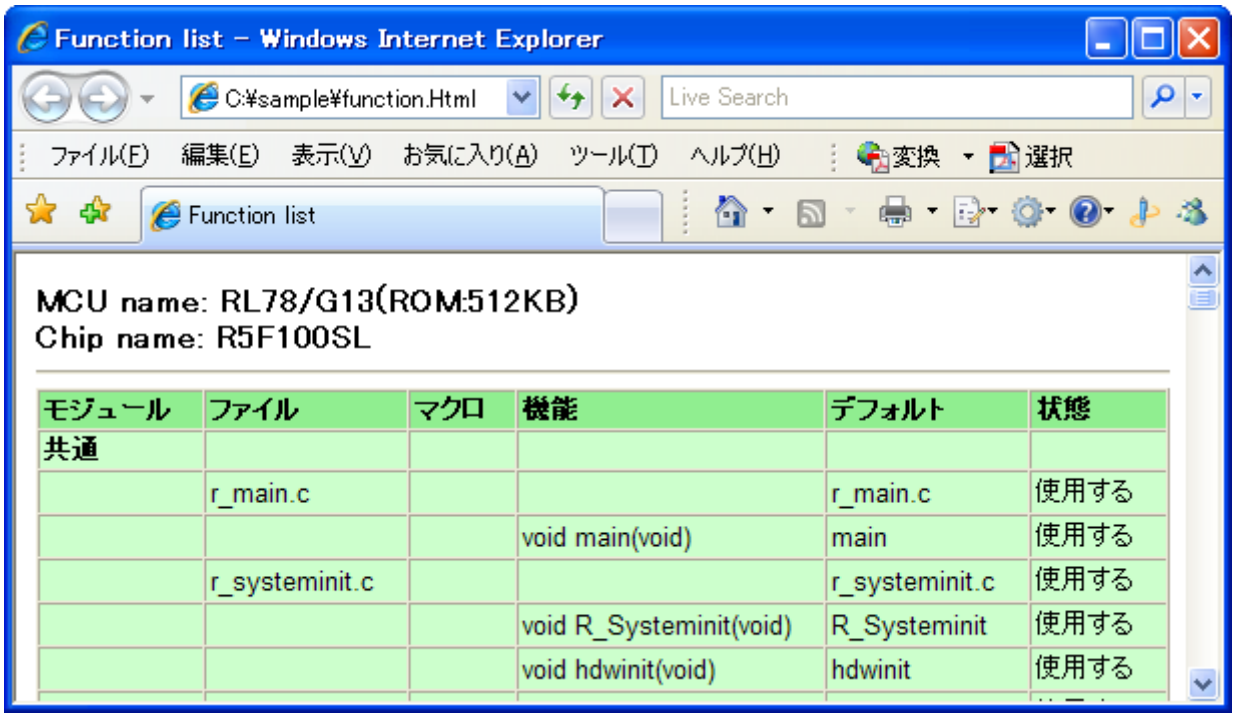


図 2—13 レポート・ファイル function の出力例

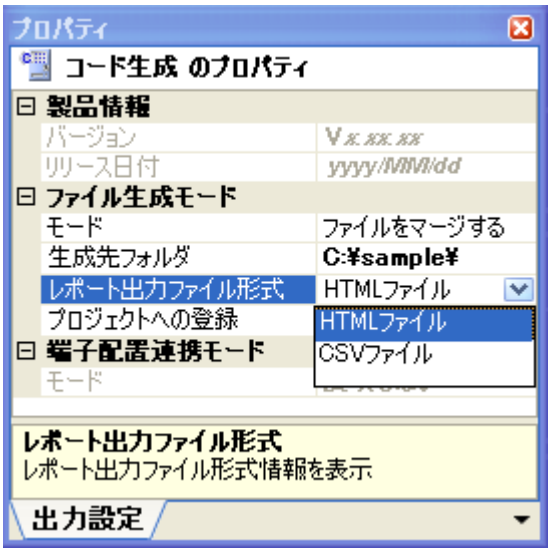


モジュール	ファイル	マクロ	機能	デフォルト	状態
共通					
	r_main.c			r_main.c	使用する
			void main(void)	main	使用する
	r_systeminit.c			r_systeminit.c	使用する
			void R_Systeminit(void)	R_Systeminit	使用する
			void hdwinit(void)	hdwinit	使用する

2. 6. 1 出力形式の変更

コード生成では，プロパティ パネルの［出力設定］タブ→［レポート出力ファイル形式］でレポート・ファイルの出力形式（HTML ファイル，CSV ファイル）を変更することができます。

図 2—14 出力形式の変更



備考 出力形式の選択は，以下の 2 種類から行います。

表 2—7 ソース・コードの出力モード

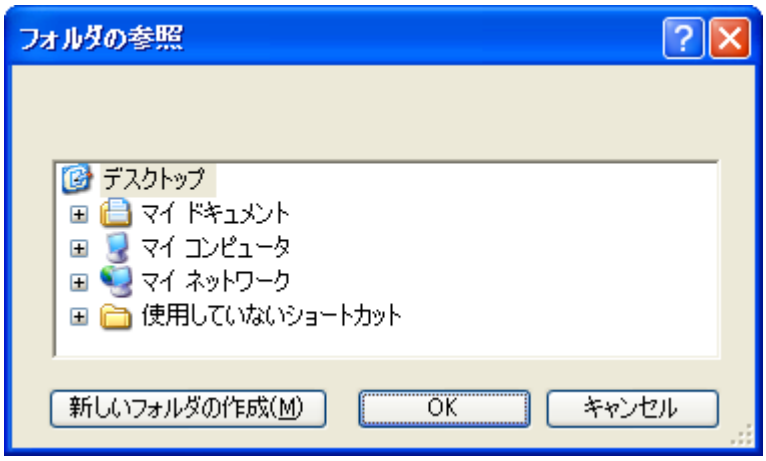
出力形式	概要
HTML ファイル	HTML 形式でレポート・ファイルを出力します。
CSV ファイル	CSV 形式でレポート・ファイルを出力します。

2. 6. 2　出力先の変更

コード生成では，プロパティ パネルの［出力設定］タブ→［生成先フォルダ］でレポート・ファイルの出力先を変更することができます。

なお，出力先の変更は，［生成先フォルダ］の［...］ボタンをクリックすることによりオープンするフォルダの参照 ダイアログで行います。

図 2—15　出力先の変更



付録 A　ウインドウ・リファレンス

本付録では，設計ツールのウインドウ／パネル／ダイアログについて説明します。

A. 1　説　　明

以下に，設計ツールのウインドウ／パネル／ダイアログの一覧を示します。

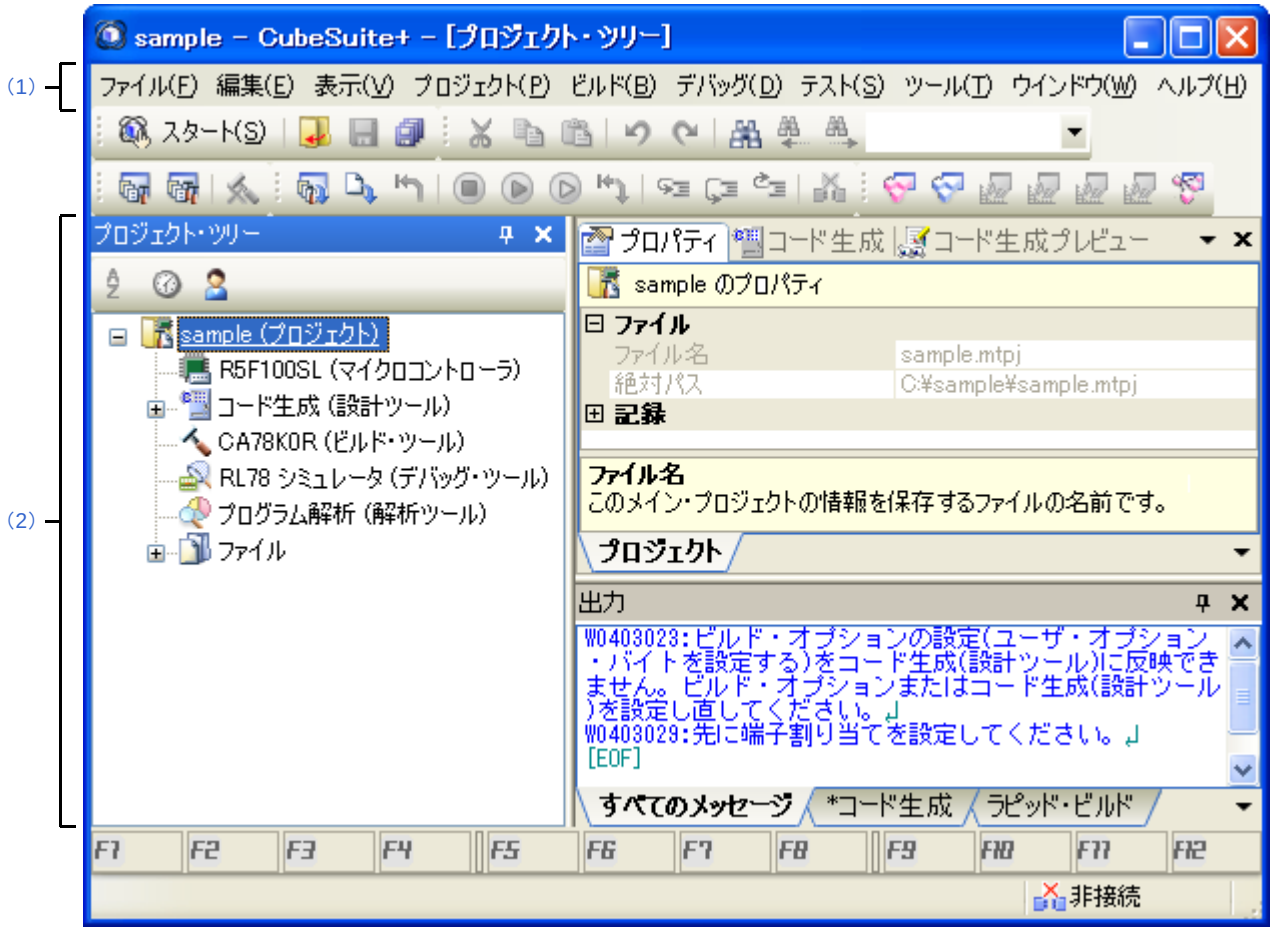
表 A—1　ウインドウ／パネル／ダイアログの一覧

ウインドウ／パネル／ダイアログ名	機能概要
メイン・ウインドウ	CubeSuite+ を起動した際，最初にオープンするウインドウであり，本ウインドウから CubeSuite+ が提供している各種コンポーネント（設計ツール，ビルド・ツールなど）に対する操作を行います。
プロジェクト・ツリー パネル	プロジェクトの構成要素（マイクロコントローラ，設計ツール，ビルド・ツールなど）をツリー形式で表示します。
プロパティ パネル	プロジェクト・ツリー パネルで選択したノード，コード生成 パネルでクリックした周辺機能ボタン，コード生成プレビュー パネルで選択したファイルの種類に対応した情報の表示，および設定の変更を行います。
コード生成 パネル	マイクロコントローラが提供している周辺機能を制御するうえで必要な情報を設定します。
コード生成プレビュー パネル	コード生成 パネルの コード生成(G) ボタンをクリックした際に出力されるソース・コード（デバイス・ドライバ・プログラム）の出力有無を API 関数単位で確認／設定するとともに，コード生成 パネルで設定した情報に応じたソース・コードの確認を行います。
出力 パネル	CubeSuite+ が提供している各種コンポーネント（設計ツール，ビルド・ツールなど）の操作ログを表示します。
フォルダの参照 ダイアログ	ファイル（ソース・コード，レポート・ファイルなど）の出力先を設定します。
名前を付けて保存 ダイアログ	ファイル（レポート・ファイルなど）に名前を付けて保存します。

メイン・ウインドウ

CubeSuite+ を起動した際，最初にオープンするウインドウであり，本ウインドウから CubeSuite+ が提供している各種コンポーネント（設計ツール，ビルド・ツールなど）に対する操作を行います。

図 A―1　メイン・ウインドウ



ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］

［オープン方法］

- Windows の［スタート］メニューから［プログラム］→［Renesas Electronics CubeSuite+］→［CubeSuite+］を選択

［各エリアの説明］

(1) メニューバー

本エリアは，以下に示したメニュー群から構成されています。

(a)［ファイル］メニュー

コード生成レポート を保存	コード生成 パネル／コード生成プレビュー パネル専用部分 レポート・ファイル（コード生成を用いて設定した情報を保持したファイル，ソース・コードに関する情報を保持したファイル）を出力します。 - レポート・ファイルの出力形式は，プロパティ パネルの［出力設定］タブ→［レポート出力ファイル形式］で選択された形式（HTML 形式，または CSV 形式）となります。 - レポート・ファイルの出力先は，プロパティ パネルの［出力設定］タブ→［生成先フォルダ］で指定されたフォルダとなります。
出力 - *タブ名* を保存	出力 パネル専用部分 該当タブのメッセージを既存のファイルに上書き保存します。
名前を付けて 出力 - *タブ名* を保存 ...	出力 パネル専用部分 該当タブのメッセージに名前を付けて保存するための名前を付けて保存 ダイアログをオープンします。

(b)［編集］メニュー

元に戻す	プロパティ パネル専用部分 直前に行った編集作業を取り消します。
切り取り	プロパティ パネル専用部分 選択している文字列を切り取り，クリップ・ボードに保存します。
コピー	プロパティ パネル／出力 パネル専用部分 選択している文字列をクリップ・ボードに保存します。
貼り付け	プロパティ パネル専用部分 指定された箇所に，クリップ・ボードの内容を挿入します。
削除	プロパティ パネル専用部分 選択している文字列を削除します。
すべて選択	プロパティ パネル／出力 パネル専用部分 編集中の項目に表示されている全文字列，またはメッセージ・エリアに表示されている全文字列を選択します。
検索 ...	コード生成プレビュー パネル／出力 パネル専用部分 文字列検索を行うための検索・置換 ダイアログを［クイック検索］タブが選択された状態でオープンします。
置換 ...	出力 パネル専用部分 文字列置換を行うための検索・置換 ダイアログを［一括置換］タブが選択された状態でオープンします。

(c)［ヘルプ］メニュー

プロジェクト・ツリー パネルのヘルプを開く	プロジェクト・ツリー パネル専用部分 プロジェクト・ツリー パネルのヘルプを表示します。
プロパティ パネルのヘルプを開く	プロパティ パネル専用部分 プロパティ パネルのヘルプを表示します。
コード生成 パネルのヘルプを開く	コード生成 パネル専用部分 コード生成 パネルのヘルプを表示します。
コード生成プレビュー パネルのヘルプを開く	コード生成プレビュー パネル専用部分 コード生成プレビュー パネルのヘルプを表示します。
出力 パネルのヘルプを開く	出力 パネル専用部分 出力 パネルのヘルプを表示します。

(2) パネル表示エリア

本エリアは，用途別に用意された各種パネルから構成されています。

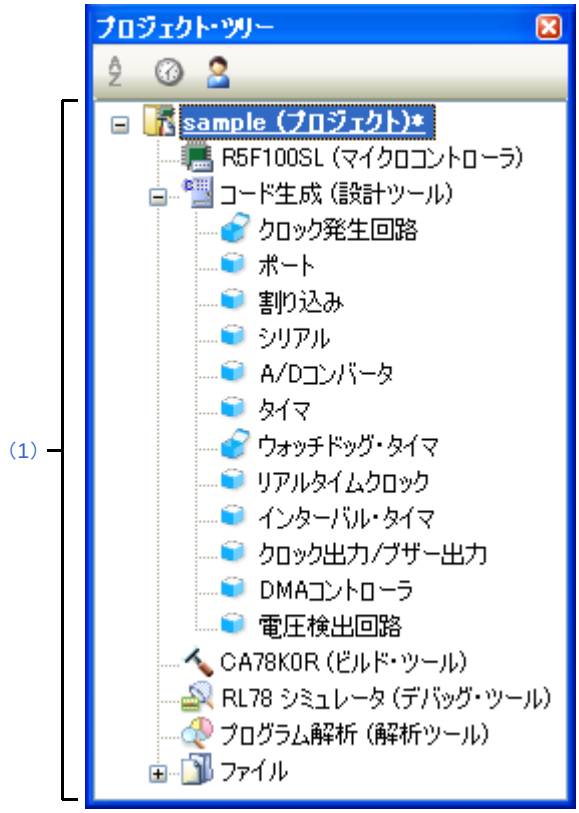
本エリアについての詳細は，以下を参照してください。

- プロジェクト・ツリー パネル
- プロパティ パネル
- コード生成 パネル
- コード生成プレビュー パネル
- 出力 パネル

プロジェクト・ツリー パネル

プロジェクトの構成要素（マイクロコントローラ，設計ツール，ビルド・ツールなど）をツリー形式で表示します。

図 A―2　プロジェクト・ツリー パネル



ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］
- ［［ヘルプ］メニュー（プロジェクト・ツリー パネル専用部分）］
- ［コンテキスト・メニュー］

［オープン方法］

- ［表示］メニュー→［プロジェクト・ツリー］を選択

［各エリアの説明］

(1) プロジェクト・ツリー・エリア

プロジェクトの構成要素（マイクロコントローラ，設計ツール，ビルド・ツールなど）をツリー形式で表示します。

(a) コード生成（設計ツール）

本ノードは，以下に示した周辺機能ノードから構成されています。

なお，対象マイクロコントローラが未サポートの周辺機能については，該当周辺機能ノードが表示されません。

クロック発生回路	マイクロコントローラが提供している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な情報を設定するための［クロック発生回路］をオープンします。
ポート	マイクロコントローラが提供しているポートの機能を制御するうえで必要な情報を設定するための［ポート］をオープンします。
割り込み	マイクロコントローラが提供している割り込み／キー割り込みの機能を制御するうえで必要な情報を設定するための［割り込み］をオープンします。
シリアル	マイクロコントローラが提供しているシリアル・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な情報を設定するための［シリアル］をオープンします。
A/D コンバータ	マイクロコントローラが提供している A/D コンバータの機能を制御するうえで必要な情報を設定するための［A/D コンバータ］をオープンします。
D/A コンバータ	マイクロコントローラが提供している D/A コンバータの機能を制御するうえで必要な情報を設定するための［D/A コンバータ］をオープンします。
タイマ	マイクロコントローラが提供しているタイマ・アレイ・ユニットの機能を制御するうえで必要な情報を設定するための［タイマ］をオープンします。
ウォッチドッグ・タイマ	マイクロコントローラが提供しているウォッチドッグ・タイマの機能を制御するうえで必要な情報を設定するための［ウォッチドッグ・タイマ］をオープンします。
リアルタイムクロック	マイクロコントローラが提供しているリアルタイムクロックの機能を制御するうえで必要な情報を設定するための［リアルタイムクロック］をオープンします。
インターバル・タイマ	マイクロコントローラが提供しているインターバル・タイマの機能を制御するうえで必要な情報を設定するための［インターバル・タイマ］をオープンします。
コンパレータ	マイクロコントローラが提供しているコンパレータの機能を制御するうえで必要な情報を設定するための［コンパレータ］をオープンします。
クロック出力／ブザー出力	マイクロコントローラが提供しているクロック出力／ブザー出力制御回路の機能を制御するうえで必要な情報を設定するための［クロック出力／ブザー出力］をオープンします。
データトランスファコントローラ	マイクロコントローラが提供している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な情報を設定するための［データトランスファコントローラ］をオープンします。

イベントリンクコントローラ	マイクロコントローラが提供している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な情報を設定するための［イベントリンクコントローラ］をオープンします。
DMA コントローラ	マイクロコントローラが提供している DMA コントローラの機能を制御するうえで必要な情報を設定するための［DMA コントローラ］をオープンします。
電圧検出回路	マイクロコントローラが提供している電圧検出回路の機能を制御するうえで必要な情報を設定するための［電圧検出回路］をオープンします。

（b）アイコン

周辺機能ノードの各文字列の直前に表示されているアイコンは，以下の意味を持ちます。

	該当コード生成 パネルに対する操作を実施済み。
	該当コード生成 パネルに対する操作が未実施。
	他の周辺機能ノードに対する操作の影響を受け，設定内容に問題が発生。

[[ヘルプ］メニュー（プロジェクト・ツリー パネル専用部分)]

プロジェクト・ツリー パネルのヘルプを開く	本パネルのヘルプを表示します。

[コンテキスト・メニュー]

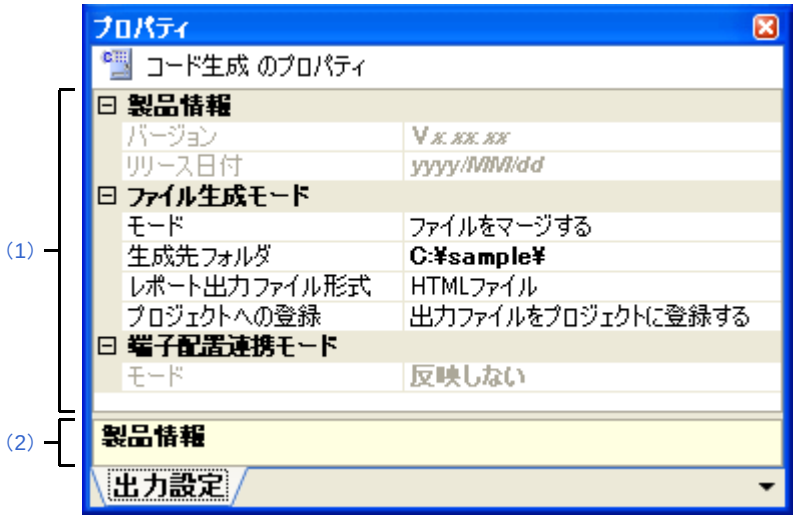
マウスを右クリックすることにより表示されるコンテキスト・メニューは，以下のとおりです。

リセット時の設定に戻す	選択された周辺機能ノードに対応した情報をデフォルトの状態に戻します。
プロパティ	選択されたノード（[コード生成（設計ツール)]）に対応した情報を保持したプロパティ パネルをオープンします。

プロパティ パネル

プロジェクト・ツリー パネルで選択したノード，コード生成 パネルでクリックした周辺機能ボタン，コード生成プレビュー パネルで選択したファイルの種類に対応した情報の表示，および設定の変更を行います。

図 A―3　プロパティ パネル（[コード生成（設計ツール）] 選択）



ここでは，次の項目について説明します。

- [オープン方法]
- [各エリアの説明]
- [[編集] メニュー（プロパティ パネル専用部分）]
- [[ヘルプ] メニュー（プロパティ パネル専用部分）]
- [コンテキスト・メニュー]

[オープン方法]

- プロジェクト・ツリー パネルにおいて，ノード（[コード生成（設計ツール）]，周辺機能ノード“[クロック発生回路]，[ポート] など”）を選択したのち，[表示] メニュー→ [プロパティ] を選択
- プロジェクト・ツリー パネルにおいて，ノード（[コード生成（設計ツール）]，周辺機能ノード“[クロック発生回路]，[ポート] など”）を選択したのち，コンテキスト・メニューから [プロパティ] を選択
- コード生成プレビュー パネルにおいて，ファイルを選択したのち，[表示] メニュー→ [プロパティ] を選択
- コード生成プレビュー パネルにおいて，ファイルを選択したのち，コンテキスト・メニューから [プロパティ] を選択

備考 1.　すでに本パネルがオープンしていた場合，プロジェクト・ツリー パネルのノード（[コード生成（設計ツール）]，周辺機能ノード“[クロック発生回路]，[ポート] など”）を選択することにより，詳細情報表示／変更エリア，および説明エリアの表示内容が該当ノードに対応したものへと切り替わります。

2. すでに本パネルがオープンしていた場合，コード生成 パネルの周辺機能ボタン（  ，  など）をクリックすることにより，詳細情報表示／変更エリア，および説明エリアの表示内容が該当ボタンに対応したものへと切り替わります。
3. すでに本パネルがオープンしていた場合，コード生成プレビュー パネルのファイルを選択することにより，詳細情報表示／変更エリア，および説明エリアの表示内容が該当ファイルに対応したものへと切り替わります。

[各エリアの説明]

(1) 詳細情報表示／変更エリア

プロジェクト・ツリー パネルで選択したノード（[コード生成（設計ツール)]，周辺機能ノード“[クロック発生回路]，[ポート] など”），コード生成 パネルでクリックした周辺機能ボタン（  ，  など），コード生成プレビュー パネルで選択したファイルの種類に対応した情報の表示，および設定の変更を行います。

なお，本エリアの表示内容については，プロジェクト・ツリー パネルで選択したノード，コード生成 パネルでクリックした周辺機能ボタン，およびコード生成プレビュー パネルで選択したファイルの種類により異なります。

各カテゴリの直前に表示されている ⊞，および ⊟ は，以下の意味を持ちます。

⊞	カテゴリ内の項目が“折りたたみ表示”されていることを示します。
⊟	カテゴリ内の項目が“展開表示”されていることを示します。

備考 1. 本エリアの表示内容についての詳細は，[出力設定] タブ，[マクロ設定] タブ，[ファイル設定] タブを参照してください。

2. ⊞ と ⊟ の切り替えは，本マークのクリック，またはカテゴリ名のダブルクリックにより実現されます。

(2) 説明エリア

詳細情報表示／変更エリアで選択されたカテゴリ，または項目に関する“簡単な説明”が表示されます。

[[編集] メニュー（プロパティ パネル専用部分)]

元に戻す	直前に行った編集作業を取り消します。
切り取り	選択している文字列を切り取り，クリップ・ボードに保存します。
コピー	選択している文字列をクリップ・ボードに保存します。
貼り付け	指定された箇所に，クリップ・ボードの内容を挿入します。
削除	選択している文字列を削除します。
すべて選択	編集中の項目に表示されている全文字列を選択します。

[[ヘルプ]メニュー（プロパティ パネル専用部分)]

プロパティ パネルのヘルプを開く	本パネルのヘルプを表示します。

[コンテキスト・メニュー]

マウスを右クリックすることにより表示されるコンテキスト・メニューは，以下のとおりです。

(1) 項目を編集中の場合

元に戻す	直前に行った編集作業を取り消します。
切り取り	選択している文字列を切り取り，クリップ・ボードに保存します。
コピー	選択している文字列をクリップ・ボードに保存します。
貼り付け	指定された箇所に，クリップ・ボードの内容を挿入します。
削除	選択している文字列を削除します。
すべて選択	編集中の項目に表示されている全文字列を選択します。

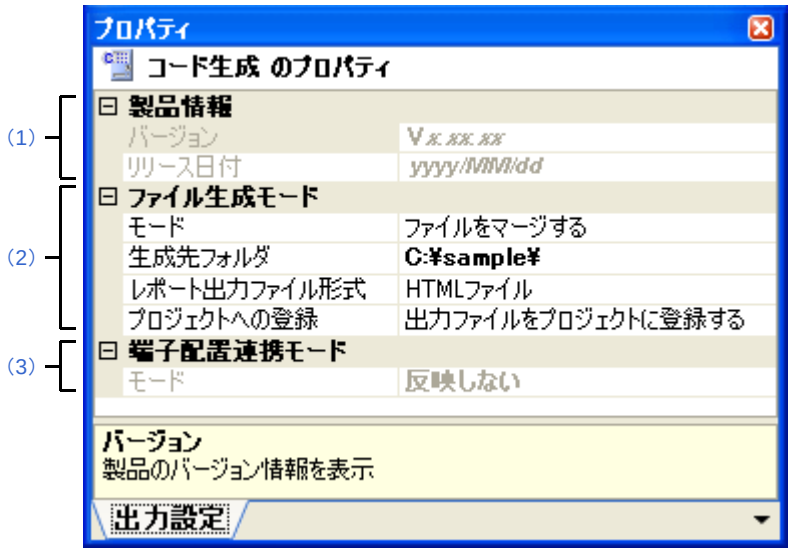
(2) 項目を編集中以外の場合

デフォルトに戻す	選択された項目をデフォルトの状態に戻します。
すべてデフォルトに戻す	すべての項目をデフォルトの状態に戻します。

［出力設定］タブ

プロジェクト・ツリー パネルで選択した［コード生成（設計ツール）］に対応した情報（製品情報，ファイル生成モード）の表示，および設定の変更を行います。

図 A—4　［出力設定］タブ



ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］

［オープン方法］

- プロジェクト・ツリー パネルにおいて，［*Project name*（プロジェクト）］→［コード生成（設計ツール）］を選択したのち，［表示］メニュー→［プロパティ］を選択
- プロジェクト・ツリー パネルにおいて，［*Project name*（プロジェクト）］→［コード生成（設計ツール）］を選択したのち，コンテキスト・メニューから［プロパティ］を選択

備考　すでに本パネルがオープンしていた場合，プロジェクト・ツリー パネルの［コード生成（設計ツール）］を選択することにより，表示内容が切り替わります。

［各エリアの説明］

(1)［製品情報］カテゴリ

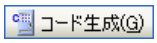
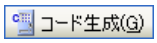
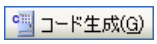
コード生成に関する製品情報（バージョン，リリース日付）の表示を行います。

バージョン	コード生成のバージョンを表示します。

リリース日付	コード生成のリリース日付を表示します。

(2)［ファイル生成モード］カテゴリ

コード生成のファイル生成モード（モード，生成先フォルダ，レポート出力ファイル形式，プロジェクトへの登録）の表示，および設定の変更を行います。

モード	 ボタンをクリックした際の動作モードを表示／選択します。 なお，［ファイル］メニュー→［コード生成レポートを保存］を選択した際の動作モードは，“ファイルを上書きする”となります。	
	ファイルを上書きする	同一のファイル名を有するファイルが既存していた場合，該当ファイルを上書きします。
	ファイルをマージする	同一のファイル名を有するファイルが既存していた場合，該当ファイルをマージします。 なお，マージする部位については， /* Start user code …. Do not edit comment generated here */ から /* End user code. Do not edit comment generated here */ で囲まれた部位に限られます。
	すでにファイルがあれば何もしない	同一のファイル名を有するファイルが既存していた場合，該当ファイルの出力を行いません。
生成先フォルダ	 ボタンをクリックした際，［ファイル］メニュー→［コード生成レポートを保存］を選択した際に出力する各種ファイル（ソース・コード，レポート・ファイル）の出力先を表示／選択します。	
レポート出力ファイル形式	［ファイル］メニュー→［コード生成レポートを保存］を選択した際に出力するレポート・ファイル（コード生成を用いて設定した情報を保持したファイル，ソース・コードに関する情報を保持したファイル）の形式を表示／選択します。	
	HTML ファイル	HTML 形式でレポート・ファイルを出力します。
	CSV ファイル	CSV 形式でレポート・ファイルを出力します。
プロジェクトへの登録	 ボタンをクリックした際に出力するソース・コードをプロジェクトに登録するか否かを選択します。	
	出力ファイルをプロジェクトに登録する	出力したソース・コードをプロジェクトに登録します。 なお，登録されたソース・コードは，プロジェクト・ツリー パネルの［ファイル］-［コード生成］ノードの直下に表示されます。
	出力ファイルをプロジェクトに登録しない	出力したソース・コードをプロジェクトに登録しません。

備考　出力先の変更は，本エリア内の［...］ボタンをクリックすることによりオープンするフォルダの参照ダイアログで行います。

(3)［端子配置連携モード］カテゴリ

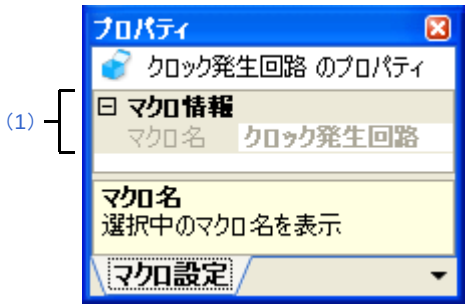
本版では，サポートしていません。

なお，本カテゴリは，グレー表記となります。

［マクロ設定］タブ

プロジェクト・ツリー パネルで選択した周辺機能ノード“[クロック発生回路]，[ポート]など”，コード生成 パネルでクリックした周辺機能ボタンの種類（  ，  など）に対応した情報（マクロ情報）の表示，および設定の変更を行います。

図 A―5 ［マクロ設定］タブ



ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］

［オープン方法］

- プロジェクト・ツリー パネルにおいて，［*Project name*（プロジェクト）］→［コード生成（設計ツール）］→周辺機能ノード“[クロック発生回路]，[ポート]など”を選択したのち，［表示］メニュー→［プロパティ］を選択
- プロジェクト・ツリー パネルにおいて，［*Project name*（プロジェクト）］→［コード生成（設計ツール）］→周辺機能ノード“[クロック発生回路]，[ポート]など”を選択したのち，コンテキスト・メニューから［プロパティ］を選択

備考 1. すでに本パネルがオープンしていた場合，プロジェクト・ツリー パネルの周辺機能ノード“[クロック発生回路]，[ポート]など”を選択することにより，表示内容が該当ノードに対応したものへと切り替わります。

2. すでに本パネルがオープンしていた場合，コード生成 パネルの周辺機能ボタン（  ，  など）をクリックすることにより，表示内容が該当ボタンに対応したものへと切り替わります。

［各エリアの説明］

(1)［マクロ情報］カテゴリ

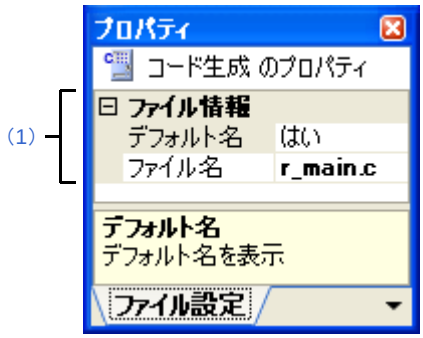
プロジェクト・ツリー パネルで選択した周辺機能ノード（[クロック発生回路]，[ポート]など），コード生成 パネルでクリックした周辺機能ボタンに関する情報（マクロ名）の表示，および設定の変更を行います。

マクロ名	プロジェクト・ツリー パネルで選択した周辺機能ノードの種類，コード生成パネルでクリックした周辺機能ボタンの種類を表示します。

［ファイル設定］タブ

コード生成プレビュー パネルで選択したファイルの種類に対応した情報（ファイル情報）の表示，および設定の変更を行います。

図 A―6 ［ファイル設定］タブ



ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］

［オープン方法］

- コード生成プレビュー パネルにおいて，ファイルを選択したのち，［表示］メニュー→［プロパティ］を選択
- コード生成プレビュー パネルにおいて，ファイルを選択したのち，コンテキスト・メニューから［プロパティ］を選択

備考　すでに本パネルがオープンしていた場合，コード生成プレビュー パネルのファイルを選択することにより，表示内容が該当ファイルに対応したものへと切り替わります。

［各エリアの説明］

(1)［ファイル情報］カテゴリ

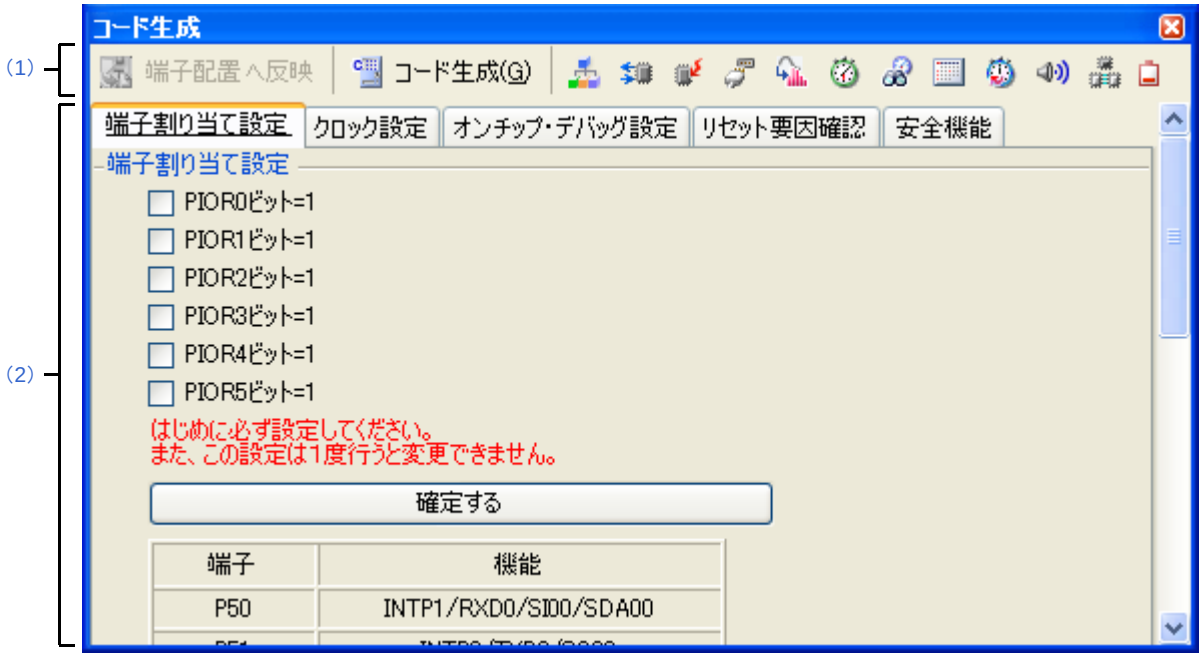
コード生成プレビュー パネルで選択したファイルに関する情報（デフォルト名，ファイル名）の表示，および設定の変更を行います。

デフォルト名	コード生成プレビュー パネルで選択したファイルのファイル名がデフォルトの名前であるか否かを表示／選択します。	
	はい	該当ファイル名は，デフォルトの名前です。 本領域を“いいえ”から“はい”へと変更した際には，該当ファイル名がデフォルトの名前へと変更されます。
	いいえ	該当ファイル名は，デフォルトの名前ではありません。
ファイル名	コード生成プレビュー パネルで選択したファイルのファイル名を表示／変更します。	

コード生成 パネル

マイクロコントローラが提供している周辺機能を制御するうえで必要な情報を設定します。

図 A―7　コード生成 パネル：[クロック発生回路]



ここでは，次の項目について説明します。

- [オープン方法]
- [各エリアの説明]
- [[ファイル] メニュー（コード生成 パネル専用部分）]
- [[ヘルプ] メニュー（コード生成 パネル専用部分）]

[オープン方法]

- プロジェクト・ツリー パネルの［*Project name*（プロジェクト）］→［コード生成（設計ツール）］→周辺機能ノード（[クロック発生回路]，［ポート］など）を選択

備考　すでに本パネルがオープンしていた場合，周辺機能ボタン（  ，  など）をクリックすることにより，情報設定エリアの表示内容が該当ボタンに対応したものへと切り替わります。

[各エリアの説明]

(1) ツールバー

本エリアは，以下に示したボタン群“周辺機能ボタン”から構成されています。

なお，対象マイクロコントローラが未サポートの周辺機能については，該当周辺機能ボタンが表示されません。

端子配置へ反映	本版では，サポートしていません。 なお，本ボタンは，グレー表記（非選択状態）となります。
コード生成(G)	プロパティ パネルの［出力設定］タブ→［生成先フォルダ］で指定されたフォルダにソース・コード（デバイス・ドライバ・プログラム）を出力します。
	情報設定エリアの表示内容を“マイクロコントローラが提供している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な情報を設定するための［クロック発生回路］”へと切り替えます。
	情報設定エリアの表示内容を“マイクロコントローラが提供しているポートの機能を制御するうえで必要な情報を設定するための［ポート］”へと切り替えます。
	情報設定エリアの表示内容を“マイクロコントローラが提供している割り込み／キー割り込みの機能を制御するうえで必要な情報を設定するための［割り込み］”へと切り替えます。
	情報設定エリアの表示内容を“マイクロコントローラが提供しているシリアル・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な情報を設定するための［シリアル］”へと切り替えます。
	情報設定エリアの表示内容を“マイクロコントローラが提供している A/D コンバータの機能を制御するうえで必要な情報を設定するための［A/D コンバータ］”へと切り替えます。
	情報設定エリアの表示内容を“マイクロコントローラが提供している D/A コンバータの機能を制御するうえで必要な情報を設定するための［D/A コンバータ］”へと切り替えます。
	情報設定エリアの表示内容を“マイクロコントローラが提供しているタイマ・アレイ・ユニットの機能を制御するうえで必要な情報を設定するための［タイマ］”へと切り替えます。
	情報設定エリアの表示内容を“マイクロコントローラが提供しているウォッチドッグ・タイマの機能を制御するうえで必要な情報を設定するための［ウォッチドッグ・タイマ］”へと切り替えます。
	情報設定エリアの表示内容を“マイクロコントローラが提供しているリアルタイムクロックの機能を制御するうえで必要な情報を設定するための［リアルタイムクロック］”へと切り替えます。
	情報設定エリアの表示内容を“マイクロコントローラが提供しているインターバル・タイマの機能を制御するうえで必要な情報を設定するための［インターバル・タイマ］”へと切り替えます。
	情報設定エリアの表示内容を“マイクロコントローラが提供しているコンパレータの機能を制御するうえで必要な情報を設定するための［コンパレータ］”へと切り替えます。
	情報設定エリアの表示内容を“マイクロコントローラが提供しているクロック出力／ブザー出力制御回路の機能を制御するうえで必要な情報を設定するための［クロック出力／ブザー出力］”へと切り替えます。

	情報設定エリアの表示内容を“マイクロコントローラが提供している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な情報を設定するための［データトランスファコントローラ］”へと切り替えます。
	情報設定エリアの表示内容を“マイクロコントローラが提供している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な情報を設定するための［イベントリンクコントローラ］”へと切り替えます。
	情報設定エリアの表示内容を“マイクロコントローラが提供している DMA コントローラの機能を制御するうえで必要な情報を設定するための［DMA コントローラ］”へと切り替えます。
	情報設定エリアの表示内容を“マイクロコントローラが提供している電圧検出回路の機能を制御するうえで必要な情報を設定するための［電圧検出回路］”へと切り替えます。

（2）情報設定エリア

本エリアの表示内容については，本パネルをオープンする際に選択／クリックする“周辺機能ノード”，または“周辺機能ボタン”の種類により異なります。

なお，設定項目についての詳細は，マイクロコントローラのユーザーズ・マニュアルを参照してください。

［［ファイル］メニュー（コード生成 パネル専用部分）］

コード生成レポート を保存	レポート・ファイル（コード生成を用いて設定した情報を保持したファイル，ソース・コードに関する情報を保持したファイル）を出力します。

備考 1. レポート・ファイルの出力形式はプロパティ パネルの［出力設定］タブ→［レポート出力ファイル形式］で選択された形式（HTML 形式，または CSV 形式）となります。

2. レポート・ファイルの出力先は，プロパティ パネルの［出力設定］タブ→［生成先フォルダ］で指定されたフォルダとなります。

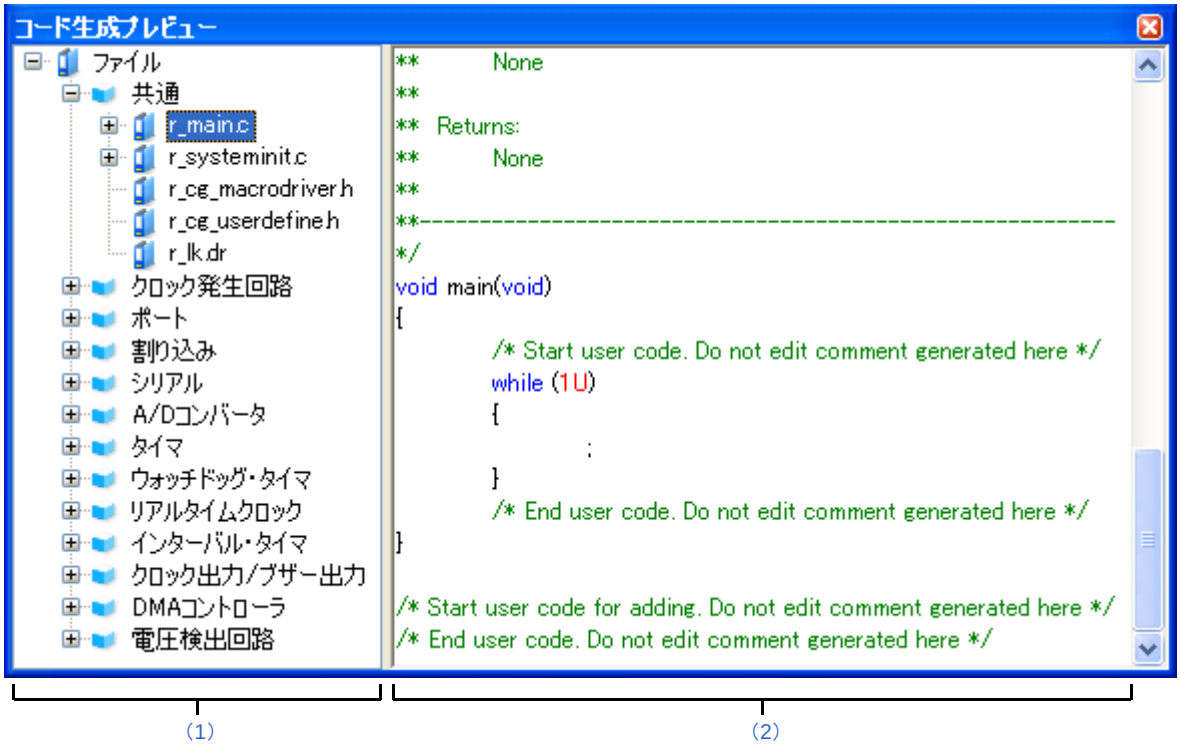
［［ヘルプ］メニュー（コード生成 パネル専用部分）］

コード生成 パネルのヘルプを開く	本パネルのヘルプを表示します。

コード生成プレビュー パネル

コード生成 パネルの コード生成(G) ボタンをクリックした際に出力されるソース・コード（デバイス・ドライバ・プログラム）の出力有無を API 関数単位で確認／設定するとともに，コード生成 パネルで設定した情報に応じたソース・コードの確認を行います。

図 A―8　コード生成プレビュー パネル



ここでは，次の項目について説明します。

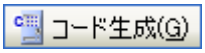
- ［オープン方法］
- ［各エリアの説明］
- ［［ファイル］メニュー（コード生成プレビュー パネル専用部分）］
- ［［ヘルプ］メニュー（コード生成プレビュー パネル専用部分）］
- ［コンテキスト・メニュー］

［オープン方法］

- ［表示］メニュー→［コード生成プレビュー］を選択

［各エリアの説明］

(1) プレビュー・ツリー

コード生成 パネルの  ボタンをクリックした際に出力されるソース・コード（デバイス・ドライバ・プログラム）の出力有無を API 関数単位で確認／設定します。

備考 1. 本ツリー内のソース・ファイル名，または API 関数名を選択することにより，ソース・コードの表示を切り替えることができます。

2. 出力有無の設定は，ツリー内のアイコン上にマウス・カーソルを移動した際，マウスを右クリックすることにより表示されるコンテキスト・メニュー（コードを生成する，コードを生成しない）から行います。

3. 出力有無の設定状況については，アイコン種別により確認することができます。

表 A—2　ソース・コードの出力有無

アイコン種別	概要
	該当 API 関数のソース・コードは，出力されます。 なお，本アイコンが表示されている API 関数は，ソース・コードの出力が必須（  への変更不可）となります。
	該当 API 関数のソース・コードは，出力されます。
	該当 API 関数のソース・コードは，出力されません。

(2) ソース・コード表示エリア

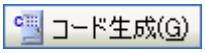
コード生成 パネルで設定した情報に応じたソース・コード（デバイス・ドライバ・プログラム）の確認を行います。

なお，本エリアに表示されるソース・コードの文字色は，以下の意味を持ちます。

表 A—3　ソース・コードの文字色

文字色	概要
緑	コメント文
青	C コンパイラの予約語
赤	数値
黒	コード部
グレー	ファイル名

備考 1. 本パネル内でソース・コードを編集することはできません。

2. 一部の API 関数（シリアル・アレイ・ユニット用 API 関数など）については，ソース・コードの出力時（コード生成 パネルの  ボタンをクリックした際）にレジスタ値 SFR などが計算され確定するものがあります。このため，本パネルに表示されるソース・コードは，実際に出力されるソース・コードと一致しない場合があります。

3. プレビュー・ツリー内のソース・ファイル名，またはAPI関数名を選択することにより，ソース・コードの表示を切り替えることができます。

[[ファイル]メニュー（コード生成プレビュー パネル専用部分）]

コード生成レポート を保存	レポート・ファイル（コード生成を用いて設定した情報を保持したファイル，ソース・コードに関する情報を保持したファイル）を出力します。

備考 1. レポート・ファイルの出力形式はプロパティ パネルの［出力設定］タブ→［レポート出力ファイル形式］で選択された形式（HTML形式，またはCSV形式）となります。

2. レポート・ファイルの出力先は，プロパティ パネルの［出力設定］タブ→［生成先フォルダ］で指定されたフォルダとなります。

[[ヘルプ]メニュー（コード生成プレビュー パネル専用部分）]

コード生成プレビュー パネルのヘルプを開く	本パネルのヘルプを表示します。

[コンテキスト・メニュー]

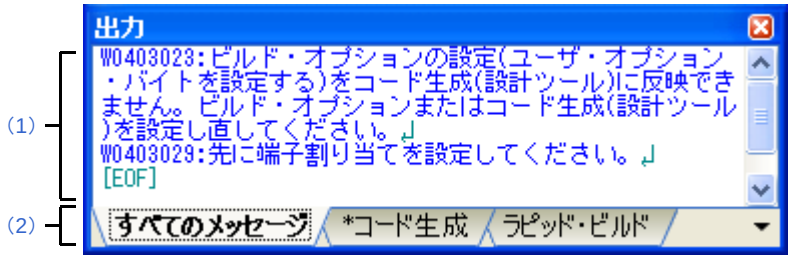
マウスを右クリックすることにより表示されるコンテキスト・メニューは，以下のとおりです。

コードを生成する	選択されたAPI関数のソース・コードをプロパティ パネルの［出力設定］タブ→［生成先フォルダ］で指定されたフォルダに出力するための設定を行います。 なお，本コンテキスト・メニューをクリックすることにより，該当API関数のアイコンは，  から  へと変化します。
コードを生成しない	選択されたAPI関数のソース・コードをコード生成 パネルの  ボタンがクリックされた際に出力しないための設定を行います。 なお，本コンテキスト・メニューをクリックすることにより，該当API関数のアイコンは，  から  へと変化します。
名前を変更する	選択されたファイル名／API関数名の部位が該当名称を編集するためにエディット・ボックス化されます。 該当エディット・ボックスの文字列を編集することにより，ファイル名／API関数名が変更されます。
名前を元に戻す	選択されたファイル名／API関数名を編集前の状態に戻します。
プロパティ	選択されたファイルに対応した情報を保持したプロパティ パネルをオープンします。

出力 パネル

CubeSuite+ が提供している各種コンポーネント（設計ツール，ビルド・ツールなど）の操作ログを表示します。

図 A—9　出力 パネル



ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］
- ［［ファイル］メニュー（出力 パネル専用部分）］
- ［［編集］メニュー（出力 パネル専用部分）］
- ［［ヘルプ］メニュー（出力 パネル専用部分）］
- ［コンテキスト・メニュー］

［オープン方法］

- ［表示］メニュー→［出力］を選択

［各エリアの説明］

（1）メッセージ・エリア

CubeSuite+ が提供している各種コンポーネント（設計ツール，ビルド・ツールなど）の操作ログを表示します。

なお，本エリアに表示されるメッセージの文字色／背景色は，以下の意味を持ちます。

表 A—4　メッセージの文字色／背景色

文字色／背景色	概要
黒／白	通常メッセージ 何らかの情報を通知する際に表示されます。
青／標準色	警告メッセージ 何らかの警告を通知する際に表示されます。

文字色／背景色	概要
赤／薄グレー	エラー・メッセージ 致命的なエラーの発生を通知する際，または操作ミスにより実行が不可能となった場合に表示されます。

備考　本エリアの表示内容についての詳細は，［すべてのメッセージ］タブ，［コード生成］タブを参照してください。

(2) タブ選択エリア

メッセージの出力元を選択します。

備考　新しいメッセージが出力された場合，タブ名の直前に“*”マークが表示されます。

［［ファイル］メニュー（出力 パネル専用部分）］

出力 - *タブ名* を保存	該当タブのメッセージを既存のファイルに上書き保存します。
名前を付けて 出力 - *タブ名* を保存 ...	該当タブのメッセージに名前を付けて保存するための名前を付けて保存 ダイアログをオープンします。

［［編集］メニュー（出力 パネル専用部分）］

コピー	選択している文字列をクリップ・ボードに保存します。
すべて選択	メッセージ・エリアに表示されている全文字列を選択します。
検索 ...	文字列検索を行うための検索・置換 ダイアログを［クイック検索］タブが選択された状態でオープンします。
置換 ...	文字列置換を行うための検索・置換 ダイアログを［一括置換］タブが選択された状態でオープンします。

［［ヘルプ］メニュー（出力 パネル専用部分）］

出力 パネルのヘルプを開く	本パネルのヘルプを表示します。

［コンテキスト・メニュー］

マウスを右クリックすることにより表示されるコンテキスト・メニューは，以下のとおりです。

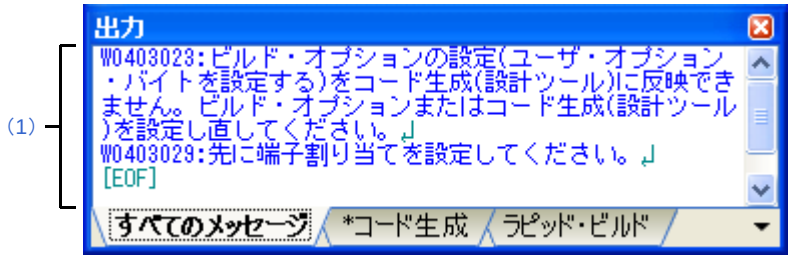
コピー	選択している文字列をクリップ・ボードに保存します。
すべて選択	メッセージ・エリアに表示されている全文字列を選択します。
クリア	メッセージ・エリアに表示されている全文字列を消去します。
検索の中止	実行中の文字列検索を中止します。 文字列検索を非実行中の場合，本項目は無効となります。

メッセージに関するヘルプ	メッセージに対応したヘルプを表示します。 ただし，本項目の選択は，キャレットが警告メッセージ／エラー・メッセージの表示行にある場合に限られます。

［すべてのメッセージ］タブ

CubeSuite+ が提供している全コンポーネント（設計ツール，ビルド・ツールなど）の操作ログを表示します。

図 A—10　［すべてのメッセージ］タブ



ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］

［オープン方法］

- ［表示］メニュー→［出力］を選択

［各エリアの説明］

(1) メッセージ・エリア

CubeSuite+ が提供している全コンポーネント（設計ツール，ビルド・ツールなど）の操作ログを表示します。

なお，本エリアに表示されるメッセージの文字色／背景色は，以下の意味を持ちます。

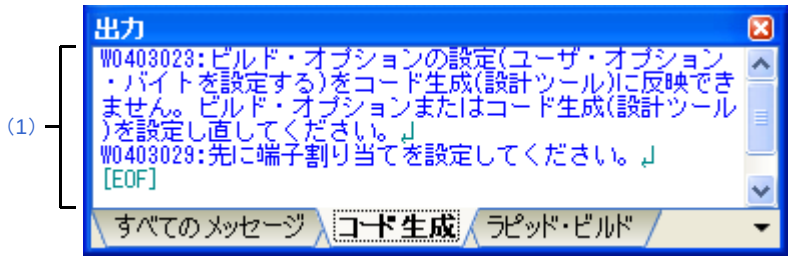
表 A—5　メッセージの文字色／背景色

文字色／背景色	概要
黒／白	通常メッセージ 何らかの情報を通知する際に表示されます。
青／標準色	警告メッセージ 何らかの警告を通知する際に表示されます。
赤／薄グレー	エラー・メッセージ 致命的なエラーの発生を通知する際，または操作ミスにより実行が不可能となった場合に表示されます。

［コード生成］タブ

CubeSuite+ が提供している各種コンポーネント（設計ツール，ビルド・ツールなど）のうち，コード生成に限定した操作ログを表示します。

図 A—11 ［コード生成］タブ



ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］

［オープン方法］

- ［表示］メニュー→［出力］を選択

［各エリアの説明］

(1) メッセージ・エリア

CubeSuite+ が提供している各種コンポーネント（設計ツール，ビルド・ツールなど）のうち，コード生成に限定した操作ログを表示します。

なお，本エリアに表示されるメッセージの文字色／背景色は，以下の意味を持ちます。

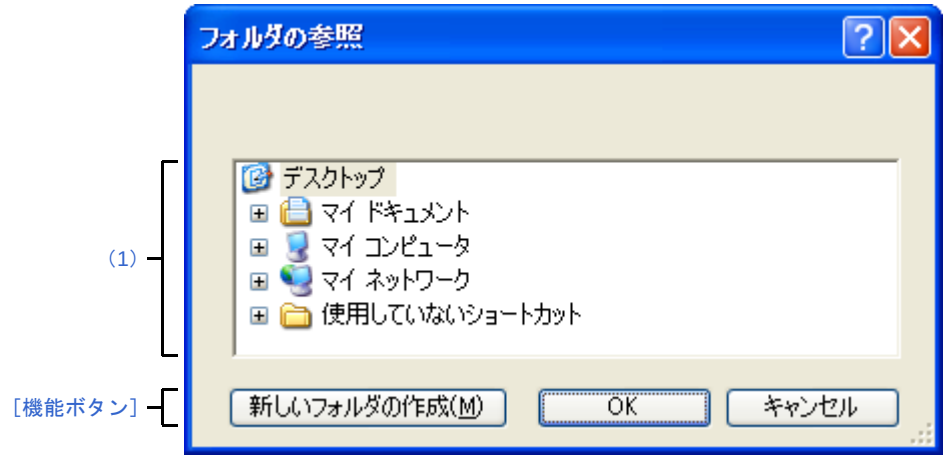
表 A—6　メッセージの文字色／背景色

文字色／背景色	概要
黒／白	通常メッセージ 何らかの情報を通知する際に表示されます。
青／標準色	警告メッセージ 何らかの警告を通知する際に表示されます。
赤／薄グレー	エラー・メッセージ 致命的なエラーの発生を通知する際，または操作ミスにより実行が不可能となった場合に表示されます。

フォルダの参照 ダイアログ

ファイル（ソース・コード，レポート・ファイルなど）の出力先を設定します。

図 A—12　フォルダの参照 ダイアログ



ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］
- ［機能ボタン］

［オープン方法］

- プロパティ パネルの［出力設定］タブにおいて，［生成先フォルダ］の［...］ボタンをクリック

［各エリアの説明］

（1）保存する場所エリア

ファイル（ソース・コード，レポート・ファイルなど）を出力するフォルダを選択します。

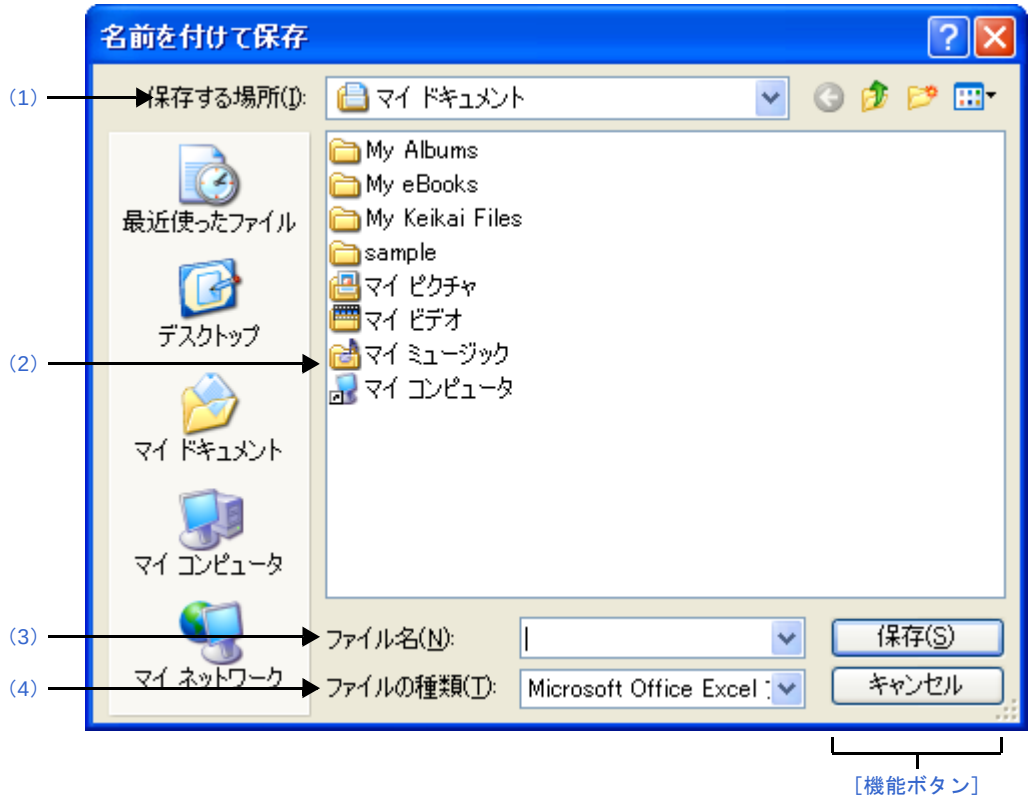
［機能ボタン］

ボタン	機能
新しいフォルダの作成	保存する場所エリアで選択されたフォルダの直下に“新しいフォルダ”を新規に作成します。
OK	ファイルの出力先を保存する場所エリアで選択されたフォルダに設定します。
キャンセル	本ダイアログをクローズします。

名前を付けて保存 ダイアログ

ファイル（レポート・ファイルなど）に名前を付けて保存します。

図 A—13　名前を付けて保存 ダイアログ



ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］
- ［機能ボタン］

［オープン方法］

- ［ファイル］メニュー→［名前を付けて *＜対象＞* を保存］を選択

［各エリアの説明］

(1) ［保存する場所］

ファイル（レポート・ファイルなど）を出力するフォルダを選択します。

(2) ファイルの一覧エリア

［保存する場所］，および［ファイルの種類］で選択された条件に合致するファイルの一覧を表示します。

(3)［ファイル名］

出力するファイルのファイル名を指定します。

(4)［ファイルの種類］

出力するファイルの種類（ファイル・タイプ）を選択します。

Microsoft Office Excel ブック（*.xls）	Microsoft Office Excel ブック形式
ビットマップ（*.bmp）	ビット・マップ形式
PNG（*.png）	PNG 形式
JPEG（*.jpg）	JPEG 形式
EMF（*.emf）	EMF 形式

［機能ボタン］

ボタン	機能
保存	［保存する場所］で指定されたフォルダに［ファイル名］，および［ファイルの種類］で指定された名前のファイルを出力します。
キャンセル	本ダイアログをクローズします。

付録 B　出力ファイル

本付録では，コード生成が出力するファイルについて説明します。

B. 1　概　　要

以下に，コード生成が出力するファイルの一覧を示します。

表 B—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_ *周辺機能名* .c	初期化関数，API 関数
	r_ *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_macrodriver.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

B. 2　出力ファイル

以下に，コード生成が出力するファイル（各周辺機能）を示します。

表 B—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_cgc.c	R_CGC_Create R_CGC_Set_ClockMode R_CGC_Set_CRCOn
	r_cgc_user.c	R_CGC_Create_UserInit R_CGC_Get_ResetSource
	r_cg_cgc.h	—
ポート	r_port.c	R_PORT_Create
	r_port_user.c	R_PORT_Create_UserInit
	r_cg_port.h	—
割り込み	r_intc.c	R_INTC_Create R_INTCn_Start

周辺機能	ファイル名	含まれる API 関数名
割り込み	r_intc.c	R_INTCn_Stop R_KEY_Create R_KEY_Start R_KEY_Stop
	r_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_serial.c	R_SAUm_Create R_SAUm_Set_PowerOff R_SAU0_Set_SnoozeOn R_SAU0_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_IICAn_Create R_IICAn_StopCondition R_IICAn_Stop R_IICAn_Set_PowerOff R_IICAn_Master_Send R_IICAn_Master_Receive R_IICAn_Slave_Send R_IICAn_Slave_Receive
	r_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

周辺機能	ファイル名	含まれる API 関数名
シリアル	r_serial_user.c	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_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_adc.c	R_ADC_Create R_ADC_Set_ComparatorOn R_ADC_Set_ComparatorOff 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 R_ADC_Get_Result R_ADC_Get_Result_8bit
	r_adc_user.c	R_ADC_Create_UserInit R_ADC_Interrupt
	r_cg_adc.h	—
D/A コンバータ	r_dac.c	R_DAC_Create R_DACn_Start R_DACn_Stop R_DAC_Set_PowerOff R_DACn_Set_ConversionValue
	r_dac_user.c	R_DAC_Create_UserInit
	r_cg_dac.h	—
タイマ	r_timer.c	R_TAUm_Create

周辺機能	ファイル名	含まれる API 関数名
タイマ	r_timer.c	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_Get_PulseWidth R_TMR_RGn_Create R_TMR_RGn_Start R_TMR_RGn_Stop R_TMR_RGn_Set_PowerOff R_TMR_RGn_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_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_RGn_Create_UserInit R_TMR_RGn_Interrupt R_TMR_RJ0_Create_UserInit R_TMR_RJ0_Interrupt
	r_cg_timer.h	—
ウォッチドッグ・タイマ	r_wdt.c	R_WDT_Create R_WDT_Restart
	r_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_rtc.c	R_RTC_Create R_RTC_Start R_RTC_Stop R_RTC_Set_PowerOff

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_rtc.c	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_rtc_user.c	R_RTC_Create_UserInit R_RTC_Interrupt R_RTC_Callback_ConstPeriod R_RTC_Callback_Alarm
	r_cg_rtc.h	—
インターバル・タイマ	r_it.c	R_IT_Create R_IT_Start R_IT_Stop R_IT_Set_PowerOff
	r_it_user.c	R_IT_Create_UserInit R_IT_Interrupt
	r_cg_it.h	—
コンパレータ	r_comp.c	R_COMP_Create R_COMPn_Start R_COMPn_Stop
	r_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_pclbuz.c	R_PCLBUZn_Create R_PCLBUZn_Start R_PCLBUZn_Stop
	r_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_dtc.c	R_DTC_Create R_DTCn_Start R_DTCn_Stop
	r_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_elc.c	R_ELC_Create R_ELC_Stop

周辺機能	ファイル名	含まれる API 関数名
イベントリンクコントローラ	r_elc_user.c	R_ELC_Create_UserInit
	r_cg_elc.h	—
DMA コントローラ	r_dmac.c	R_DMACn_Create R_DMACn_Start R_DMACn_Stop R_DMACn_SoftwareTriggerOn
	r_dmac_user.c	R_DMACn_Create_UserInit R_DMACn_Interrupt
	r_cg_dmac.h	—
電圧検出回路	r_lvd.c	R_LVD_Create R_LVD_InterruptMode_Start
	r_lvd_user.c	R_LVD_Create_UserInit R_LVD_Interrupt
	r_cg_lvd.h	—

付録 C　API 関数

本付録では，コード生成が出力する API 関数について説明します。

C. 1　概　　要

以下に，コード生成が API 関数を出力する際の命名規則を示します。

- マクロ名

  すべて大文字。

  なお，先頭に“数字”が付与されている場合，該当数字（16 進数値）とマクロ値は同値。

- ローカル変数名

  すべて小文字。

- グローバル変数名

  先頭に“g”を付与し，構成単語の先頭のみ大文字。

- ローカル変数へのポインタ名

  先頭に“p”を付与し，すべて小文字。

- 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。

C. 2　出力関数

以下に，コード生成が出力する API 関数の一覧を示します。

表 C—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。

周辺機能	API 関数名	機能概要
クロック発生回路	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。
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_SAU0_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_SAU0_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。

周辺機能	API 関数名	機能概要
シリアル	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。
	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。

周辺機能	API 関数名	機能概要
シリアル	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_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。
	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。

周辺機能	API 関数名	機能概要
A/D コンバータ	R_ADC_Set_ComparatorOn	電圧コンパレータを動作許可状態に設定します。
	R_ADC_Set_ComparatorOff	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。
	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。

周辺機能	API 関数名	機能概要
タイマ	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	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	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	タイマ RD*n* のカウント処理を開始します。
	R_TMR_RDn_Stop	タイマ RD*n* のカウント処理を終了します。
	R_TMR_RDn_Set_PowerOff	タイマ RD*n* に対するクロック供給を停止します。
	R_TMR_RDn_Get_PulseWidth	タイマ RD*n* のパルス幅を読み出します。
	R_TMR_RGn_Create	タイマ RG*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_RGn_Create_UserInit	タイマ RG*n* に関するユーザ独自の初期化処理を行います。
	R_TMR_RGn_Interrupt	タイマ割り込みの発生に伴う処理を行います。
	R_TMR_RGn_Start	タイマ RG*n* のカウント処理を開始します。
	R_TMR_RGn_Stop	タイマ RG*n* のカウント処理を終了します。
	R_TMR_RGn_Set_PowerOff	タイマ RG*n* に対するクロック供給を停止します。
	R_TMR_RGn_Get_PulseWidth	タイマ RG*n* のパルス幅を読み出します。
	R_TMR_RJ0_Create	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	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	タイマ RJ0 のカウント処理を開始します。
	R_TMR_RJ0_Stop	タイマ RJ0 のカウント処理を終了します。
	R_TMR_RJ0_Set_PowerOff	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	タイマ RJ0 のパルス幅を読み出します。

周辺機能	API 関数名	機能概要
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。
	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。

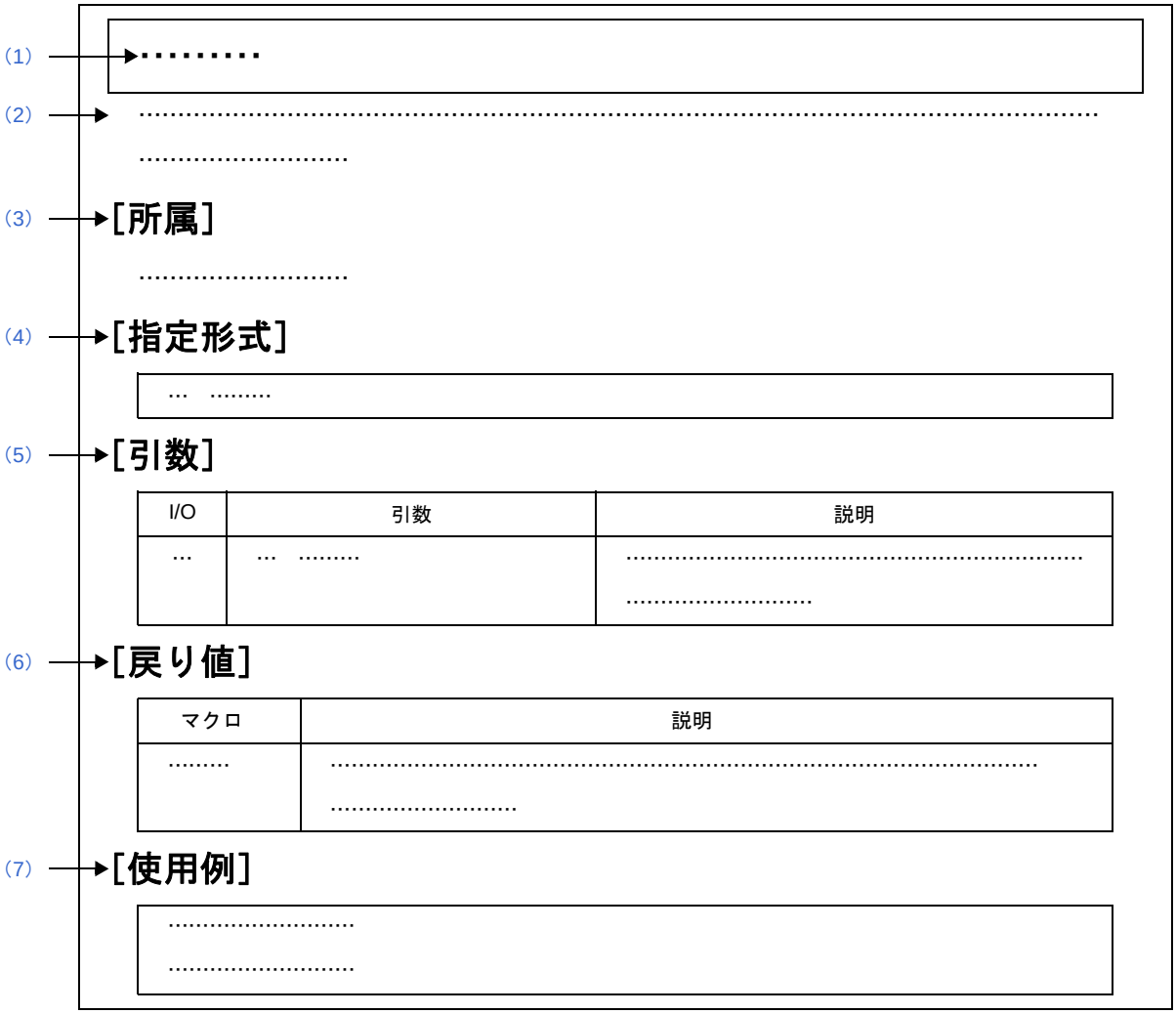
周辺機能	API 関数名	機能概要
リアルタイムクロック	R_RTC_Set_RTC1HZOff	RTC1HZ 端子に対する補正クロック（1 Hz）の出力を禁止します。
インターバル・タイマ	R_IT_Create	インターバル・タイマの機能を制御するうえで必要となる初期化処理を行います。
	R_IT_Create_UserInit	インターバル・タイマに関するユーザ独自の初期化処理を行います。
	R_IT_Interrupt	インターバル・タイマ割り込み INTIT の発生に伴う処理を行います。
	R_IT_Start	インターバル・タイマのカウントを開始します。
	R_IT_Stop	インターバル・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。
	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。
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。

周辺機能	API 関数名	機能概要
イベントリンクコントローラ	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_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。
	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）。

C. 3 関数リファレンス

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

図 C—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。

C. 3. 1 クロック発生回路

以下に，コード生成がクロック発生回路用として出力する API 関数の一覧を示します。

表 C—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_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_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_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_cgc.c

[指定形式]

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

[引数]

I/O	引数	説明
I	enum ClockMode *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

[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_cgc.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

C. 3. 2 ポート

以下に，コード生成がポート用として出力する API 関数の一覧を示します。

表 C—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_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_port_user.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

C. 3. 3 割り込み

以下に，コード生成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 C—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_intc.c

[指定形式]

```
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_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_intc_user.c

[指定形式]

```
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_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_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_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_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_intc_user.c

[指定形式]

```
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_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_intc.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

C. 3. 4 シリアル

以下に，コード生成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 C—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_SAU0_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_SAU0_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_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。
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_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_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_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_SAU0_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_serial.c

[指定形式]

```
void    R_SAU0_Set_SnoozeOn ( void );
```

[引数]

なし

[戻り値]

なし

R_SAU0_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_serial.c

[指定形式]

```
void    R_SAU0_Set_SnoozeOff ( void );
```

[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_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_serial_user.c

[指定形式]

```
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_serial_user.c

[指定形式]

```
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

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として呼び出されます。

[所属]

r_serial_user.c

[指定形式]

```
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_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_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 *txbuf* で指定されたバッファから 1 バイト単位の UART 送信を引数 *txnum* で指定された回数だけ繰り返し行います。

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_serial.c

[指定形式]

```
#include    "r_cg_macrodriver.h"
MD_STATUS   R_UARTn_Send ( uint8_t *txbuf, uint16_t txnum );
```

備考 *n* は，チャネル番号を意味します。

[引数]

I/O	引数	説明
I	`uint8_t *txbuf;`	送信するデータを格納したバッファへのポインタ
I	`uint16_t txnum;`	送信するデータの総数

[戻り値]

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

R_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 *rxnum* で指定された回数だけ繰り返し行い，引数 *rxbuf* で指定されたバッファに格納します。

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_serial.c

[指定形式]

```
#include    "r_cg_macrodriver.h"
MD_STATUS   R_UARTn_Receive ( uint8_t *rxbuf, uint16_t rxnum );
```

備考 *n* は，チャネル番号を意味します。

[引数]

I/O	引数	説明
O	`uint8_t *rxbuf;`	受信したデータを格納するバッファへのポインタ
I	`uint16_t rxnum;`	受信するデータの総数

[戻り値]

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

R_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 *txnum* 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_serial_user.c

[指定形式]

```
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 *rxnum* 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_serial_user.c

[指定形式]

```
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

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 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_serial_user.c

[指定形式]

```
#include    "r_cg_macrodriver.h"
void    R_UART*n*_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*;`	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 *rxnum* 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_serial_user.c

[指定形式]

```
#include    "r_cg_macrodriver.h"
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 *rxnum* 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_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_serial_user.c

[指定形式]

```
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_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_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 *txbuf* 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 *txnum* 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_serial.c

[指定形式]

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

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 *txbuf;`	送信するデータを格納したバッファへのポインタ
I	`uint16_t txnum;`	送信するデータの総数

[戻り値]

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

R_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 *rxnum* で指定された回数だけ繰り返し行い，引数 *rxbuf* で指定されたバッファに格納します。

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_serial.c

[指定形式]

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

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 *rxbuf;	受信したデータを格納するバッファへのポインタ
I	uint16_t rxnum;	受信するデータの総数

[戻り値]

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

R_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 *txbuf* 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 *txnum* 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 *txnum* で指定された回数だけ繰り返し行い，引数 *rxbuf* で指定されたバッファに格納します。

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_serial.c

[指定形式]

```
#include    "r_cg_macrodriver.h"
MD_STATUS   R_CSImn_Send_Receive ( uint8_t *txbuf, uint16_t txnum, uint8_t *rxbuf );
```

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 *txbuf;	送信するデータを格納したバッファへのポインタ
I	uint16_t txnum;	送受信するデータの総数
O	uint8_t *rxbuf;	受信したデータを格納するバッファへのポインタ

[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 *txnum* 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_serial_user.c

[指定形式]

```
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 *rxnum* 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_serial_user.c

[指定形式]

```
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_serial_user.c

[指定形式]

```
#include    "r_cg_macrodriver.h"
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*;	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_serial.c

[指定形式]

```
void    R_IICmn_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_serial_user.c

[指定形式]

```
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_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_serial.c

[指定形式]

```
void    R_IIC*mn*_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_serial.c

[指定形式]

```
void    R_IIC*mn*_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 *txbuf* 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 *txnum* で指定された回数だけ繰り返し行います。

[所属]

r_serial.c

[指定形式]

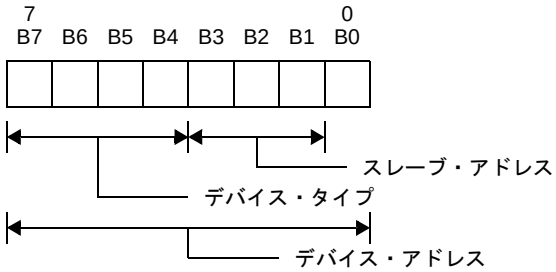
```
#include    "r_cg_macrodriver.h"
void    R_IIC mn_Master_Send ( uint8_t adr, uint8_t *txbuf, uint16_t txnum );
```

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 *txbuf;	送信するデータを格納したバッファへのポインタ
I	uint16_t *txnum*;	送信するデータの総数

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 *rxnum* で指定された回数だけ繰り返し行い，引数 *rxbuf* で指定されたバッファに格納します。

[所属]

r_serial.c

[指定形式]

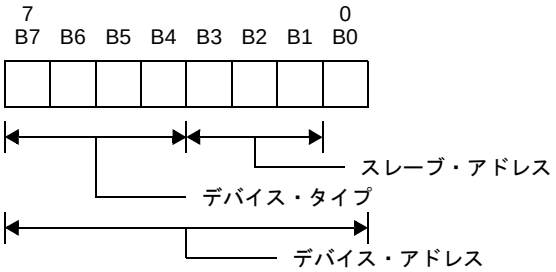
```
#include    "r_cg_macrodriver.h"
void    R_IIC mn_Master_Receive ( uint8_t adr, uint8_t *rxbuf, uint16_t rxnum );
```

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 **rxbuf*;	受信したデータを格納するバッファへのポインタ
I	uint16_t *rxnum*;	受信するデータの総数

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 *txnum* 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_serial_user.c

[指定形式]

```
void    R_IIC*mn*_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 *rxnum* 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_serial_user.c

[指定形式]

```
void    R_IIC*mn*_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_serial_user.c

[指定形式]

```
#include    "r_cg_macrodriver.h"
void    R_IIC*mn*_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_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_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_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_serial_user.c

[指定形式]

```
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_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_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_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 *txbuf* 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 *txnum* で指定された回数だけ繰り返し行います。

[所属]

r_serial.c

[指定形式]

```
#include    "r_cg_macrodriver.h"
MD_STATUS   R_IICAn_Master_Send ( uint8_t adr, uint8_t *txbuf, uint16_t txnum, 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 **txbuf*;	送信するデータを格納したバッファへのポインタ
I	uint16_t *txnum*;	送信するデータの総数
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 *rxnum* で指定された回数だけ繰り返し行い，引数 *rxbuf* で指定されたバッファに格納します。

[所属]

r_serial.c

[指定形式]

```
#include    "r_cg_macrodriver.h"
MD_STATUS   R_IICAn_Master_Receive ( uint8_t adr, uint8_t *rxbuf, uint16_t rxnum, 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 **rxbuf*;	受信したデータを格納するバッファへのポインタ
I	uint16_t *rxnum*;	受信するデータの総数
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_serial_user.c

[指定形式]

```
void    R_IICAn_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_serial_user.c

[指定形式]

```
void    R_IICAn_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_serial_user.c

[指定形式]

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

備考 *n* はチャネル番号を意味します。

[引数]

I/O	引数	説明
I	MD_STATUS *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 *txbuf* 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 *txnum* で指定された回数だけ繰り返し行います。

[所属]

r_serial.c

[指定形式]

```
#include    "r_cg_macrodriver.h"
void    R_IICAn_Slave_Send ( uint8_t *txbuf, uint16_t txnum );
```

備考 *n* はチャネル番号を意味します。

[引数]

I/O	引数	説明
I	uint8_t *txbuf;	送信するデータを格納したバッファへのポインタ
I	uint16_t txnum;	送信するデータの総数

[戻り値]

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 *rxnum* で指定された回数だけ繰り返し行い，引数 *rxbuf* で指定されたバッファに格納します。

[所属]

r_serial.c

[指定形式]

```
#include    "r_cg_macrodriver.h"
void    R_IICAn_Slave_Receive ( uint8_t *rxbuf, uint16_t rxnum );
```

備考 *n* はチャネル番号を意味します。

[引数]

I/O	引数	説明
O	`uint8_t *rxbuf;`	受信したデータを格納するバッファへのポインタ
I	`uint16_t rxnum;`	受信するデータの総数

[戻り値]

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_serial_user.c

[指定形式]

```
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_serial_user.c

[指定形式]

```
void    R_IICAn_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_serial_user.c

[指定形式]

```
#include    "r_cg_macrodriver.h"
void    R_IICAn_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_serial_user.c

[指定形式]

```
void    R_IICAn_Callback_GetStopCondition ( void );
```

備考 *n* はチャネル番号を意味します。

[引数]

なし

[戻り値]

なし

C. 3. 5 A/D コンバータ

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

表 C—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_ComparatorOn	電圧コンパレータを動作許可状態に設定します。
R_ADC_Set_ComparatorOff	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_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_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_adc_user.c

[指定形式]

```
void    R_ADC_Interrupt ( void );
```

[引数]

なし

[戻り値]

なし

R_ADC_Set_ComparatorOn

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_adc.c

［指定形式］

```
void    R_ADC_Set_ComparatorOn ( void );
```

［引数］

なし

［戻り値］

なし

R_ADC_Set_ComparatorOff

電圧コンパレータを動作停止状態に設定します。

[所属]

r_adc.c

[指定形式]

```
void    R_ADC_Set_ComparatorOff ( 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_ComparatorOn 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_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_ComparatorOff を呼び出す必要があります。

[所属]

r_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_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_adc.c

［指定形式］

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

［引数］

I/O	引数	説明
I	enum ADChannel *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_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_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_adc.c

[指定形式]

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

[引数]

I/O	引数	説明
I	enum TestChannel *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_ad.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_adc.c

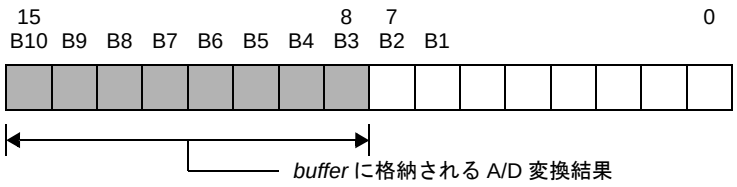
[指定形式]

```
#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

C. 3. 6 D/A コンバータ

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

表 C―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_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_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_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_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_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_dac.c

[指定形式]

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

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 regvalue;`	D/A 変換値（0x0 ～ 0xFF）

[戻り値]

なし

C. 3. 7 タ イ マ

以下に，コード生成がタイマ用として出力する API 関数の一覧を示します。

表 C—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	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	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	タイマ RD*n* のカウント処理を開始します。
R_TMR_RDn_Stop	タイマ RD*n* のカウント処理を終了します。
R_TMR_RDn_Set_PowerOff	タイマ RD*n* に対するクロック供給を停止します。
R_TMR_RDn_Get_PulseWidth	タイマ RD*n* のパルス幅を読み出します。
R_TMR_RGn_Create	タイマ RG*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_RGn_Create_UserInit	タイマ RG*n* に関するユーザ独自の初期化処理を行います。
R_TMR_RGn_Interrupt	タイマ割り込みの発生に伴う処理を行います。
R_TMR_RGn_Start	タイマ RG*n* のカウント処理を開始します。
R_TMR_RGn_Stop	タイマ RG*n* のカウント処理を終了します。

API 関数名	機能概要
R_TMR_RGn_Set_PowerOff	タイマ RG*n* に対するクロック供給を停止します。
R_TMR_RGn_Get_PulseWidth	タイマ RG*n* のパルス幅を読み出します。
R_TMR_RJ0_Create	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	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	タイマ RJ0 のカウント処理を開始します。
R_TMR_RJ0_Stop	タイマ RJ0 のカウント処理を終了します。
R_TMR_RJ0_Set_PowerOff	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	タイマ RJ0 のパルス幅を読み出します。

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_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_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_timer_user.c

[指定形式]

```
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_timer_user.c

[指定形式]

```
void    R_TAUm_Channeln_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_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_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_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_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_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_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_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_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_timer.c

[指定形式]

```
void    R_TAUm_Channeln_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

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_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

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_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_timer_user.c

[指定形式]

```
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

タイマ RD*n* のカウント処理を開始します。

[所属]

r_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

タイマ RD*n* のカウント処理を終了します。

[所属]

r_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

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，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_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*_Get_PulseWidth

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，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_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 *activewidth, uint32_t *inactivewidth, enum TMChannel 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 *activewidth;	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 *inactivewidth;	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	enum TMChannel 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_RG*n*_Create

タイマ RG*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_timer.c

[指定形式]

```
void    R_TMR_RGn_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_RG*n*_Create_UserInit

タイマ RG*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_RGn_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_timer_user.c

[指定形式]

```
void    R_TMR_RGn_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_RG*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_timer_user.c

[指定形式]

```
void    R_TMR_RGn_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_RG*n*_Start

タイマ RG*n* のカウント処理を開始します。

[所属]

r_timer.c

[指定形式]

```
void    R_TMR_RGn_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_RG*n*_Stop

タイマ RG*n* のカウント処理を終了します。

[所属]

r_timer.c

[指定形式]

```
void    R_TMR_RGn_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_RG*n*_Set_PowerOff

タイマ RG*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，タイマ RG*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_timer.c

[指定形式]

```
void    R_TMR_RGn_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_RG*n*_Get_PulseWidth

タイマ RG*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，タイマ RG*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_timer.c

[指定形式]

```
#include    "r_cg_macrodriver.h"
#include    "r_cg_timer.h"
MD_STATUS   R_TMR_RGn_Get_PulseWidth ( uint32_t *activewidth, uint32_t *inactivewidth, enum TMChannel 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 *activewidth;`	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 *inactivewidth;`	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	`enum channel;`	読み出し対象端子 TMCHANNELA： TRGIOA*n* 端子 TMCHANNELB： TRGIOB*n* 端子

[戻り値]

マクロ	説明
MD_OK	正常終了

R_TMR_RJ0_Create

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_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

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_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_timer_user.c

[指定形式]

```
void    R_TMR_RJ0_Interrupt ( void );
```

[引数]

なし

[戻り値]

なし

R_TMR_RJ0_Start

タイマ RJ0 のカウント処理を開始します。

[所属]

r_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

タイマ RJ0 のカウント処理を終了します。

[所属]

r_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

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，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_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

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，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_timer.c

[指定形式]

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

[引数]

I/O	引数	説明
O	uint32_t *activewidth;	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

C. 3. 8 ウォッチドッグ・タイマ

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

表 C—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_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_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_wdt_user.c

[指定形式]

```
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_wdt.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

C. 3. 9 リアルタイムクロック

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

表 C—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_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_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_rtc_user.c

[指定形式]

```
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_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_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_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_rtc.c

［指定形式］

```
#include    "r_cg_macrodriver.h"
#include    "r_cg_rtc.h"
MD_STATUS   R_RTC_Set_HourSystem ( enum RTCHourSystem hoursystem );
```

［引数］

I/O	引数	説明
I	enum RTCHourSystem *hoursystem*;	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_rtc.c

[指定形式]

```
#include    "r_cg_macrodriver.h"
#include    "r_cg_rtc.h"
MD_STATUS   R_RTC_Set_CounterValue ( struct RTCCounterValue counterwriteval );
```

[引数]

I/O	引数	説明
I	struct RTCCounterValue *counterwriteval*;	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 RTCCounterValue の構成を示します。

```
struct  RTCCounterValue {
    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 */
};
```

[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_rtc.c

[指定形式]

```
#include    "r_cg_macrodriver.h"
#include    "r_cg_rtc.h"
MD_STATUS   R_RTC_Get_CounterValue ( struct RTCCounterValue *counterreadval );
```

[引数]

I/O	引数	説明
O	struct RTCCounterValue *counterreadval;	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 RTCCounterValue 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_rtc.c

[指定形式]

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

[引数]

I/O	引数	説明
I	enum RTCINTPeriod *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_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_rtc_user.c

[指定形式]

```
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_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_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_rtc.c

［指定形式］

```
#include    "r_cg_macrodriver.h"
#include    "r_cg_rtc.h"
void    R_RTC_Set_AlarmValue ( struct RTCAlarmValue alarmval );
```

［引数］

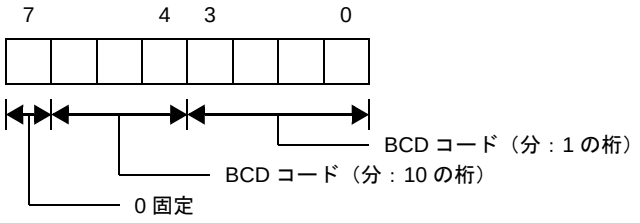
I/O	引数	説明
I	struct RTCAlarmValue *alarmval*;	アラームの発生条件（曜日，時，分）

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

```
struct  RTCAlarmValue {
    uint8_t Alarmwm;    /* 分 */
    uint8_t Alarmwh;    /* 時 */
    uint8_t Alarmww;    /* 曜日 */
};
```

- 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。



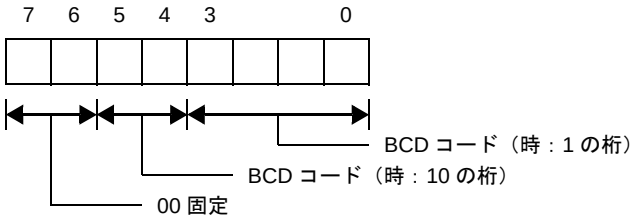
- Alarmwh（時）

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。

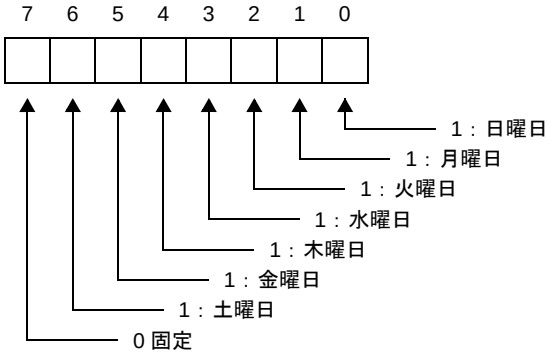
0： 午前

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_rtc.c

[指定形式]

```
#include    "r_cg_macrodriver.h"
#include    "r_cg_rtc.h"
void    R_RTC_Get_AlarmValue ( struct RTCAlarmValue *alarmval );
```

備考 アラームの発生条件 RTCAlarmValue 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	`struct RTCAlarmValue *alarmval;`	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_rtc_user.c

[指定形式]

```
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_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_rtc.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

C. 3. 10 インターバル・タイマ

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

表 C―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_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_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_it_user.c

[指定形式]

```
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_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_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_it.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

C. 3. 11 コンパレータ

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

表 C—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_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_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_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_comp_user.c

[指定形式]

```
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_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_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

C. 3. 12 クロック出力／ブザー出力

以下に，コード生成がクロック出力／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 C—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_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_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_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_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

C. 3. 13 データトランスファコントローラ

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

表 C—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_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_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_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_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_dtc.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

C. 3. 14 イベントリンクコントローラ

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

表 C—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_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_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_elc.c

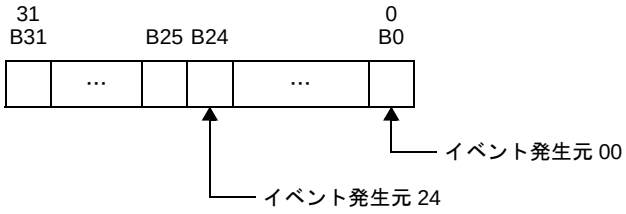
[指定形式]

```
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

C. 3. 15 DMA コントローラ

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

表 C—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_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_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_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_dmac_user.c

[指定形式]

```
void    R_DMAC n_Interrupt ( void );
```

備考 *n* は，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

R_DMAC*n*_Start

チャネル *n* を動作許可状態に設定します。

[所属]

r_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_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*_SoftwareTriggerOn

DMA 動作許可状態の際，DMA 転送を開始します。

[所属]

r_dmac.c

[指定形式]

```
void    R_DMACn_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

C. 3. 16 電圧検出回路

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

表 C—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_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_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_lvd_user.c

[指定形式]

```
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_lvd.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

付録D 索 引

【あ行】

【か行】

【さ行】

【た行】

【な行】

【は行】

【ま行】

【ら行】

【わ行】

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2011.06.01	―	初版発行

CubeSuite+ V1.00.01 ユーザーズマニュアル
RL78 設計編

発行年月日　2011 年 6 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

CubeSuite+ V1.00.01

ルネサス エレクトロニクス株式会社

R20UT0548JJ0100