

# CubeSuite+ V1.03.00

統合開発環境

ユーザーズマニュアル V850 設計編

対象デバイス

V850 ファミリ

本資料に記載の全ての情報は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。

## ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して、お客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
2. 本資料に記載されている情報は、正確を期すため慎重に作成したのですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
3. 本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害に関し、当社は、何らの責任を負うものではありません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を改造、改変、複製等しないでください。かかる改造、改変、複製等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。  
標準水準：            コンピュータ、OA 機器、通信機器、計測機器、AV 機器、  
                                 家電、工作機械、パーソナル機器、産業用ロボット等  
高品質水準：        輸送機器（自動車、電車、船舶等）、交通用信号機器、  
                                 防災・防犯装置、各種安全装置等  
当社製品は、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（原子力制御システム、軍事機器等）に使用されることを意図しておらず、使用することはできません。たとえ、意図しない用途に当社製品を使用したことによりお客様または第三者に損害が生じて、当社は一切その責任を負いません。なお、ご不明点がある場合は、当社営業にお問い合わせください。
6. 当社製品をご使用の際は、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他の保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
9. 本資料に記載されている当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。また、当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途に使用しないでください。当社製品または技術を輸出する場合は、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。
10. お客様の転売等により、本ご注意書き記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は何らの責任も負わず、お客様にてご負担して頂きますのでご了承ください。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。

注 1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

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

# このマニュアルの使い方

このマニュアルは、V850 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、V850 ファミリの統合開発環境（ソフトウェア開発における、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+ の持つソフトウェ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ェア、ソフトウェ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。

[第 1 章 概 説](#)

[第 2 章 機能（端子配置）](#)

[第 3 章 機能（コード生成）](#)

[付録 A ウィンドウ・リファレンス](#)

[付録 B 出力ファイル](#)

[付録 C API 関数](#)

[付録 D 索 引](#)

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

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

関連資料

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

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	R20UT2133J	R20UT2133E
	V850 設計編	このマニュアル	R20UT2134E
	RL78 設計編	R20UT2136J	R20UT2136E
	78K0R 設計編	R20UT2137J	R20UT2137E
	78K0 設計編	R20UT2138J	R20UT2138E
	RX コーディング編	R20UT0767J	R20UT0767E
	V850 コーディング編	R20UT0553J	R20UT0553E
	コーディング編 (CX コンパイラ)	R20UT2139J	R20UT2139E
	RL78, 78K0R コーディング編	R20UT2140J	R20UT2140E
	78K0 コーディング編	R20UT2141J	R20UT2141E
	RX ビルド編	R20UT0768J	R20UT0768E
	V850 ビルド編	R20UT0557J	R20UT0557E
	ビルド編 (CX コンパイラ)	R20UT2142J	R20UT2142E
	RL78, 78K0R ビルド編	R20UT2143J	R20UT2143E
	78K0 ビルド編	R20UT0783J	R20UT0783E
	RX デバッグ編	R20UT2175J	R20UT2175E
	V850 デバッグ編	R20UT2144J	R20UT2144E
	RL78 デバッグ編	R20UT2145J	R20UT2145E
	78K0R デバッグ編	R20UT0732J	R20UT0732E
	78K0 デバッグ編	R20UT0731J	R20UT0731E
解析編	R20UT2146J	R20UT2146E	
メッセージ編	R20UT2147J	R20UT2147E	

**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。

# 目 次

## 第1章 概 説 … 7

- 1.1 概 要 … 7
- 1.2 特 長 … 7

## 第2章 機能（端子配置） … 8

- 2.1 概 要 … 8
- 2.2 端子配置表 パネルのオープン … 10
  - 2.2.1 表示項目の選択 … 11
  - 2.2.2 表示順序の変更 … 12
  - 2.2.3 列の追加 … 13
  - 2.2.4 列の削除 … 13
- 2.3 端子配置図 パネルのオープン … 14
  - 2.3.1 マイクロコントローラの形状選択 … 15
  - 2.3.2 表示色の選択 … 16
  - 2.3.3 ポップアップ情報の選択 … 17
  - 2.3.4 付加情報の選択 … 18
- 2.4 情報の記述 … 19
- 2.5 レポート・ファイルの出力 … 20
  - 2.5.1 端子配置表の出力 … 20
  - 2.5.2 端子配置図の出力 … 21

## 第3章 機能（コード生成） … 22

- 3.1 概 要 … 22
- 3.2 コード生成 パネルのオープン … 23
- 3.3 情報の設定 … 24
  - 3.3.1 入力規約 … 24
  - 3.3.2 入力不備箇所に対するアイコン表示 … 25
  - 3.3.3 端子の競合に対するアイコン表示 … 26
- 3.4 ソース・コードの確認 … 27
- 3.5 ソース・コードの出力 … 28
  - 3.5.1 出力有無の設定 … 29
  - 3.5.2 ファイル名の変更 … 30
  - 3.5.3 API 関数名の変更 … 31
  - 3.5.4 出力モードの変更 … 32
  - 3.5.5 出力先の変更 … 33
- 3.6 レポート・ファイルの出力 … 34
  - 3.6.1 出力形式の変更 … 36
  - 3.6.2 出力先の変更 … 37

**付録 A ウィンドウ・リファレンス … 38**

A.1 説 明 … 38

**付録 B 出力ファイル … 89**

B.1 概 要 … 89

B.2 出力ファイル … 89

**付録 C API 関数 … 97**

C.1 概 要 … 97

C.2 出力関数 … 97

C.3 関数リファレンス … 109

C.3.1 システム … 111

C.3.2 外部バス … 126

C.3.3 ポート … 129

C.3.4 割り込み … 135

C.3.5 シリアル … 146

C.3.6 A/D コンバータ … 228

C.3.7 D/A コンバータ … 238

C.3.8 タイマ … 245

C.3.9 時計タイマ … 307

C.3.10 リアルタイム・カウンタ … 312

C.3.11 リアルタイム出力機能 … 351

C.3.12 DMA コントローラ … 366

C.3.13 低電圧検出回路 … 374

**付録 D 索 引 … 381**

# 第1章 概 説

CubeSuite+ は、ルネサス エレクトロニクス製マイクロコントローラ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（端子配置／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の端子配置状況（端子配置表、端子配置図）／マイクロコントローラが提供している周辺機能（クロック発生回路、I/O ポートなど）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（端子配置／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。

### - レポート機能

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

### - リネーム機能

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

## 第2章 機能（端子配置）

本章では、設計ツール（端子配置）が提供している主な機能を実践手順とともに説明します。

### 2.1 概要

端子配置は、マイクロコントローラの端子配置状況を入力することにより、端子配置表、端子配置図といったレポート・ファイルを出力させることができます。

なお、端子配置の実践手順は、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+ 統合開発環境 ユーザーズマニュアル 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+ 統合開発環境 ユーザーズマニュアル 起動編」を参照してください。

#### (3) 端子配置表 パネルのオープン

マイクロコントローラの各端子に関する情報を記述するための[端子配置表 パネル](#)をオープンします。

##### (a) 表示項目の選択

端子配置表に表示する項目を選択します。

##### (b) 表示順序の変更

端子配置表に表示する項目の順序を変更します。

##### (c) 列の追加

端子配置表に対する列の追加を行います。

##### (d) 列の削除

端子配置表に対する列の削除を行います。

**(4) 端子配置図 パネルのオープン**

端子に関する情報の記述状況を確認するための端子配置図 パネルをオープンします。

**(a) マイクロコントローラの形状選択**

端子配置図 パネルに表示するマイクロコントローラの形状を選択します。

**(b) 表示色の選択**

端子配置図 パネルの各端子（電源端子、特殊端子、使用端子など）に関する情報の記述状況を確認するための表示色を選択します。

**(c) ポップアップ情報の選択**

端子配置図 パネルの各端子上にマウス・カーソルを移動した際、ポップアップ表示させる情報の種類を選択します。

**(d) 付加情報の選択**

端子配置図 パネルの端子部分に表示させる情報の種類を選択します。

**(5) 情報の記述**

端子配置表 パネルでマイクロコントローラの各端子に関する情報を記述します。

**(6) レポート・ファイルの出力**

レポート・ファイル（端子配置を用いて設定した情報を保持したファイル：端子配置表、端子配置図）を指定されたフォルダに出力します。

**(a) 端子配置表の出力**

端子配置表を出力します。

**(b) 端子配置図の出力**

端子配置図を出力します。

**(7) 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+ 統合開発環境 ユーザーズマニュアル 起動編」を参照してください。

## 2.2 端子配置表 パネルのオープン

マイクロコントローラの各端子に関する情報を記述するための端子配置表 パネルをオープンします。

なお、端子配置表 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)] → [端子配置 (設計ツール)] → [端子配置表] をダブルクリックすることにより行います。

図 2—1 端子配置表 パネルのオープン



**備考 1.** 端子配置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 “[端子配置 (設計ツール)] ノード” は表示されません。

**2.** 端子配置表 パネルは3個のタブから構成され、タブを選択することにより、“マイクロコントローラの各端子に関する情報” の表示順序が切り替わります。

- [端子番号] タブ

マイクロコントローラの各端子に関する情報を端子番号順で表示

- [マクロ] タブ

マイクロコントローラの各端子に関する情報を周辺機能単位にグルーピングされた順序で表示

- [外部周辺] タブ

外部周辺に接続された端子に関する情報を外部周辺部品単位にグルーピングされた順序で表示

### 2.2.1 表示項目の選択

端子配置では、端子配置表の左上に設けられた  ボタンで端子配置表の表示項目を選択することができます。

なお、表示項目の選択は、端子配置表の左上に設けられた  ボタンをクリックすることによりオープンする [列の選択 ダイアログ](#)で行います。

図 2—2 表示項目の選択



**備考** 表示項目の選択は、該当チェック・ボックスをクリックすることにより行います。

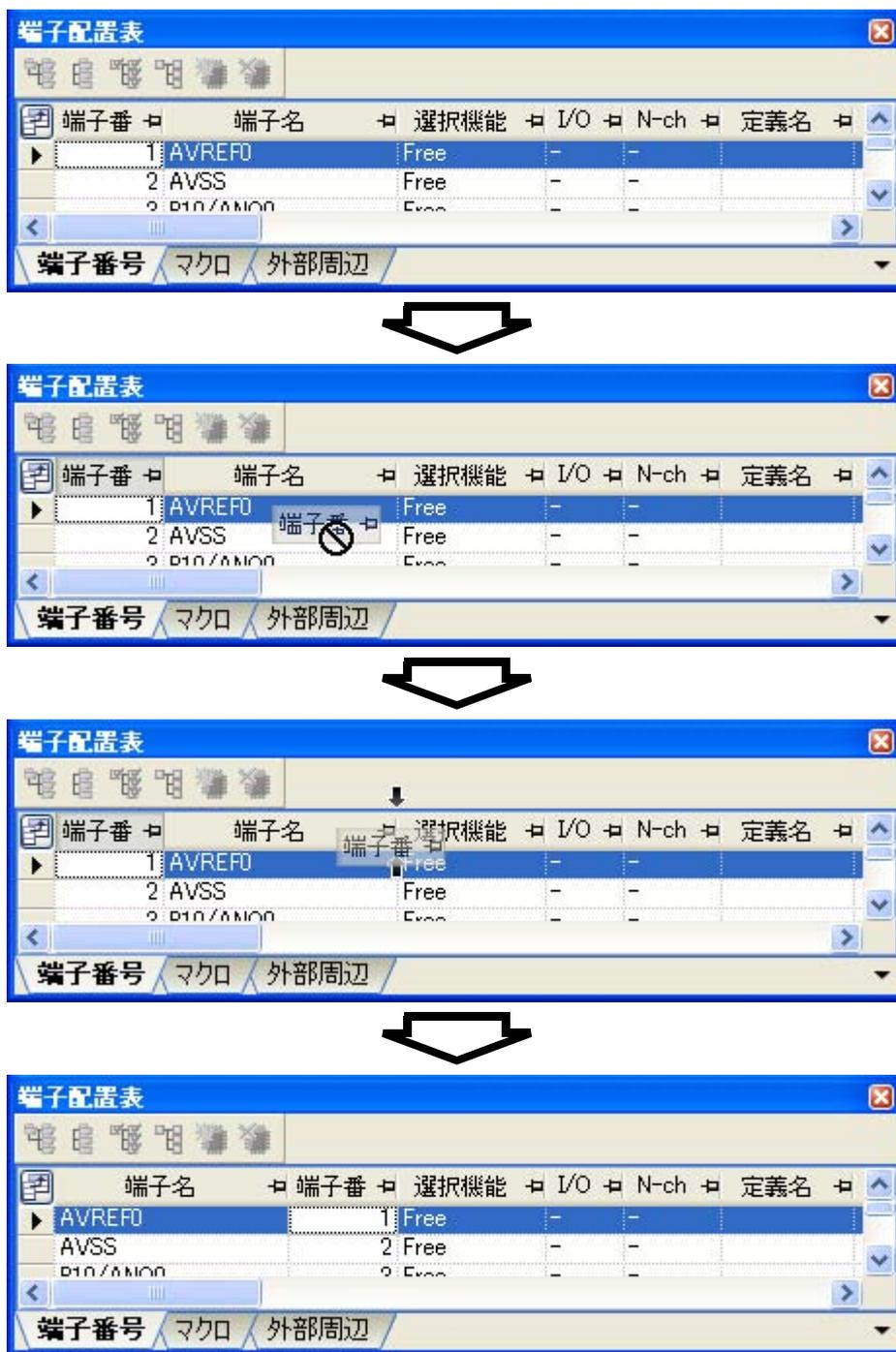
表 2—1 表示項目の選択

チェック状態	該当項目を端子配置表に表示します。
非チェック状態	該当項目を端子配置表から非表示とします。

### 2.2.2 表示順序の変更

端子配置では、端子配置表の列をドラッグしたのち、移動先にドロップすることにより、表示項目の表示順序を変更（列を移動）することができます。

図 2—3 表示順序の変更



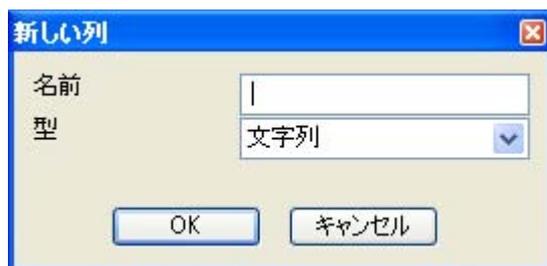
**備考** 表示順序の変更は、端子配置表の左上に設けられた  ボタンをクリックすることによりオープンする列の選択ダイアログの表示項目選択エリアに表示されている項目をドラッグしたのち、端子配置表の移動先にドロップすることでも、表示項目の表示順序を変更することができます。

### 2.2.3 列の追加

端子配置では、端子配置表の左上に設けられた  ボタンをクリックすることによりオープンする [列の選択 ダイアログ](#) の [新しい列 ...] ボタンで“ユーザ独自の列”を端子配置表に追加することができます。

なお、列の追加は、[列の選択 ダイアログ](#) の [新しい列 ...] ボタンをクリックすることによりオープンする [新しい列 ダイアログ](#) で行います。

図 2—4 列の追加



**備考** 端子配置表 “[マクロ] タブ, [外部周辺] タブの第 1 階層”については、列の追加が制限されています。

### 2.2.4 列の削除

端子配置では、端子配置表の左上に設けられた  ボタンをクリックすることによりオープンする [列の選択 ダイアログ](#) の [列の削除] ボタンで“ユーザ独自の列”を端子配置表から削除することができます。

なお、列の削除は、[列の選択 ダイアログ](#) の表示項目選択エリアで削除対象列を選択したのち、[列の削除] ボタンをクリックすることにより行います。

図 2—5 列の削除



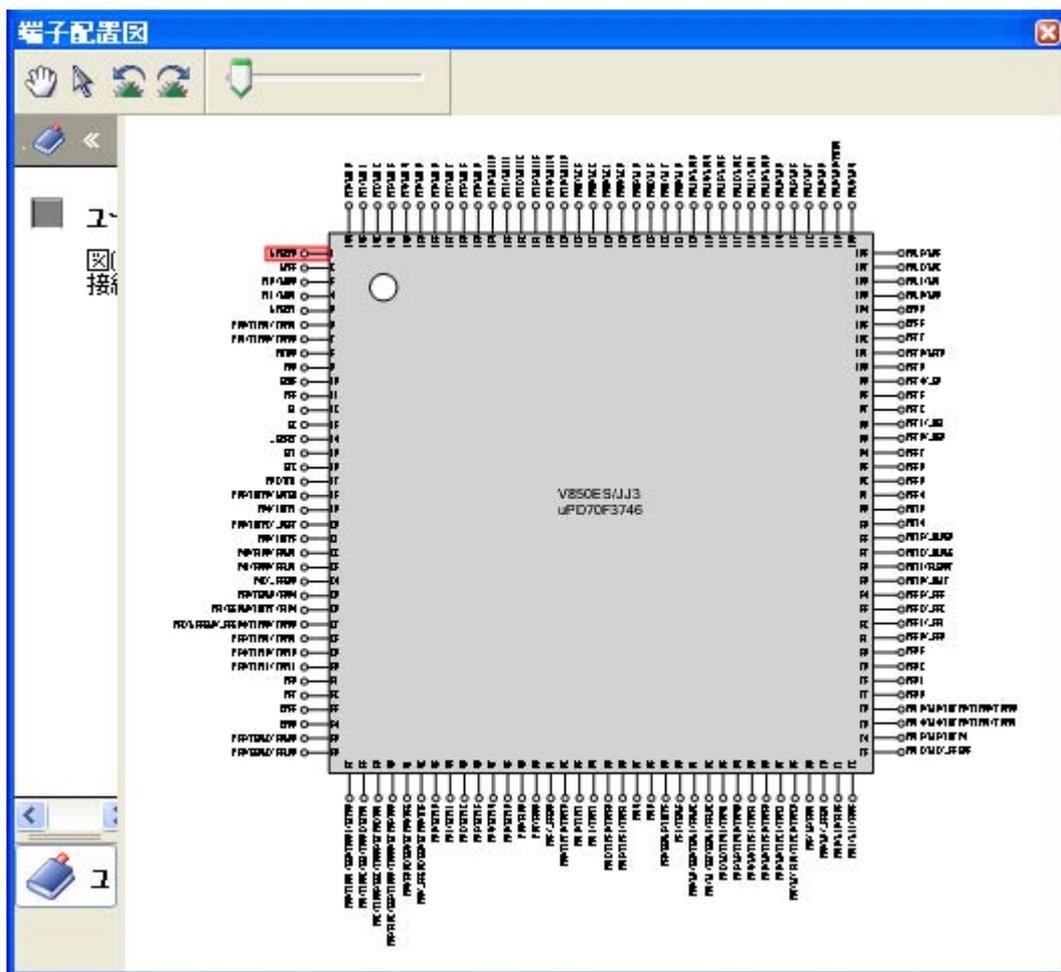
**備考** 削除可能な列は、[新しい列 ダイアログ](#) でユーザが独自に追加した列に限られます。

## 2.3 端子配置図 パネルのオープン

マイクロコントローラの各端子に関する情報の記述状況を確認するための端子配置図パネルをオープンします。

なお、端子配置図パネルのオープンには、プロジェクト・ツリーパネルの [Project name (プロジェクト)] → [端子配置 (設計ツール)] → [端子配置図] をダブルクリックすることにより行います。

図 2—6 端子配置図 パネルのオープン



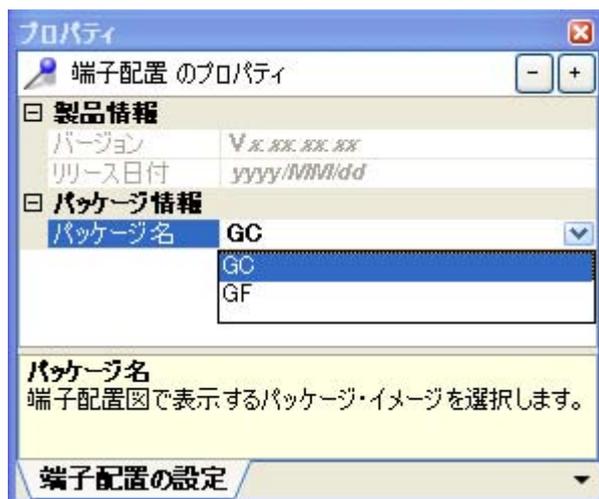
**備考** プロパティパネルの [端子配置の設定] タブでパッケージ名に“BGA”を選択している場合、端子配置図パネルをオープンすることができません。

### 2.3.1 マイクロコントローラの形状選択

「2.3 端子配置図 パネルのオープン」でオープンした端子配置図 パネルに表示するマイクロコントローラの形状を選択します。

なお、マイクロコントローラの形状選択は、プロパティ パネルの [端子配置の設定] タブ→ [パッケージ名] で該当形状を選択することにより行います。

図 2-7 マイクロコントローラの形状選択



**備考** マイクロコントローラの形状選択は、オーダ名称（GC、GF など）で行います。

### 2.3.2 表示色の選択

「2.3 端子配置図 パネルのオープン」でオープンした端子配置図 パネルの各端子（電源端子，特殊端子，未使用端子など）に関する情報の記述状況を確認するための表示色を選択します。

なお，表示色の選択は，プロパティ パネルの [端子配置図の設定] タブ→ [色設定] からオープンするカラー・パレットで該当色を選択することにより行います。

図 2—8 表示色の選択



**備考** 表示色の選択は，以下の8種類に対して行います。

表 2—2 表示色の選択

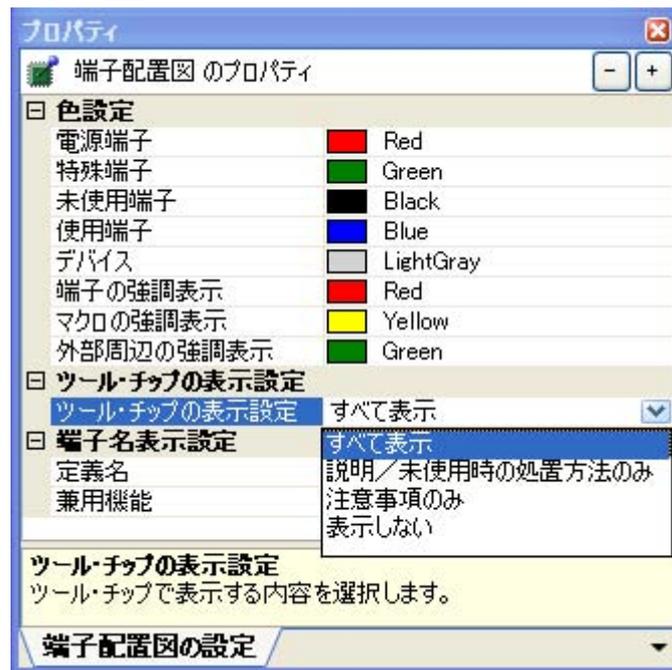
設定対象	概要
電源端子	電源端子（用途が電源に限定されている端子）の色設定を選択します。
特殊端子	特殊端子（用途が規定されている端子）の色設定を選択します。
未使用端子	未使用端子（端子配置表 パネルにおいて，用途が未設定の兼用端子）の色設定を選択します。
使用端子	使用端子（端子配置表 パネルにおいて，用途が設定済みの兼用端子）の色設定を選択します。
デバイス	マイクロコントローラ本体部の色設定を選択します。
端子の強調表示	端子配置表 パネルで選択した端子を端子配置図 パネルの端子配置図で強調表示する際の色設定を選択します。
マクロの強調表示	端子配置表 パネルの [マクロ] タブで選択された項目に対応した端子の背景色を選択します。
外部周辺の強調表示	端子配置表 パネルの [外部周辺] タブで選択された項目に対応した端子の背景色を選択します。

### 2.3.3 ポップアップ情報の選択

「2.3 端子配置図 パネルのオープン」でオープンした端子配置図 パネルの各端子上にマウス・カーソルを移動した際にポップアップ表示させる情報の種類を選択します。

なお、ポップアップ情報の選択は、プロパティ パネルの [端子配置図の設定] タブ→ [ツール・チップの表示設定] で該当種類を選択することにより行います。

図 2—9 ポップアップ情報の選択



**備考** ポップアップ情報の選択は、以下の4種類から行います。

表 2—3 ポップアップ情報の選択

ポップアップ情報	概要
すべて表示	対応する端子配置表の“説明”，“未使用時の処置方法”，“注意事項”に記載されている文字列を表示します。
説明／未使用時の処置方法のみ	端子配置表の“説明”，“未使用時の処置方法”に記載されている文字列を表示します。
注意事項のみ	対応する端子配置表の“注意事項”に記載されている文字列を表示します。
表示しない	端子上にマウス・カーソルを移動しても、何も表示しません。

### 2.3.4 付加情報の選択

「2.3 端子配置図 パネルのオープン」でオープンした端子配置図 パネルの端子部分に表示させる情報の種類を選択します。

なお、付加情報の選択は、プロパティ パネルの [端子配置図の設定] タブ→ [端子名表示設定] で該当情報を選択することにより行います。

図 2—10 付加情報の選択



**備考 1.** 定義名（端子配置表の“定義名”に記載された文字列を付与した形式で表示するか否か）については、以下の2種類から選択します。

表示する	端子配置表の“定義名”に記載されている文字列を付与した形式で表示します。
表示しない	端子配置表の“定義名”に記載されている文字列を付与しません。

2. 兼用機能（端子配置表の“選択機能”で機能を選択した際、非選択機能についても表示するか否か）については、以下の2種類から選択します。

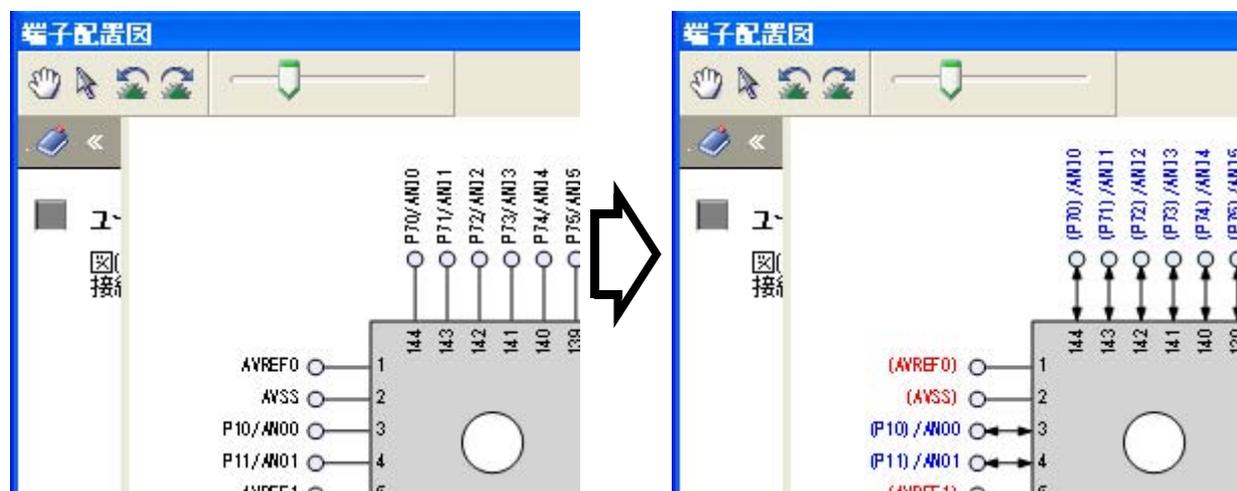
すべて	端子配置表の“選択機能”で選択された機能をかっこで括った形式で表示します。
選択機能のみ	端子配置表の“選択機能”で選択された機能のみを端子配置図に表示します。

## 2.4 情報の記述

「2.2 端子配置表 パネルのオープン」でオープンした端子配置表 パネルでマイクロコントローラの各端子に関する情報を記述します。

- 備考 1.** 端子配置表の“端子番号”、“端子名”、“説明”、“未使用時の処置方法”、“注意事項”については、固定化された情報のため、該当欄に情報を追記することはできません。
- 2.** “兼用端子名”欄の Free を固有端子名に変更した場合、端子配置図 パネルの該当端子色がプロパティ パネルの [端子配置図の設定] タブ→ [色設定] で選択された“未使用端子の表示色”から“使用端子の表示”へと変化します。

図 2—11 表示色の変化



## 2.5 レポート・ファイルの出力

レポート・ファイル（端子配置を用いて設定した情報を保持したファイル：端子配置表、端子配置図）を指定されたフォルダに出力します。

### 2.5.1 端子配置表の出力

[ファイル]メニュー→[名前を付けて 端子配置表 を保存 ...]を選択し、レポート・ファイル（端子配置を用いて設定した情報を保持したファイル：端子配置表）を出力します。

なお、端子配置表の出力先は、[ファイル]メニュー→[名前を付けて 端子配置表 を保存 ...]を選択することによりオープンする名前を付けて保存 ダイアログで指定されたフォルダとなります。

図 2—12 端子配置表の出力



- 備考 1.** すでに端子配置表が出力されていた場合、[ファイル]メニュー→[端子配置表 を保存]を選択することにより、該当表を上書きします。
- 2.** 端子配置表の出力形式は、Microsoft Office Excel ブック形式に限られます。

## 2.5.2 端子配置図の出力

[ファイル] メニュー→ [名前を付けて 端子配置図 を保存 ...] を選択し、レポート・ファイル（端子配置を用いて設定した情報を保持したファイル：端子配置図）を出力します。

なお、端子配置図の出力先は、[ファイル] メニュー→ [名前を付けて 端子配置図 を保存 ...] を選択することによりオープンする名前を付けて保存 ダイアログで指定されたフォルダとなります。

図 2—13 端子配置図の出力



**備考** すでに端子配置図が出力されていた場合、[ファイル] メニュー→ [端子配置図 を保存] を選択することにより、該当図を上書きします。

## 第3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実践手順とともに説明します。

### 3.1 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、I/Oポートなど）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、コード生成の実践手順は、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+ 統合開発環境 ユーザーズマニュアル 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+ 統合開発環境 ユーザーズマニュアル 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、I/Oポートなど）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。

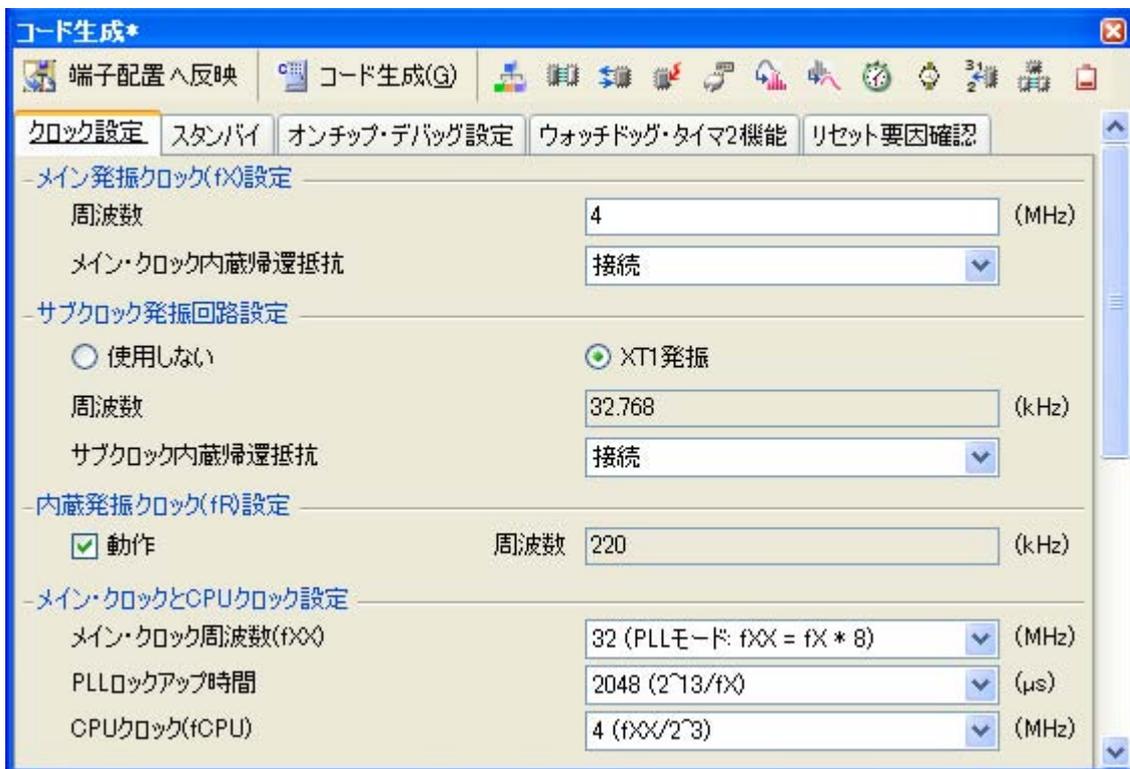
**備考** “プロジェクトの保存” についての詳細は、「CubeSuite+ 統合開発環境 ユーザーズマニュアル 起動編」を参照してください。

**3.2 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、I/Oポートなど）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 “[システム], [ポート] など” をダブルクリックすることにより行います。

図 3—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。

### 3.3 情報の設定

「3.2 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、「3.2 コード生成 パネルのオープン」から「3.3 情報の設定」の操作を繰り返し行うことになります。

#### 3.3.1 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 3—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 3—2 進数の一覧

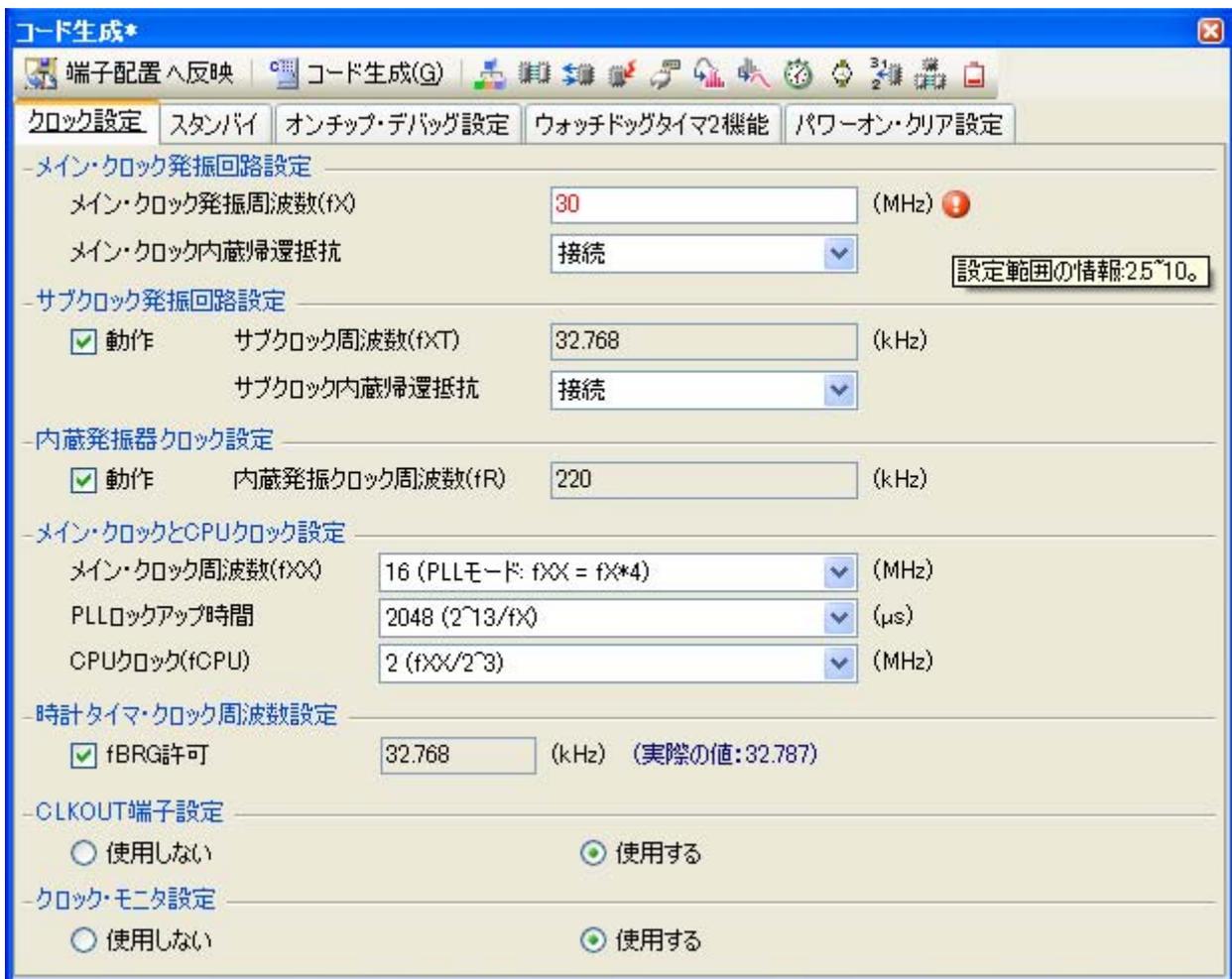
進数表記	概要
10 進数	1～9の数字で始まり0～9の数字が続く数値、および0
16 進数	0xで始まり0～9の数字、およびa～fの英字が続く数値（英字の大文字／小文字については、不問）

### 3.3.2 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、および入力が必要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 3—2 入力不備箇所に対するアイコン表示



### 3.3.3 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 3—3 端子の競合に対するアイコン表示

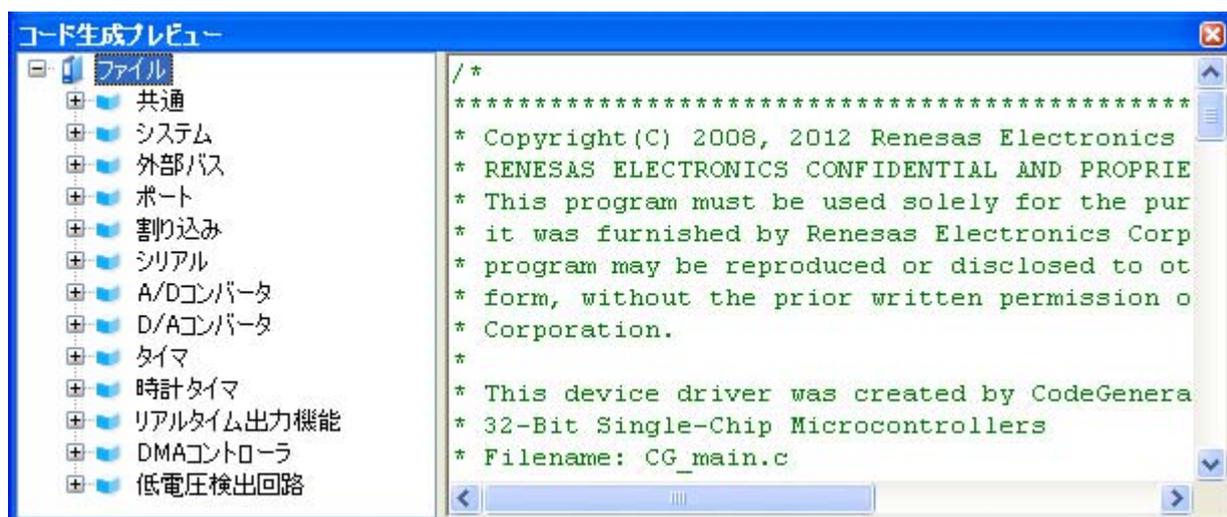


### 3.4 ソース・コードの確認

「3.3 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3—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3—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

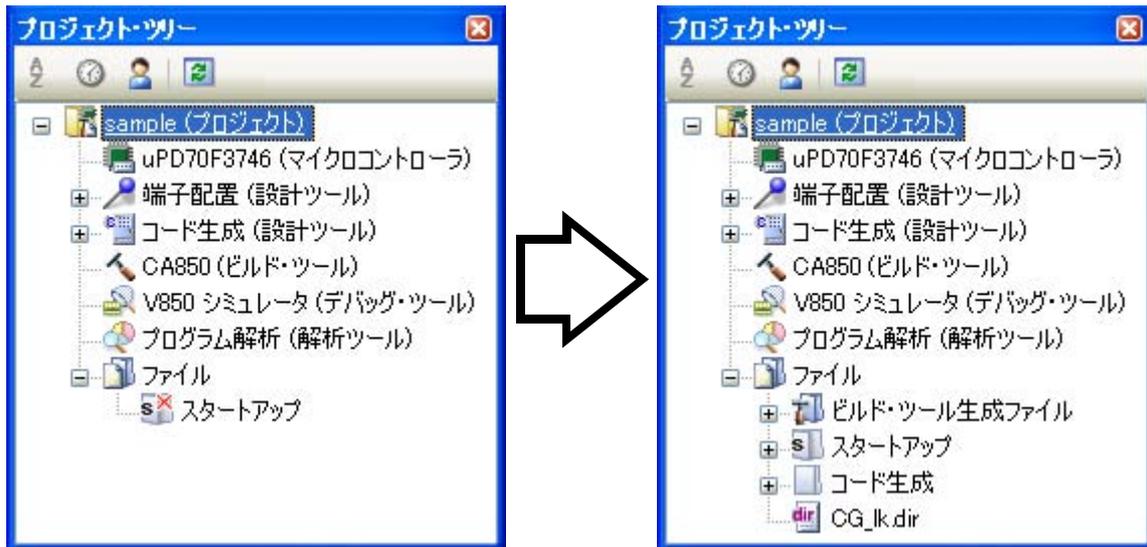
- 3.** コード生成プレビューパネル内でソース・コードを編集することはできません。
- 4.** 一部のAPI関数（シリアル・アレイ・ユニット用API関数など）については、ソース・コードの出力時（コード生成パネルの  コード生成(G) ボタンをクリックした際）にレジスタ値SFRなどが計算され確定するものがあります。このため、コード生成プレビューパネルに表示されるソース・コードは、実際に出力されるソース・コードと一致しない場合があります。

### 3.5 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。

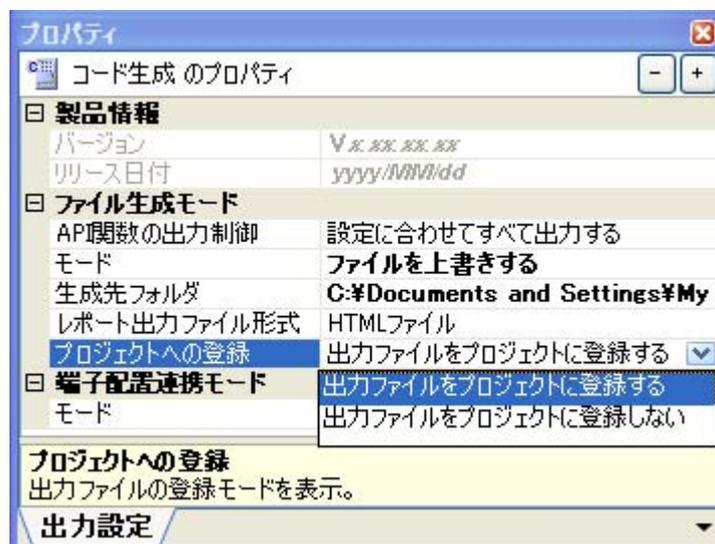
なお、ソース・コードの出力先は、プロパティ パネルの [出力設定] タブ→ [生成先フォルダ] で指定されたフォルダとなります。

図 3—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。

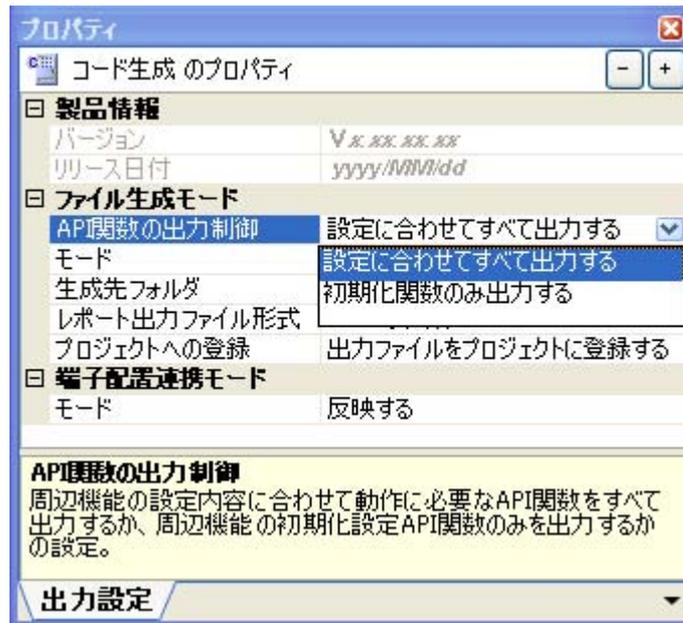
図 3—6 登録有無の設定



### 3.5.1 出力有無の設定

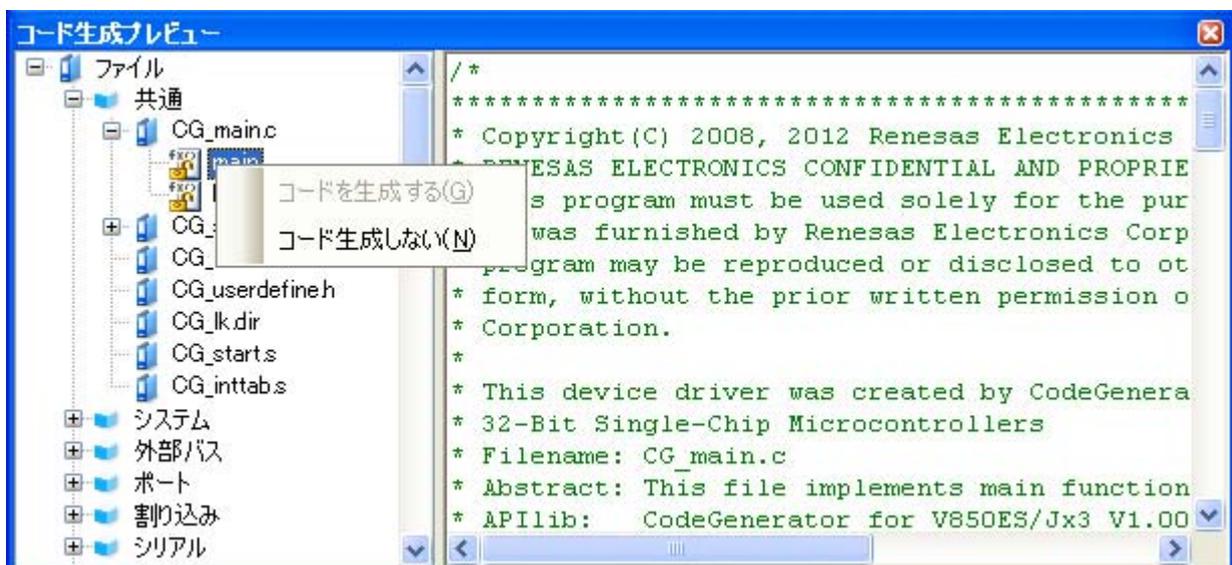
コード生成では、**プロパティパネル**の**[出力設定] タブ**→**[API関数の出力制御]**で“設定に合わせてすべて出力する／初期化関数のみ出力する”を選択することにより、出力するAPI関数の種類（全API関数、初期化用API関数のみ）を設定することができます。

図 3—7 出力有無の設定



また、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”についても設定することができます。

図 3—8 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 3—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。

### 3.5.2 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 3—9 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。

### 3.5.3 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 3—10 API 関数名の変更

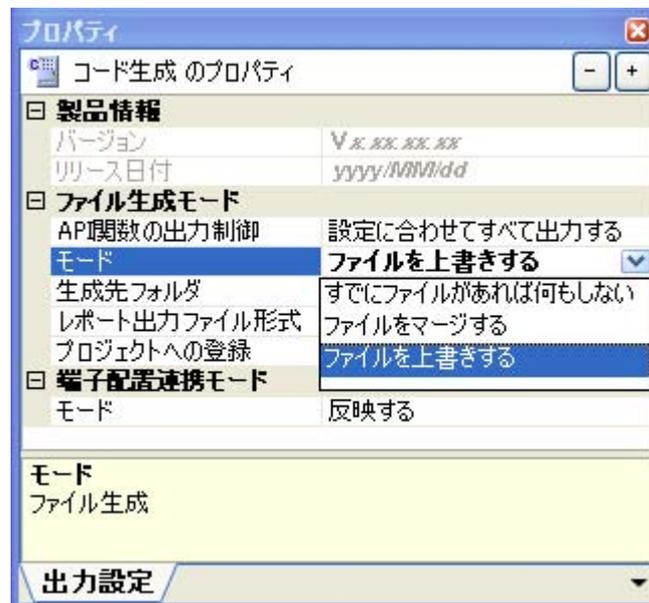


**備考** コード生成が規定しているデフォルトAPI関数名に戻す際には、コンテキスト・メニューから“名前を元に戻す”を選択します。

### 3.5.4 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（すでにファイルがあれば何もしない、ファイルをマージする、ファイルを上書きする）を変更することができます。

図 3—11 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 3—5 ソース・コードの出力モード

出力モード	概要
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。
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。
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。

### 3.5.5 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 3—12 出力先の変更



### 3.6 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 3—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 3—13 レポート・ファイル macro の出力例

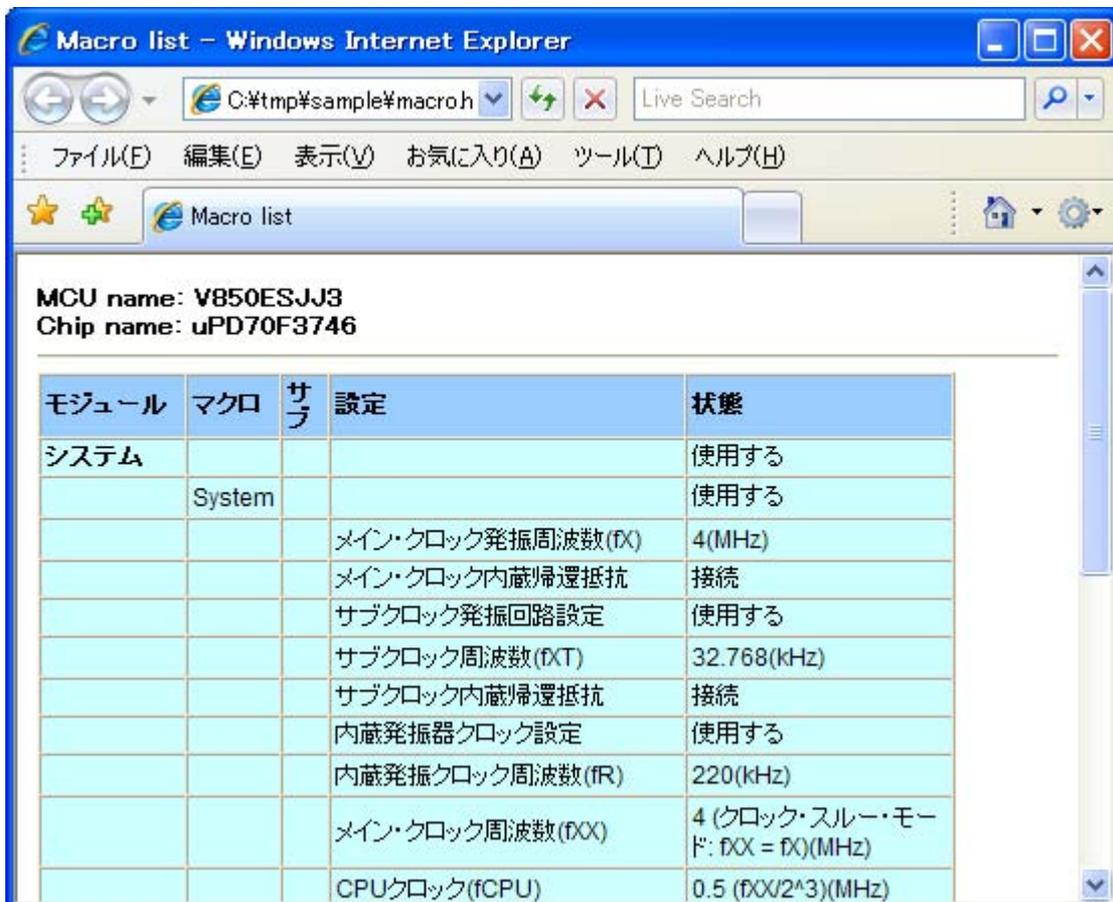


図 3—14 レポート・ファイル function の出力例

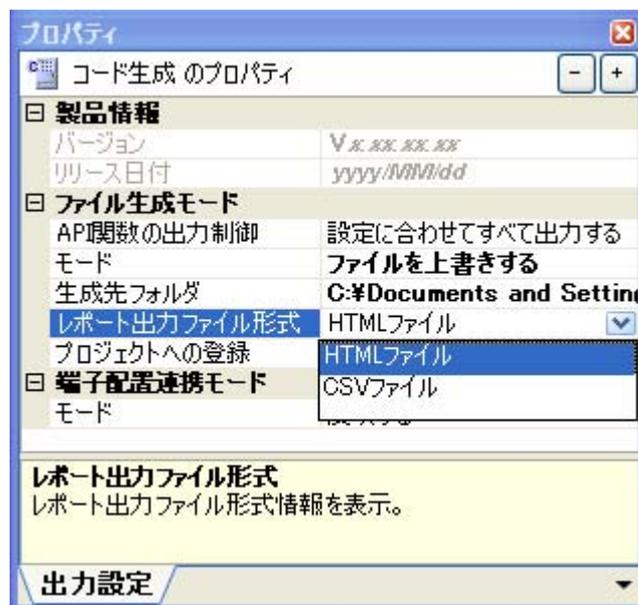
MCU name: V850ES/JJ3  
Chip name: uPD70F3746

モジュール	ファイル	マクロ	機能	デフォルト	状態
<b>共通</b>					
	CG_main.c			CG_main.c	使用する
			void main(void)	main	使用する
			void R_MAIN_UserInit(void)	R_MAIN_UserInit	使用する
	CG_systeminit.c			CG_systeminit.c	使用する
			void systeminit(void)	systeminit	使用する
	CG_macrodriver.h			CG_macrodriver.h	使用する
	CG_userdefine.h			CG_userdefine.h	使用する
	CG_ik.dir			CG_ik.dir	使用する
	CG_start.s			CG_start.s	使用する
	CG_inttab.s			CG_inttab.s	使用する
<b>システム</b>					
	CG_system.c			CG_system.c	使用する
			void CLOCK_Init(void)	CLOCK_Init	使用する

### 3.6.1 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 3—15 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 3—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。

### 3.6.2 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 3—16 出力先の変更



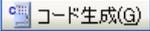
## 付録 A ウィンドウ・リファレンス

本付録では、設計ツールのウィンドウ／パネル／ダイアログについて説明します。

### A.1 説明

以下に、設計ツールのウィンドウ／パネル／ダイアログの一覧を示します。

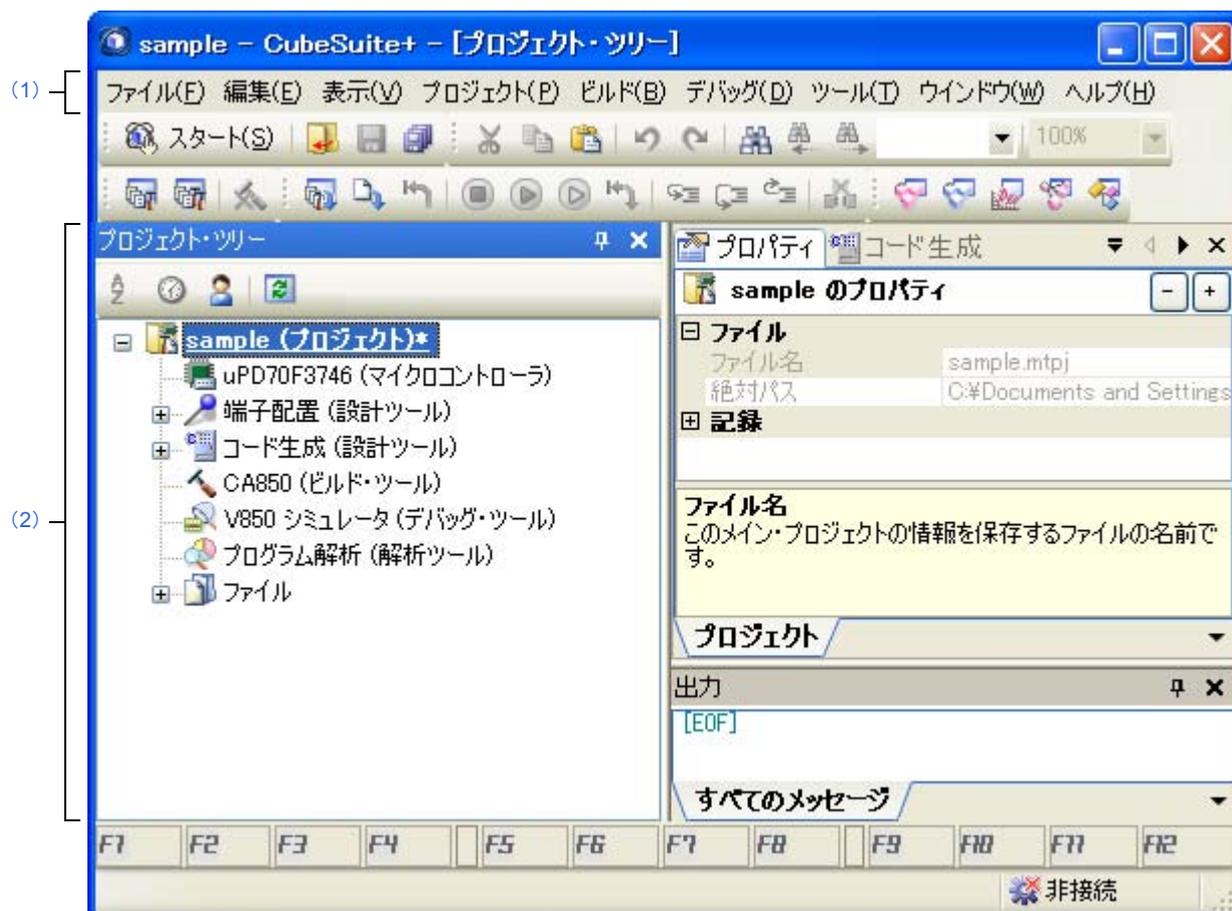
表 A—1 ウィンドウ／パネル／ダイアログの一覧

ウィンドウ／パネル／ダイアログ名	機能概要
メイン・ウィンドウ	CubeSuite+ を起動した際、最初にオープンするウィンドウであり、本ウィンドウから 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。
端子配置表 パネル	マイクロコントローラの各端子に関する情報を記述します。
端子配置図 パネル	端子配置表 パネルにおける情報の記述状況を表示します。
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。
コード生成プレビュー パネル	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。
列の選択 ダイアログ	本ダイアログに表示されている項目を端子配置表に表示するか否かの選択、および端子配置表に対する列の追加／削除を行います。
新しい列 ダイアログ	端子配置表に列を追加します。
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・ウィンドウ

CubeSuite+ を起動した際、最初にオープンするウィンドウであり、本ウィンドウから 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・ウィンドウ



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

端子配置表 を保存	端子配置表 パネル専用部分 レポート・ファイル（端子配置を用いて設定した情報を保持したファイル：端子配置表）を既存のファイルに上書き保存します。
名前を付けて 端子配置表 を保存 ...	端子配置表 パネル専用部分 レポート・ファイル（端子配置を用いて設定した情報を保持したファイル：端子配置表）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。
端子配置図 を保存	端子配置図 パネル専用部分 レポート・ファイル（端子配置を用いて設定した情報を保持したファイル：端子配置図）を既存のファイルに上書き保存します。
名前を付けて 端子配置図 を保存 ...	端子配置図 パネル専用部分 レポート・ファイル（端子配置を用いて設定した情報を保持したファイル：端子配置図）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。
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 ...	端子配置表 パネル／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。
端子配置表 パネルのヘルプを開く	端子配置表 パネル専用部分 端子配置表 パネルのヘルプを表示します。
端子配置図 パネルのヘルプを開く	端子配置図 パネル専用部分 端子配置図 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。

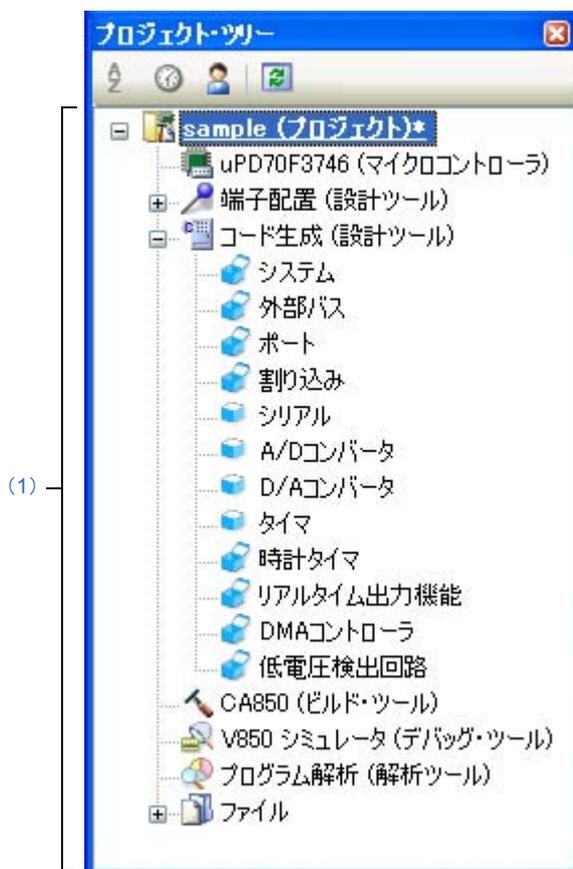
本エリアについての詳細は、以下を参照してください。

- プロジェクト・ツリー パネル
- プロパティ パネル
- 端子配置表 パネル
- 端子配置図 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) 端子配置（設計ツール）

本ノードは、以下に示した端子ノードから構成されています。

端子配置表	マイクロコントローラの各端子に関する情報を記述するための <a href="#">端子配置表パネル</a> をオープンします。
端子配置図	<a href="#">端子配置表パネル</a> における情報の記述状況を表示するための <a href="#">端子配置図パネル</a> をオープンします。

#### (b) 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。

システム	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、およびパワーオン・クリア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 <a href="#">[システム]</a> をオープンします。
外部バス	マイクロコントローラが提供している外部バス・インタフェースの機能（外部バスを内蔵ROM、RAM、SFR以外の領域に接続する機能）を制御するうえで必要な情報を設定するための <a href="#">[外部バス]</a> 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 <a href="#">[ポート]</a> 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 <a href="#">[割り込み]</a> 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 <a href="#">[シリアル]</a> 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 href="#">[A/Dコンバータ]</a> 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 <a href="#">[D/Aコンバータ]</a> 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 <a href="#">[タイマ]</a> をオープンします。

時計タイマ	マイクロコントローラが提供している時計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 [時計タイマ] をオープンします。
リアルタイム・カウンタ	マイクロコントローラが提供しているリアルタイム・カウン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 [リアルタイム・カウンタ] をオープンします。
リアルタイム出力機能	マイクロコントローラが提供しているリアルタイム出力機能を制御するうえで必要な情報を設定するための [リアルタイム出力機能] 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。
低電圧検出回路	マイクロコントローラが提供している低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 [低電圧検出回路] をオープンします。

(c) 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]

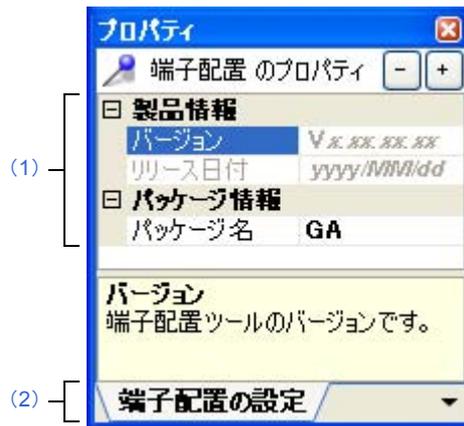
マウスを右クリックすることにより表示されるコンテキスト・メニューは、以下のとおりです。

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 ([端子配置 (設計ツール)], [端子配置表], [端子配置図], [コード生成 (設計ツール)], 周辺機能ノード “[システム], [ポート] など”) に対応した情報を保持した <a href="#">プロパティ パネル</a> 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 ([端子配置 (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)]
- [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 ([端子配置 (設計ツール)], [端子配置表], [端子配置図], [コード生成 (設計ツール)], 周辺機能ノード “[システム], [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 ([端子配置 (設計ツール)], [端子配置表], [端子配置図], [コード生成 (設計ツール)], 周辺機能ノード “[システム], [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 ([端子配置 (設計ツール)], [端子配置表], [端子配置図], [コード生成 (設計ツール)], 周辺機能ノード “[システム], [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と切り替わります。

- 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と切り替わります。
- 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（[端子配置（設計ツール）]、[端子配置表]、[端子配置図]、[コード生成（設計ツール）]、周辺機能ノード “[システム]、[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**  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) タブ選択エリア

タブを選択することにより、詳細情報を示すカテゴリが切り替わります。

このパネルには、次のタブが存在します（各タブ上における表示内容／設定方法についての詳細は、該当するタブの項を参照してください）。

- [端子配置の設定] タブ
- [端子配置表の情報] タブ
- [端子配置図の設定] タブ
- [出力設定] タブ
- [マクロ設定] タブ
- [ファイル設定] タブ

## [[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。

## [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。

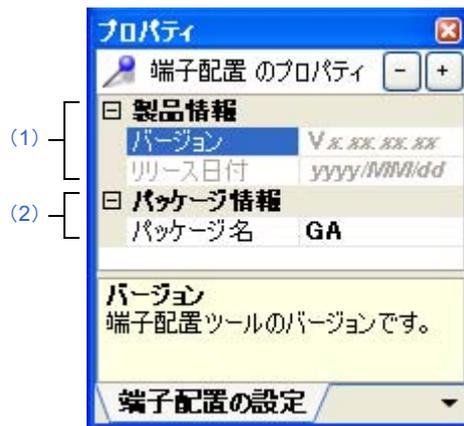
### (2) 項目を編集中外の場合

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。

## [端子配置の設定] タブ

プロジェクト・ツリーパネルで選択した [端子配置 (設計ツール)] に対応した情報 (製品情報, パッケージ情報) の表示を行います。

図 A—4 [端子配置の設定] 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)] → [端子配置 (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)] → [端子配置 (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 [端子配置 (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

端子配置に関する製品情報 (バージョン, リリース日付) の表示を行います。

バージョン	端子配置 (端子配置プラグイン) のバージョンを表示します。
リリース日付	端子配置 (端子配置プラグイン) のリリース日付を表示します。

## (2) [パッケージ情報] カテゴリ

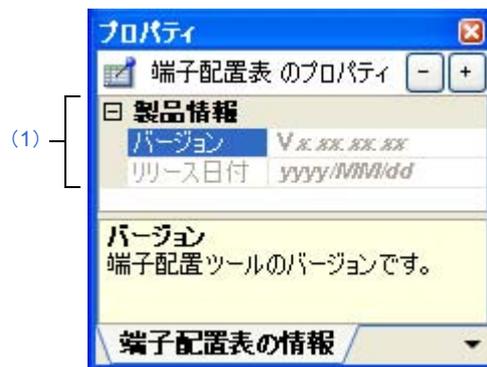
端子配置図 [パネル](#)に端子配置図として表示するマイクロコントローラの形状（パッケージ名）を選択します。

パッケージ名	端子配置図として表示するマイクロコントローラの形状を選択します。
--------	----------------------------------

## [端子配置表の情報] タブ

プロジェクト・ツリーパネルで選択した [端子配置表] に対応した情報（製品情報）の表示を行います。

図 A—5 [端子配置表の情報] 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)] → [端子配置 (設計ツール)] → [端子配置表] 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)] → [端子配置 (設計ツール)] → [端子配置表] 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 [端子配置表] 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

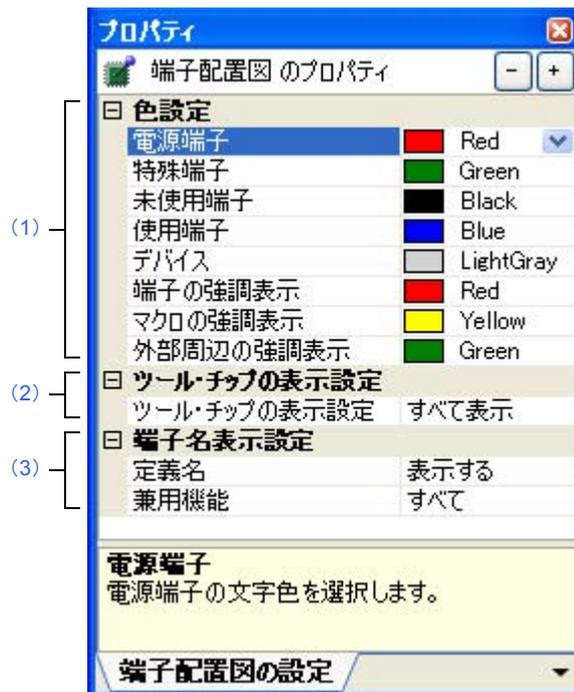
端子配置に関する製品情報（バージョン、リリース日付）の表示を行います。

バージョン	端子配置（端子配置プラグイン）のバージョンを表示します。
リリース日付	端子配置（端子配置プラグイン）のリリース日付を表示します。

## [端子配置図の設定] タブ

プロジェクト・ツリーパネルで選択した [端子配置図] に対応した情報（色設定、ツール・チップの表示設定、端子名表示設定）の表示、および設定の変更を行います。

図 A—6 [端子配置図の設定] 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)] → [端子配置 (設計ツール)] → [端子配置図] 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)] → [端子配置 (設計ツール)] → [端子配置図] 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 [端子配置図] 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) [色設定] カテゴリ

端子配置図の端子をグループ単位（電源端子、特殊端子など）に区別するための表示色を選択します。

電源端子	電源端子（用途が電源に限定されている端子）の表示色を選択します。
特殊端子	特殊端子（用途が規定されている端子）の表示色を選択します。
未使用端子	未使用端子（端子配置表 パネルにおいて、用途が未設定の兼用端子）の表示色を選択します。
使用端子	使用端子（端子配置表 パネルにおいて、用途が設定済みの兼用端子）の表示色を選択します。
デバイス	マイクロコントローラ本体部の表示色を選択します。
端子の強調表示	端子配置表 パネルの [端子番号] タブで選択された項目に対応した端子の背景色を選択します。
マクロの強調表示	端子配置表 パネルの [マクロ] タブで選択された項目に対応した端子の背景色を選択します。
外部周辺の強調表示	端子配置表 パネルの [外部周辺] タブで選択された項目に対応した端子の背景色を選択します。

**備考** 色設定の変更は、本エリア内のドロップダウン・リストを選択することによりオープンする以下のカラー・パレットで行います。

図 A-7 カラー・パレット



### (2) [ツール・チップの表示設定] カテゴリ

端子配置図の端子上にマウス・カーソルを移動した際、該当端子に関する情報をポップアップ表示させるか否かを選択します。

ツール・チップの表示設定	端子配置図の端子上にマウス・カーソルを移動した際、該当端子に関する情報をポップアップ表示させるか否かを選択します。	
	すべて表示	端子配置表の“説明”、“未使用時の処置方法”、“注意事項”に記載されている文字列を表示します。
	説明／未使用時の処置方法のみ	端子配置表の“説明”、“未使用時の処置方法”に記載されている文字列を表示します。
	注意事項のみ	端子配置表の“注意事項”に記載されている文字列を表示します。
	表示しない	端子上にマウス・カーソルを移動しても、何も表示しません。

## (3) [端子名表示設定] カテゴリ

端子の付加情報を端子配置図に表示するか否かを選択します。

定義名	端子配置図の端子を、端子配置表の“定義名”に記載された文字列を付与した形式で表示するか否かを選択します。	
	表示する	端子配置表の“定義名”に記載されている文字列を付与した形式で表示します。
	表示しない	端子配置表の“定義名”に記載されている文字列を付与しません。
兼用機能	端子配置表の“選択機能”で機能を選択した際、非選択機能についても端子配置図に表示するか否かを選択します。	
	すべて	端子配置表の“選択機能”で選択された機能をかっこで括った形式で表示します。
	選択機能のみ	端子配置表の“選択機能”で選択された機能のみを端子配置図に表示します。

## [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, 端子配置連携モード) の表示, および設定の変更を行います。

図 A—8 [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]

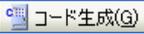
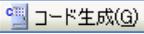
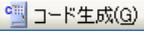
#### (1) [製品情報] カテゴリ

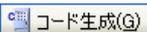
コード生成に関する製品情報 (バージョン, リリース日付) の表示を行います。

バージョン	コード生成（コードライブラリ）のバージョンを表示します。
リリース日付	コード生成（コードライブラリ）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（API 関数の出力制御、モードなど）の表示、および設定の変更を行います。

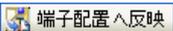
API 関数の出力制御	 ボタンをクリックした際に出力する API 関数の種類（全 API 関数、初期化用 API 関数のみ）を表示／選択します。	
	設定に合わせてすべて出力する	全 API 関数の出力となります。
	初期化関数のみ出力する	初期化用 API 関数のみの出力となります。
モード	 ボタンをクリックした際の動作モードを表示／選択します。 なお、[ファイル] メニュー→[コード生成レポートを保存] を選択した際の動作モードは、“ファイルを上書きする”となります。	
	すでにファイルがあれば何もしない	同一のファイル名を有するファイルが既存していた場合、該当ファイルの出力を行いません。
	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、 <pre>/* Start user code ... Do not edit comment generated here */</pre> から <pre>/* End user code. Do not edit comment generated here */</pre> で囲まれた部位に限られます。
	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。
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。

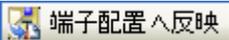
プロジェクトへの登録	 ボタンをクリックした際に出力するソース・コードをプロジェクトに登録するか否かを選択します。	
	出力ファイルをプロジェクトに登録する	出力したソース・コードをプロジェクトに登録します。 なお、登録されたソース・コードは、 <a href="#">プロジェクト・ツリーパネル</a> の [ファイル] - [コード生成] ノードの直下に表示されます。
	出力ファイルをプロジェクトに登録しない	出力したソース・コードをプロジェクトに登録しません。

**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

コード生成と端子配置の情報連携（モード）に関する設定を行います。

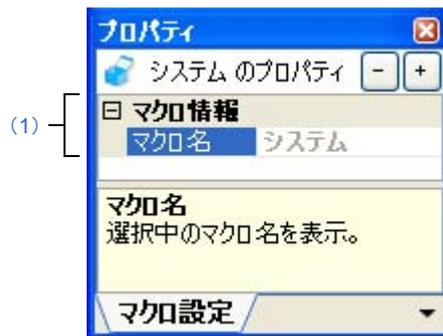
モード	 ボタンをクリックした際、 <a href="#">コード生成パネル</a> で設定した各種情報を <a href="#">端子配置表パネル</a> に反映するか否かを選択します。	
	反映する	<a href="#">コード生成パネル</a> で設定した各種情報を <a href="#">端子配置表パネル</a> に反映します。
	反映しない	<a href="#">コード生成パネル</a> で設定した各種情報を <a href="#">端子配置表パネル</a> に反映しません。

**備考** “反映しない” を選択した際には、 ボタンがグレー表記（非選択状態）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 “[システム], [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—9 [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 “[システム], [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 “[システム], [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 “[システム], [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

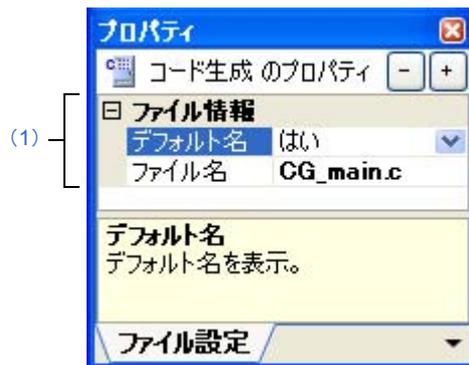
プロジェクト・ツリー パネルで選択した周辺機能ノード “[システム], [ポート] など”, コード生成 パネルでクリックした周辺機能ボタンに関する情報（マクロ名）の表示、および設定の変更を行います。

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—10 [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。	

## 端子配置表 パネル

マイクロコントローラの各端子に関する情報を記述します。

**備考** ツールバーの 100% , または [Ctrl] キーを押下しながらマウス・ホイールを操作することにより、**端子配置表エリア**の内容を拡大／縮小することができます。

図 A—11 端子配置表 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 (端子配置表 パネル専用部分)]
- [[ヘルプ] メニュー (端子配置表 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)] → [端子配置 (設計ツール)] → [端子配置表] をダブルクリック
- プロジェクト・ツリーパネルの [Project name (プロジェクト)] → [端子配置 (設計ツール)] → [端子配置表] を選択したのち、[Enter] キーを押下
- [表示] メニュー → [端子配置] → [端子配置表] を選択

### [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から構成されています。

	端子配置表エリアの情報を展開表示します。
	端子配置表エリアの情報を折りたたみ表示します。

	[マクロ] タブの第 1 階層に表示されている周辺機能を選択したのち、本ボタンをクリックすることにより、選択機能、I/O、N-ch などといった各欄に対する情報の設定処理が自動実行されます。
	[マクロ] タブの第 1 階層に表示されている周辺機能を選択したのち、本ボタンをクリックすることにより、選択機能、I/O、N-ch などといった各欄の情報が初期化されます。
	本ボタンをクリックすることにより、[外部周辺] タブに該当情報が、端子配置図パネルに外部周辺コントローラが作成表示されます。
	[外部周辺] タブの第 1 階層に表示されている外部周辺コントローラを選択したのち、本ボタンをクリックすることにより、該当情報が削除されます。

- 備考 1.  ボタンをクリックした際には、[端子番号] タブ、および [マクロ] タブの“外部周辺”列の選択肢として該当情報が追加されます。
2.  ボタンをクリックした際には、端子配置図パネルの端子配置図エリアから該当外部周辺部品が削除されます。

## (2) 端子配置表エリア

マイクロコントローラの各端子に関する情報を記述するための“端子配置表”を表示します。

## (3) タブ選択エリア

タブを選択することにより、“マイクロコントローラの各端子に関する情報”の表示順序が切り替わります。本パネルには、次のタブが存在します。

- [端子番号] タブ

マイクロコントローラの各端子に関する情報を端子番号順で表示

- [マクロ] タブ

マイクロコントローラの各端子に関する情報を周辺機能単位にグルーピングされた順序で表示

- [外部周辺] タブ

外部周辺に接続された端子に関する情報を外部周辺部品単位にグルーピングされた順序で表示

## [[ファイル] メニュー (端子配置表 パネル専用部分)]

端子配置表 を保存	レポート・ファイル (端子配置を用いて設定した情報を保持したファイル: 端子配置表) を既存のファイルに上書き保存します。
名前を付けて 端子配置表 を保存 ...	レポート・ファイル (端子配置を用いて設定した情報を保持したファイル: 端子配置表) 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。

## [[ヘルプ] メニュー (端子配置表 パネル専用部分)]

端子配置表 パネルのヘルプを開く	本パネルのヘルプを表示します。
------------------	-----------------

## [端子番号] タブ

マイクロコントローラの各端子に関する情報を端子番号順で表示します。

図 A—12 [端子番号] 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・ツリーパネルの [Project name (プロジェクト)] → [端子配置 (設計ツール)] → [端子配置表] をダブルクリック
- プロジェクト・ツリーパネルの [Project name (プロジェクト)] → [端子配置 (設計ツール)] → [端子配置表] を選択したのち、[Enter] キーを押下
- [表示] メニュー → [端子配置] → [端子配置表] を選択

### [各エリアの説明]

#### (1) 端子配置表エリア

マイクロコントローラの各端子に関する情報を記述するための“端子配置表”を表示します。

なお、本エリアの端子配置表は、端子番号順となっています。

以下に、端子配置表を構成する列を示します。

列の見出し	概要
端子番号	該当端子の端子番号を表示します。

列の見出し	概要
端子名	該当端子の端子名を表示します。
選択機能	該当端子が複数の機能を有している際、“どのような機能で利用するのか”を選択するための領域です。
I/O	該当端子の入出力モードを選択するための領域です。
N-ch	該当端子を出力モードで使用する際、“どのような出力モードで利用するのか”を選択するための領域です。
定義名	該当端子に“ユーザ独自の端子名”を付与するための領域です。 なお、定義名として入力可能な文字数は、256文字までに限られます。
説明	該当端子の機能概要を表示します。
未使用時の処置方法	該当端子を使用しない場合の処置方法を表示します。 なお、本欄は、“兼用端子名”欄で Free が選択されている場合に限り表示されます。
注意事項	該当端子を使用するうえで注意すべき事項を表示します。
外部周辺	該当端子を“どの外部周辺コントローラに接続するのか”を選択するための領域です。

- 備考 1.** “端子番号”，“端子名”，“説明”，“未使用時の処置方法”，“注意事項”については、固定化された情報のため、該当欄に情報を追記することはできません。
- “選択機能”欄の Free を固有端子名に変更した場合、[端子配置図パネル](#)の該当端子色が[プロパティパネル](#)の[\[端子配置図の設定\] タブ](#)→[\[色設定\]](#)で選択された“未使用端子の表示色”から“使用端子の表示”へと変化します。
  - 列の移動（表示順序の変更）は、端子配置表の該当列をドラッグしたのち、移動先にドロップすることにより行います。
  - “ユーザ独自の列”を追加する場合、端子配置表の左上に設けられた  ボタンをクリックすることによりオープンする[列の選択ダイアログ](#)の[\[新しい列 ...\]](#) ボタンをクリックすることによりオープンする[新しい列ダイアログ](#)で行います。

## [マクロ] タブ

マイクロコントローラの各端子に関する情報を周辺機能単位にグルーピングされた順序で表示します。

図 A—13 [マクロ] 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・ツリーパネルの [Project name (プロジェクト)] → [端子配置 (設計ツール)] → [端子配置表] をダブルクリック
- プロジェクト・ツリーパネルの [Project name (プロジェクト)] → [端子配置 (設計ツール)] → [端子配置表] を選択したのち、[Enter] キーを押下
- [表示] メニュー → [端子配置] → [端子配置表] を選択

### [各エリアの説明]

#### (1) 端子配置表エリア

マイクロコントローラの各端子に関する情報を記述するための“端子配置表”を表示します。  
 なお、本エリアの端子配置表は、周辺機能単位にグルーピングされた順序となっています。

#### (a) 第1階層

以下に、端子配置表を構成する列を示します。

列の見出し	概要
マクロ名	周辺機能の名称を表示します。
総数	周辺機能に対して割り当てられている端子の総数を表示します。
使用中	用途が設定済みの端子の総数を表示します。
他で使用中	他の周辺機能で用途が設定済みの端子の総数を表示します。

## (b) 第2階層

列の見出し	概要
端子番号	該当端子の端子番号を表示します。
端子名	該当端子の端子名を表示します。
選択機能	該当端子が複数の機能を有している際、“どのような機能で利用するのか”を選択するための領域です。
I/O	該当端子の入出力モードを選択するための領域です。
N-ch	該当端子を出力モードで使用する際、“どのような出力モードで利用するのか”を選択するための領域です。
定義名	該当端子に“ユーザ独自の端子名”を付与するための領域です。 なお、定義名として入力可能な文字数は、256文字までに限られます。
説明	該当端子の機能概要を表示します。
未使用時の処置方法	該当端子を使用しない場合の処置方法を表示します。 なお、本欄は、“選択機能”欄でFreeが選択されている場合に限り表示されます。
注意事項	該当端子を使用するうえで注意すべき事項を表示します。
外部周辺	該当端子を“どの外部周辺コントローラに接続するのか”を選択するための領域です。

- 備考 1.** “マクロ名”，“総数”，“使用中”，“他で使用中”，“端子番号”，“端子名”，“説明”，“未使用時の処置方法”，“注意事項”については、固定化された情報のため、該当欄に情報を追記することはできません。
- 2.** “選択機能”欄のFreeを固有端子名に変更した場合、[端子配置図パネル](#)の該当端子色が[プロパティパネル](#)の[\[端子配置図の設定\]タブ](#)→[\[色設定\]](#)で選択された“未使用端子の表示色”から“使用端子の表示”へと変化します。
- 3.** 列の移動（表示順序の変更）は、端子配置表の該当列をドラッグしたのち、移動先にドロップすることにより行います。
- 4.** “ユーザ独自の列”を追加する場合、端子配置表の左上に設けられた ボタンをクリックすることによりオープンする[列の選択ダイアログ](#)の[\[新しい列 ...\]](#) ボタンをクリックすることによりオープンする[新しい列ダイアログ](#)で行います。

## [外部周辺] タブ

外部周辺に接続された端子に関する情報を外部周辺部品単位にグルーピングされた順序で表示します。

図 A—14 [外部周辺] 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・ツリーパネルの [Project name (プロジェクト)] → [端子配置 (設計ツール)] → [端子配置表] をダブルクリック
- プロジェクト・ツリーパネルの [Project name (プロジェクト)] → [端子配置 (設計ツール)] → [端子配置表] を選択したのち、[Enter] キーを押下
- [表示] メニュー → [端子配置] → [端子配置表] を選択

### [各エリアの説明]

#### (1) 端子配置表エリア

外部周辺コントローラの各端子に関する情報を記述するための“端子配置表”を表示します。

なお、本エリアの端子配置表は、外部周辺コントローラ単位にグルーピングされた順序となっています。

#### (a) 第1階層

以下に、端子配置表を構成する列を示します。

列の見出し	概要
外部周辺名	外部周辺コントローラの名称を表示します。 なお、名称を変更する場合は、本欄を選択したのち、[F2] キーを押下することにより行います。
総数	マイクロコントローラとの接続用に割り当てられている端子の総数を表示します。

## (b) 第2階層

列の見出し	概要
端子番号	該当端子の端子番号を表示します。
端子名	該当端子の端子名を表示します。
選択機能	該当端子が複数の機能を有している際、“どのような機能で利用するのか”を選択するための領域です。
I/O	該当端子の入出力モードを選択するための領域です。
N-ch	該当端子を出力モードで使用する際、“どのような出力モードで利用するのか”を選択するための領域です。
定義名	該当端子に“ユーザ独自の端子名”を付与するための領域です。 なお、定義名として入力可能な文字数は、256文字までに限られます。
説明	該当端子の機能概要を表示します。
未使用時の処置方法	該当端子を使用しない場合の処置方法を表示します。 なお、本欄は、“選択機能”欄で Free が選択されている場合に限り表示されます。
注意事項	該当端子を使用するうえで注意すべき事項を表示します。

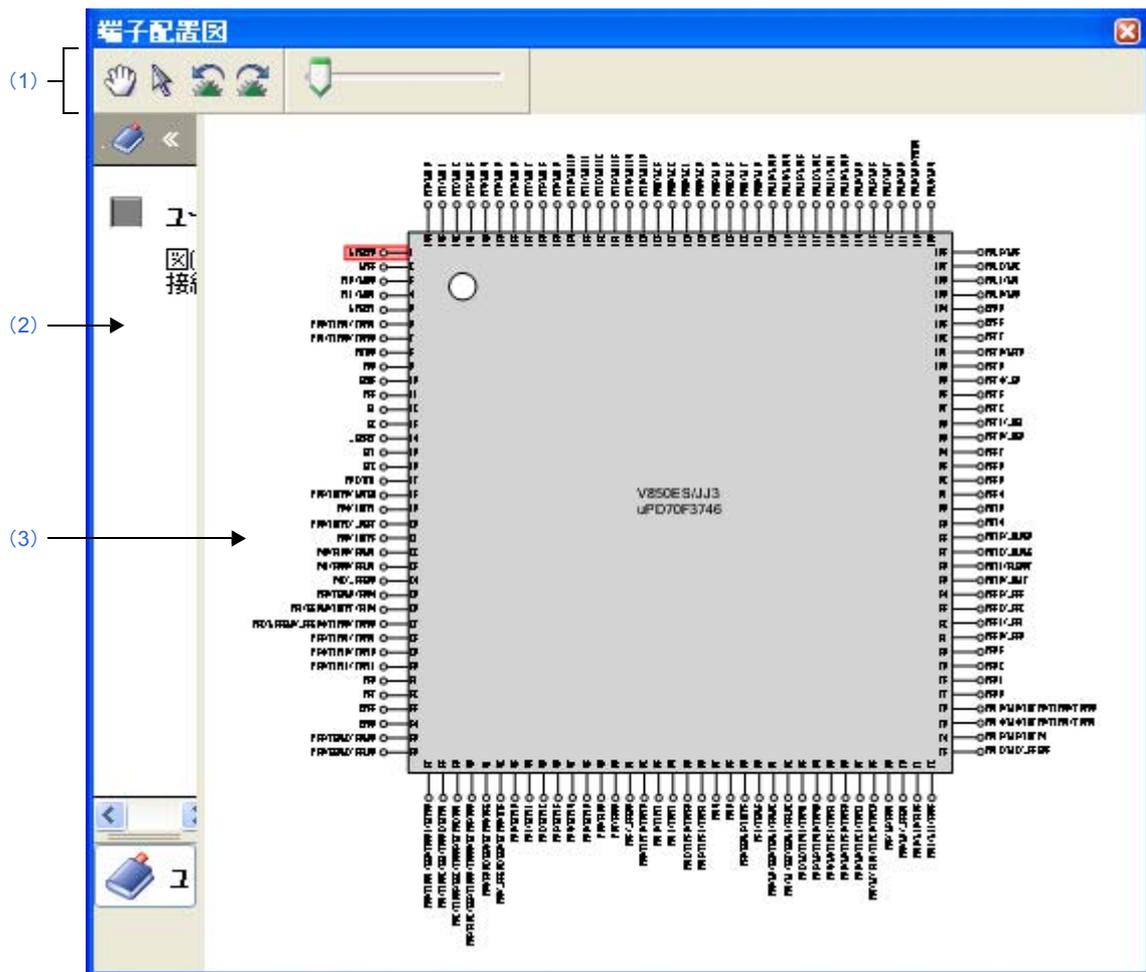
- 備考 1.** “接続数”，“端子番号”，“端子名”，“説明”，“未使用時の処置方法”，“注意事項”については、固定化された情報のため、該当欄に情報を追記することはできません。
- 2.** “選択機能”欄の Free を固有端子名に変更した場合、[端子配置図パネル](#)の該当端子色が[プロパティパネル](#)の[\[端子配置図の設定\] タブ](#)→[\[色設定\]](#)で選択された“未使用端子の表示色”から“使用端子の表示”へと変化します。
- 3.** 列の移動（表示順序の変更）は、端子配置表の該当列をドラッグしたのち、移動先にドロップすることにより行います。
- 4.** “ユーザ独自の列”を追加する場合、端子配置表の左上に設けられた ボタンをクリックすることによりオープンする[列の選択ダイアログ](#)の[\[新しい列 ...\]](#) ボタンをクリックすることによりオープンする[新しい列ダイアログ](#)で行います。

# 端子配置図 パネル

端子配置表 パネルにおける情報の記述状況を表示します。

備考 ツールバーの  により、端子配置図エリアの内容を拡大／縮小することができます。

図 A—15 端子配置図 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 (端子配置図 パネル専用部分)]
- [[ヘルプ] メニュー (端子配置図 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・ツリーパネルの [Project name (プロジェクト)] → [端子配置 (設計ツール)] → [端子配置図] をダブルクリック
- プロジェクト・ツリーパネルの [Project name (プロジェクト)] → [端子配置 (設計ツール)] → [端子配置図] を選択したのち, [Enter] キーを押下
- [表示] メニュー → [端子配置] → [端子配置図] を選択

**備考** プロパティパネルの [端子配置の設定] タブでパッケージ名に“BGA”を選択している場合, 本パネルをオープンすることができません。

## [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から構成されています。

	本ボタンをクリックすることにより, ドラッグ・アンド・ドロップで端子配置図エリアの表示部分を変更することが可能となります。 なお, 本ボタンのクリックにより, 端子配置図エリア内におけるマウス・カーソルの形状が矢印から手形へと変化します。
	本ボタンをクリックすることにより, 端子配置図エリアに表示されている外部周辺部品を任意の位置に移動したり, 端子を選択したりすることが可能となります。 なお, 本ボタンのクリックにより,  ボタンのクリックにより変化したマウス・カーソルの形状が手形から矢印へと戻ります。
	端子配置図エリアの内容を左に 90 度回転します。
	端子配置図エリアの内容を右に 90 度回転します。
	端子配置図エリアの内容を拡大/縮小します。

### (2) [ユーザ定義] エリア

本エリア内の  ボタンを端子配置図エリアにドラッグ・アンド・ドロップすることにより, 外部周辺コントローラが作成表示されます。

### (3) 端子配置図エリア

マイクロコントローラの端子配置状況を表示します。

なお, 端子配置の設定状況については, プロパティパネルの [端子配置図の設定] タブ → [色設定] で指定された色での表示となります。

**備考** 図中の端子名をダブルクリックした際には, 端子配置表パネルがオープンし, 表中の該当端子にフォーカスが遷移します。

## [[ファイル] メニュー (端子配置図 パネル専用部分)]

端子配置図 を保存	レポート・ファイル (端子配置を用いて設定した情報を保持したファイル: 端子配置図) を既存のファイルに上書き保存します。
名前を付けて 端子配置図 を保存 ...	レポート・ファイル (端子配置を用いて設定した情報を保持したファイル: 端子配置図) 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。

## [[ヘルプ] メニュー (端子配置図 パネル専用部分)]

端子配置図 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]

端子配置図エリアの端子上、または外部周辺コントローラ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・メニューは、以下のとおりです。

## (1) 端子上で右クリックした場合

機能選択	該当端子が複数の機能を有している際、“どのような機能で利用するのか”を選択します。
外部周辺選択	該当端子を“どの外部周辺コントローラに接続するのか”を選択します。

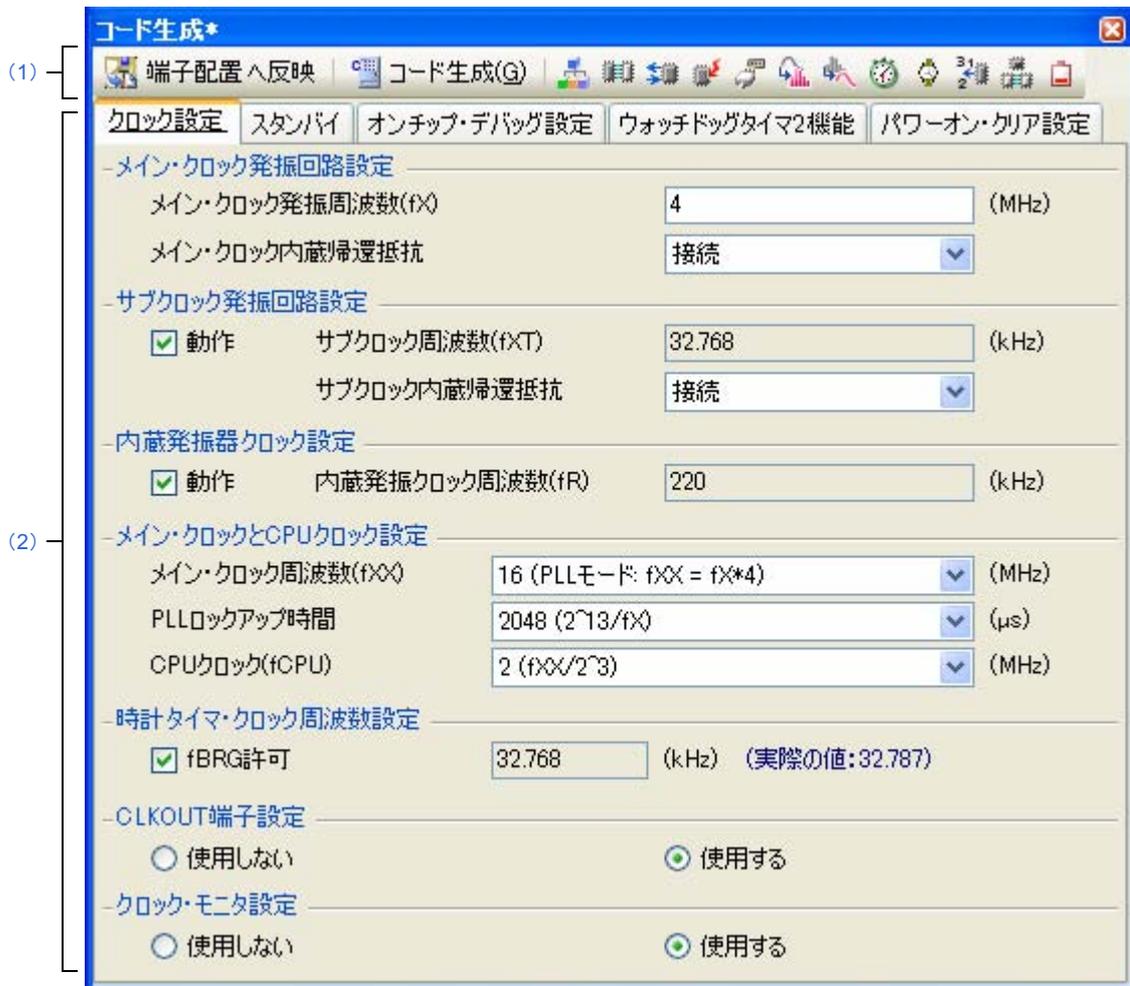
## (2) 外部周辺コントローラ上で右クリックした場合

接続端子の切断	該当端子との接続を切断します。
外部周辺削除	外部周辺コントローラを削除します。

## 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—16 コード生成 パネル : [システム]



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 “[システム], [ポート] など” をダブルクリック
- 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 “[システム], [ポート] など” を選択したのち, [Enter] キーを押下

**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	本パネルで設定した各種情報を <a href="#">端子配置表</a> パネルに反映するとともに、 <a href="#">出力パネル</a> に変更内容を出力します。 なお、本ボタンは、 <a href="#">[出力設定] タブ</a> の <a href="#">[端子配置連携モード]</a> カテゴリで“反映しない”が選択されている際には、グレー表記（非選択状態）となります。
 コード生成(G)	<a href="#">プロパティ</a> パネルの <a href="#">[出力設定] タブ</a> → <a href="#">[生成先フォルダ]</a> 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。
	<a href="#">情報設定エリア</a> 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機能、スタンバイ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 <a href="#">[システム]</a> ”へと切り替えます。
	<a href="#">情報設定エリア</a> の表示内容を“マイクロコントローラが提供している外部バス・インタフェースの機能（外部メモリ領域に接続する機能）を制御するうえで必要な情報を設定するための <a href="#">[外部バス]</a> ”へと切り替えます。
	<a href="#">情報設定エリア</a> 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 <a href="#">[ポート]</a> ”へと切り替えます。
	<a href="#">情報設定エリア</a> の表示内容を“マイクロコントローラが提供している外部割り込み／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 <a href="#">[割り込み]</a> ”へと切り替えます。
	<a href="#">情報設定エリア</a> の表示内容を“マイクロコントローラが提供しているアシンクロナス・シリアル・インタフェース A (UARTA), 3 線式可変長シリアル I/O (CSIB) など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 href="#">[シリアル]</a> ”へと切り替えます。
	<a href="#">情報設定エリア</a> 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 href="#">[A/D コンバータ]</a> ”へと切り替えます。
	<a href="#">情報設定エリア</a> 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 <a href="#">[D/A コンバータ]</a> ”へと切り替えます。
	<a href="#">情報設定エリア</a> の表示内容を“マイクロコントローラが提供している 16 ビット・タイマ／イベントカウンタ P (TMP), 16 ビット・タイマ／イベント・カウンタ Q (TMQ), および 16 ビット・インターバル・タイマ (TMM) 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 href="#">[タイマ]</a> ”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時計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 [時計タイマ]”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・カウン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 [リアルタイム・カウンタ]”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リアルタイム出力機能を制御するうえで必要な情報を設定するための [リアルタイム出力機能]”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低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 [低電圧検出回路]”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

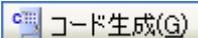
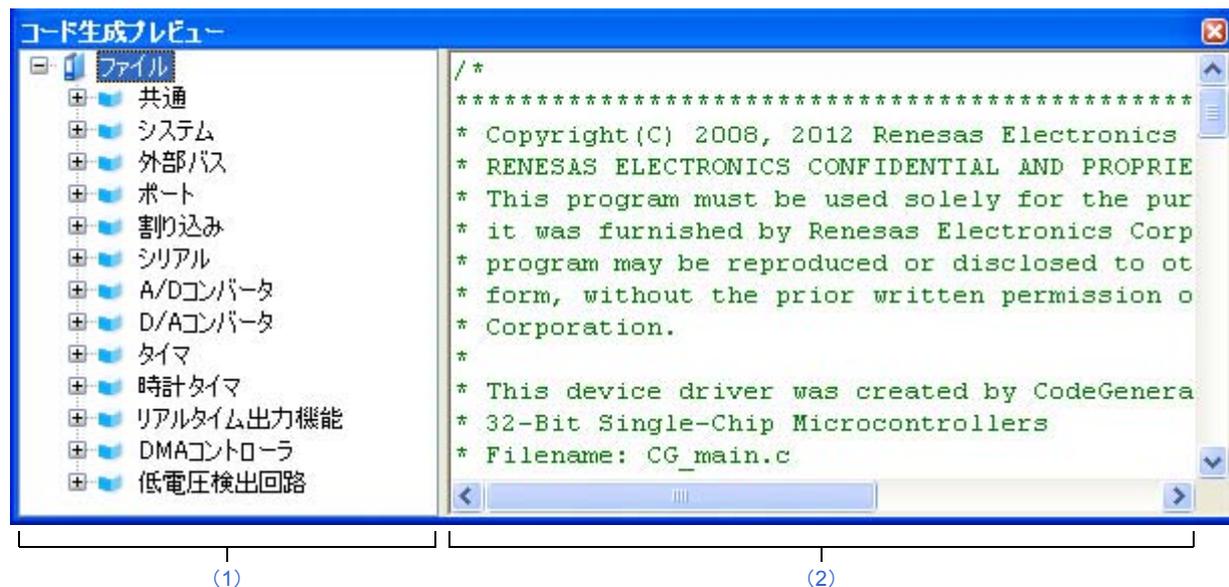
コード生成パネルの  ボタンをクリックした際に出力されるソース・コード（デバイス・ドライバ・プログラム）の出力有無を API 関数単位で確認／設定するとともに、コード生成パネルで設定した情報に応じたソース・コードの確認を行います。

図 A—17 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。

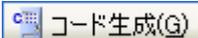
- [オープン方法]
- [各エリアの説明]
- [[ファイル] メニュー（コード生成プレビューパネル専用部分）]
- [[ヘルプ] メニュー（コード生成プレビューパネル専用部分）]
- [コンテキスト・メニュー]

### [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  **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、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	<p>選択された API 関数のソース・コードを<b>プロパティ パネル</b>の <b>[出力設定] タブ</b>→ [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。</p> <p>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。</p> <p>本項目は, 選択された API 関数が初期化用 API 関数以外であり, <b>プロパティ パネル</b>の <b>[出力設定] タブ</b>→ [API 関数の出力制御] で “初期化関数のみ出力する” が選択されている際には, グレー表記 (非選択状態) となります。</p>
コードを生成しない	<p>選択された API 関数のソース・コードを<b>コード生成 パネル</b>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。</p> <p>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。</p>
名前を変更する	<p>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。</p> <p>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。</p>
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 <b>プロパティ パネル</b> をオープンします。

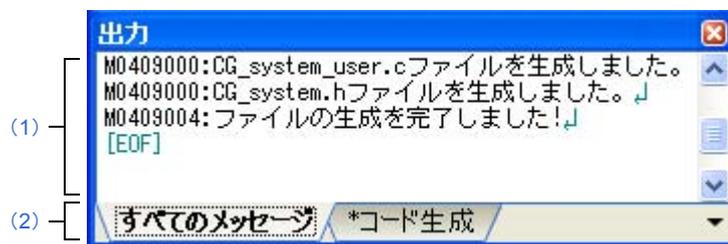
## 出力パネル

CubeSuite+ が提供している各種コンポーネント（設計ツールを含む、ビルド・ツール／デバッグ・ツール／解析ツールなど）から出力されるメッセージの表示を行います。

メッセージは、出力元のメッセージごとに分類されたタブ上でそれぞれ個別に表示されます。

**備考** ツールバーの 100%、または [Ctrl] キーを押下しながらマウス・ホイールを操作することにより、[メッセージ・エリア](#)の内容を拡大／縮小することができます。

図 A—18 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）]
- [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

各ツールから出力されたメッセージを表示します。

なお、メッセージの表示色は、出力メッセージの種別により、次のように異なります（表示の際の文字色／背景色はオプション ダイアログにおける [全般 - フォントと色] カテゴリの設定に依存）。

メッセージ種別	表示例（デフォルト）		説明	
通常メッセージ	Aaああアア亜宇	文字色	黒	何らかの情報を通知する際に表示されます。
		背景色	白	
警告メッセージ	Aaああアア亜宇	文字色	青	操作に対して、何らかの警告を通知する際に表示されます。
		背景色	標準色	

メッセージ種別	表示例 (デフォルト)		説明
エラー・メッセージ		文字色	赤
		背景色	グレー
			致命的なエラー、または操作ミスにより実行が不可能な場合に表示されます。

(2) タブ選択エリア

メッセージの出力元を示すタブを選択します。

設計ツールでは、次のタブを使用します。

タブ名	説明
すべてのメッセージ	CubeSuite+ が提供している全コンポーネント (設計ツールを含む、ビルド・ツール/デバッグ・ツール/解析ツールなど) から出力されるメッセージを表示します (ラピッド・ビルドの実行によるメッセージを除く)。
コード生成	CubeSuite+ が提供している各種コンポーネント (設計ツールを含む、ビルド・ツール/デバッグ・ツール/解析ツールなど) から出力されるメッセージのうち、コード生成が出力するメッセージを表示します。

**注意** 新たなメッセージが非選択状態のタブ上に出力されても、自動的なタブの表示切り替えは行いません。  
 この場合、タブ名の先頭に “\*” が付加し、新たなメッセージが出力されていることを示します。

[[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。

[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、カー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。

# 列の選択 ダイアログ

本ダイアログに表示されている項目を端子配置表に表示するか否かの選択、および端子配置表に対する列の追加／削除を行います。

図 A—19 列の選択 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]

- 端子配置表 パネルの [端子番号] タブにおいて、 ボタンをクリック
- 端子配置表 パネルの [マクロ] タブにおいて、 ボタンをクリック
- 端子配置表 パネルの [外部周辺] タブにおいて、 ボタンをクリック

## [各エリアの説明]

### (1) 操作対象選択エリア

本ダイアログの操作対象となる端子配置表を選択します。

端子番号	[端子番号] タブの端子配置表を操作対象とします。
マクロ	[マクロ] タブの第 1 階層の端子配置表を操作対象とします。
マクロ - 端子	[マクロ] タブの第 2 階層の端子配置表を操作対象とします。

外部周辺	[外部周辺] タブの第1階層の端子配置表を操作対象とします。
外部周辺 - 端子	[外部周辺] タブの第2階層の端子配置表を操作対象とします。

図 A—20 操作対象 ([端子番号] タブ)



図 A—21 操作対象 ([マクロ] タブ: 第1階層)



図 A—22 操作対象 ([マクロ] タブ: 第2階層)

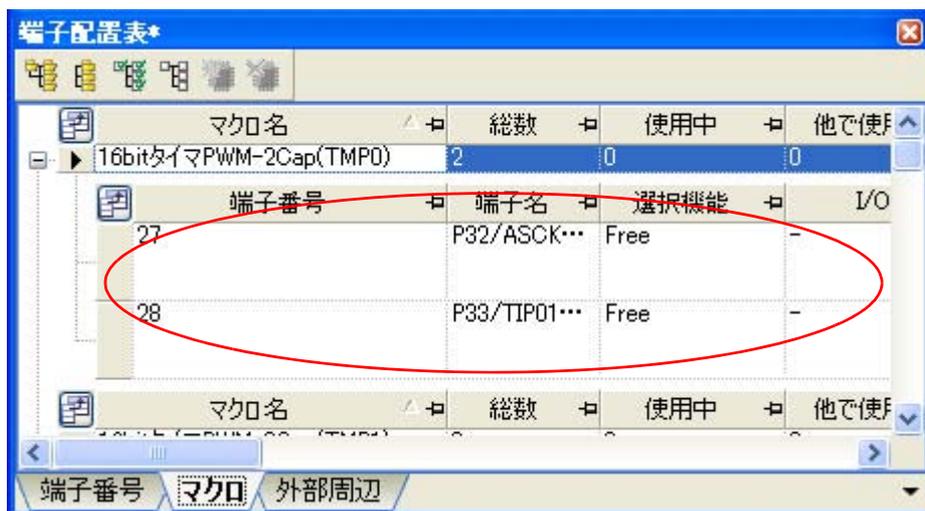


図 A—23 操作対象 ([外部周辺] タブ: 第1階層)

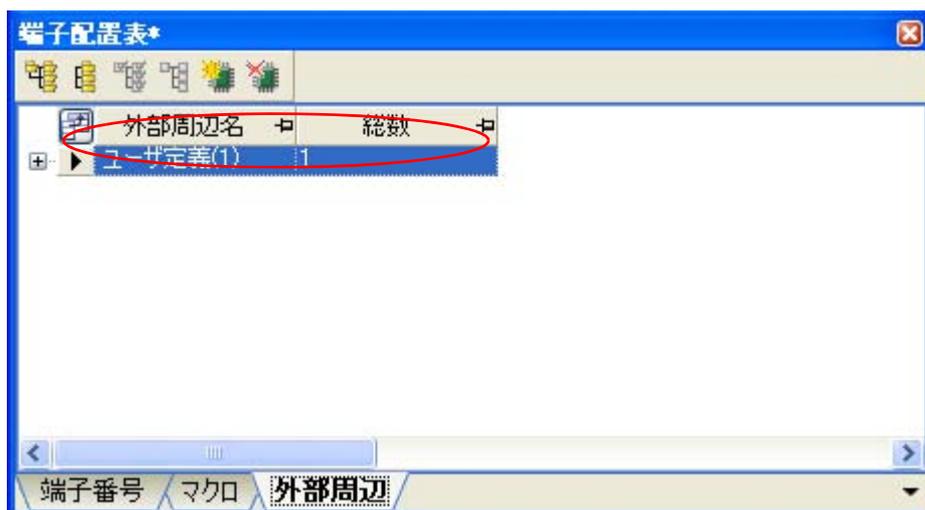
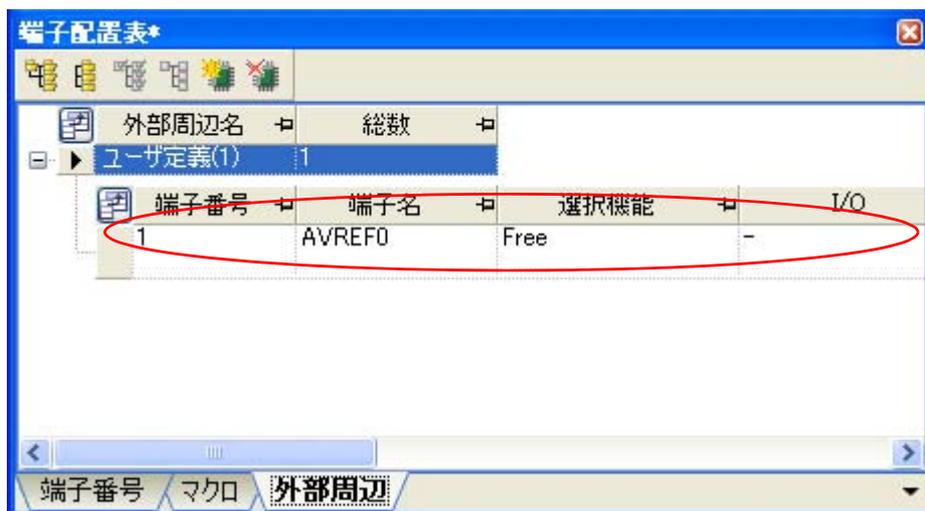


図 A—24 操作対象（[外部周辺] タブ：第2階層）



(2) 表示項目選択エリア

該当項目を操作対象選択エリアで選択された端子配置表に表示するか否かを選択します。

チェック状態	該当項目を端子配置表に表示します。
非チェック状態	該当項目を端子配置表から非表示とします。

[機能ボタン]

ボタン	機能
新しい列 ...	端子配置表に列を追加するための新しい列ダイアログをオープンします。
列の削除	選択された列を端子配置表から削除します。 なお、削除可能な列は、新しい列ダイアログでユーザが独自に追加した列に限られます。
デフォルト	列の並び順を初期状態に戻します。
閉じる	本ダイアログをクローズします。

## 新しい列 ダイアログ

端子配置表に列を追加します。

図 A—25 新しい列 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]

- 列の選択 ダイアログの [新しい列 ...] ボタンをクリック

### [各エリアの説明]

#### (1) [名前]

端子配置表に追加する列の見出しを入力します。

なお、名前として入力可能な文字数は、256文字までに限られます。

#### (2) [型]

端子配置表に追加する列の入力フォームを選択します。

文字列	文字列のみ入力可能な列となります。
チェック・ボックス	チェック・ボックスの設けられた列となります。
整数	整数のみ入力可能な列となります。
実数	実数のみ入力可能な列となります。
日付	年月日形式の日付のみ入力可能な列となります。

**[機能ボタン]**

ボタン	機能
OK	[名前] で指定された見出しを有する列を端子配置表の右端に追加します。
キャンセル	本ダイアログをクローズします。

# 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。

図 A-26 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]

- ブロパティ 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。

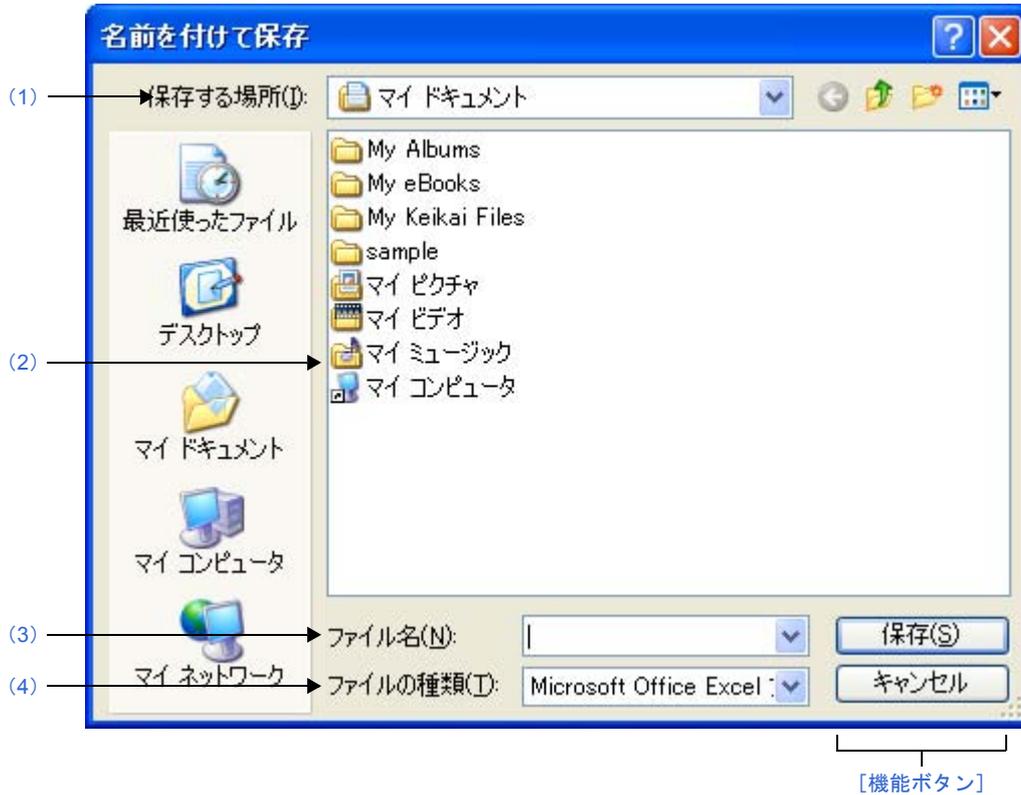
## [機能ボタン]

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—27 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	周辺機能名.c	初期化関数, API 関数
	周辺機能名_user.c	割り込み関数 (MD_INTxxx), コールバック関数
	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	CG_main.c	main 関数, R_MAIN_UserInit 関数
	CG_systeminit.c	各周辺機能の初期化関数コール CG_ReadResetSource のコール
	CG_macrodriver.h	全ソース・ファイルで共通使用するマクロを定義
	CG_userdefine.h	空ファイル (ユーザ定義用)
	CG_lk.dir	リンク・ディレクティブ
	CG_start.s	スタートアップ・ルーチン
	CG_inttab.s	割り込みベクタ・テーブル

### 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
システム	CG_system.c	CLOCK_Init CG_ChangeClockMode CG_ChangeFrequency CG_SelectPowerSaveMode CG_SelectStabTime CG_SelectPIIMode CG_SelectSSCGMode WDT2_Restart CRC_Start CRC_SetData

周辺機能	ファイル名	含まれる API 関数名
システム	CG_system.c	CRC_GetResult
	CG_system_user.c	CLOCK_UserInit CG_ReadResetSource MD_INTWDT2
	CG_system.h	—
外部バス	CG_bus.c	BUS_Init
	CG_bus_user.c	BUS_UserInit
	CG_bus.h	—
ポート	CG_port.c	PORT_Init PORT_ChangePmnInput PORT_ChangePmnOutput
	CG_port_user.c	PORT_UserInit
	CG_port.h	—
割り込み	CG_int.c	INTP_Init KEY_Init INT_MaskableInterruptEnable INTPn_Disable INTPn_Enable KEY_Disable KEY_Enable
	CG_int_user.c	INTP_UserInit KEY_UserInit MD_INTNMI MD_INTPn MD_INTKR
	CG_int.h	—
シリアル	CG_serial.c	UARTAn_Init UARTAn_Start UARTAn_Stop UARTAn_SendData UARTAn_ReceiveData UARTBn_Init UARTBn_Start UARTBn_Stop UARTBn_SendData UARTBn_ReceiveData UARTCn_Init UARTCn_Start UARTCn_Stop UARTCn_SendData UARTCn_ReceiveData CSIBn_Init

周辺機能	ファイル名	含まれる API 関数名
シリアル	CG_serial.c	CSIBn_Start CSIBn_Stop CSIBn_SendData CSIBn_ReceiveData CSIBn_SendReceiveData CSIBn_Init CSIBn_Start CSIBn_Stop CSIBn_SendData CSIBn_ReceiveData CSIBn_SendReceiveData CSIFn_Init CSIFn_Start CSIFn_Stop CSIFn_SendData CSIFn_ReceiveData CSIFn_SendReceiveData IIC0n_Init IIC0n_Stop IIC0n_StopCondition IIC0n_MasterSendStart IIC0n_MasterReceiveStart IIC0n_SlaveSendStart IIC0n_SlaveReceiveStart
	CG_serial_user.c	UARTAn_UserInit UARTAn_SendEndCallback UARTAn_ReceiveEndCallback UARTAn_ErrorCallback UARTAn_SoftOverRunCallback UARTBn_UserInit UARTBn_SendEndCallback UARTBn_ReceiveEndCallback UARTBn_SingleErrorCallback UARTBn_FIFOErrorCallback UARTBn_TimeoutErrorCallback UARTBn_SoftOverRunCallback UARTCn_UserInit UARTCn_SendEndCallback UARTCn_ReceiveEndCallback UARTCn_ErrorCallback UARTCn_SoftOverRunCallback CSIBn_UserInit CSIBn_SendEndCallback CSIBn_ReceiveEndCallback CSIBn_ErrorCallback

周辺機能	ファイル名	含まれる API 関数名
シリアル	CG_serial_user.c	CSIEn_UserInit CSIEn_SendEndCallback CSIEn_ReceiveEndCallback CSIEn_ErrorCallback CSIFn_UserInit CSIFn_SendEndCallback CSIFn_ReceiveEndCallback CSIFn_ErrorCallback IIC0n_UserInit IIC0n_MasterSendEndCallback IIC0n_MasterReceiveEndCallback IIC0n_MasterErrorCallback IIC0n_SlaveSendEndCallback IIC0n_SlaveReceiveEndCallback IIC0n_SlaveErrorCallback IIC0n_GetStopConditionCallback MD_INTUAnT MD_INTUAnR MD_INTUBnTIT MD_INTUBnTIF MD_INTUBnTIR MD_INTUBnTIRE MD_INTUBnTITO MD_INTCBnT MD_INTCBnR MD_INTUCnT MD_INTUCnR MD_INTCEnT MD_INTCEnTIOF MD_INTCFnT MD_INTCFnR MD_INTIICn
	CG_serial.h	—
A/D コンバータ	CG_ad.c	AD_Init AD_Start AD_Stop AD_SelectADChannel AD_SetPFTCondition AD_Read AD_ReadByte
	CG_ad_user.c	AD_UserInit MD_INTAD
	CG_ad.h	—
D/A コンバータ	CG_da.c	DAn_Init

周辺機能	ファイル名	含まれる API 関数名
D/A コンバータ	CG_da.c	DAn_Start DAn_Stop DAn_SetValue
	CG_da_user.c	DAn_UserInit
	CG_da.h	—
タイマ	CG_timer.c	TMPn_Init TMPn_Start TMPn_Stop TMPn_ChangeTimerCondition TMPn_GetPulseWidth TMPn_GetFreeRunningValue TMPn_ChangeDuty TMPn_SoftwareTriggerOn TMQ0_Init TMQ0_Start TMQ0_Stop TMQ0_ChangeTimerCondition TMQ0_GetPulseWidth TMQ0_GetFreeRunningValue TMQ0_ChangeDuty TMQ0_SoftwareTriggerOn TAAAn_Init TAAAn_Start TAAAn_Stop TAAAn_ChangeTimerCondition TAAAn_ControlOutputToggle TAAAn_GetPulseWidth TAAAn_GetFreeRunningValue TAAAn_ChangeDuty TAAAn_SoftwareTriggerOn TABn_Init TABn_Start TABn_Stop TABn_ChangeTimerCondition TABn_ControlOutputToggle TABn_GetPulseWidth TABn_GetFreeRunningValue TABn_ChangeDuty TABn_SoftwareTriggerOn TMT0_Init TMT0_Start TMT0_Stop TMT0_ChangeTimerCondition TMT0_GetPulseWidth

周辺機能	ファイル名	含まれる API 関数名
タイマ	CG_timer.c	TMT0_GetFreeRunningValue TMT0_ChangeDuty TMT0_SoftwareTriggerOn TMT0_EnableHold TMT0_DisableHold TMT0_ChangeCountValue TMMn_Init TMMn_Start TMMn_Stop TMMn_ChangeTimerCondition
	CG_timer_user.c	TMPn_UserInit TMQ0_UserInit TAAAn_UserInit TABn_UserInit TMT0_UserInit TMMn_UserInit MD_INTTPnOV MD_INTTPnCCm MD_INTTQ0OV MD_INTTQ0CCm MD_INTTAAAnOV MD_INTTAAAnCCm MD_INTTABnOV MD_INTTABnCCm MD_INTTT0EC MD_INTTT0OV MD_INTTT0CCm MD_INTTMnEQ0
	CG_timer.h	—
時計タイマ	CG_wt.c	WT_Init WT_Start WT_Stop
	CG_wt_user.c	WT_UserInit MD_INTWT MD_INTWTI
	CG_wt.h	—
リアルタイム・カウンタ	CG_rtc.c	RTC_Init RTC_CounterEnable RTC_CounterDisable RTC_SetHourSystem RTC_CounterSet RTC_CounterGet RTC_ConstPeriodInterruptEnable

周辺機能	ファイル名	含まれる API 関数名
リアルタイム・カウンタ	CG_rtc.c	RTC_ConstPeriodInterruptDisable RTC_AlarmEnable RTC_AlarmDisable RTC_AlarmSet RTC_AlarmGet RTC_IntervalStart RTC_IntervalStop RTC_IntervalInterruptEnable RTC_IntervalInterruptDisable RTC_RC1CK1HZ_OutputEnable RTC_RC1CK1HZ_OutputDisable RTC_RC1CKO_OutputEnable RTC_RC1CKO_OutputDisable RTC_RC1CKDIV_OutputEnable RTC_RC1CKDIV_OutputDisable RTC_RTC1HZ_OutputEnable RTC_RTC1HZ_OutputDisable RTC_RTCCL_OutputEnable RTC_RTCCL_OutputDisable RTC_RTCDIV_OutputEnable RTC_RTCDIV_OutputDisable RTC_ChangeCorrectionValue
	CG_rtc_user.c	RTC_UserInit MD_INTRTCn
	CG_rtc.h	—
リアルタイム出力機能	CG_rto.c	RTO_n_Init RTO_n_Enable RTO_n_Disable RTO_n_Set2BitsData RTO_n_Set4BitsData RTO_n_Set6BitsData RTO_n_Set8BitsData RTO_n_SetHigh2BitsData RTO_n_SetLow2BitsData RTO_n_SetHigh4BitsData RTO_n_SetLow4BitsData RTO_n_GetValue
	CG_rto_user.c	RTO_n_UserInit
	CG_rto.h	—
DMA コントローラ	CG_dma.c	DMA_n_Init DMA_n_Enable DMA_n_Disable DMA_n_CheckStatus

周辺機能	ファイル名	含まれる API 関数名
DMA コントローラ	CG_dma.c	DMA <sub>n</sub> _SetData DMA <sub>n</sub> _SoftwareTriggerOn
	CG_dma_user.c	DMA <sub>n</sub> _UserInit MD_INTDMA <sub>n</sub>
	CG_dma.h	—
低電圧検出回路	CG_lvi.c	LVI_Init LVI_InterruptModeStart LVI_ResetModeStart LVI_Start LVI_Stop
	CG_lvi_user.c	LVI_UserInit MD_INTLVI
	CG_lvi.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。

- 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
システム	CLOCK_Init	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。
	CLOCK_UserInit	クロックに関するユーザ独自の初期化処理を行います。
	CG_ReadResetSource	リセットの発生に伴う処理を行います。
	CG_ChangeClockMode	CPUクロックを変更します。
	CG_ChangeFrequency	CPUクロックの分周比を変更します。
	CG_SelectPowerSaveMode	CPUのスタンバイ・モードを設定します。

周辺機能	API 関数名	機能概要
システム	CG_SelectStabTime	STOP モードが解除された際に必要となる X1 発振回路の発振安定時間を選択します。
	CG_SelectPllMode	PLL 機能の動作モードを選択します。
	CG_SelectSSCGMode	SSCG (Spread Spectrum Clock Generator) の動作状態を選択します。
	WDT2_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。
	CRC_Start	データ・ブロックの誤り検出動作を開始します。
	CRC_SetData	CRC インพุット・レジスタ (CRCIN) にデータを設定します。
	CRC_GetResult	CRC データ・レジスタ (CRCD) に格納されている演算結果を読み出します。
外部バス	BUS_Init	外部バス・インタフェースの機能 (外部バスを内蔵 ROM, RAM, SFR 以外の領域に接続する機能) を制御するうえで必要となる初期化処理を行います。
	BUS_UserInit	外部バス・インタフェースに関するユーザ独自の初期化処理を行います。
ポート	PORT_Init	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。
	PORT_UserInit	ポートに関するユーザ独自の初期化処理を行います。
	PORT_ChangePmnInput	端子の入出力モードを出力モードから入力モードへと切り替えます。
	PORT_ChangePmnOutput	端子の入出力モードを入力モードから出力モードへと切り替えます。
割り込み	INTP_Init	外部割り込み INTP <sub>n</sub> 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。
	INTP_UserInit	外部割り込み INTP <sub>n</sub> に関するユーザ独自の初期化処理を行います。
	KEY_Init	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。
	KEY_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。
	INT_MaskableInterruptEnable	マスクブル割り込みの受け付けを禁止/許可します。
	INTPn_Disable	マスクブル割り込み (外部割り込み要求) INTP <sub>n</sub> の受け付けを禁止します。
	INTPn_Enable	マスクブル割り込み (外部割り込み要求) INTP <sub>n</sub> の受け付けを許可します。
	KEY_Disable	キー割り込み INTKR の受け付けを禁止します。
	KEY_Enable	キー割り込み INTKR の受け付けを許可します。

周辺機能	API 関数名	機能概要
シリアル	UARTAn_Init	アシンクロナス・シリアル・インタフェース A (UARTA) 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。
	UARTAn_UserInit	アシンクロナス・シリアル・インタフェース A (UARTA) に関するユーザ独自の初期化処理を行います。
	UARTAn_Start	アシンクロナス・シリアル・インタフェース A (UARTA) を動作許可状態へと移行します。
	UARTAn_Stop	アシンクロナス・シリアル・インタフェース A (UARTA) を動作禁止状態へと移行します。
	UARTAn_SendData	データの UARTAn 送信を開始します。
	UARTAn_ReceiveData	データの UARTAn 受信を開始します。
	UARTAn_SendEndCallback	UARTAn の連続送信許可割り込み INTUANrT の発生に伴う処理を行います。
	UARTAn_ReceiveEndCallback	UARTAn の受信終了割り込み INTUANrR の発生に伴う処理を行います。
	UARTAn_ErrorCallback	UARTAn 受信エラー割り込み INTUANrR の発生に伴う処理を行います。
	UARTAn_SoftOverRunCallback	オーバラン・エラーの発生に伴う処理を行います。
	UARTBn_Init	アシンクロナス・シリアル・インタフェース B (UARTB) 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。
	UARTBn_UserInit	アシンクロナス・シリアル・インタフェース B (UARTB) に関するユーザ独自の初期化処理を行います。
	UARTBn_Start	アシンクロナス・シリアル・インタフェース B (UARTB) を動作許可状態へと移行します。
	UARTBn_Stop	アシンクロナス・シリアル・インタフェース B (UARTB) を動作禁止状態へと移行します。
	UARTBn_SendData	データの UARTBn 送信を開始します。
	UARTBn_ReceiveData	データの UARTBn 受信を開始します。
	UARTBn_SendEndCallback	UARTBn の送信許可割り込み INTUBnTIT, および FIFO 送信完了割り込み INTUBnTIF の発生に伴う処理を行います。
	UARTBn_ReceiveEndCallback	UARTBn の受信完了割り込み INTUBnTIR の発生に伴う処理を行います。
	UARTBn_SingleErrorCallback	受信エラー割り込み INTUBnTIRE (オーバラン・エラー, フレーミング・エラー, パリティ・エラー) 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	UARTBn_FIFOErrorCallback	受信エラー割り込み INTUBnTIRE (オーバラン・エラー, フレーミング・エラー, パリティ・エラー) の発生に伴う処理を行います。
	UARTBn_TimeoutErrorCallback	受信タイムアウト割り込み INTUBnTITO の発生に伴う処理を行います。
	UARTBn_SoftOverRunCallback	オーバラン・エラーの発生に伴う処理を行います。
	UARTCn_Init	アシンクロナス・シリアル・インタフェース C (UARTC) 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。
	UARTCn_UserInit	アシンクロナス・シリアル・インタフェース C (UARTC) に関するユーザ独自の初期化処理を行います。
	UARTCn_Start	アシンクロナス・シリアル・インタフェース C (UARTC) を動作許可状態へと移行します。
	UARTCn_Stop	アシンクロナス・シリアル・インタフェース C (UARTC) を動作禁止状態へと移行します。
	UARTCn_SendData	データの UARTCn 送信を開始します。
	UARTCn_ReceiveData	データの UARTCn 受信を開始します。
	UARTCn_SendEndCallback	UARTCn の連続送信許可割り込み INTUCnT の発生に伴う処理を行います。
	UARTCn_ReceiveEndCallback	UARTCn の受信終了割り込み INTUCnR の発生に伴う処理を行います。
	UARTCn_ErrorCallback	UARTCn 受信エラー割り込み INTUCnR の発生に伴う処理を行います。
	UARTCn_SoftOverRunCallback	オーバラン・エラーの発生に伴う処理を行います。
	CSIBn_Init	3 線式可変長シリアル I/O B (CSIB) 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。
	CSIBn_UserInit	3 線式可変長シリアル I/O B (CSIB) に関するユーザ独自の初期化処理を行います。
	CSIBn_Start	3 線式可変長シリアル I/O B (CSIB) を動作許可状態へと移行します。
	CSIBn_Stop	3 線式可変長シリアル I/O B (CSIB) を動作禁止状態へと移行します。
	CSIBn_SendData	データの CSIB 送信を開始します。
	CSIBn_ReceiveData	データの CSIB 受信を開始します。
	CSIBn_SendReceiveData	データの CSIB 送受信を開始します。
	CSIBn_SendEndCallback	CSIBn の受信終了割り込み INTCBnR, および CSIBn の連続送信書き込み許可割り込み INTCBnT 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	CSIBn_ReceiveEndCallback	CSIBn の受信終了割り込み INTCBnR の発生に伴う処理を行います。
	CSIBn_ErrorCallback	CSIBn の受信エラー割り込み INTCBnR (オーバーラン・エラー) の発生に伴う処理を行います。
	CSIEn_Init	3 線式可変長シリアル I/O E (CSIE) 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。
	CSIEn_UserInit	3 線式可変長シリアル I/O E (CSIE) に関するユーザ独自の初期化処理を行います。
	CSIEn_Start	3 線式可変長シリアル I/O E (CSIE) を動作許可状態へと移行します。
	CSIEn_Stop	3 線式可変長シリアル I/O E (CSIE) を動作禁止状態へと移行します。
	CSIEn_SendData	データの CSIE 送信を開始します。
	CSIEn_ReceiveData	データの CSIE 受信を開始します。
	CSIEn_SendReceiveData	データの CSIE 送受信を開始します。
	CSIEn_SendEndCallback	CSIEn の送受信完了割り込み INTCEnT の発生に伴う処理を行います。
	CSIEn_ReceiveEndCallback	CSIEn の送受信完了割り込み INTCEnT の発生に伴う処理を行います。
	CSIEn_ErrorCallback	CSIEn の CSIEnBUF オーバフロー割り込み INTCEnTIOF の発生に伴う処理を行います。
	CSIFn_Init	3 線式可変長シリアル I/O F (CSIF) 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。
	CSIFn_UserInit	3 線式可変長シリアル I/O F (CSIF) に関するユーザ独自の初期化処理を行います。
	CSIFn_Start	3 線式可変長シリアル I/O F (CSIF) を動作許可状態へと移行します。
	CSIFn_Stop	3 線式可変長シリアル I/O F (CSIF) を動作禁止状態へと移行します。
	CSIFn_SendData	データの CSIF 送信を開始します。
	CSIFn_ReceiveData	データの CSIF 受信を開始します。
	CSIFn_SendReceiveData	データの CSIF 送受信を開始します。
	CSIFn_SendEndCallback	CSIFn の送受信完了割り込み INTCFnT の発生に伴う処理を行います。
	CSIFn_ReceiveEndCallback	CSIFn の送受信完了割り込み INTCFnT の発生に伴う処理を行います。
	CSIFn_ErrorCallback	CSIFn の受信エラー割り込み INTCFnR (オーバーラン・エラー) の発生に伴う処理を行います。
	IIC0n_Init	IIC バ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。

周辺機能	API 関数名	機能概要
シリアル	IIC0n_UserInit	IIC バスに関するユーザ独自の初期化処理を行います。
	IIC0n_Stop	IIC0n 通信を終了します。
	IIC0n_StopCondition	ストップ・コンディションを生成します。
	IIC0n_MasterSendStart	データの IIC0n マスタ送信を開始します。
	IIC0n_MasterReceiveStart	データの IIC0n マスタ受信を開始します。
	IIC0n_MasterSendEndCallback	IIC0n マスタ送信の転送終了割り込み INTIICn の発生に伴う処理を行います。
	IIC0n_MasterReceiveEndCallback	IIC0n マスタ受信の転送終了割り込み INTIICn の発生に伴う処理を行います。
	IIC0n_MasterErrorCallback	IIC0n マスタ通信エラーの検出に伴う処理を行います。
	IIC0n_SlaveSendStart	データの IIC0n スレーブ送信を開始します。
	IIC0n_SlaveReceiveStart	データの IIC0n スレーブ受信を開始します。
	IIC0n_SlaveSendEndCallback	IIC0n スレーブ送信の転送終了割り込み INTIICn の発生に伴う処理を行います。
	IIC0n_SlaveReceiveEndCallback	IIC0n スレーブ受信の転送終了割り込み INTIICn の発生に伴う処理を行います。
	IIC0n_SlaveErrorCallback	IIC0n スレーブ通信エラーの検出に伴う処理を行います。
IIC0n_GetStopConditionCallback	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	AD_Init	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。
	AD_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。
	AD_Start	A/D 変換を開始します。
	AD_Stop	A/D 変換を終了します。
	AD_SelectADChannel	A/D 変換するアナログ電圧の入力端子を設定します。
	AD_SetPFTCondition	パワー・フェイル比較モードで動作する際の情報（比較値、A/D 変換終了割り込み INTAD の発生要因）を設定します。
	AD_Read	A/D 変換結果（10 ビット）を読み出します。
	AD_ReadByte	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	DAn_Init	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。
	DAn_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。
	DAn_Start	D/A 変換を開始します。

周辺機能	API 関数名	機能概要
D/A コンバータ	DAn_Stop	D/A 変換を終了します。
	DAn_SetValue	ANOn 端子に出力するアナログ電圧値を設定します。
タイマ	TMPn_Init	16 ビット・タイマ/イベント・カウンタ P (TMP) 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。
	TMPn_UserInit	16 ビット・タイマ/イベント・カウンタ P (TMP) に関するユーザ独自の初期化処理を行います。
	TMPn_Start	16 ビット・タイマ/イベント・カウンタ P (TMP) のカウントを開始します。
	TMPn_Stop	16 ビット・タイマ/イベント・カウンタ P (TMP) のカウントを終了します。
	TMPn_ChangeTimerCondition	16 ビット・タイマ/イベント・カウンタ P (TMP) のカウント値を変更します。
	TMPn_GetPulseWidth	16 ビット・タイマ/イベント・カウンタ P (TMP) のパルス幅 (ハイ・レベル幅, ロウ・レベル幅) を読み出します。
	TMPn_GetFreeRunningValue	16 ビット・タイマ/イベント・カウンタ P (TMP) がキャプチャした値を読み出します。
	TMPn_ChangeDuty	PWM 信号のデューティ比を変更します。
	TMPn_SoftwareTriggerOn	タイマ出力のためのトリガ (ソフトウェア・トリガ) を発生させます。
	TMQ0_Init	16 ビット・タイマ/イベント・カウンタ Q (TMQ) 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。
	TMQ0_UserInit	16 ビット・タイマ/イベント・カウンタ Q (TMQ) に関するユーザ独自の初期化処理を行います。
	TMQ0_Start	16 ビット・タイマ/イベント・カウンタ Q (TMQ) のカウントを開始します。
	TMQ0_Stop	16 ビット・タイマ/イベント・カウンタ Q (TMQ) のカウントを終了します。
	TMQ0_ChangeTimerCondition	16 ビット・タイマ/イベント・カウンタ Q (TMQ) のカウント値を変更します。
	TMQ0_GetPulseWidth	16 ビット・タイマ/イベント・カウンタ Q (TMQ) のパルス幅 (ハイ・レベル幅, ロウ・レベル幅) を読み出します。
	TMQ0_GetFreeRunningValue	16 ビット・タイマ/イベント・カウンタ Q (TMQ) がキャプチャした値を読み出します。
	TMQ0_ChangeDuty	PWM 信号のデューティ比を変更します。
	TMQ0_SoftwareTriggerOn	タイマ出力のためのトリガ (ソフトウェア・トリガ) を発生させます。

周辺機能	API 関数名	機能概要
タイマ	TAA <sub>n</sub> _Init	16 ビット・タイマ/イベント・カウンタ AA (TAA) 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。
	TAA <sub>n</sub> _UserInit	16 ビット・タイマ/イベント・カウンタ AA (TAA) に関するユーザ独自の初期化処理を行います。
	TAA <sub>n</sub> _Start	16 ビット・タイマ/イベント・カウンタ AA (TAA) のカウントを開始します。
	TAA <sub>n</sub> _Stop	16 ビット・タイマ/イベント・カウンタ AA (TAA) のカウントを終了します。
	TAA <sub>n</sub> _ChangeTimerCondition	16 ビット・タイマ/イベント・カウンタ AA (TAA) のカウント値を変更します。
	TAA <sub>n</sub> _ControlOutputToggle	16 ビット・タイマ/イベント・カウンタ AA (TAA) のトグル制御を変更します。
	TAA <sub>n</sub> _GetPulseWidth	16 ビット・タイマ/イベント・カウンタ AA (TAA) のパルス幅 (ハイ・レベル幅, ロウ・レベル幅) を読み出します。
	TAA <sub>n</sub> _GetFreeRunningValue	16 ビット・タイマ/イベント・カウンタ AA (TAA) がキャプチャした値を読み出します。
	TAA <sub>n</sub> _ChangeDuty	PWM 信号のデューティ比を変更します。
	TAA <sub>n</sub> _SoftwareTriggerOn	タイマ出力のためのトリガ (ソフトウェア・トリガ) を発生させます。
	TAB <sub>n</sub> _Init	16 ビット・タイマ/イベント・カウンタ AB (TAB) 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。
	TAB <sub>n</sub> _UserInit	16 ビット・タイマ/イベント・カウンタ AB (TAB) に関するユーザ独自の初期化処理を行います。
	TAB <sub>n</sub> _Start	16 ビット・タイマ/イベント・カウンタ AB (TAB) のカウントを開始します。
	TAB <sub>n</sub> _Stop	16 ビット・タイマ/イベント・カウンタ AB (TAB) のカウントを終了します。
	TAB <sub>n</sub> _ChangeTimerCondition	16 ビット・タイマ/イベント・カウンタ AB (TAB) のカウント値を変更します。
	TAB <sub>n</sub> _ControlOutputToggle	16 ビット・タイマ/イベント・カウンタ AB (TAB) のトグル制御を変更します。
	TAB <sub>n</sub> _GetPulseWidth	16 ビット・タイマ/イベント・カウンタ AB (TAB) のパルス幅 (ハイ・レベル幅, ロウ・レベル幅) を読み出します。
	TAB <sub>n</sub> _GetFreeRunningValue	16 ビット・タイマ/イベント・カウンタ AB (TAB) がキャプチャした値を読み出します。
	TAB <sub>n</sub> _ChangeDuty	PWM 信号のデューティ比を変更します。

周辺機能	API 関数名	機能概要
タイマ	TABn_SoftwareTriggerOn	タイマ出力のためのトリガ（ソフトウェア・トリガ）を発生させます。
	TMT0_Init	16ビット・タイマ/イベント・カウンタ T（TMT）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。
	TMT0_UserInit	16ビット・タイマ/イベント・カウンタ T（TMT）に関するユーザ独自の初期化処理を行います。
	TMT0_Start	16ビット・タイマ/イベント・カウンタ T（TMT）のカウントを開始します。
	TMT0_Stop	16ビット・タイマ/イベント・カウンタ T（TMT）のカウントを終了します。
	TMT0_ChangeTimerCondition	16ビット・タイマ/イベント・カウンタ T（TMT）のカウント値を変更します。
	TMT0_GetPulseWidth	16ビット・タイマ・イベント・カウンタ T（TMT）のパルス幅（ハイ・レベル幅、ロウ・レベル幅）を読み出します。
	TMT0_GetFreeRunningValue	16ビット・タイマ/イベント・カウンタ T（TMT）がキャプチャした値を読み出します。
	TMT0_ChangeDuty	PWM 信号のデューティ比を変更します。
	TMT0_SoftwareTriggerOn	タイマ出力のためのトリガ（ソフトウェア・トリガ）を発生させます。
	TMT0_EnableHold	16ビット・タイマ/イベント・カウンタ T（TMT）のエンコーダ・カウンタ制御を保持動作へと変更します。
	TMT0_DisableHold	16ビット・タイマ/イベント・カウンタ T（TMT）のエンコーダ・カウンタ制御を通常動作へと変更します。
	TMT0_ChangeCountValue	16ビット・タイマ/イベント・カウンタ T（TMT）の初期カウント値を変更します。
	TMMn_Init	16ビット・インターバル・タイマ M（TMM）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。
	TMMn_UserInit	16ビット・インターバル・タイマ M（TMM）に関するユーザ独自の初期化処理を行います。
	TMMn_Start	16ビット・インターバル・タイマ M（TMM）のカウントを開始します。
	TMMn_Stop	16ビット・インターバル・タイマ M（TMM）のカウントを終了します。
TMMn_ChangeTimerCondition	16ビット・インターバル・タイマ M（TMM）のカウント値を変更します。	
時計タイマ	WT_Init	時計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。

周辺機能	API 関数名	機能概要
時計タイマ	WT_UserInit	時計タイマに関するユーザ独自の初期化処理を行います。
	WT_Start	時計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。
	WT_Stop	時計タイマのカウント処理を中断します。
リアルタイム・カウンタ	RTC_Init	リアルタイム・カウン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。
	RTC_UserInit	リアルタイム・カウンタに関するユーザ独自の初期化処理を行います。
	RTC_CounterEnable	リアルタイム・カウンタ（年、月、曜日、日、時、分、秒）のカウントを開始します。
	RTC_CounterDisable	リアルタイム・カウンタ（年、月、曜日、日、時、分、秒）のカウントを終了します。
	RTC_SetHourSystem	リアルタイム・カウンタの時間制（12時間制、24時間制）を設定します。
	RTC_CounterSet	リアルタイム・カウンタにカウント値（年、月、曜日、日、時、分、秒）を設定します。
	RTC_CounterGet	リアルタイム・カウンタのカウント値（年、月、曜日、日、時、分、秒）を読み出します。
	RTC_ConstPeriodInterruptEnable	割り込み INTRTC0 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。
	RTC_ConstPeriodInterruptDisable	定周期割り込み機能を終了します。
	RTC_AlarmEnable	アラーム割り込み機能を開始します。
	RTC_AlarmDisable	アラーム割り込み機能を終了します。
	RTC_AlarmSet	アラームの発生条件（曜日、時、分）を設定します。
	RTC_AlarmGet	アラームの発生条件（曜日、時、分）を読み出します。
	RTC_IntervalStart	インターバル割り込み機能を開始します。
	RTC_IntervalStop	インターバル割り込み機能を終了します。
	RTC_IntervallInterruptEnable	割り込み INTRTC2 の発生周期を設定したのち、インターバル割り込み機能を開始します。
	RTC_IntervallInterruptDisable	インターバル割り込み機能を終了します。
	RTC_RC1CK1HZ_OutputEnable	RC1CK1HZ 端子に対するリアルタイム・カウンタ補正クロック（1 Hz）の出力を許可します。
	RTC_RC1CK1HZ_OutputDisable	RC1CK1HZ 端子に対するリアルタイム・カウンタ補正クロック（1 Hz）の出力を禁止します。
	RTC_RC1CKO_OutputEnable	RC1CKO 端子に対するリアルタイム・カウンタ・クロック（32 kHz 原発）の出力を許可します。
RTC_RC1CKO_OutputDisable	RC1CKO 端子に対するリアルタイム・カウンタ・クロック（32 kHz 原発）の出力を禁止します。	

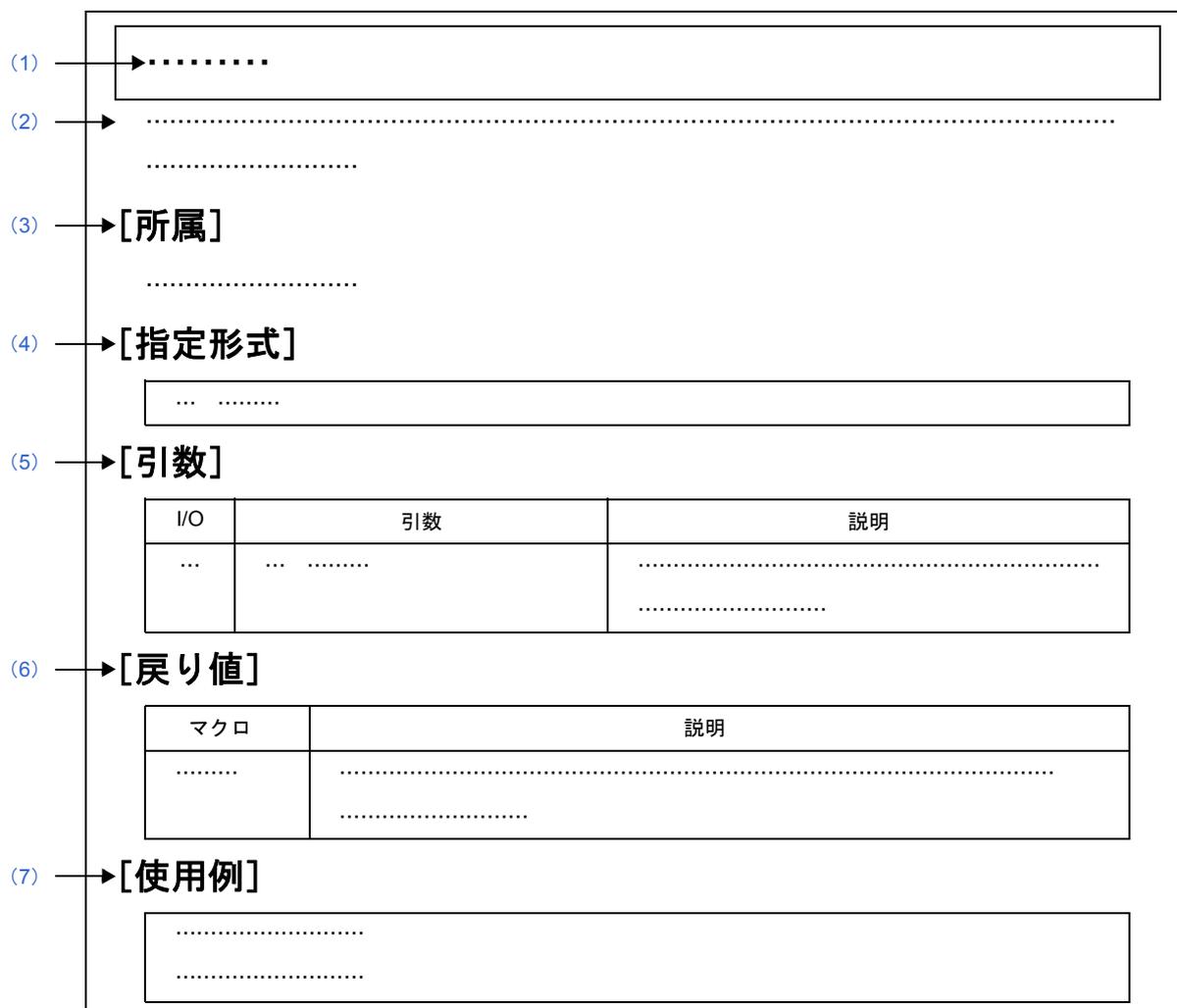
周辺機能	API 関数名	機能概要
リアルタイム・カウンタ	RTC_RC1CKDIV_OutputEnable	RC1CKDIV 端子に対するリアルタイム・カウンタ・クロック (32 kHz 分周) の出力を許可します。
	RTC_RC1CKDIV_OutputDisable	RC1CKDIV 端子に対するリアルタイム・カウンタ・クロック (32 kHz 分周) の出力を禁止します。
	RTC_RTC1HZ_OutputEnable	RTC1HZ 端子に対するリアルタイム・カウンタ補正クロック (1 Hz) の出力を許可します。
	RTC_RTC1HZ_OutputDisable	RTC1HZ 端子に対するリアルタイム・カウンタ補正クロック (1 Hz) の出力を禁止します。
	RTC_RTCCL_OutputEnable	RTCCL 端子に対するリアルタイム・カウンタ・クロック (32 kHz 原発) の出力を許可します。
	RTC_RTCCL_OutputDisable	RTCCL 端子に対するリアルタイム・カウンタ・クロック (32 kHz 原発) の出力を禁止します。
	RTC_RTCDIV_OutputEnable	RTCDIV 端子に対するリアルタイム・カウンタ・クロック (32 kHz 分周) の出力を許可します。
	RTC_RTCDIV_OutputDisable	RTCDIV 端子に対するリアルタイム・カウンタ・クロック (32 kHz 分周) の出力を禁止します。
	RTC_ChangeCorrectionValue	時計誤差を補正するタイミング、および補正値を変更します。
リアルタイム出力機能	RTOOn_Init	リアルタイム出力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。
	RTOOn_UserInit	リアルタイム出力に関するユーザ独自の初期化処理を行います。
	RTOOn_Enable	リアルタイム出力を許可します。
	RTOOn_Disable	リアルタイム出力を禁止します。
	RTOOn_Set2BitsData	リアルタイム出力する 2 ビット・データを設定します。
	RTOOn_Set4BitsData	リアルタイム出力する 4 ビット・データを設定します。
	RTOOn_Set6BitsData	リアルタイム出力する 6 ビット・データを設定します。
	RTOOn_Set8BitsData	リアルタイム出力する 8 ビット・データを設定します。
	RTOOn_SetHigh2BitsData	リアルタイム出力する上位 2 ビット・データを設定します。
	RTOOn_SetLow2BitsData	リアルタイム出力する下位 2 ビット・データを設定します。
	RTOOn_SetHigh4BitsData	リアルタイム出力する上位 4 ビット・データを設定します。
	RTOOn_SetLow4BitsData	リアルタイム出力する下位 4 ビット・データを設定します。

周辺機能	API 関数名	機能概要
リアルタイム出力機能	RTOn_GetValue	リアルタイム出力しているデータを読み出します。
DMA コントローラ	DMAAn_Init	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。
	DMAAn_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。
	DMAAn_Enable	チャンネル $n$ を動作許可状態に設定します。
	DMAAn_Disable	チャンネル $n$ を動作停止状態に設定します。
	DMAAn_CheckStatus	転送状態（転送終了、転送中）を読み出します。
	DMAAn_SetData	転送先／転送元の RAM アドレス、およびデータの転送回数を設定します。
	DMAAn_SoftwareTriggerOn	DMA 転送の起動要因として、ソフトウェア・トリガを使用します。
低電圧検出回路	LVI_Init	低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。
	LVI_UserInit	低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。
	LVI_InterruptModeStart	低電圧検出動作を開始します（割り込み発生モード時）。
	LVI_ResetModeStart	低電圧検出動作を開始します（内部リセット・モード時）。
	LVI_Start	低電圧検出動作を開始します。
	LVI_Stop	低電圧検出動作を停止します。

### 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 システム

以下に、コード生成がシステム用として出力するAPI関数の一覧を示します。

表 C—2 システム用 API 関数

API 関数名	機能概要
CLOCK_Init	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。
CLOCK_UserInit	クロックに関するユーザ独自の初期化処理を行います。
CG_ReadResetSource	リセットの発生に伴う処理を行います。
CG_ChangeClockMode	CPU クロックを変更します。
CG_ChangeFrequency	CPU クロックの分周比を変更します。
CG_SelectPowerSaveMode	CPU のスタンバイ・モードを設定します。
CG_SelectStabTime	STOP モードが解除された際に必要となる X1 発振回路の発振安定時間を選択します。
CG_SelectPIIMode	PLL 機能の動作モードを選択します。
CG_SelectSSCGMode	SSCG (Spread Spectrum Clock Generator) の動作状態を選択します。
WDT2_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。
CRC_Start	データ・ブロックの誤り検出動作を開始します。
CRC_SetData	CRC インプット・レジスタ (CRCIN) にデータを設定します。
CRC_GetResult	CRC データ・レジスタ (CRCD) に格納されている演算結果を読み出します。

## CLOCK\_Init

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]

CG\_system.c

### [指定形式]

```
void    CLOCK_Init ( void );
```

### [引数]

なし

### [戻り値]

なし

## CLOCK\_UserInit

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、[CLOCK\\_Init](#) のコールバック・ルーチンとして呼び出されます。

### [所属]

CG\_system\_user.c

### [指定形式]

```
void    CLOCK_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

## CG\_ReadResetSource

リセットの発生に伴う処理を行います。

### [所属]

CG\_system\_user.c

### [指定形式]

```
void CG_ReadResetSource ( void );
```

### [引数]

なし

### [戻り値]

なし

### [使用例]

以下に、リセットの発生要因別に異なる処理を実行する際の例を示します。

#### 【CG\_systeminit.c】

```
void systeminit ( void ) {
    CG_ReadResetSource ();      /* リセットの発生要因別に処理を実行 */
    .....
}
```

#### 【CG\_system\_user.c】

```
#include "CG_macrodriver.h"
void CG_ReadResetSource ( void ) {
    UCHAR resetflag = RESF; /* リセット・コントロール・フラグ・レジスタ : RESF の内容確保 */
    if ( resetflag & 0x1 ) { /* 発生要因の判別 : LVIRF フラグのチェック */
        ..... /* 低電圧検出回路が低電圧を検出した際の処理 */
    } else if ( resetflag & 0x2 ) { /* 発生要因の判別 : CLMRF フラグのチェック */
        ..... /* クロック・モニタの発振停止を検出した際の処理 */
    } else if ( resetflag & 0x10 ) { /* 発生要因の判別 : WDT2RF フラグのチェック */
        ..... /* ウォッチドッグ・タイマ 2 のオーバフローを検出した際の処理 */
    }
    .....
}
```

```
}  
}
```

## CG\_ChangeClockMode

CPUクロックを変更します。

### [所属]

CG\_system.c

### [指定形式]

```
#include "CG_macrodriver.h"
#include "CG_system.h"
MD_STATUS CG_ChangeClockMode ( enum ClockMode mode );
```

### [引数]

I/O	引数	説明
I	enum ClockMode mode;	CPUクロックの種類 MAINOSCCLK : メイン・クロック発振回路 (fxx) SUBCLK : サブクロック発振回路 (fXT)

### [戻り値]

マクロ	説明
MD_OK	正常終了
MD_ERROR1	異常終了 - サブクロック動作からメイン・クロック動作への変更ができませんでした
MD_ERROR2	異常終了 - メイン・クロック動作からサブクロック動作への変更ができませんでした
MD_ARGERROR	引数の指定が不正

## CG\_ChangeFrequency

CPUクロックの分周比を変更します。

### [所属]

CG\_system.c

### [指定形式]

```
#include "CG_macrodriver.h"
#include "CG_system.h"
MD_STATUS CG_ChangeFrequency ( enum CPUClock clock );
```

### [引数]

I/O	引数	説明
I	enum CPUClock <i>clock</i> ;	分周比の種類 SYSTEMCLOCK : fxx SYSONEHALF : fxx/2 SYSONEFOURTH : fxx/4 SYSONEIGHTH : fxx/8 SYSONESIXTEENTH : fxx/16 SYSONETHIRTYSECOND : fxx/32

備考 fxx は、メイン・クロック発振回路の周波数を意味します。

### [戻り値]

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

## CG\_SelectPowerSaveMode

CPUのスタンバイ・モードを選択します。

### [所属]

CG\_system.c

### [指定形式]

```
#include "CG_macrodriver.h"
#include "CG_system.h"
MD_STATUS CG_SelectPowerSaveMode ( enum PSLevel level );
```

### [引数]

I/O	引数	説明
I	enum PSLevel level;	スタンバイ・モードの種類 【E/Sx3-H】 【ES/Jx3-E】 【ES/Jx3-H】 PSSTOP : STOP モード PSHALT : HALT モード PSIDLE1 : IDLE1 モード PSIDLE2 : IDLE2 モード 【ES/Jx3】 【ES/Jx3-L】 PSSTOP : STOP モード PSHALT : HALT モード

### [戻り値]

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

## CG\_SelectStabTime

STOP モードが解除された際に必要となる X1 発振回路の発振安定時間を選択します。

### [所属]

CG\_system.c

### [指定形式]

```
#include "CG_macrodriver.h"
#include "CG_system.h"
MD_STATUS CG_SelectStabTime ( enum StabTime waittime );
```

### [引数]

I/O	引数	説明
I	enum StabTime waittime;	発振安定時間の種類 STLEVEL0 : 2 <sup>10</sup> /fx STLEVEL1 : 2 <sup>11</sup> /fx STLEVEL2 : 2 <sup>12</sup> /fx STLEVEL3 : 2 <sup>13</sup> /fx STLEVEL4 : 2 <sup>14</sup> /fx STLEVEL5 : 2 <sup>15</sup> /fx STLEVEL6 : 2 <sup>16</sup> /fx

### [戻り値]

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

## CG\_SelectPllMode

PLL 機能の動作モードを選択します。

### [所属]

CG\_system.c

### [指定形式]

```
#include "CG_macrodriver.h"
#include "CG_system.h"
MD_STATUS CG_SelectPllMode ( enum PllMode pllmode );
```

### [引数]

I/O	引数	説明
I	enum PllMode pllmode;	動作モードの種類 【E/Sx3-H】【ES/Jx3】【ES/Jx3-L】 SYSPLLOFF : クロック・スルー・モード SYS4PLL : 4 通倍 (PLL 機能使用時) SYS8PLL : 8 通倍 (PLL 機能使用時) 【ES/Jx3-E】【ES/Jx3-H】 SYSPLLOFF : PLL 停止 SYSPLLON : PLL 動作

### [戻り値]

マクロ	説明
MD_OK	正常終了
MD_ERROR	異常終了【ES/Jx3】【ES/Jx3-E】【ES/Jx3-H】【ES/Jx3-L】
MD_ERROR1	異常終了【E/Sx3-H】 - 動作モードの変更はできません。
MD_ERROR2	異常終了【E/Sx3-H】 - 4 通倍への変更はできません。
MD_ERROR3	異常終了【E/Sx3-H】 - 8 通倍への変更はできません。
MD_ARGERROR	引数の指定が不正

## CG\_SelectSSCGMode

SSCG (Spread Spectrum Clock Generator) の動作状態を選択します。

### [所属]

CG\_system.c

### [指定形式]

```
#include "CG_macrodriver.h"
#include "CG_system.h"
MD_STATUS CG_SelectSSCGMode ( enum SSCGMode sscgmode );
```

### [引数]

I/O	引数	説明
I	enum SSCGMode <i>sscgmode</i> ;	動作状態の種類 SYSSSCGON : 動作許可状態 SYSSSCGOFF : 動作禁止状態

### [戻り値]

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

## WDT2\_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]

CG\_system.c

### [指定形式]

```
void WDT2_Restart ( void );
```

### [引数]

なし

### [戻り値]

なし

## CRC\_Start

データ・ブロックの誤り検出動作を開始します。

### [所属]

CG\_system.c

### [指定形式]

```
void CRC_Start ( void );
```

### [引数]

なし

### [戻り値]

なし

## CRC\_SetData

CRC インプット・レジスタ (CRCIN) にデータを設定します。

### [所属]

CG\_system.c

### [指定形式]

```
#include "CG_macrodriver.h"
void CRC_SetData ( UCHAR data );
```

### [引数]

I/O	引数	説明
I	UCHAR data;	設定するデータ

### [戻り値]

なし

## CRC\_GetResult

CRC データ・レジスタ (CRCD) に格納されている演算結果を読み出します。

### [所属]

CG\_system.c

### [指定形式]

```
#include "CG_macrodriver.h"
void CRC_GetResult ( USHORT *result );
```

### [引数]

I/O	引数	説明
O	USHORT *result;	読み出した演算結果を格納する領域へのポインタ

### [戻り値]

なし

### C.3.2 外部バス

以下に、コード生成が外部バス用として出力するAPI関数の一覧を示します。

表 C—3 外部バス用 API 関数

API 関数名	機能概要
BUS_Init	外部バス・インタフェースの機能（外部バスを内蔵 ROM, RAM, SFR 以外の領域に接続する機能）を制御するうえで必要となる初期化処理を行います。
BUS_UserInit	外部バス・インタフェースに関するユーザ独自の初期化処理を行います。

## BUS\_Init

外部バス・インタフェースの機能（外部バスを内蔵 ROM, RAM, SFR 以外の領域に接続する機能）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]

CG\_bus.c

### [指定形式]

```
void    BUS_Init ( void );
```

### [引数]

なし

### [戻り値]

なし

## BUS\_UserInit

外部バス・インタフェースに関するユーザ独自の初期化処理を行います。

**備考** 本API関数は、[BUS\\_Init](#)のコールバック・ルーチンとして呼び出されます。

### [所属]

CG\_bus\_user.c

### [指定形式]

```
void    BUS_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

### C.3.3 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—4 ポート用 API 関数

API 関数名	機能概要
PORT_Init	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。
PORT_UserInit	ポートに関するユーザ独自の初期化処理を行います。
PORT_ChangePmnInput	端子の入出力モードを出力モードから入力モードへと切り替えます。
PORT_ChangePmnOutput	端子の入出力モードを入力モードから出力モードへと切り替えます。

## PORT\_Init

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]

CG\_port.c

### [指定形式]

```
void PORT_Init ( void );
```

### [引数]

なし

### [戻り値]

なし

## PORT\_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、[PORT\\_Init](#) のコールバック・ルーチンとして呼び出されます。

### [所属]

CG\_port\_user.c

### [指定形式]

```
void PORT_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

## PORT\_ChangePmnInput

端子の入出力モードを出力モードから入力モードへと切り替えます。

### [所属]

CG\_port.c

### [指定形式]

```
void PORT_ChangePmnInput ( void );
```

**備考** *mn* は、ポート番号を意味します。

### [引数]

なし

### [戻り値]

なし

### [使用例]

以下に、P00 端子の入出力モードを出力モードから入力モードへと切り替える際の例を示します。

【CG\_main.c】

```
void main ( void ) {  
    .....  
    PORT_ChangeP00Input ( );          /* 入出力モードの切り替え */  
    .....  
}
```

## PORT\_ChangePmnOutput

端子の入出力モードを入力モードから出力モードへと切り替えます。

### [所属]

CG\_port.c

### [指定形式]

本 API 関数の指定形式は、対象端子で N-ch オープン・ドレイン出力が行われるか否かにより異なります。

- 【N-ch オープン・ドレイン出力：なし】

```
#include "CG_macrodriver.h"
void PORT_ChangePmnOutput ( BOOL initialvalue );
```

- 【N-ch オープン・ドレイン出力：あり】

```
#include "CG_macrodriver.h"
void PORT_ChangePmnOutput ( BOOL enablench, BOOL initialvalue );
```

**備考** *nm* は、ポート番号を意味します。

### [引数]

I/O	引数	説明
I	BOOL <i>enablench</i> ;	出力モードの種類 MD_TRUE : N-ch オープン・ドレイン出力 (V <sub>DD</sub> 耐圧) モード MD_FALSE : 通常出力モード
I	BOOL <i>initialvalue</i> ;	初期出力値 MD_SET : High レベル “1” を出力 MD_CLEAR : Low レベル “0” を出力

### [戻り値]

なし

### [使用例 1]

以下に、P00 端子 (N-ch オープン・ドレイン出力：なし) を

入出力モードの種類： 出力モード

初期出力値： High レベル “1” を出力

に変更する際の例を示します。

**【CG\_main.c】**

```
#include "CG_macrodriver.h"
void main ( void ) {
    .....
    PORT_ChangeP00Output ( MD_SET ); /* 入出力モードの切り替え */
    .....
}
```

**[使用例 2]**

以下に、P04 端子 (N-ch オープン・ドレイン出力：あり) を

入出力モードの種類： 出力モード

出力モードの種類： N-ch オープン・ドレイン出力 (VDD 耐圧) モード

初期出力値： Low レベル “0” を出力

に変更する際の例を示します。

**【CG\_main.c】**

```
#include "CG_macrodriver.h"
void main ( void ) {
    .....
    PORT_ChangeP04Output ( MD_TRUE, MD_CLEAR ); /* 入出力モードの切り替え */
    .....
}
```

### C.3.4 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—5 割り込み用 API 関数

API 関数名	機能概要
INTP_Init	外部割り込み 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。
INTP_UserInit	外部割り込み INTP $n$ に関するユーザ独自の初期化処理を行います。
KEY_Init	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。
KEY_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。
INT_MaskableInterruptEnable	マスクブル割り込みの受け付けを禁止／許可します。
INTPn_Disable	マスクブル割り込み（外部割り込み要求）INTP $n$ の受け付けを禁止します。
INTPn_Enable	マスクブル割り込み（外部割り込み要求）INTP $n$ の受け付けを許可します。
KEY_Disable	キー割り込み INTKRの受け付けを禁止します。
KEY_Enable	キー割り込み INTKRの受け付けを許可します。

## INTP\_Init

外部割り込み 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]

CG\_int.c

### [指定形式]

```
void    INTP_Init ( void );
```

### [引数]

なし

### [戻り値]

なし

## INTP\_UserInit

外部割り込み 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、INTP\_Init のコールバック・ルーチンとして呼び出されます。

### [所属]

CG\_int\_user.c

### [指定形式]

```
void INTP_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

## KEY\_Init

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]

CG\_int.c

### [指定形式]

```
void KEY_Init ( void );
```

### [引数]

なし

### [戻り値]

なし

## KEY\_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、KEY\_Init のコールバック・ルーチンとして呼び出されます。

### [所属]

CG\_int\_user.c

### [指定形式]

```
void KEY_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

## INT\_MaskableInterruptEnable

マスカブル割り込みの受け付けを禁止／許可します。

### [所属]

CG\_int.c

### [指定形式]

- 【E/Sx3-H】 【ES/Jx3-E】 【ES/Jx3-H】

```
#include "CG_macrodriver.h"
#include "CG_int.h"
MD_STATUS INT_MaskableInterruptEnable ( enum MaskableSource name, BOOL enableflag );
```

- 【ES/Jx3】 【ES/Jx3-L】

```
#include "CG_macrodriver.h"
#include "CG_int.h"
void INT_MaskableInterruptEnable ( enum MaskableSource name, BOOL enableflag );
```

### [引数]

I/O	引数	説明
I	enum MaskableSource name;	マスカブル割り込みの種類 INT_xxx: マスカブル割り込み
I	BOOL enableflag;	受け付けの禁止／許可 MD_TRUE: 受け付けを許可 MD_FALSE: 受け付けを禁止

**備考** マスカブル割り込みの種類 INT\_xxx についての詳細は、ヘッダ・ファイル CG\_int.h を参照してください。

### [戻り値]

- 【E/Sx3-H】 【ES/Jx3-E】 【ES/Jx3-H】

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

- 【ES/Jx3】 【ES/Jx3-L】

なし

## [使用例 1]

以下に、マスカブル割り込み INTP0 の受け付けを“禁止”に設定する際の例を示します。

【CG\_main.c】

```
#include "CG_macrodriver.h"
#include "CG_int.h"
void main ( void ) {
    .....
    INT_MaskableInterruptEnable ( INT_INTP0, MD_FALSE ); /* マスカブル割り込み INTP0 の受け付け禁止
*/
    .....
}
```

## [使用例 2]

以下に、マスカブル割り込み INTP0 の受け付けを“許可”に設定する際の例を示します。

【CG\_main.c】

```
#include "CG_macrodriver.h"
#include "CG_int.h"
void main ( void ) {
    .....
    INT_MaskableInterruptEnable ( INT_INTP0, MD_TRUE ); /* マスカブル割り込み INTP0 の受け付け許可
*/
    .....
}
```

## INTP $n$ \_Disable

マスクブル割り込み（外部割り込み要求）INTP $n$ の受け付けを禁止します。

### [所属]

CG\_int.c

### [指定形式]

```
void    INTP $n$ _Disable ( 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

## INTP $n$ \_Enable

マスクブル割り込み（外部割り込み要求）INTP $n$ の受け付けを許可します。

### [所属]

CG\_int.c

### [指定形式]

```
void ITPn_Enable ( 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

## KEY\_Disable

キー割り込み INTKR の受け付けを禁止します。

### [所属]

CG\_int.c

### [指定形式]

```
void KEY_Disable ( void );
```

### [引数]

なし

### [戻り値]

なし

## KEY\_Enable

キー割り込み INTKR の受け付けを許可します。

### [所属]

CG\_int.c

### [指定形式]

```
void KEY_Enable ( void );
```

### [引数]

なし

### [戻り値]

なし

### C.3.5 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—6 シリアル用 API 関数

API 関数名	機能概要
UARTAn_Init	アシンクロナス・シリアル・インタフェース A (UARTA) 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。
UARTAn_UserInit	アシンクロナス・シリアル・インタフェース A (UARTA) に関するユーザ独自の初期化処理を行います。
UARTAn_Start	アシンクロナス・シリアル・インタフェース A (UARTA) を動作許可状態へと移行します。
UARTAn_Stop	アシンクロナス・シリアル・インタフェース A (UARTA) を動作禁止状態へと移行します。
UARTAn_SendData	データの UARTAn 送信を開始します。
UARTAn_ReceiveData	データの UARTAn 受信を開始します。
UARTAn_SendEndCallback	UARTAn の連続送信許可割り込み INTUANt の発生に伴う処理を行います。
UARTAn_ReceiveEndCallback	UARTAn の受信終了割り込み INTUANr の発生に伴う処理を行います。
UARTAn_ErrorCallback	UARTAn 受信エラー割り込み INTUANR (オーバラン・エラー, フレーミング・エラー, パリティ・エラー) の発生に伴う処理を行います。
UARTAn_SoftOverRunCallback	オーバラン・エラーの発生に伴う処理を行います。
UARTBn_Init	アシンクロナス・シリアル・インタフェース B (UARTB) 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。
UARTBn_UserInit	アシンクロナス・シリアル・インタフェース B (UARTB) に関するユーザ独自の初期化処理を行います。
UARTBn_Start	アシンクロナス・シリアル・インタフェース B (UARTB) を動作許可状態へと移行します。
UARTBn_Stop	アシンクロナス・シリアル・インタフェース B (UARTB) を動作禁止状態へと移行します。
UARTBn_SendData	データの UARTBn 送信を開始します。
UARTBn_ReceiveData	データの UARTBn 受信を開始します。
UARTBn_SendEndCallback	UARTBn の送信許可割り込み INTUBnTIT, および FIFO 送信完了割り込み INTUBnTIF の発生に伴う処理を行います。
UARTBn_ReceiveEndCallback	UARTBn の受信完了割り込み INTUBnTIR の発生に伴う処理を行います。
UARTBn_SingleErrorCallback	受信エラー割り込み INTUBnTIRE (オーバラン・エラー, フレーミング・エラー, パリティ・エラー) の発生に伴う処理を行います。
UARTBn_FIFOErrorCallback	受信エラー割り込み INTUBnTIRE (オーバラン・エラー, フレーミング・エラー, パリティ・エラー) の発生に伴う処理を行います。
UARTBn_TimeoutErrorCallback	受信タイムアウト割り込み INTUBnTITO の発生に伴う処理を行います。
UARTBn_SoftOverRunCallback	オーバラン・エラーの発生に伴う処理を行います。

API 関数名	機能概要
UARTCn_Init	アシンクロナス・シリアル・インタフェース C (UARTC) 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。
UARTCn_UserInit	アシンクロナス・シリアル・インタフェース C (UARTC) に関するユーザ独自の初期化処理を行います。
UARTCn_Start	アシンクロナス・シリアル・インタフェース C (UARTC) を動作許可状態へと移行します。
UARTCn_Stop	アシンクロナス・シリアル・インタフェース C (UARTC) を動作禁止状態へと移行します。
UARTCn_SendData	データの UARTCn 送信を開始します。
UARTCn_ReceiveData	データの UARTCn 受信を開始します。
UARTCn_SendEndCallback	UARTCn の連続送信許可割り込み INTUCnT の発生に伴う処理を行います。
UARTCn_ReceiveEndCallback	UARTCn の受信終了割り込み INTUCnR の発生に伴う処理を行います。
UARTCn_ErrorCallback	UARTCn 受信エラー割り込み INTUCnR (オーバラン・エラー, フレーミング・エラー, パリティ・エラー) の発生に伴う処理を行います。
UARTCn_SoftOverRunCallback	オーバラン・エラーの発生に伴う処理を行います。
CSIBn_Init	3 線式可変長シリアル I/O B (CSIB) 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。
CSIBn_UserInit	3 線式可変長シリアル I/O B (CSIB) に関するユーザ独自の初期化処理を行います。
CSIBn_Start	3 線式可変長シリアル I/O B (CSIB) を動作許可状態へと移行します。
CSIBn_Stop	3 線式可変長シリアル I/O B (CSIB) を動作禁止状態へと移行します。
CSIBn_SendData	データの CSIB 送信を開始します。
CSIBn_ReceiveData	データの CSIB 受信を開始します。
CSIBn_SendReceiveData	データの CSIB 送受信を開始します。
CSIBn_SendEndCallback	CSIBn の受信終了割り込み INTCBnR, および CSIBn の連続送信書き込み許可割り込み INTCBnT の発生に伴う処理を行います。
CSIBn_ReceiveEndCallback	CSIBn の受信終了割り込み INTCBnR の発生に伴う処理を行います。
CSIBn_ErrorCallback	CSIBn の受信エラー割り込み INTCBnR (オーバラン・エラー) の発生に伴う処理を行います。
CSIEn_Init	3 線式可変長シリアル I/O E (CSIE) 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。
CSIEn_UserInit	3 線式可変長シリアル I/O E (CSIE) に関するユーザ独自の初期化処理を行います。
CSIEn_Start	3 線式可変長シリアル I/O E (CSIE) を動作許可状態へと移行します。
CSIEn_Stop	3 線式可変長シリアル I/O E (CSIE) を動作禁止状態へと移行します。
CSIEn_SendData	データの CSIE 送信を開始します。
CSIEn_ReceiveData	データの CSIE 受信を開始します。
CSIEn_SendReceiveData	データの CSIE 送受信を開始します。
CSIEn_SendEndCallback	CSIEn の送受信完了割り込み INTCEnT の発生に伴う処理を行います。

API 関数名	機能概要
CSIE $n$ _ReceiveEndCallback	CSIE $n$ の送受信完了割り込み INTCE $n$ T の発生に伴う処理を行います。
CSIE $n$ _ErrorCallback	CSIE $n$ の CSIE $n$ BUF オーバフロー割り込み INTCE $n$ TIOF の発生に伴う処理を行います。
CSIF $n$ _Init	3 線式可変長シリアル I/O F (CSIF) 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。
CSIF $n$ _UserInit	3 線式可変長シリアル I/O F (CSIF) に関するユーザ独自の初期化処理を行います。
CSIF $n$ _Start	3 線式可変長シリアル I/O F (CSIF) を動作許可状態へと移行します。
CSIF $n$ _Stop	3 線式可変長シリアル I/O F (CSIF) を動作禁止状態へと移行します。
CSIF $n$ _SendData	データの CSIF 送信を開始します。
CSIF $n$ _ReceiveData	データの CSIF 受信を開始します。
CSIF $n$ _SendReceiveData	データの CSIF 送受信を開始します。
CSIF $n$ _SendEndCallback	CSIF $n$ の送受信完了割り込み INTCF $n$ T の発生に伴う処理を行います。
CSIF $n$ _ReceiveEndCallback	CSIF $n$ の送受信完了割り込み INTCF $n$ T の発生に伴う処理を行います。
CSIF $n$ _ErrorCallback	CSIF $n$ の受信エラー割り込み INTCF $n$ R (オーバラン・エラー) の発生に伴う処理を行います。
IIC0 $n$ _Init	IIC バ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。
IIC0 $n$ _UserInit	IIC バスに関するユーザ独自の初期化処理を行います。
IIC0 $n$ _Stop	IIC0 $n$ 通信を終了します。
IIC0 $n$ _StopCondition	ストップ・コンディションを生成します。
IIC0 $n$ _MasterSendStart	データの IIC0 $n$ マスタ送信を開始します。
IIC0 $n$ _MasterReceiveStart	データの IIC0 $n$ マスタ受信を開始します。
IIC0 $n$ _MasterSendEndCallback	IIC0 $n$ マスタ送信の転送終了割り込み INTIIC $n$ の発生に伴う処理を行います。
IIC0 $n$ _MasterReceiveEndCallback	IIC0 $n$ マスタ受信の転送終了割り込み INTIIC $n$ の発生に伴う処理を行います。
IIC0 $n$ _MasterErrorCallback	IIC0 $n$ マスタ通信エラーの検出に伴う処理を行います。
IIC0 $n$ _SlaveSendStart	データの IIC0 $n$ スレーブ送信を開始します。
IIC0 $n$ _SlaveReceiveStart	データの IIC0 $n$ スレーブ受信を開始します。
IIC0 $n$ _SlaveSendEndCallback	IIC0 $n$ スレーブ送信の転送終了割り込み INTIIC $n$ の発生に伴う処理を行います。
IIC0 $n$ _SlaveReceiveEndCallback	IIC0 $n$ スレーブ受信の転送終了割り込み INTIIC $n$ の発生に伴う処理を行います。
IIC0 $n$ _SlaveErrorCallback	IIC0 $n$ スレーブ通信エラーの検出に伴う処理を行います。
IIC0 $n$ _GetStopConditionCallback	ストップ・コンディションの検出に伴う処理を行います。

## UARTAn\_Init

アシンクロナス・シリアル・インタフェース A (UARTA) 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]

CG\_serial.c

### [指定形式]

```
void UARTAn_Init ( void );
```

**備考**  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## UARTAn\_UserInit

アシンクロナス・シリアル・インタフェース A (UARTA) 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、[UARTAn\\_Init](#) のコールバック・ルーチンとして呼び出されます。

### [所属]

CG\_serial\_user.c

### [指定形式]

```
void    UARTAn_UserInit ( void );
```

**備考** *n* は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## UARTAn\_Start

アシンクロナス・シリアル・インタフェース A (UARTA) を動作許可状態へと移行します。

### [所属]

CG\_serial.c

### [指定形式]

```
void    UARTAn_Start ( void );
```

**備考** *n*は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## UARTAn\_Stop

アシンクロナス・シリアル・インタフェース A (UARTA) を動作禁止状態へと移行します。

### [所属]

CG\_serial.c

### [指定形式]

```
void    UARTAn_Stop ( void );
```

**備考** *n*は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## UARTAn\_SendData

データの UARTAn 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 UARTAn 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.** UARTAn 送信を行う際には、本 API 関数の呼び出し以前に [UARTAn\\_Start](#) を呼び出す必要があります。

### [所属]

CG\_serial.c

### [指定形式]

```
#include "CG_macrodriver.h"
MD_STATUS UARTAn_SendData ( UCHAR *txbuf, USHORT txnum );
```

**備考** *n* は、チャンネル番号を意味します。

### [引数]

I/O	引数	説明
I	UCHAR * <i>txbuf</i> ;	送信するデータを格納したバッファへのポインタ
I	USHORT <i>txnum</i> ;	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
MD_DATAEXISTS	異常終了 - UAnTX レジスタに“次に転送すべきデータ”が存在する (データの送信中)

## UARTAn\_ReceiveData

データの UARTAn 受信を開始します。

- 備考 1.** 本 API 関数では、1 バイト単位の UARTAn 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 UARTAn 受信は、本 API 関数の呼び出し後、[UARTAn\\_Start](#) を呼び出すことにより開始されます。

### [所属]

CG\_serial.c

### [指定形式]

```
#include "CG_macrodriver.h"
MD_STATUS UARTAn_ReceiveData ( UCHAR *rxbuf, USHORT rxnum );
```

**備考** *n* は、チャンネル番号を意味します。

### [引数]

I/O	引数	説明
O	UCHAR * <i>rxbuf</i> ;	受信したデータを格納するバッファへのポインタ
I	USHORT <i>rxnum</i> ;	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

## UARTAn\_SendEndCallback

UARTAnの連続送信許可割り込み INTUANnT の発生に伴う処理を行います。

**備考** 本 API 関数は、UARTAnの連続送信許可割り込み INTUANnT に対応した割り込み処理 MD\_INTUANnT のコールバック・ルーチン (UARTAn\_SendData の引数 *txnum* で指定された総数の UARTAn 送信が完了した際の処理) として呼び出されます。

### [所属]

CG\_serial\_user.c

### [指定形式]

```
void    UARTAn_SendEndCallback ( void );
```

**備考** *n* は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## UARTAn\_ReceiveEndCallback

UARTAnの受信終了割り込み INTUANR の発生に伴う処理を行います。

**備考** 本 API 関数は、UARTAnの受信終了割り込み INTUANR に対応した割り込み処理 MD\_INTUANR のコールバック・ルーチン (UARTAn\_ReceiveData の引数 *rxnum* で指定された総数の UARTAn 受信が完了した際の処理) として呼び出されます。

### [所属]

CG\_serial\_user.c

### [指定形式]

```
void    UARTAn_ReceiveEndCallback ( void );
```

**備考** *n* は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## UARTAn\_ErrorCallback

UARTAn 受信エラー割り込み INTUANR (オーバラン・エラー, フレーミング・エラー, パリティ・エラー) の発生に伴う処理を行います。

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

### [所属]

CG\_serial\_user.c

### [指定形式]

```
#include "CG_macrodriver.h"
void UARTAn_ErrorCallback ( UCHAR 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	UCHAR <i>err_type</i> ;	エラー割り込みの発生要因 00000xx1B : オーバラン・エラー 00000x1xB : フレーミング・エラー 000001xxB : パリティ・エラー

### [戻り値]

なし

## UARTn\_SoftOverRunCallback

オーバラン・エラーの発生に伴う処理を行います。

**備考** 本API関数は、UARTnの受信エラー割り込みINTUAnRに対応した割り込み処理MD\_INTUAnRのコールバック・ルーチン（UARTn\_ReceiveData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]

CG\_serial\_user.c

### [指定形式]

```
#include "CG_macrodriver.h"
void UARTn_SoftOverRunCallback ( UCHAR rx_data );
```

**備考** nは、チャンネル番号を意味します。

### [引数]

I/O	引数	説明
O	UCHAR rx_data;	受信したデータ

### [戻り値]

なし

## UARTB $n$ \_Init

アシンクロナス・シリアル・インタフェース A (UARTB) 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]

CG\_serial.c

### [指定形式]

```
void UARTB $n$ _Init ( void );
```

**備考**  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## UARTB $n$ \_UserInit

アシンクロナス・シリアル・インタフェース B (UARTB) 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、UARTB $n$ \_Init のコールバック・ルーチンとして呼び出されます。

### [所属]

CG\_serial\_user.c

### [指定形式]

```
void UARTB $n$ _UserInit ( void );
```

**備考**  $n$  は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## UARTB $n$ \_Start

アシンクロナス・シリアル・インタフェース B (UARTB) を動作許可状態へと移行します。

### [所属]

CG\_serial.c

### [指定形式]

```
void    UARTBn_Start ( void );
```

**備考**  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## UARTB $n$ \_Stop

アシンクロナス・シリアル・インタフェース B (UARTB) を動作禁止状態へと移行します。

### [所属]

CG\_serial.c

### [指定形式]

```
void    UARTBn_Stop ( void );
```

**備考**  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## UARTBn\_SendData

データの UARTBn 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 UARTBn 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.** UARTBn 送信（シングル・モード）を行う際には、本 API 関数の呼び出し以前に [UARTBn\\_Start](#) を呼び出す必要があります。
- 3.** UARTBn 送信（FIFO モード）を行う際には、本 API 関数の呼び出し後に [UARTBn\\_Start](#) を呼び出す必要があります。

### [所属]

CG\_serial.c

### [指定形式]

```
#include "CG_macrodriver.h"
MD_STATUS UARTBn_SendData ( UCHAR *txbuf, USHORT txnum );
```

**備考** *n* は、チャンネル番号を意味します。

### [引数]

I/O	引数	説明
I	UCHAR * <i>txbuf</i> ;	送信するデータを格納したバッファへのポインタ
I	USHORT <i>txnum</i> ;	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 - <i>txnum</i> が送信 FIFO トリガ数の倍数でない
MD_DATAEXISTS	異常終了 - UBnTX レジスタに“次に転送すべきデータ”が存在する（データの送信中）

## UARTBn\_ReceiveData

データの UARTBn 受信を開始します。

- 備考 1.** 本 API 関数では、1 バイト単位の UARTBn 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 UARTBn 受信は、本 API 関数の呼び出し後、[UARTBn\\_Start](#) を呼び出すことにより開始されます。

### [所属]

CG\_serial.c

### [指定形式]

```
#include "CG_macrodriver.h"
MD_STATUS UARTBn_ReceiveData ( UCHAR *rxbuf, USHORT rxnum );
```

**備考** *n* は、チャンネル番号を意味します。

### [引数]

I/O	引数	説明
O	UCHAR * <i>rxbuf</i> ;	受信したデータを格納するバッファへのポインタ
I	USHORT <i>rxnum</i> ;	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 - <i>rxnum</i> が受信 FIFO トリガ数の倍数でない

## UARTB $n$ \_SendEndCallback

UARTB $n$ の送信許可割り込み INTUB $n$ TIT, および FIFO 送信完了割り込み INTUB $n$ TIF の発生に伴う処理を行います。

**備考** 本 API 関数は, UARTB $n$  (シングル・モード) の送信許可割り込み INTUB $n$ TIT に対応した割り込み処理 MD\_INTUB $n$ TIT, および UARTB $n$  (FIFO モード) の FIFO 送信完了割り込み INTUB $n$ TIF に対応した割り込み処理 MD\_INTUB $n$ TIF のコールバック・ルーチン (UARTB $n$ \_SendData の引数 *txnum* で指定された総数の UARTB $n$  送信が完了した際の処理) として呼び出されます。

### [所属]

CG\_serial\_user.c

### [指定形式]

```
void UARTB $n$ _SendEndCallback ( 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

## UARTBn\_ReceiveEndCallback

UARTBnの受信完了割り込み INTUBnTIR の発生に伴う処理を行います。

**備考** 本 API 関数は、UARTBnの受信完了割り込み INTUBnTIR に対応した割り込み処理 MD\_INTUBnTIR のコールバック・ルーチン (UARTBn\_ReceiveData の引数 rxnum で指定された総数の UARTBn 受信が完了した際の処理) として呼び出されます。

### [所属]

CG\_serial\_user.c

### [指定形式]

```
void UARTBn_ReceiveEndCallback ( void );
```

**備考** nは、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## UARTBn\_SingleErrorCallback

受信エラー割り込み INTUBnTIRE（オーバラン・エラー，フレーミング・エラー，パリティ・エラー）の発生に伴う処理を行います。

**備考** 本 API 関数は，UARTBn（シングル・モード）の受信エラー割り込み INTUBnTIRE に対応した割り込み処理 MD\_INTUBnTIRE のコールバック・ルーチンとして呼び出されます。

### [所属]

CG\_serial\_user.c

### [指定形式]

```
#include "CG_macrodriver.h"
void UARTBn_SingleErrorCallback ( UCHAR err_type );
```

**備考** *n* は，チャンネル番号を意味します。

### [引数]

I/O	引数	説明
O	UCHAR <i>err_type</i> ;	受信エラー割り込みの発生要因 00000xx1B：オーバラン・エラー 00000x1xB：フレーミング・エラー 000001xxB：パリティ・エラー

### [戻り値]

なし

## UARTBn\_FIFOErrorCallback

受信エラー割り込み INTUBnTIRE（オーバラン・エラー，フレーミング・エラー，パリティ・エラー）の発生に伴う処理を行います。

**備考** 本 API 関数は，UARTBn（FIFO モード）の受信エラー割り込み INTUBnTIRE に対応した割り込み処理 MD\_INTUBnTIRE のコールバック・ルーチンとして呼び出されます。

### [所属]

CG\_serial\_user.c

### [指定形式]

```
#include "CG_macrodriver.h"
void UARTBn_FIFOErrorCallback ( UCHAR err_type1, UCHAR err_type2 );
```

**備考** *n* は，チャンネル番号を意味します。

### [引数]

I/O	引数	説明
○	UCHAR <i>err_type1</i> ;	受信エラー割り込みの発生要因 00001000B : オーバラン・エラー
○	UCHAR <i>err_type2</i> ;	受信エラー割り込みの発生要因 000000x1B : フレーミング・エラー 0000001xB : パリティ・エラー

### [戻り値]

なし

## UARTB $n$ \_TimeoutErrorCallback

受信タイムアウト割り込み INTUB $n$ TITO の発生に伴う処理を行います。

**備考** 本 API 関数は、UARTB $n$  (FIFO モード) の受信タイムアウト割り込み INTUBA $n$ TITO に対応した割り込み処理 MD\_INTUB $n$ TITO のコールバック・ルーチンとして呼び出されます。

### [所属]

CG\_serial\_user.c

### [指定形式]

```
void UARTB $n$ _TimeoutErrorCallback ( void );
```

**備考**  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## UARTBn\_SoftOverRunCallback

オーバーラン・エラーの発生に伴う処理を行います。

**備考** 本API関数は、UARTBnの受信エラー割り込み INTUBnTIRE に対応した割り込み処理 MD\_INTUBnTIRE のコールバック・ルーチン（UARTBn\_ReceiveData 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]

CG\_serial\_user.c

### [指定形式]

```
#include "CG_macrodriver.h"
void UARTBn_SoftOverRunCallback ( UCHAR rx_data );
```

**備考** *n*は、チャンネル番号を意味します。

### [引数]

I/O	引数	説明
O	UCHAR <i>rx_data</i> ;	受信したデータ

### [戻り値]

なし

## UARTC $n$ \_Init

アシンクロナス・シリアル・インタフェース C (UARTC) 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]

CG\_serial.c

### [指定形式]

```
void UARTC $n$ _Init ( void );
```

**備考**  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## UARTCn\_UserInit

アシンクロナス・シリアル・インタフェース（UARTC）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、UARTCn\_Initのコールバック・ルーチンとして呼び出されます。

### [所属]

CG\_serial\_user.c

### [指定形式]

```
void UARTCn_UserInit ( void );
```

**備考** *n*は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## UARTC $n$ \_Start

アシンクロナス・シリアル・インタフェース C (UARTC) を動作許可状態へと移行します。

### [所属]

CG\_serial.c

### [指定形式]

```
void    UARTCn_Start ( void );
```

**備考**  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## UARTCn\_Stop

アシンクロナス・シリアル・インタフェース C (UARTC) を動作禁止状態へと移行します。

### [所属]

CG\_serial.c

### [指定形式]

```
void    UARTCn_Stop ( void );
```

**備考** *n*は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## UARTCn\_SendData

データの UARTCn 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 UARTCn 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.** UARTCn 送信を行う際には、本 API 関数の呼び出し以前に [UARTCn\\_Start](#) を呼び出す必要があります。

### [所属]

CG\_serial.c

### [指定形式]

```
#include "CG_macrodriver.h"
MD_STATUS UARTCn_SendData ( UCHAR *txbuf, USHORT txnum );
```

**備考** *n* は、チャンネル番号を意味します。

### [引数]

I/O	引数	説明
I	UCHAR * <i>txbuf</i> ;	送信するデータを格納したバッファへのポインタ
I	USHORT <i>txnum</i> ;	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
MD_DATAEXISTS	異常終了 - UCnTX レジスタに“次に転送すべきデータ”が存在する（データの送信中）

## UARTCn\_ReceiveData

データの UARTCn 受信を開始します。

- 備考 1.** 本 API 関数では、1 バイト単位の UARTCn 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 UARTCn 受信は、本 API 関数の呼び出し後、[UARTCn\\_Start](#) を呼び出すことにより開始されます。

### [所属]

CG\_serial.c

### [指定形式]

```
#include "CG_macrodriver.h"
MD_STATUS UARTCn_ReceiveData ( UCHAR *rxbuf, USHORT rxnum );
```

**備考** *n* は、チャンネル番号を意味します。

### [引数]

I/O	引数	説明
O	UCHAR * <i>rxbuf</i> ;	受信したデータを格納するバッファへのポインタ
I	USHORT <i>rxnum</i> ;	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

## UARTCn\_SendEndCallback

UARTCnの連続送信許可割り込み INTUCnT の発生に伴う処理を行います。

**備考** 本 API 関数は、UARTCnの連続送信許可割り込み INTUCnT に対応した割り込み処理 MD\_INTUCnT のコールバック・ルーチン (UARTCn\_SendData の引数 *txnum* で指定された総数の UARTCn 送信が完了した際の処理) として呼び出されます。

### [所属]

CG\_serial\_user.c

### [指定形式]

```
void    UARTCn_SendEndCallback ( void );
```

**備考** *n*は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## UARTCn\_ReceiveEndCallback

UARTCnの受信終了割り込み INTUCnR の発生に伴う処理を行います。

**備考** 本 API 関数は、UARTCnの受信終了割り込み INTUCnRに対応した割り込み処理 MD\_INTUCnR のコールバック・ルーチン (UARTCn\_ReceiveData の引数 *rxnum* で指定された総数の UARTCn 受信が完了した際の処理) として呼び出されます。

### [所属]

CG\_serial\_user.c

### [指定形式]

```
void    UARTCn_ReceiveEndCallback ( void );
```

**備考** *n*は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## UARTCn\_ErrorCallback

UARTCn 受信エラー割り込み INTUCnR (オーバラン・エラー, フレーミング・エラー, パリティ・エラー) の発生に伴う処理を行います。

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

### [所属]

CG\_serial\_user.c

### [指定形式]

```
#include "CG_macrodriver.h"
void UARTCn_ErrorCallback ( UCHAR 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	UCHAR <i>err_type</i> ;	エラー割り込みの発生要因 00000xx1B : オーバラン・エラー 00000x1xB : フレーミング・エラー 000001xxB : パリティ・エラー

### [戻り値]

なし

## UARTCn\_SoftOverRunCallback

オーバラン・エラーの発生に伴う処理を行います。

**備考** 本API関数は、UARTCnの受信エラー割り込み INTUCnRに対応した割り込み処理 MD\_INTUCnRのコールバック・ルーチン (UARTCn\_ReceiveData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]

CG\_serial\_user.c

### [指定形式]

```
#include "CG_macrodriver.h"
void UARTCn_SoftOverRunCallback ( UCHAR 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	UCHAR <i>rx_data</i> ;	受信したデータ

### [戻り値]

なし

## CSIB $n$ \_Init

3線式可変長シリアルI/O B (CSIB) 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]

CG\_serial.c

### [指定形式]

```
void CSIB $n$ _Init ( void );
```

**備考**  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## CSIB $n$ \_UserInit

3 線式可変長シリアル I/O B (CSIB) 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、CSIB $n$ \_Init のコールバック・ルーチンとして呼び出されます。

### [所属]

CG\_serial\_user.c

### [指定形式]

```
void CSIB $n$ _UserInit ( void );
```

**備考**  $n$  は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## CSIB $n$ \_Start

3線式可変長シリアルI/O B (CSIB) を動作許可状態へと移行します。

### [所属]

CG\_serial.c

### [指定形式]

```
void CSIBn_Start ( void );
```

**備考**  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## CSIB $n$ \_Stop

3線式可変長シリアルI/O B (CSIB) を動作禁止状態へと移行します。

### [所属]

CG\_serial.c

### [指定形式]

```
void CSIB $n$ _Stop ( void );
```

**備考**  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## CSIBn\_SendData

データの CSIBn 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 CSIBn 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.** CSIBn 送信を行う際には、本 API 関数の呼び出し以前に [CSIBn\\_Start](#) を呼び出す必要があります。

### [所属]

CG\_serial.c

### [指定形式]

```
#include "CG_macrodriver.h"
MD_STATUS CSIBn_SendData ( UCHAR *txbuf, USHORT txnum );
```

**備考** *n* は、チャンネル番号を意味します。

### [引数]

I/O	引数	説明
I	UCHAR * <i>txbuf</i> ;	送信するデータを格納したバッファへのポインタ
I	USHORT <i>txnum</i> ;	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

## CSIBn\_ReceiveData

データの CSIBn 受信を開始します。

- 備考 1.** 本 API 関数では、1 バイト単位の CSIBn 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.** CSIBn 受信を行う際には、本 API 関数の呼び出し以前に [CSIBn\\_Start](#) を呼び出す必要があります。

### [所属]

CG\_serial.c

### [指定形式]

```
#include "CG_macrodriver.h"
MD_STATUS CSIBn_ReceiveData ( UCHAR *rxbuf, USHORT rxnum );
```

**備考** *n* は、チャンネル番号を意味します。

### [引数]

I/O	引数	説明
O	UCHAR * <i>rxbuf</i> ;	受信したデータを格納するバッファへのポインタ
I	USHORT <i>rxnum</i> ;	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

## CSIBn\_SendReceiveData

データの CSIBn 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 CSIBn 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 CSIBn 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.** CSIBn 送受信を行う際には、本 API 関数の呼び出し以前に [CSIBn\\_Start](#) を呼び出す必要があります。

### [所属]

CG\_serial.c

### [指定形式]

```
#include "CG_macrodriver.h"
MD_STATUS CSIBn_SendReceiveData ( UCHAR *txbuf, USHORT txnum, UCHAR *rxbuf );
```

**備考** *n* は、チャンネル番号を意味します。

### [引数]

I/O	引数	説明
O	UCHAR * <i>txbuf</i> ;	受信したデータを格納するバッファへのポインタ
I	USHORT <i>txnum</i> ;	送受信するデータの総数
I	UCHAR * <i>rxbuf</i> ;	送信するデータを格納した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

## CSIBn\_SendEndCallback

CSIBnの受信終了割り込み INTCBnR, および CSIBnの連続書き込み許可割り込み INTCBnT の発生に伴う処理を行います。

**備考** 本 API 関数は、CSIBnの受信終了割り込み INTCBnR に対応した割り込み処理 MD\_INTCBnR, および CSIBnの連続書き込み許可割り込み INTCBnT に対応した割り込み処理 MD\_INTCBnT のコールバック・ルーチン (CSIBn\_SendData, または CSIBn\_SendReceiveData の引数 *txnum* で指定された総数の CSIBn 送信が完了した際の処理) として呼び出されます。

### [所属]

CG\_serial\_user.c

### [指定形式]

```
void CSIBn_SendEndCallback ( void );
```

**備考** *n*は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## CSIBn\_ReceiveEndCallback

CSIBnの受信終了割り込み INTCBnR の発生に伴う処理を行います。

**備考** 本 API 関数は、CSIBnの受信終了割り込み INTCBnR に対応した割り込み処理 MD\_INTCBnR のコールバック・ルーチン（CSIBn\_ReceiveData, または CSIBn\_SendReceiveData の引数 rxnum で指定された総数の CSIBn 受信が完了した際の処理）として呼び出されます。

### [所属]

CG\_serial\_user.c

### [指定形式]

```
void CSIBn_ReceiveEndCallback ( void );
```

**備考** nは、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## CSIBn\_ErrorCallback

CSIBnの受信エラー割り込み INTCBnR（オーバラン・エラー）の発生に伴う処理を行います。

**備考** 本API関数は、CSIBnの受信エラー割り込み INTCBnRに対応した割り込み処理 MD\_INTCBnRのコールバック・ルーチンとして呼び出されます。

### [所属]

CG\_serial\_user.c

### [指定形式]

```
#include "CG_macrodriver.h"
void CSIBn_ErrorCallback ( void );
```

**備考** nは、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## CSIE $n$ \_Init

3線式可変長シリアル I/O E (CSIE) 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]

CG\_serial.c

### [指定形式]

```
void CSIE $n$ _Init ( void );
```

**備考**  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## CSIEn\_UserInit

3 線式可変長シリアル I/O E (CSIE) 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、CSIEn\_Init のコールバック・ルーチンとして呼び出されます。

### [所属]

CG\_serial\_user.c

### [指定形式]

```
void CSIEn_UserInit ( void );
```

**備考** *n* は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## CSIE $n$ \_Start

3 線式可変長シリアル I/O E (CSIE) を動作許可状態へと移行します。

### [所属]

CG\_serial.c

### [指定形式]

```
void CSIE $n$ _Start ( void );
```

**備考**  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## CSIE*n*\_Stop

3 線式可変長シリアル I/O E (CSIE) を動作禁止状態へと移行します。

### [所属]

CG\_serial.c

### [指定形式]

```
void CSIEn_Stop ( void );
```

**備考** *n*は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## CSIE<sub>n</sub>\_SendData

データの CSIE<sub>n</sub> 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 CSIE<sub>n</sub> 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.** CSIE<sub>n</sub> 送信を行う際には、本 API 関数の呼び出し以前に [CSIE<sub>n</sub>\\_Start](#) を呼び出す必要があります。

### [所属]

CG\_serial.c

### [指定形式]

```
#include "CG_macrodriver.h"
MD_STATUS CSIEn_SendData ( UCHAR *txbuf, USHORT txnum );
```

**備考** *n* は、チャンネル番号を意味します。

### [引数]

I/O	引数	説明
I	UCHAR * <i>txbuf</i> ;	送信するデータを格納したバッファへのポインタ
I	USHORT <i>txnum</i> ;	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

## CSIE<sub>n</sub>\_ReceiveData

データの CSIE<sub>n</sub> 受信を開始します。

- 備考 1.** 本 API 関数では、1 バイト単位の CSIE<sub>n</sub> 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.** CSIE<sub>n</sub> 受信を行う際には、本 API 関数の呼び出し以前に [CSIE<sub>n</sub>\\_Start](#) を呼び出す必要があります。

### [所属]

CG\_serial.c

### [指定形式]

```
#include "CG_macrodriver.h"
MD_STATUS CSIEn_ReceiveData ( UCHAR *rxbuf, USHORT rxnum );
```

**備考** *n* は、チャンネル番号を意味します。

### [引数]

I/O	引数	説明
O	UCHAR * <i>rxbuf</i> ;	受信したデータを格納するバッファへのポインタ
I	USHORT <i>rxnum</i> ;	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

## CSIE<sub>n</sub>\_SendReceiveData

データの CSIE<sub>n</sub> 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 CSIE<sub>n</sub> 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 CSIE<sub>n</sub> 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.** CSIE<sub>n</sub> 送受信を行う際には、本 API 関数の呼び出し以前に [CSIE<sub>n</sub>\\_Start](#) を呼び出す必要があります。

### [所属]

CG\_serial.c

### [指定形式]

```
#include "CG_macrodriver.h"
MD_STATUS CSIEn_SendReceiveData ( UCHAR *txbuf, USHORT txnum, UCHAR *rxbuf );
```

**備考** *n* は、チャンネル番号を意味します。

### [引数]

I/O	引数	説明
O	UCHAR * <i>txbuf</i> ;	受信したデータを格納するバッファへのポインタ
I	USHORT <i>txnum</i> ;	送受信するデータの総数
I	UCHAR * <i>rxbuf</i> ;	送信するデータを格納した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

## CSIE $n$ \_SendEndCallback

CSIE $n$ の送受信完了割り込み INTCE $n$ Tの発生に伴う処理を行います。

**備考** 本API関数は、CSIE $n$ の送受信完了割り込み INTCE $n$ Tに対応した割り込み処理 MD\_INTCE $n$ Tのコールバック・ルーチン（CSIE $n$ \_SendData、またはCSIE $n$ \_SendReceiveDataの引数 *txnum* で指定された総数のCSIE $n$ 送信が完了した際の処理）として呼び出されます。

### [所属]

CG\_serial\_user.c

### [指定形式]

```
void CSIE $n$ _SendEndCallback ( void );
```

**備考**  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## CSIE*n*\_ReceiveEndCallback

CSIE*n*の送受信完了割り込み INTCE*n*Tの発生に伴う処理を行います。

**備考** 本API関数は、CSIE*n*の送受信完了割り込み INTCE*n*Tに対応した割り込み処理 MD\_INTCE*n*Tのコールバック・ルーチン（CSIE*n*\_ReceiveData、またはCSIE*n*\_SendReceiveDataの引数 *rxnum*で指定された総数のCSIE*n*受信が完了した際の処理）として呼び出されます。

### [所属]

CG\_serial\_user.c

### [指定形式]

```
void CSIEn_ReceiveEndCallback ( void );
```

**備考** *n*は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## CSIE $n$ \_ErrorCallback

CSIE $n$ の CSIE $n$ BUF オーバフロー割り込み INTCE $n$ TIOF の発生に伴う処理を行います。

**備考** 本 API 関数は、CSIE $n$ の CSIE $n$ BUF オーバフロー割り込み INTCE $n$ TIOF に対応した割り込み処理 MD\_INTCE $n$ TIOF のコールバック・ルーチンとして呼び出されます。

### [所属]

CG\_serial\_user.c

### [指定形式]

```
void CSIE $n$ _ErrorCallback ( void );
```

**備考**  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## CSIF $n$ \_Init

3線式可変長シリアルI/O F (CSIF) 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]

CG\_serial.c

### [指定形式]

```
void CSIFn_Init ( void );
```

**備考**  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## CSIF $n$ \_UserInit

3線式可変長シリアル I/O F (CSIF) 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、CSIF $n$ \_Init のコールバック・ルーチンとして呼び出されます。

### [所属]

CG\_serial\_user.c

### [指定形式]

```
void CSIF $n$ _UserInit ( void );
```

**備考**  $n$  は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## CSIF $n$ \_Start

3 線式可変長シリアル I/O F (CSIF) を動作許可状態へと移行します。

### [所属]

CG\_serial.c

### [指定形式]

```
void CSIFn_Start ( void );
```

**備考**  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## CSIF $n$ \_Stop

3線式可変長シリアルI/O F (CSIF) を動作禁止状態へと移行します。

### [所属]

CG\_serial.c

### [指定形式]

```
void CSIFn_Stop ( void );
```

**備考**  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## CSIFn\_SendData

データのCSIFn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CSIFn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.** CSIFn送信を行う際には、本API関数の呼び出し以前に [CSIFn\\_Start](#) を呼び出す必要があります。

### [所属]

CG\_serial.c

### [指定形式]

```
#include "CG_macrodriver.h"
MD_STATUS CSIFn_SendData ( UCHAR *txbuf, USHORT txnum );
```

**備考** *n* は、チャンネル番号を意味します。

### [引数]

I/O	引数	説明
I	UCHAR * <i>txbuf</i> ;	送信するデータを格納したバッファへのポインタ
I	USHORT <i>txnum</i> ;	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

## CSIFn\_ReceiveData

データのCSIFn受信を開始します。

- 備考 1.** 本API関数では、1バイト単位のCSIFn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.** CSIFn 受信を行う際には、本API関数の呼び出し以前に [CSIFn\\_Start](#) を呼び出す必要があります。

### [所属]

CG\_serial.c

### [指定形式]

```
#include "CG_macrodriver.h"
MD_STATUS CSIFn_ReceiveData ( UCHAR *rxbuf, USHORT rxnum );
```

**備考** *n* は、チャンネル番号を意味します。

### [引数]

I/O	引数	説明
O	UCHAR * <i>rxbuf</i> ;	受信したデータを格納するバッファへのポインタ
I	USHORT <i>rxnum</i> ;	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

## CSIFn\_SendReceiveData

データのCSIFn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CSIFn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CSIFn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.** CSIFn送受信を行う際には、本API関数の呼び出し以前に [CSIFn\\_Start](#) を呼び出す必要があります。

### [所属]

CG\_serial.c

### [指定形式]

```
#include "CG_macrodriver.h"
MD_STATUS CSIFn_SendReceiveData ( UCHAR *txbuf, USHORT txnum, UCHAR *rxbuf );
```

**備考** *n* は、チャンネル番号を意味します。

### [引数]

I/O	引数	説明
O	UCHAR * <i>txbuf</i> ;	受信したデータを格納するバッファへのポインタ
I	USHORT <i>txnum</i> ;	送受信するデータの総数
I	UCHAR * <i>rxbuf</i> ;	送信するデータを格納した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

## CSIF $n$ \_SendEndCallback

CSIF $n$ の送受信完了割り込み INTCF $n$ T の発生に伴う処理を行います。

**備考** 本 API 関数は、CSIF $n$ の送受信完了割り込み INTCF $n$ T に対応した割り込み処理 MD\_INTCF $n$ T のコールバック・ルーチン（CSIF $n$ \_SendData, または CSIF $n$ \_SendReceiveData の引数  $txnum$  で指定された総数の CSIF $n$  送信が完了した際の処理）として呼び出されます。

### [所属]

CG\_serial\_user.c

### [指定形式]

```
void CSIF $n$ _SendEndCallback ( void );
```

**備考**  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## CSIFn\_ReceiveEndCallback

CSIFnの送受信完了割り込み INTCFnT の発生に伴う処理を行います。

**備考** 本 API 関数は、CSIFnの送受信完了割り込み INTCFnT に対応した割り込み処理 MD\_INTCFnT のコールバック・ルーチン（CSIFn\_ReceiveData, または CSIFn\_SendReceiveData の引数 rxnum で指定された総数の CSIFn 受信が完了した際の処理）として呼び出されます。

### [所属]

CG\_serial\_user.c

### [指定形式]

```
void CSIFn_ReceiveEndCallback ( void );
```

**備考** nは、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## CSIFn\_ErrorCallback

CSIFnの受信エラー割り込み INTCFnR（オーバラン・エラー）の発生に伴う処理を行います。

**備考** 本API関数は、CSIFnの受信エラー割り込み INTCFnRに対応した割り込み処理 MD\_INTCFnRのコールバック・ルーチンとして呼び出されます。

### [所属]

CG\_serial\_user.c

### [指定形式]

```
void CSIFn_ErrorCallback ( void );
```

**備考** *n*は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## IIC0n\_Init

IICバ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]

CG\_serial.c

### [指定形式]

```
void IIC0n_Init ( void );
```

**備考** *n*は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## IIC0n\_UserInit

IICバ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、IIC0n\_Initのコールバック・ルーチンとして呼び出されます。

### [所属]

CG\_serial\_user.c

### [指定形式]

```
void IIC0n_UserInit ( void );
```

**備考** *n*は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## IIC0n\_Stop

IIC0n通信を終了します。

### [所属]

CG\_serial.c

### [指定形式]

```
void IIC0n_Stop ( void );
```

**備考** *n*は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## IIC0n\_StopCondition

ストップ・コンディションを生成します。

### [所属]

CG\_serial.c

### [指定形式]

```
void IIC0n_StopCondition ( void );
```

**備考**  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## IIC0n\_MasterSendStart

データのIIC0n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IIC0n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]

CG\_serial.c

### [指定形式]

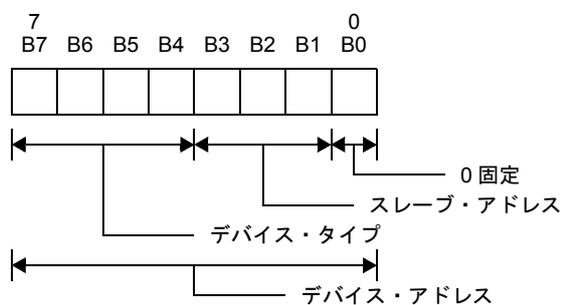
```
#include "CG_macrodriver.h"
MD_STATUS IIC0n_MasterSendStart ( UCHAR adr, UCHAR *txbuf, USHORT txnum, UCHAR wait );
```

**備考** *n*は、チャンネル番号を意味します。

### [引数]

I/O	引数	説明
I	UCHAR <i>adr</i> ;	デバイス・アドレス
I	UCHAR * <i>txbuf</i> ;	送信するデータを格納したバッファへのポインタ
I	USHORT <i>txnum</i> ;	送信するデータの総数
I	UCHAR <i>wait</i> ;	スタート・コンディションのセットアップ時間

**備考** デバイス・アドレス *adr*は、デバイス・タイプとスレーブ・アドレスから構成されます。



### [戻り値]

マクロ	説明
MD_OK	正常終了

---

マクロ	説明
MD_ERROR	異常終了

## IIC0n\_MasterReceiveStart

データのIIC0n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IIC0n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]

CG\_serial.c

### [指定形式]

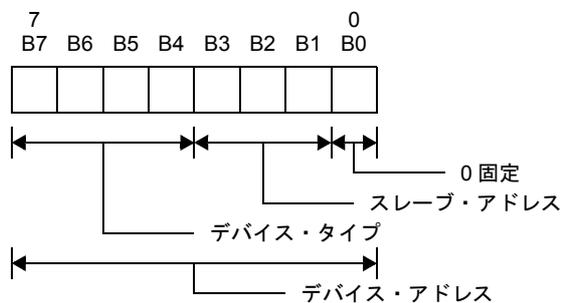
```
#include "CG_macrodriver.h"
MD_STATUS IIC0n_MasterReceiveStart ( UCHAR adr, UCHAR *rxbuf, USHORT rxnum, UCHAR wait );
```

**備考** *n*は、チャンネル番号を意味します。

### [引数]

I/O	引数	説明
I	UCHAR <i>adr</i> ;	デバイス・アドレス
O	UCHAR * <i>rxbuf</i> ;	受信したデータを格納するバッファへのポインタ
I	USHORT <i>rxnum</i> ;	受信するデータの総数
I	UCHAR <i>wait</i> ;	スタート・コンディションのセットアップ時間

**備考** デバイス・アドレス *adr*は、デバイス・タイプとスレーブ・アドレスから構成されます。



### [戻り値]

マクロ	説明
MD_OK	正常終了

---

マクロ	説明
MD_ERROR	異常終了

## IIC0n\_MasterSendEndCallback

IIC0n マスタ送信の転送終了割り込み INTIICn の発生に伴う処理を行います。

**備考** 本 API 関数は、IIC0n マスタ送信の転送終了割り込み INTIICn に対応した割り込み処理 MD\_INTIICn のコールバック・ルーチン（IIC0n\_MasterSendStart の引数 *txnum* で指定された総数の IIC0n マスタ送信が完了した際の処理）として呼び出されます。

### [所属]

CG\_serial\_user.c

### [指定形式]

```
void IIC0n_MasterSendEndCallback ( void );
```

**備考** *n* は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## IIC0n\_MasterReceiveEndCallback

IIC0n マスタ受信の転送終了割り込み INTIICn の発生に伴う処理を行います。

**備考** 本 API 関数は、IIC0n マスタ受信の転送終了割り込み INTIICn に対応した割り込み処理 MD\_INTIICn のコールバック・ルーチン（IIC0n\_MasterReceiveStart の引数 *rxnum* で指定された総数の IIC0n マスタ受信が完了した際の処理）として呼び出されます。

### [所属]

CG\_serial\_user.c

### [指定形式]

```
void IIC0n_MasterReceiveEndCallback ( void );
```

**備考** *n* は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## IIC0n\_MasterErrorCallback

IIC0n マスタ通信エラーの検出に伴う処理を行います。

**備考** 本 API 関数は、IIC0n の転送終了割り込み INTIICn に対応した割り込み処理 MD\_INTIICn のコールバック・ルーチン（IIC0n マスタ通信エラーを検出した際の処理）として呼び出されます。

### [所属]

CG\_serial\_user.c

### [指定形式]

```
#include "CG_macrodriver.h"
void IIC0n_MasterErrorCallback ( MD_STATUS flag );
```

**備考** n は、チャンネル番号を意味します。

### [引数]

I/O	引数	説明
○	MD_STATUS flag;	IIC0n マスタ通信エラーの発生要因 MD_SPT : IIC0n マスタ通信中にストップ・コンディションを検出 MD_NACK : アクノリッジを未検出

### [戻り値]

なし

## IIC0n\_SlaveSendStart

データの IIC0n 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 IIC0n 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]

CG\_serial.c

### [指定形式]

```
#include "CG_macrodriver.h"
void IIC0n_SlaveSendStart ( UCHAR *txbuf, USHORT txnum );
```

**備考** *n* は、チャンネル番号を意味します。

### [引数]

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

### [戻り値]

なし

## IIC0n\_SlaveReceiveStart

データの IIC0n 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 IIC0n 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]

CG\_serial.c

### [指定形式]

```
#include "CG_macrodriver.h"
void IIC0n_SlaveReceiveStart ( UCHAR *rxbuf, USHORT rxnum );
```

**備考** *n* は、チャンネル番号を意味します。

### [引数]

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

### [戻り値]

なし

## IIC0n\_SlaveSendEndCallback

IIC0n スレーブ送信の転送終了割り込み INTIICn の発生に伴う処理を行います。

**備考** 本 API 関数は、IIC0n スレーブ送信の転送終了割り込み INTIICn に対応した割り込み処理 MD\_INTIICn のコールバック・ルーチン（IIC0n\_SlaveSendStart の引数 *txnum* で指定された総数の IIC0n スレーブ送信が完了した際の処理）として呼び出されます。

### [所属]

CG\_serial\_user.c

### [指定形式]

```
void IIC0n_SlaveSendEndCallback ( void );
```

**備考** *n* は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## IIC0n\_SlaveReceiveEndCallback

IIC0n スレーブ受信の転送終了割り込み INTIICn の発生に伴う処理を行います。

**備考** 本 API 関数は、IIC0n スレーブ受信の転送終了割り込み INTIICn に対応した割り込み処理 MD\_INTIICn のコールバック・ルーチン（IIC0n\_SlaveReceiveStart の引数 *rxnum* で指定された総数の IIC0n スレーブ受信が完了した際の処理）として呼び出されます。

### [所属]

CG\_serial\_user.c

### [指定形式]

```
void IIC0n_SlaveReceiveEndCallback ( void );
```

**備考** *n* は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## IIC0n\_SlaveErrorCallback

IIC0n スレーブ通信エラーの検出に伴う処理を行います。

**備考** 本 API 関数は、IIC0n の転送終了割り込み INTIICn に対応した割り込み処理 MD\_INTIICn のコールバック・ルーチン（IIC0n スレーブ通信エラーを検出した際の処理）として呼び出されます。

### [所属]

CG\_serial\_user.c

### [指定形式]

```
#include "CG_macrodriver.h"
void IIC0n_SlaveErrorCallback ( MD_STATUS flag );
```

**備考** *n* は、チャンネル番号を意味します。

### [引数]

I/O	引数	説明
○	MD_STATUS <i>flag</i> ;	IIC0n マスタ通信エラーの発生要因 MD_ERROR: アドレスの不一致を検出 MD_NACK: アクノリッジを未検出

### [戻り値]

なし

## IIC0n\_GetStopConditionCallback

ストップ・コンディションの検出に伴う処理を行います。

**備考** 本API関数は、IIC0nの転送終了割り込み INTIICnに対応した割り込み処理 MD\_INTIICnのコールバック・ルーチン（ストップ・コンディションを検出した際の処理）として呼び出されます。

### [所属]

CG\_serial\_user.c

### [指定形式]

```
void IIC0n_GetStopConditionCallback ( void );
```

**備考** nは、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

### C.3.6 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—7 A/D コンバータ用 API 関数

API 関数名	機能概要
AD_Init	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。
AD_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。
AD_Start	A/D 変換を開始します。
AD_Stop	A/D 変換を終了します。
AD_SelectADChannel	A/D 変換するアナログ電圧の入力端子を設定します。
AD_SetPFTCondition	パワー・フェイル比較モードで動作する際の情報（比較値、A/D 変換終了割り込み INTAD の発生要因）を設定します。
AD_Read	A/D 変換結果（10 ビット）を読み出します。
AD_ReadByte	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。

## AD\_Init

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]

CG\_ad.c

### [指定形式]

```
void AD_Init ( void );
```

### [引数]

なし

### [戻り値]

なし

## AD\_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、[AD\\_Init](#) のコールバック・ルーチンとして呼び出されます。

### [所属]

CG\_ad\_user.c

### [指定形式]

```
void AD_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

## AD\_Start

A/D 変換を開始します。

**備考** 本 API 関数の呼び出しから [AD\\_Stop](#) が呼び出されるまでの間、A/D 変換処理が繰り返し実行されます。

### [所属]

CG\_ad.c

### [指定形式]

```
void AD_Start ( void );
```

### [引数]

なし

### [戻り値]

なし

### [使用例]

以下に、アナログ電圧を A/D 変換する際の例を示します。

【CG\_main.c】

```
#include "CG_macrodriver.h"

BOOL gFlag; /* A/D 変換完了フラグ */
void main ( void ) {
    USHORT buffer = 0;
    int wait = 100;
    gFlag = 1; /* A/D 変換完了フラグの初期化 */
    .....

    AD_Start (); /* A/D 変換の開始 */
    while ( gFlag ); /* 割り込み INTAD の発生待ち */
    AD_Read ( &buffer ); /* A/D 変換結果の読み出し */
    AD_Stop (); /* A/D 変換の終了 */
    .....
}
```

## 【CG\_ad\_user.c】

```
#include "CG_macrodriver.h"
extern BOOL gFlag; /* A/D 変換完了フラグ */
__interrupt void MD_INTAD ( void ) { /* 割り込み INTAD 発生時の割り込み処理 */
    gFlag = 0; /* A/D 変換完了フラグの設定 */
}
```

## AD\_Stop

A/D 変換を終了します。

### [所属]

CG\_ad.c

### [指定形式]

```
void AD_Stop ( void );
```

### [引数]

なし

### [戻り値]

なし

## AD\_SelectADChannel

A/D 変換するアナログ電圧の入力端子を設定します。

### [所属]

CG\_ad.c

### [指定形式]

```
#include "CG_macrodriver.h"
#include "CG_ad.h"
MD_STATUS AD_SelectADChannel ( enum ADChannel channel );
```

### [引数]

I/O	引数	説明
I	enum ADChannel <i>channel</i> ;	アナログ電圧の入力端子 ADCHANNEL <i>n</i> : 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 CG\_ad.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

## AD\_SetPFTCondition

パワー・フェイル比較モードで動作する際の情報（比較値、A/D変換終了割り込みINTADの発生要因）を設定します。

### [所属]

CG\_ad.c

### [指定形式]

```
#include "CG_macrodriver.h"
#include "CG_ad.h"
MD_STATUS AD_SetPFTCondition ( UCHAR pftvalue, enum ADPFTMode mode );
```

### [引数]

I/O	引数	説明
I	UCHAR <i>pftvalue</i> ;	比較値
I	enum ADPFTMode <i>mode</i> ;	A/D変換終了割り込みINTADの発生要因 EACHEND : A/D変換の終了時にINTADを発生 PFTHIGHER : ADA0CRnH $\geq$ ADA0PFTの際にINTADを発生 PFTLOWER : ADA0CRnH < ADA0PFTの際にINTADを発生

**備考** 引数 *pftvalue* に指定された値は、引数 *mode* に PFTHIGHER、また PFTLOWER が指定された際、パワー・フェイル比較しきい値レジスタ (ADA0PFT) に設定され、A/D変換結果レジスタ *nH* (ADA0CRnH) との比較に使用されます。

### [戻り値]

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

## AD\_Read

A/D 変換結果（10 ビット）を読み出します。

### [所属]

CG\_ad.c

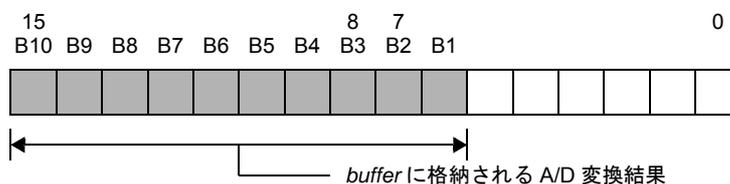
### [指定形式]

```
#include "CG_macrodriver.h"
MD_STATUS AD_Read ( USHORT *buffer );
```

### [引数]

I/O	引数	説明
O	USHORT *buffer;	読み出した A/D 変換結果（10 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]

マクロ	説明
MD_OK	正常終了
MD_ERROR	異常終了



### C.3.7 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—8 D/A コンバータ用 API 関数

API 関数名	機能概要
DAn_Init	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。
DAn_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。
DAn_Start	D/A 変換を開始します。
DAn_Stop	D/A 変換を終了します。
DAn_SetValue	ANOn 端子に出力するアナログ電圧値を設定します。

## DAn\_Init

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]

CG\_da.c

### [指定形式]

```
void DAn_Init ( void );
```

**備考**  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## DAn\_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、DAn\_Init のコールバック・ルーチンとして呼び出されます。

### [所属]

CG\_da\_user.c

### [指定形式]

```
void DAn_UserInit ( void );
```

**備考** *n* は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## DAn\_Start

D/A 変換を開始します。

### [所属]

CG\_da.c

### [指定形式]

```
void DAn_Start ( void );
```

**備考**  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## DAn\_Stop

D/A 変換を終了します。

### [所属]

CG\_da.c

### [指定形式]

```
void DAn_Stop ( void );
```

**備考** *n*は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## DAn\_SetValue

ANOn 端子に出力するアナログ電圧値を設定します。

### [所属]

CG\_da.c

### [指定形式]

```
#include "CG_macrodriver.h"
void DAn_SetValue ( UCHAR value );
```

**備考** nは、チャンネル番号を意味します。

### [引数]

I/O	引数	説明
I	UCHAR value;	アナログ電圧値 (0x0 ~ 0xff)

### [戻り値]

なし

### [使用例]

以下に、チャンネル0、およびチャンネル1に“アナログ電圧値”を設定する際の例を示します。

#### 【CG\_main.c】

```
#include "CG_macrodriver.h"
void main ( void ) {
    .....
    DA0_Start (); /* D/A 変換の開始 */
    DA1_Start (); /* D/A 変換の開始 */
    .....
    DA0_SetValue ( 0x7f ); /* アナログ電圧値の設定 */
    .....
}
```

## 【CG\_tau\_user.c】

```
#include "CG_macrodriver.h"
UCHAR gValue = 0;
__interrupt void MD_INTP2CC0 ( void ) { /* 割り込み INTP2CC0 発生時の割り込み処理 */
    DA1_SetValue ( gValue++ ); /* アナログ電圧値の設定 */
}
```

### C.3.8 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—9 タイマ用 API 関数

API 関数名	機能概要
TMPn_Init	16 ビット・タイマ/イベント・カウンタ P (TMP) 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。
TMPn_UserInit	16 ビット・タイマ/イベント・カウンタ P (TMP) に関するユーザ独自の初期化処理を行います。
TMPn_Start	16 ビット・タイマ/イベント・カウンタ P (TMP) のカウントを開始します。
TMPn_Stop	16 ビット・タイマ/イベント・カウンタ P (TMP) のカウントを終了します。
TMPn_ChangeTimerCondition	16 ビット・タイマ/イベント・カウンタ P (TMP) のカウント値を変更します。
TMPn_GetPulseWidth	16 ビット・タイマ/イベント・カウンタ P (TMP) のパルス幅 (ハイ・レベル幅, ロウ・レベル幅) を読み出します。
TMPn_GetFreeRunningValue	16 ビット・タイマ/イベント・カウンタ P (TMP) がキャプチャした値を読み出します。
TMPn_ChangeDuty	PWM 信号のデューティ比を変更します。
TMPn_SoftwareTriggerOn	タイマ出力のためのトリガ (ソフトウェア・トリガ) を発生させます。
TMQ0_Init	16 ビット・タイマ/イベント・カウンタ Q (TMQ) 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。
TMQ0_UserInit	16 ビット・タイマ/イベント・カウンタ Q (TMQ) に関するユーザ独自の初期化処理を行います。
TMQ0_Start	16 ビット・タイマ/イベント・カウンタ Q (TMQ) のカウントを開始します。
TMQ0_Stop	16 ビット・タイマ/イベント・カウンタ Q (TMQ) のカウントを終了します。
TMQ0_ChangeTimerCondition	16 ビット・タイマ/イベント・カウンタ Q (TMQ) のカウント値を変更します。
TMQ0_GetPulseWidth	16 ビット・タイマ/イベント・カウンタ Q (TMQ) のパルス幅 (ハイ・レベル幅, ロウ・レベル幅) を読み出します。
TMQ0_GetFreeRunningValue	16 ビット・タイマ/イベント・カウンタ Q (TMQ) がキャプチャした値を読み出します。
TMQ0_ChangeDuty	PWM 信号のデューティ比を変更します。
TMQ0_SoftwareTriggerOn	タイマ出力のためのトリガ (ソフトウェア・トリガ) を発生させます。
TAAAn_Init	16 ビット・タイマ/イベント・カウンタ AA (TAA) 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。
TAAAn_UserInit	16 ビット・タイマ/イベント・カウンタ AA (TAA) に関するユーザ独自の初期化処理を行います。

API 関数名	機能概要
TAAAn_Start	16 ビット・タイマ/イベント・カウンタ AA (TAA) のカウントを開始します。
TAAAn_Stop	16 ビット・タイマ/イベント・カウンタ AA (TAA) のカウントを終了します。
TAAAn_ChangeTimerCondition	16 ビット・タイマ/イベント・カウンタ AA (TAA) のカウント値を変更します。
TAAAn_ControlOutputToggle	16 ビット・タイマ/イベント・カウンタ AA (TAA) のトグル制御を変更します。
TAAAn_GetPulseWidth	16 ビット・タイマ/イベント・カウンタ AA (TAA) のパルス幅 (ハイ・レベル幅, ロウ・レベル幅) を読み出します。
TAAAn_GetFreeRunningValue	16 ビット・タイマ/イベント・カウンタ AA (TAA) がキャプチャした値を読み出します。
TAAAn_ChangeDuty	PWM 信号のデューティ比を変更します。
TAAAn_SoftwareTriggerOn	タイマ出力のためのトリガ (ソフトウェア・トリガ) を発生させます。
TABn_Init	16 ビット・タイマ/イベント・カウンタ AB (TAB) 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。
TABn_UserInit	16 ビット・タイマ/イベント・カウンタ AB (TAB) に関するユーザ独自の初期化処理を行います。
TABn_Start	16 ビット・タイマ/イベント・カウンタ AB (TAB) のカウントを開始します。
TABn_Stop	16 ビット・タイマ/イベント・カウンタ AB (TAB) のカウントを終了します。
TABn_ChangeTimerCondition	16 ビット・タイマ/イベント・カウンタ AB (TAB) のカウント値を変更します。
TABn_ControlOutputToggle	16 ビット・タイマ/イベント・カウンタ AB (TAB) のトグル制御を変更します。
TABn_GetPulseWidth	16 ビット・タイマ/イベント・カウンタ AB (TAB) のパルス幅 (ハイ・レベル幅, ロウ・レベル幅) を読み出します。
TABn_GetFreeRunningValue	16 ビット・タイマ/イベント・カウンタ AB (TAB) がキャプチャした値を読み出します。
TABn_ChangeDuty	PWM 信号のデューティ比を変更します。
TABn_SoftwareTriggerOn	タイマ出力のためのトリガ (ソフトウェア・トリガ) を発生させます。
TMT0_Init	16 ビット・タイマ/イベント・カウンタ T (TMT) 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。
TMT0_UserInit	16 ビット・タイマ/イベント・カウンタ T (TMT) に関するユーザ独自の初期化処理を行います。
TMT0_Start	16 ビット・タイマ/イベント・カウンタ T (TMT) のカウントを開始します。
TMT0_Stop	16 ビット・タイマ/イベント・カウンタ T (TMT) のカウントを終了します。

API 関数名	機能概要
TMT0_ChangeTimerCondition	16 ビット・タイマ/イベント・カウンタ T (TMT) のカウント値を変更します。
TMT0_GetPulseWidth	16 ビット・タイマ/イベント・カウンタ T (TMT) のパルス幅 (ハイ・レベル幅, ロウ・レベル幅) を読み出します。
TMT0_GetFreeRunningValue	16 ビット・タイマ/イベント・カウンタ T (TMT) がキャプチャした値を読み出します。
TMT0_ChangeDuty	PWM 信号のデューティ比を変更します。
TMT0_SoftwareTriggerOn	タイマ出力のためのトリガ (ソフトウェア・トリガ) を発生させます。
TMT0_EnableHold	16 ビット・タイマ/イベント・カウンタ T (TMT) のエンコーダ・カウンタ制御を保持動作へと変更します。
TMT0_DisableHold	16 ビット・タイマ/イベント・カウンタ T (TMT) のエンコーダ・カウンタ制御を通常動作へと変更します。
TMT0_ChangeCountValue	16 ビット・タイマ/イベント・カウンタ T (TMT) の初期カウント値を変更します。
TMMn_Init	16 ビット・インターバル・タイマ M (TMM) 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。
TMMn_UserInit	16 ビット・インターバル・タイマ M (TMM) に関するユーザ独自の初期化処理を行います。
TMMn_Start	16 ビット・インターバル・タイマ M (TMM) のカウントを開始します。
TMMn_Stop	16 ビット・インターバル・タイマ M (TMM) のカウントを終了します。
TMMn_ChangeTimerCondition	16 ビット・インターバル・タイマ M (TMM) のカウント値を変更します。

## TMP $n$ \_Init

16ビット・タイマ/イベント・カウンタP (TMP) 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]

CG\_timer.c

### [指定形式]

```
void    TMP $n$ _Init ( void );
```

**備考**  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## TMP $n$ \_UserInit

16 ビット・タイマ/イベント・カウンタ P (TMP) 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、[TMP \$n\$ \\_Init](#) のコールバック・ルーチンとして呼び出されます。

### [所属]

CG\_timer\_user.c

### [指定形式]

```
void    TMP $n$ _UserInit ( void );
```

**備考**  $n$  は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## TMP $n$ \_Start

16 ビット・タイマ/イベント・カウンタ P (TMP) 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、外部イベント・カウンタ、外部トリガ・パルス出力など) により異なります。

### [所属]

CG\_timer.c

### [指定形式]

```
void    TMPn_Start ( void );
```

**備考**  $n$  は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## TMP $n$ \_Stop

16 ビット・タイマ/イベント・カウンタ P (TMP) のカウントを終了します。

**備考** 本 API 関数を呼び出してからカウント処理を終了するまでの時間は、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、外部トリガ・パルス出力など) により異なります。

### [所属]

CG\_timer.c

### [指定形式]

```
void    TMP $n$ _Stop ( void );
```

**備考**  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## TMP $n$ \_ChangeTimerCondition

16ビット・タイマ/イベント・カウンタ P (TMP) のカウント値を変更します。

**備考** 引数 *array\_reg* に指定された値は、TMP $n$  キャプチャ/コンペア・レジスタ *m* (TPnCCR $m$ ) に設定されます。

### [所属]

CG\_timer.c

### [指定形式]

```
#include "CG_macrodriver.h"
MD_STATUS TMPn_ChangeTimerCondition ( USHORT *array_reg, UCHAR array_num );
```

**備考** *n* は、チャンネル番号を意味します。

### [引数]

I/O	引数	説明
I	USHORT *array_reg;	カウント値 (0x0 ~ 0xffff) を格納した領域へのポインタ
I	UCHAR array_num;	変更対象レジスタ 1: TPnCCR0 2: TPnCCR0, TPnCCR1

### [戻り値]

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

### [使用例]

以下に、インターバル時間を半分に変更する際の例を示します。

なお、下記は、チャンネル0がインターバル・タイマ用として選択された場合の例となっています。

【CG\_main.c】

```
#include "CG_macrodriver.h"
void main ( void ) {
    int flag_finish = 1;
```

```
USHORT  array_reg = TMP_TPOCCR0_VALUE >> 1; /* TMP_TPOCCR0_VALUE : 現在のインターバル時間 */
UCHAR   array_num = 1;
.....
TMP0_Start ();                               /* カウントの開始 */
while ( flag_finish );                       /* タイムアップの確認 */
.....
TMP0_ChannelTimerCondition ( &array_reg, array_num ); /* カウント値の変更 */
.....
}
```

## TMP $n$ \_GetPulseWidth

16ビット・タイマ/イベント・カウンタP (TMP) のパルス幅 (ハイ・レベル幅, ロウ・レベル幅) を読み出します。

- 備考 1.** 本API関数の呼び出しは, 16ビット・タイマ/イベント・カウンタP (TMP) をパルス幅測定用として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ーフロー (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]

CG\_timer.c

### [指定形式]

```
#include "CG_macrodriver.h"
void TMPn_GetPulseWidth ( ULONG *activewith, ULONG *inactivewidth );
```

**備考**  $n$ は, チャンネル番号を意味します。

### [引数]

I/O	引数	説明
○	ULONG *activewith;	読み出したハイ・レベル幅 (0x0 ~ 0x1ffff) を格納する領域へのポインタ
○	ULONG *inactivewidth;	読み出したロウ・レベル幅 (0x0 ~ 0x1ffff) を格納する領域へのポインタ

### [戻り値]

なし

## TMP $n$ \_GetFreeRunningValue

16ビット・タイマ/イベント・カウンタ P (TMP) がキャプチャした値を読み出します。

**備考** 本API関数の呼び出しは、16ビット・タイマ/イベント・カウンタ P (TMP) をフリー・ランニング・タイマ用として使用し、TMP $n$ キャプチャ/コンペア・レジスタ  $m$  (TP $n$ CCR $m$ ) がキャプチャ・レジスタとして選択されている場合に限られます。

### [所属]

CG\_timer.c

### [指定形式]

```
#include "CG_macrodriver.h"
#include "CG_timer.h"
MD_STATUS TMPn_GetFreeRunningValue ( ULONG *count, enum TMChannel channel );
```

**備考**  $n$ は、チャンネル番号を意味します。

### [引数]

I/O	引数	説明
O	ULONG *count;	読み出した幅を格納する領域へのポインタ
I	enum TMChannel channel;	読み出し対象チャンネル TMCHANNEL0: チャンネル0 (TP $n$ CCR0) TMCHANNEL1: チャンネル1 (TP $n$ CCR1)

### [戻り値]

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

## TMP $n$ \_ChangeDuty

PWM 信号のデューティ比を変更します。

**備考** 本 API 関数の呼び出しは、16 ビット・タイマ/イベント・カウンタ P (TMP) を外部トリガ・パルス出力/PWM 出力用として使用している場合に限られます。

### [所属]

CG\_timer.c

### [指定形式]

```
#include "CG_macrodriver.h"
void TMPn_ChangeDuty ( UCHAR array_duty );
```

**備考**  $n$  は、チャンネル番号を意味します。

### [引数]

I/O	引数	説明
I	UCHAR array_duty;	デューティ比 (0 ~ 100, 単位: %)

**備考** デューティ比 array\_duty に設定する値は、10 進数に限られます。

### [戻り値]

なし

### [使用例]

以下に、デューティ比を 25% に変更する際の例を示します。

【CG\_main.c】

```
#include "CG_macrodriver.h"
void main ( void ) {
    int flagStatus = 1;
    UCHAR array_duty = 25;
    .....
    TMP0_Start (); /* カウントの開始 */
    while ( flagStatus );
```

```
TMP0_ChangeDuty ( array_duty );    /* デューティ比の変更 */  
.....  
}
```

## TMP $n$ \_SoftwareTriggerOn

タイマ出力のためのトリガ（ソフトウェア・トリガ）を発生させます。

**備考** 本API関数の呼び出しは、16ビット・タイマ/イベント・カウンタP（TMP）を外部トリガ・パルス出力/ワンショット・パルス出力用として使用している場合に限られます。

### [所属]

CG\_timer.c

### [指定形式]

```
void    TMP $n$ _SoftwareTriggerOn ( void );
```

**備考**  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## TMQ0\_Init

16ビット・タイマ/イベント・カウンタ (TMQ) 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]

CG\_timer.c

### [指定形式]

```
void    TMQ0_Init ( void );
```

### [引数]

なし

### [戻り値]

なし

## TMQ0\_UserInit

16 ビット・タイマ/イベント・カウンタ (TMQ) 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、[TMQ0\\_Init](#) のコールバック・ルーチンとして呼び出されます。

### [所属]

CG\_timer\_user.c

### [指定形式]

```
void    TMQ0_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

## TMQ0\_Start

16ビット・タイマ/イベント・カウンタ (TMQ) 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、外部イベント・カウンタ、外部トリガ・パルス出力など) により異なります。

### [所属]

CG\_timer.c

### [指定形式]

```
void    TMQ0_Start ( void );
```

### [引数]

なし

### [戻り値]

なし

## TMQ0\_Stop

16ビット・タイマ/イベント・カウンタ (TMQ) のカウントを終了します。

**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、外部イベント・カウンタ、外部トリガ・パルス出力など) により異なります。

### [所属]

CG\_timer.c

### [指定形式]

```
void    TMQ0_Stop ( void );
```

### [引数]

なし

### [戻り値]

なし

## TMQ0\_ChangeTimerCondition

16ビット・タイマ/イベント・カウンタ (TMQ) のカウント値を変更します。

**備考** 引数 *array\_reg* に指定された値は、TMQ0 キャプチャ/コンペア・レジスタ *m* (TQ0CCR*m*) に設定されます。

### [所属]

CG\_timer.c

### [指定形式]

```
#include "CG_macrodriver.h"
MD_STATUS TMQ0_ChangeTimerCondition ( USHORT *array_reg, UCHAR array_num );
```

### [引数]

I/O	引数	説明
I	USHORT *array_reg;	カウント値 (0x0 ~ 0xffff) を格納した領域へのポインタ
I	UCHAR array_num;	変更対象レジスタ 1: TQ0CCR0 2: TQ0CCR0, TQ0CCR1 3: TQ0CCR0, TQ0CCR1, TQ0CCR2 4: TQ0CCR0, TQ0CCR1, TQ0CCR2, TQ0CCR3

### [戻り値]

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

### [使用例]

以下に、インターバル時間を半分に変更する際の例を示します。

なお、下記は、チャンネル0がインターバル・タイマ用として選択された場合の例となっています。

【CG\_main.c】

```
#include "CG_macrodriver.h"
void main ( void ) {
    int flag_finish = 1;
```

```
USHORT array_reg = TMQ_TQ0CCR0_VALUE >> 1; /* TMQ_TP0CCR0_VALUE : 現在のインターバル時間 */
UCHAR array_num = 1;
.....
TMQ0_Start (); /* カウントの開始 */
while ( flag_finish ); /* タイムアップの確認 */
.....
TMQ0_ChannelTimerCondition ( &array_reg, array_num ); /* カウント値の変更 */
.....
}
```

## TMQ0\_GetPulseWidth

16ビット・タイマ/イベント・カウンタ (TMQ) のパルス幅 (ハイ・レベル幅, ロウ・レベル幅) を読み出します。

- 備考 1.** 本 API 関数の呼び出しは, 16ビット・タイマ/イベント・カウンタ (TMQ) をパルス幅測定用として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ーフロー (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]

CG\_timer.c

### [指定形式]

```
#include "CG_macrodriver.h"
void TMQ0_GetPulseWidth ( ULONG *activewidth, ULONG *inactivewidth );
```

### [引数]

I/O	引数	説明
○	ULONG *activewidth;	読み出したハイ・レベル幅 (0x0 ~ 0x1ffff) を格納する領域へのポインタ
○	ULONG *inactivewidth;	読み出したロウ・レベル幅 (0x0 ~ 0x1ffff) を格納する領域へのポインタ

### [戻り値]

なし

## TMQ0\_GetFreeRunningValue

16 ビット・タイマ/イベント・カウンタ (TMQ) がキャプチャした値を読み出します。

**備考** 本 API 関数の呼び出しは、16 ビット・タイマ/イベント・カウンタ (TMQ) をフリー・ランニング・タイマ用として使用し、TMQ0 キャプチャ/コンペア・レジスタ *m* (TQ0CCR*m*) がキャプチャ・レジスタとして選択されている場合に限られます。

### [所属]

CG\_timer.c

### [指定形式]

```
#include "CG_macrodriver.h"
#include "CG_timer.h"
MD_STATUS TMQ0_GetFreeRunningValue ( ULONG *count, enum TMChannel channel );
```

### [引数]

I/O	引数	説明
O	ULONG *count;	読み出した幅を格納する領域へのポインタ
I	enum TMChannel channel;	読み出し対象チャネル TMCHANNEL0 : チャネル 0 (TQ0CCR0) TMCHANNEL1 : チャネル 1 (TQ0CCR1) TMCHANNEL2 : チャネル 2 (TQ0CCR2) TMCHANNEL3 : チャネル 3 (TQ0CCR3)

### [戻り値]

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

## TMQ0\_ChangeDuty

PWM 信号のデューティ比を変更します。

**備考** 本 API 関数の呼び出しは、16 ビット・タイマ/イベント・カウンタ (TMQ) を外部トリガ・パルス出力/PWM 出力用として使用している場合に限られます。

### [所属]

CG\_timer.c

### [指定形式]

```
#include "CG_macrodriver.h"
MD_STATUS TMQ0_ChangeDuty ( UCHAR *array_duty, UCHAR array_num );
```

### [引数]

I/O	引数	説明
I	UCHAR *array_duty;	デューティ比 (0 ~ 100, 単位 : %) を格納した領域へのポインタ
I	UCHAR array_num;	変更対象レジスタ 1: TQ0CCR1 2: TQ0CCR1, TQ0CCR2 3: TQ0CCR1, TQ0CCR2, TQ0CCR3

**備考** デューティ比 array\_duty に設定する値は、10 進数に限られます。

### [戻り値]

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

### [使用例]

以下に、デューティ比を 25% に変更する際の例を示します。

【CG\_main.c】

```
#include "CG_macrodriver.h"
void main ( void ) {
```

```
int    flagStatus = 1;
UCHAR  array_duty = 25;
UCHAR  array_num = 1;
.....
TMQ0_Start ();                                /* カウントの開始 */
while ( flagStatus );
TMQ0_ChangeDuty ( &array_duty, array_num );  /* デューティ比の変更 */
.....
}
```

## TMQ0\_SoftwareTriggerOn

タイマ出力のためのトリガ（ソフトウェア・トリガ）を発生させます。

**備考** 本API関数の呼び出しは、16ビット・タイマ/イベント・カウンタ（TMQ）を外部トリガ・パルス出力/ワンショット・パルス出力用として使用している場合に限られます。

### [所属]

CG\_timer.c

### [指定形式]

```
void    TMQ0_SoftwareTriggerOn ( void );
```

### [引数]

なし

### [戻り値]

なし

## TAA $n$ \_Init

16ビット・タイマ/イベント・カウンタ AA (TAA) 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]

CG\_timer.c

### [指定形式]

```
void TAA $n$ _Init ( void );
```

**備考**  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## TAA $n$ \_UserInit

16ビット・タイマ/イベント・カウンタ AA (TAA) 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、TAA $n$ \_Initのコールバック・ルーチンとして呼び出されます。

### [所属]

CG\_timer\_user.c

### [指定形式]

```
void TAA $n$ _UserInit ( void );
```

**備考**  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## TAA $n$ \_Start

16ビット・タイマ/イベント・カウンタ AA (TAA) 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、外部イベント・カウンタ、外部トリガ・パルス出力など）により異なります。

### [所属]

CG\_timer.c

### [指定形式]

```
void TAA $n$ _Start ( void );
```

**備考**  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## TAA $n$ \_Stop

16 ビット・タイマ/イベント・カウンタ AA (TAA) のカウントを終了します。

**備考** 本 API 関数を呼び出してからカウント処理を終了するまでの時間は、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、外部トリガ・パルス出力など）により異なります。

### [所属]

CG\_timer.c

### [指定形式]

```
void TAA $n$ _Stop ( void );
```

**備考**  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## TAA<sub>n</sub>\_ChangeTimerCondition

16ビット・タイマ/イベント・カウンタ AA (TAA) のカウント値を変更します。

**備考** 引数 *array\_reg* に指定された値は、TAA<sub>n</sub> キャプチャ/コンペア・レジスタ *m* (TAA<sub>n</sub>CCR*m*) に設定されます。

### [所属]

CG\_timer.c

### [指定形式]

```
#include "CG_macrodriver.h"
MD_STATUS TAAn_ChangeTimerCondition ( USHORT *array_reg, UCHAR array_num );
```

**備考** *n* は、チャネル番号を意味します。

### [引数]

I/O	引数	説明
I	USHORT *array_reg;	カウント値 (0x0 ~ 0xffff) を格納した領域へのポインタ
I	UCHAR array_num;	変更対象レジスタ 1: TAA <sub>n</sub> CCR0 2: TAA <sub>n</sub> CCR0, TAA <sub>n</sub> CCR1

### [戻り値]

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

## TAA<sub>n</sub>\_ControlOutputToggle

16ビット・タイマ/イベント・カウンタ AA (TAA) のトグル制御を変更します。

**備考** 本API関数の呼び出しは、16ビット・タイマ/イベント・カウンタ AA (TAA) をインターバル・タイマ/フリー・ランニング・タイマ用として使用している場合に限られます。

### [所属]

CG\_timer.c

### [指定形式]

```
#include "CG_macrodriver.h"
#include "CG_timer.h"
MD_STATUS TAAn_ControlOutputToggle ( enum TMOOutput toggle, enum TMChannel channel );
```

**備考** *n*は、チャンネル番号を意味します。

### [引数]

I/O	引数	説明
I	enum TMOOutput <i>toggle</i> ;	トグル制御 STANDARD : 通常のトグル動作 INACTIVE : リセット要求 ACTIVE : セット要求 FREEZE : キープ要求
I	enum TMChannel <i>channel</i> ;	対象端子 TMCHANNEL0 : TOAA <sub>n</sub> 0 端子 TMCHANNEL1 : TOAA <sub>n</sub> 1 端子

### [戻り値]

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

## TAA<sub>n</sub>\_GetPulseWidth

16ビット・タイマ/イベント・カウンタ AA (TAA) のパルス幅 (ハイ・レベル幅, ロウ・レベル幅) を読み出します。

- 備考 1.** 本 API 関数の呼び出しは, 16ビット・タイマ/イベント・カウンタ AA (TAA) をパルス幅測定用として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]

CG\_timer.c

### [指定形式]

```
#include "CG_macrodriver.h"
void TAAn_GetPulseWidth ( ULONG *activewith, ULONG *inactivewidth );
```

**備考** *n* は, チャンネル番号を意味します。

### [引数]

I/O	引数	説明
○	ULONG *activewith;	読み出したハイ・レベル幅 (0x0 ~ 0x1ffff) を格納する領域へのポインタ
○	ULONG *inactivewidth;	読み出したロウ・レベル幅 (0x0 ~ 0x1ffff) を格納する領域へのポインタ

### [戻り値]

なし

## TAA $n$ \_GetFreeRunningValue

16 ビット・タイマ/イベント・カウンタ AA (TAA) がキャプチャした値を読み出します。

**備考** 本 API 関数の呼び出しは、16 ビット・タイマ/イベント・カウンタ AA (TAA) をフリー・ランニング・タイマ用として使用し、TAA $n$ キャプチャ/コンペア・レジスタ  $m$  (TAA $n$ CCRM) がキャプチャ・レジスタとして選択されている場合に限られます。

### [所属]

CG\_timer.c

### [指定形式]

```
#include "CG_macrodriver.h"
#include "CG_timer.h"
MD_STATUS TAA $n$ _GetFreeRunningValue ( ULONG *count, enum TMChannel channel );
```

**備考**  $n$ は、チャンネル番号を意味します。

### [引数]

I/O	引数	説明
O	ULONG *count;	読み出した幅を格納する領域へのポインタ
I	enum TMChannel channel;	読み出し対象チャンネル TMCHANNEL0: チャンネル 0 (TAA $n$ CCRO) TMCHANNEL1: チャンネル 1 (TAA $n$ CCR1)

### [戻り値]

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

## TAA $n$ \_ChangeDuty

PWM 信号のデューティ比を変更します。

**備考** 本 API 関数の呼び出しは、16 ビット・タイマ/イベント・カウンタ AA (TAA) を外部トリガ・パルス出力/PWM 出力用として使用している場合に限られます。

### [所属]

CG\_timer.c

### [指定形式]

```
#include "CG_macrodriver.h"
void TAA $n$ _ChangeDuty ( UCHAR array_duty );
```

**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	UCHAR array_duty;	デューティ比 (0 ~ 100, 単位 : %)

**備考** デューティ比 array\_duty に設定する値は、10 進数に限られます。

### [戻り値]

なし

## TAA*n*\_SoftwareTriggerOn

タイマ出力のためのトリガ（ソフトウェア・トリガ）を発生させます。

**備考** 本API関数の呼び出しは、16ビット・タイマ/イベント・カウンタAA（TAA）を外部トリガ・パルス出力/ワンショット・パルス出力用として使用している場合に限られます。

### [所属]

CG\_timer.c

### [指定形式]

```
void TAAn_SoftwareTriggerOn ( void );
```

**備考** *n*は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## TAB $n$ \_Init

16 ビット・タイマ/イベント・カウンタ AB (TAB) 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]

CG\_timer.c

### [指定形式]

```
void TAB $n$ _Init ( void );
```

**備考**  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## TAB $n$ \_UserInit

16 ビット・タイマ/イベント・カウンタ AB (TAB) 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、TAB $n$ \_Init のコールバック・ルーチンとして呼び出されます。

### [所属]

CG\_timer\_user.c

### [指定形式]

```
void TAB $n$ _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

## TAB $n$ \_Start

16 ビット・タイマ/イベント・カウンタ AB (TAB) 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、外部イベント・カウンタ、外部トリガ・パルス出力など) により異なります。

### [所属]

CG\_timer.c

### [指定形式]

```
void TAB $n$ _Start ( void );
```

**備考**  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## TAB $n$ \_Stop

16 ビット・タイマ/イベント・カウンタ AB (TAB) のカウントを終了します。

**備考** 本 API 関数を呼び出してからカウント処理を終了するまでの時間は、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、外部トリガ・パルス出力など) により異なります。

### [所属]

CG\_timer.c

### [指定形式]

```
void TAB $n$ _Stop ( void );
```

**備考**  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## TAB $n$ \_ChangeTimerCondition

16ビット・タイマ/イベント・カウンタ AB (TAB) のカウント値を変更します。

**備考** 引数 *array\_reg* に指定された値は、TAB $n$  キャプチャ/コンペア・レジスタ  $m$  (TAB $n$ CCR $m$ ) に設定されます。

### [所属]

CG\_timer.c

### [指定形式]

```
#include "CG_macrodriver.h"
MD_STATUS TABn_ChangeTimerCondition ( USHORT *array_reg, UCHAR array_num );
```

**備考**  $n$  は、チャンネル番号を意味します。

### [引数]

I/O	引数	説明
I	USHORT *array_reg;	カウント値 (0x0 ~ 0xffff) を格納した領域へのポインタ
I	UCHAR array_num;	変更対象レジスタ 1: TAB $n$ CCR0 2: TAB $n$ CCR0, TAB $n$ CCR1 3: TAB $n$ CCR0, TAB $n$ CCR1, TAB $n$ CCR2 4: TAB $n$ CCR0, TAB $n$ CCR1, TAB $n$ CCR2, TAB $n$ CCR3

### [戻り値]

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

## TAB<sub>n</sub>\_ControlOutputToggle

16ビット・タイマ/イベント・カウンタ AB (TAB) のトグル制御を変更します。

**備考** 本API関数の呼び出しは、16ビット・タイマ/イベント・カウンタ AB (TAB) をインターバル・タイマ/フリー・ランニング・タイマ用として使用している場合に限られます。

### [所属]

CG\_timer.c

### [指定形式]

```
#include "CG_macrodriver.h"
#include "CG_timer.h"
MD_STATUS TABn_ControlOutputToggle ( enum TMOOutput toggle, enum TMChannel channel );
```

**備考** *n*は、チャンネル番号を意味します。

### [引数]

I/O	引数	説明
I	enum TMOOutput <i>toggle</i> ;	トグル制御 STANDARD : 通常のトグル動作 INACTIVE : リセット要求 ACTIVE : セット要求 FREEZE : キープ要求
I	enum TMChannel <i>channel</i> ;	対象端子 TMCHANNEL0 : TOAB <sub>n</sub> 0 端子 TMCHANNEL1 : TOAB <sub>n</sub> 1 端子

### [戻り値]

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

## TAB<sub>n</sub>\_GetPulseWidth

16ビット・タイマ/イベント・カウンタ AB (TAB) のパルス幅 (ハイ・レベル幅, ロウ・レベル幅) を読み出します。

- 備考 1.** 本 API 関数の呼び出しは, 16 ビット・タイマ/イベント・カウンタ AB (TAB) をパルス幅測定用として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]

CG\_timer.c

### [指定形式]

```
#include "CG_macrodriver.h"
void TABn_GetPulseWidth ( ULONG *activewidth, ULONG *inactivewidth );
```

**備考** *n* は, チャンネル番号を意味します。

### [引数]

I/O	引数	説明
○	ULONG *activewidth;	読み出したハイ・レベル幅 (0x0 ~ 0x1ffff) を格納する領域へのポインタ
○	ULONG *inactivewidth;	読み出したロウ・レベル幅 (0x0 ~ 0x1ffff) を格納する領域へのポインタ

### [戻り値]

なし

## TAB $n$ \_GetFreeRunningValue

16 ビット・タイマ/イベント・カウンタ AB (TAB) がキャプチャした値を読み出します。

**備考** 本 API 関数の呼び出しは、16 ビット・タイマ/イベント・カウンタ AB (TAB) をフリー・ランニング・タイマ用として使用し、TAB $n$ キャプチャ/コンペア・レジスタ  $m$  (TAB $n$ CCRM) がキャプチャ・レジスタとして選択されている場合に限られます。

### [所属]

CG\_timer.c

### [指定形式]

```
#include "CG_macrodriver.h"
#include "CG_timer.h"
MD_STATUS TABn_GetFreeRunningValue ( ULONG *count, enum TMChannel channel );
```

**備考**  $n$ は、チャンネル番号を意味します。

### [引数]

I/O	引数	説明
O	ULONG *count;	読み出した幅を格納する領域へのポインタ
I	enum TMChannel channel;	読み出し対象チャンネル TMCHANNEL0: チャンネル 0 (TAB $n$ CCRO) TMCHANNEL1: チャンネル 1 (TAB $n$ CCR1) TMCHANNEL2: チャンネル 2 (TAB $n$ CCR2) TMCHANNEL3: チャンネル 3 (TAB $n$ CCR3)

### [戻り値]

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

## TAB<sub>n</sub>\_ChangeDuty

PWM 信号のデューティ比を変更します。

**備考** 本 API 関数の呼び出しは、16 ビット・タイマ/イベント・カウンタ AB (TAB) を外部トリガ・パルス出力/PWM 出力用として使用している場合に限られます。

### [所属]

CG\_timer.c

### [指定形式]

```
#include "CG_macrodriver.h"
MD_STATUS TABn_ChangeDuty ( UCHAR array_duty, UCHAR array_num );
```

**備考** *n* は、チャンネル番号を意味します。

### [引数]

I/O	引数	説明
I	UCHAR <i>array_duty</i> ;	デューティ比 (0 ~ 100, 単位: %)
I	UCHAR <i>array_num</i> ;	変更対象レジスタ 1: TAB <sub>n</sub> CCR1 2: TAB <sub>n</sub> CCR1, TAB <sub>n</sub> CCR2 3: TAB <sub>n</sub> CCR1, TAB <sub>n</sub> CCR2, TAB <sub>n</sub> CCR3

**備考** デューティ比 *array\_duty* に設定する値は、10 進数に限られます。

### [戻り値]

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

## TAB $n$ \_SoftwareTriggerOn

タイマ出力のためのトリガ（ソフトウェア・トリガ）を発生させます。

**備考** 本API関数の呼び出しは、16ビット・タイマ/イベント・カウンタ AB（TAB）を外部トリガ・パルス出力/ワンショット・パルス出力用として使用している場合に限られます。

### [所属]

CG\_timer.c

### [指定形式]

```
void TAB $n$ _SoftwareTriggerOn ( void );
```

**備考**  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## TMT0\_Init

16ビット・タイマ/イベント・カウンタ T (TMT) 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]

CG\_timer.c

### [指定形式]

```
void TMT0_Init ( void );
```

### [引数]

なし

### [戻り値]

なし

## TMT0\_UserInit

16ビット・タイマ/イベント・カウンタ T (TMT) 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、[TMT0\\_Init](#) のコールバック・ルーチンとして呼び出されます。

### [所属]

CG\_timer\_user.c

### [指定形式]

```
void TMT0_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

## TMT0\_Start

16ビット・タイマ/イベント・カウンタ T (TMT) 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、外部イベント・カウンタ、外部トリガ・パルス出力など）により異なります。

### [所属]

CG\_timer.c

### [指定形式]

```
void TMT0_Start ( void );
```

### [引数]

なし

### [戻り値]

なし

## TMT0\_Stop

16ビット・タイマ/イベント・カウンタ T (TMT) のカウントを終了します。

**備考** 本API関数を呼び出してからカウント処理を終了するまでの時間は、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、外部トリガ・パルス出力など）により異なります。

### [所属]

CG\_timer.c

### [指定形式]

```
void TMT0_Stop ( void );
```

### [引数]

なし

### [戻り値]

なし

## TMT0\_ChangeTimerCondition

16ビット・タイマ/イベント・カウンタ T (TMT) のカウント値を変更します。

**備考** 引数 *array\_reg* に指定された値は、TMT0 キャプチャ/コンペア・レジスタ *m* (TT0CCR*m*) に設定されます。

### [所属]

CG\_timer.c

### [指定形式]

```
#include "CG_macrodriver.h"
MD_STATUS TMT0_ChangeTimerCondition ( USHORT *array_reg, UCHAR array_num );
```

### [引数]

I/O	引数	説明
I	USHORT *array_reg;	カウント値 (0x0 ~ 0xffff) を格納した領域へのポインタ
I	UCHAR array_num;	変更対象レジスタ 1: TT0CCR0 2: TT0CCR0, TT0CCR1

### [戻り値]

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

## TMT0\_GetPulseWidth

16ビット・タイマ/イベント・カウンタ T (TMT) のパルス幅 (ハイ・レベル幅, ロウ・レベル幅) を読み出します。

- 備考 1.** 本 API 関数の呼び出しは, 16 ビット・タイマ/イベント・カウンタ T (TMT) をパルス幅測定用として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]

CG\_timer.c

### [指定形式]

```
#include "CG_macrodriver.h"
void TMT0_GetPulseWidth ( ULONG *activewith, ULONG *inactivewidth );
```

### [引数]

I/O	引数	説明
O	ULONG *activewith;	読み出したハイ・レベル幅 (0x0 ~ 0x1ffff) を格納する領域へのポインタ
O	ULONG *inactivewidth;	読み出したロウ・レベル幅 (0x0 ~ 0x1ffff) を格納する領域へのポインタ

### [戻り値]

なし

## TMT0\_GetFreeRunningValue

16 ビット・タイマ/イベント・カウンタ T (TMT) がキャプチャした値を読み出します。

**備考** 本 API 関数の呼び出しは、16 ビット・タイマ/イベント・カウンタ T (TMT) をフリー・ランニング・タイマ用として使用し、TMT0 キャプチャ/コンペア・レジスタ *m* (TT0CCRM) がキャプチャ・レジスタとして選択されている場合に限られます。

### [所属]

CG\_timer.c

### [指定形式]

```
#include "CG_macrodriver.h"
#include "CG_timer.h"
MD_STATUS TMT0_GetFreeRunningValue ( ULONG *count, enum TMChannel channel );
```

### [引数]

I/O	引数	説明
O	ULONG *count;	読み出した幅を格納する領域へのポインタ
I	enum TMChannel channel;	読み出し対象チャネル TMCHANNEL0: チャネル 0 (TT0CCR0) TMCHANNEL1: チャネル 1 (TT0CCR1)

### [戻り値]

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

## TMT0\_ChangeDuty

PWM 信号のデューティ比を変更します。

**備考** 本 API 関数の呼び出しは、16 ビット・タイマ/イベント・カウンタ T (TMT) を外部トリガ・パルス出力/PWM 出力用として使用している場合に限られます。

### [所属]

CG\_timer.c

### [指定形式]

```
#include "CG_macrodriver.h"
MD_STATUS TMT0_ChangeDuty ( UCHAR array_duty );
```

**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	UCHAR <i>array_duty</i> ;	デューティ比 (0 ~ 100, 単位 : %)

**備考** デューティ比 *array\_duty* に設定する値は、10 進数に限られます。

### [戻り値]

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

## TMT0\_SoftwareTriggerOn

タイマ出力のためのトリガ（ソフトウェア・トリガ）を発生させます。

**備考** 本API関数の呼び出しは、16ビット・タイマ/イベント・カウンタT（TMT）を外部トリガ・パルス出力/ワ  
ンショット・パルス出力用として使用している場合に限られます。

### [所属]

CG\_timer.c

### [指定形式]

```
void TMT0_SoftwareTriggerOn ( void );
```

**備考** *n*は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## TMT0\_EnableHold

16 ビット・タイマ/イベント・カウンタ T (TMT) のエンコーダ・カウンタ制御を保持動作へと変更します。

**備考** 本 API 関数の呼び出しは、16 ビット・タイマ/イベント・カウンタ T (TMT) をエンコーダ・カウンタ用として使用している場合に限られます。

### [所属]

CG\_timer.c

### [指定形式]

```
void TMT0_EnableHold ( void );
```

### [引数]

なし

### [戻り値]

なし

## TMT0\_DisableHold

16ビット・タイマ/イベント・カウンタ T (TMT) のエンコーダ・カウンタ制御を通常動作へと変更します。

**備考** 本API関数の呼び出しは、16ビット・タイマ/イベント・カウンタ T (TMT) をエンコーダ・カウンタ用として使用している場合に限られます。

### [所属]

CG\_timer.c

### [指定形式]

```
void TMT0_DisableHold ( void );
```

### [引数]

なし

### [戻り値]

なし

## TMT0\_ChangeCountValue

16 ビット・タイマ/イベント・カウンタ T (TMT) の初期カウント値を変更します。

**備考 1.** 引数 *regvalue* に指定された値は、TMT0 カウンタ・ライト・レジスタ (TT0TCW) に設定されます。

2. 本 API 関数の呼び出しは、16 ビット・タイマ/イベント・カウンタ T (TMT) をエンコーダ・カウント用として使用している場合に限られます。

### [所属]

CG\_timer.c

### [指定形式]

```
#include "CG_macrodriver.h"
void TMT0_ChangeCountValue ( USHORT regvalue );
```

### [引数]

I/O	引数	説明
I	USHORT <i>regvalue</i> ;	カウント値 (0x0 ~ 0xffff)

### [戻り値]

なし

## TMM $n$ \_Init

16 ビット・インターバル・タイマ M (TMM) 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]

CG\_timer.c

### [指定形式]

```
void TMM $n$ _Init ( void );
```

**備考**  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## TMM $n$ \_UserInit

16 ビット・インターバル・タイマ M (TMM) 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、TMM $n$ \_Init のコールバック・ルーチンとして呼び出されます。

### [所属]

CG\_timer\_user.c

### [指定形式]

```
void TMM $n$ _UserInit ( void );
```

**備考**  $n$  は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## TMM $n$ \_Start

16 ビット・インターバル・タイマ M (TMM) のカウントを開始します。

### [所属]

CG\_timer.c

### [指定形式]

```
void TMM $n$ _Start ( void );
```

**備考**  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## TMM $n$ \_Stop

16 ビット・インターバル・タイマ M (TMM) のカウントを終了します。

### [所属]

CG\_timer.c

### [指定形式]

```
void TMM $n$ _Stop ( void );
```

**備考**  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## TMM $n$ \_ChangeTimerCondition

16 ビット・インターバル・タイマ M (TMM) のカウント値を変更します。

**備考** 引数 *regvalue* に指定された値は、TMM $n$  コンペア・レジスタ 0 (TM $n$ CMP0) に設定されます。

### [所属]

CG\_timer.c

### [指定形式]

```
#include "CG_macrodriver.h"
void TMM $n$ _ChangeTimerCondition ( USHORT regvalue );
```

**備考**  $n$  は、チャンネル番号を意味します。

### [引数]

I/O	引数	説明
I	USHORT <i>regvalue</i> ;	カウント値 (0x0 ~ 0xffff)

### [戻り値]

なし

### C.3.9 時計タイマ

以下に、コード生成が時計タイマ用として出力するAPI関数の一覧を示します。

表 C—10 時計タイマ用 API 関数

API 関数名	機能概要
WT_Init	時計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。
WT_UserInit	時計タイマに関するユーザ独自の初期化処理を行います。
WT_Start	時計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。
WT_Stop	時計タイマのカウント処理を終了します。

## WT\_Init

時計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]

CG\_wt.c

### [指定形式]

```
void WT_Init ( void );
```

### [引数]

なし

### [戻り値]

なし

## WT\_UserInit

時計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、WT\_Initのコールバック・ルーチンとして呼び出されます。

### [所属]

CG\_wt\_user.c

### [指定形式]

```
void WT_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

## WT\_Start

時計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]

CG\_wt.c

### [指定形式]

```
void WT_Start ( void );
```

### [引数]

なし

### [戻り値]

なし

## WT\_Stop

時計タイマのカウント処理を終了します。

### [所属]

CG\_wt.c

### [指定形式]

```
void WT_Stop ( void );
```

### [引数]

なし

### [戻り値]

なし

### [使用例]

以下に、時計タイマの機能を利用する際の例を示します。

#### 【CG\_main.c】

```
#include "CG_macrodriver.h"
ULONG INT_flg = 0;
void main ( void ) {
    WT_Start ();          /* カウントの再開 */
    while ( !INT_flg );
    WT_Stop ();          /* カウントの終了 */
    .....
}
```

#### 【CG\_wt\_user.c】

```
#include "CG_macrodriver.h"
extern ULONG INT_flg;
__interrupt void MD_INTWT ( void ) { /* 割り込み INTWT 発生時の割り込み処理 */
    INT_flg = 1;
}
```

### C. 3.10 リアルタイム・カウンタ

以下に、コード生成がリアルタイム・カウンタ用として出力する API 関数の一覧を示します。

表 C—11 リアルタイム・カウンタ用 API 関数

API 関数名	機能概要
RTC_Init	リアルタイム・カウン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。
RTC_UserInit	リアルタイム・カウンタに関するユーザ独自の初期化処理を行います。
RTC_CounterEnable	リアルタイム・カウンタ（年，月，曜日，日，時，分，秒）のカウントを開始します。
RTC_CounterDisable	リアルタイム・カウンタ（年，月，曜日，日，時，分，秒）のカウントを終了します。
RTC_SetHourSystem	リアルタイム・カウンタの時間制（12 時間制，24 時間制）を設定します。
RTC_CounterSet	リアルタイム・カウンタにカウント値（年，月，曜日，日，時，分，秒）を設定します。
RTC_CounterGet	リアルタイム・カウンタのカウント値（年，月，曜日，日，時，分，秒）を読み出します。
RTC_ConstPeriodInterruptEnable	割り込み INTRTC0 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。
RTC_ConstPeriodInterruptDisable	定周期割り込み機能を終了します。
RTC_AlarmEnable	アラーム割り込み機能を開始します。
RTC_AlarmDisable	アラーム割り込み機能を終了します。
RTC_AlarmSet	アラームの発生条件（曜日，時，分）を設定します。
RTC_AlarmGet	アラームの発生条件（曜日，時，分）を読み出します。
RTC_IntervalStart	インターバル割り込み機能を開始します。
RTC_IntervalStop	インターバル割り込み機能を終了します。
RTC_IntervalInterruptEnable	割り込み INTRTC2 の発生周期を設定したのち，インターバル割り込み機能を開始します。
RTC_IntervalInterruptDisable	インターバル割り込み機能を終了します。
RTC_RC1CK1HZ_OutputEnable	RC1CK1HZ 端子に対するリアルタイム・カウンタ補正クロック（1 Hz）の出力を許可します。
RTC_RC1CK1HZ_OutputDisable	RC1CK1HZ 端子に対するリアルタイム・カウンタ補正クロック（1 Hz）の出力を禁止します。
RTC_RC1CKO_OutputEnable	RC1CKO 端子に対するリアルタイム・カウンタ・クロック（32 kHz 原発）の出力を許可します。
RTC_RC1CKO_OutputDisable	RC1CKO 端子に対するリアルタイム・カウンタ・クロック（32 kHz 原発）の出力を禁止します。
RTC_RC1CKDIV_OutputEnable	RC1CKDIV 端子に対するリアルタイム・カウンタ・クロック（32 kHz 分周）の出力を許可します。

API 関数名	機能概要
RTC_RC1CKDIV_OutputDisable	RC1CKDIV 端子に対するリアルタイム・カウンタ・クロック (32 kHz 分周) の出力を禁止します。
RTC_RTC1HZ_OutputEnable	RTC1HZ 端子に対するリアルタイム・カウンタ補正クロック (1 Hz) の出力を許可します。
RTC_RTC1HZ_OutputDisable	RTC1HZ 端子に対するリアルタイム・カウンタ補正クロック (1 Hz) の出力を禁止します。
RTC_RTCCL_OutputEnable	RTCCL 端子に対するリアルタイム・カウンタ・クロック (32 kHz 原発) の出力を許可します。
RTC_RTCCL_OutputDisable	RTCCL 端子に対するリアルタイム・カウンタ・クロック (32 kHz 原発) の出力を禁止します。
RTC_RTCDIV_OutputEnable	RTCDIV 端子に対するリアルタイム・カウンタ・クロック (32 kHz 分周) の出力を許可します。
RTC_RTCDIV_OutputDisable	RTCDIV 端子に対するリアルタイム・カウンタ・クロック (32 kHz 分周) の出力を禁止します。
RTC_ChangeCorrectionValue	時計誤差を補正するタイミング, および補正值を変更します。

## RTC\_Init

リアルタイム・カウン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]

CG\_rtc.c

### [指定形式]

```
void    RTC_Init ( void );
```

### [引数]

なし

### [戻り値]

なし

## RTC\_UserInit

リアルタイム・カウン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、[RTC\\_Init](#)のコールバック・ルーチンとして呼び出されます。

### [所属]

CG\_rtc\_user.c

### [指定形式]

```
void RTC_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

## RTC\_CounterEnable

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

### [所属]

CG\_rtc.c

### [指定形式]

```
void RTC_CounterEnable ( void );
```

### [引数]

なし

### [戻り値]

なし

## RTC\_CounterDisable

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

### [所属]

CG\_rtc.c

### [指定形式]

```
void RTC_CounterDisable ( void );
```

### [引数]

なし

### [戻り値]

なし

## RTC\_SetHourSystem

リアルタイム・カウンタの時間制（12 時間制，24 時間制）を設定します。

### [所属]

CG\_rtc.c

### [指定形式]

```
#include "CG_macrodriver.h"
#include "CG_rtc.h"
MD_STATUS RTC_SetHourSystem ( 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 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。

### [使用例]

以下に、リアルタイム・カウンタの時間制を“24 時間制”に設定する際の例を示します。

【CG\_main.c】

```
#include "CG_macrodriver.h"
#include "CG_rtc.h"
```

```
void main ( void ) {  
    .....  
    RTC_CounterEnable ();          /* カウントの開始 */  
    .....  
    RTC_SetHourSystem ( HOUR24 ); /* 時間制の設定 */  
    .....  
}
```

## RTC\_CounterSet

リアルタイム・カウンタにカウント値を設定します。

### [所属]

CG\_rtc.c

### [指定形式]

```
#include "CG_macrodriver.h"
#include "CG_rtc.h"
MD_STATUS RTC_CounterSet ( 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・カウンタのカウント値 RTCCounterValue の構成を示します。

```
struct RTCCounterValue {
    UCHAR  Sec;    /* 秒 */
    UCHAR  Min;    /* 分 */
    UCHAR  Hour;   /* 時 */
    UCHAR  Day;    /* 日 */
    UCHAR  Week;   /* 曜日 (0:日曜日, 6:土曜日) */
    UCHAR  Month;  /* 月 */
    UCHAR  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 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。

## [使用例]

以下に、リアルタイム・カウンタのカウント値として、“2008年12月25日（木）17時30分00秒”を設定する際の例を示します。

### 【CG\_main.c】

```
#include "CG_macrodriver.h"
#include "CG_rtc.h"
void main ( main ) {
    struct RTCCounterValue counterwriteval;
    .....
    RTC_CounterEnable ();          /* カウントの開始 */
    .....
    counterwriteval.Year = 0x08;
    counterwriteval.Month = 0x12;
    counterwriteval.Day = 0x25;
    counterwriteval.Week = 0x05;
    counterwriteval.Hour = 0x17;
    counterwriteval.Min = 0x30;
    counterwriteval.Sec = 0;
    RTC_SetHourSystem ( HOUR24 ); /* 時間制の設定 */
    RTC_CounterSet ( counterwriteval ); /* カウント値の設定 */
    .....
}
```

## RTC\_CounterGet

リアルタイム・カウンタのカウント値を読み出します。

### [所属]

CG\_rtc.c

### [指定形式]

```
#include "CG_macrodriver.h"
#include "CG_rtc.h"
MD_STATUS RTC_CounterGet ( 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、[RTC\\_CounterSet](#) 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 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。

### [使用例]

以下に、リアルタイム・カウンタのカウント値を読み出す際の例を示します。

【CG\_main.c】

```
#include "CG_macrodriver.h"
#include "CG_rtc.h"
```

```
void main ( void ) {  
    struct RTCCounterValue counterreadval;  
    .....  
    RTC_CounterEnable ();          /* カウントの開始 */  
    .....  
    RTC_CounterGet ( &counterreadval ); /* カウント値の読み出し */  
    .....  
}
```

## RTC\_ConstPeriodInterruptEnable

割り込み INTRTC0 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]

CG\_rtc.c

### [指定形式]

```
#include "CG_macrodriver.h"
#include "CG_rtc.h"
MD_STATUS RTC_ConstPeriodInterruptEnable ( enum RTCINTPeriod period );
```

### [引数]

I/O	引数	説明
I	enum RTCINTPeriod <i>period</i> ;	割り込み INTRTC0 の発生周期 HALFSEC : 0.5 秒 ONESEC : 1 秒 ONEMIN : 1 分 ONEHOUR : 1 時間 ONEDAY : 1 日 ONEMONTH : 1 ヶ月

### [戻り値]

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

### [使用例]

以下に、割り込み INTRTC0 の発生周期を設定したのち、定周期割り込み機能を開始する際の例を示します。

【CG\_main.c】

```
#include "CG_macrodriver.h"
#include "CG_rtc.h"
void main ( void ) {
    .....
    RTC_ConstPeriodInterruptDisable (); /* 定周期割り込み機能の終了 */
```

```
.....  
RTC_ConstPeriodInterruptEnable ( HALFSEC ); /* 定周期割り込み機能の開始 */  
.....  
}
```

## RTC\_ConstPeriodInterruptDisable

定周期割り込み機能を終了します。

### [所属]

CG\_rtc.c

### [指定形式]

```
void    RTC_ConstPeriodInterruptDisable ( void );
```

### [引数]

なし

### [戻り値]

なし

## RTC\_AlarmEnable

アラーム割り込み機能を開始します。

### [所属]

CG\_rtc.c

### [指定形式]

```
void RTC_AlarmEnable ( void );
```

### [引数]

なし

### [戻り値]

なし

## RTC\_AlarmDisable

アラーム割り込み機能を終了します。

### [所属]

CG\_rtc.c

### [指定形式]

```
void    RTC_AlarmDisable ( void );
```

### [引数]

なし

### [戻り値]

なし

## RTC\_AlarmSet

アラームの発生条件（曜日，時，分）を設定します。

### [所属]

CG\_rtc.c

### [指定形式]

```
#include "CG_macrodriver.h"
#include "CG_rtc.h"
void RTC_AlarmSet ( struct RTCArmValue alarmval );
```

### [引数]

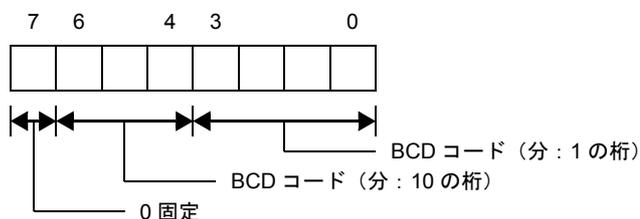
I/O	引数	説明
I	struct RTCArmValue alarmval;	アラームの発生条件（曜日，時，分）

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

```
struct RTCArmValue {
    UCHAR Alarmwm; /* 分 */
    UCHAR Alarmwh; /* 時 */
    UCHAR Alarmmw; /* 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。



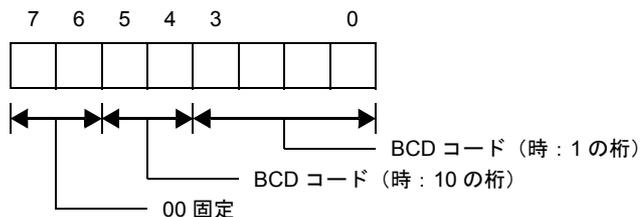
#### - Alarmwh (時)

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・カウンタが 12 時間制の場合，以下の意味となります。

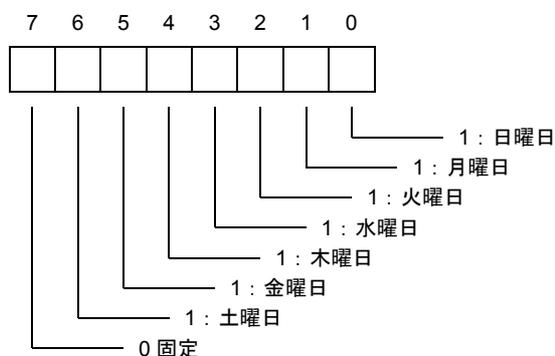
0：午前

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

## [使用例 1]

以下に、アラームの発生条件として、“月曜日／火曜日／水曜日の 17 時 30 分”を設定する際の例を示します。

【CG\_main.c】

```

#include "CG_macrodriver.h"
#include "CG_rtc.h"
void main ( void ) {
    struct RTCArmValue  alarmval;
    .....
    RTC_AlarmEnable ();          /* アラーム割り込み機能の開始 */
    RTC_CounterEnable ();       /* カウントの開始 */
    .....
    RTC_SetHourSystem ( HOUR24 ); /* 時間制の設定 */
    alarmval.Alarmww = 0xe;
    alarmval.Alarmwh = 0x17;
    alarmval.Alarmwm = 0x30;
}

```

```
RTC_AlarmSet ( alarmval );      /* 発生条件の設定 */
.....
}
```

## [使用例 2]

以下に、アラームの発生条件を“土曜日/日曜日（時分はそのまま）”に変更する際の例を示します。

### 【CG\_main.c】

```
#include "CG_macrodriver.h"
#include "CG_rtc.h"
void main ( void ) {
    struct RTCArmValue  alarmval;
    .....

    RTC_AlarmEnable ();          /* アラーム割り込み機能の開始 */
    .....

    alarmval.Alarmmw = 0x41;
    RTC_AlarmSet ( alarmval );   /* 発生条件の変更 */
    .....
}
```

## RTC\_AlarmGet

アラームの発生条件（曜日，時，分）を読み出します。

### [所属]

CG\_rtc.c

### [指定形式]

```
#include "CG_macrodriver.h"
#include "CG_rtc.h"
void RTC_AlarmGet ( struct RTCArmValue *alarmval );
```

**備考** アラームの発生条件 RTCArmValue についての詳細は、[RTC\\_AlarmSet](#) を参照してください。

### [引数]

I/O	引数	説明
O	struct RTCArmValue *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

### [使用例]

以下に、アラームの発生条件を読み出す際の例を示します。

#### 【CG\_main.c】

```
#include "CG_macrodriver.h"
#include "CG_rtc.h"
void main ( void ) {
    struct RTCArmValue alarmval;
    .....
    RTC_AlarmEnable ();          /* アラーム割り込み機能の開始 */
    .....
    RTC_AlarmGet ( &alarmval ); /* 発生条件の読み出し */
    .....
}
```

## RTC\_IntervalStart

インターバル割り込み機能を開始します。

**備考** 割り込み INTRTC2 の発生周期を設定したのち、インターバル割り込み機能を開始する場合は、  
[RTC\\_IntervalInterruptEnable](#) を呼び出します。

### [所属]

CG\_rtc.c

### [指定形式]

```
void RTC_IntervalStart ( void );
```

### [引数]

なし

### [戻り値]

なし

## RTC\_IntervalStop

インターバル割り込み機能を終了します。

### [所属]

CG\_rtc.c

### [指定形式]

```
void RTC_IntervalStop ( void );
```

### [引数]

なし

### [戻り値]

なし

## RTC\_IntervalInterruptEnable

割り込み INTRTC2 の発生周期を設定したのち、インターバル割り込み機能を開始します。

**備考** 割り込み INTRTC2 の発生周期を設定することなく、インターバル割り込み機能を開始する場合は、[RTC\\_IntervalStart](#) を呼び出します。

### [所属]

CG\_rtc.c

### [指定形式]

```
#include "CG_macrodriver.h"
#include "CG_rtc.h"
MD_STATUS RTC_IntervalInterruptEnable ( enum RTCINTInterval interval );
```

### [引数]

I/O	引数	説明
I	enum RTCINTInterval <i>interval</i> ;	割り込み INTRTC2 の発生周期 INTERVAL0 : 2 <sup>6</sup> /fRTC INTERVAL1 : 2 <sup>7</sup> /fRTC INTERVAL2 : 2 <sup>8</sup> /fRTC INTERVAL3 : 2 <sup>9</sup> /fRTC INTERVAL4 : 2 <sup>10</sup> /fRTC INTERVAL5 : 2 <sup>11</sup> /fRTC INTERVAL6 : 2 <sup>12</sup> /fRTC

**備考** fRTC は、サブシステム・クロックの周波数を意味します。

### [戻り値]

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

### [使用例]

以下に、インターバル間隔を変更したのち、インターバル割り込み機能を再開始する際の例を示します。

## 【CG\_main.c】

```
#include "CG_macrodriver.h"
#include "CG_rtc.h"
void main ( void ) {
    .....
    RTC_IntervalStart ();          /* インターバル割り込み機能の開始 */
    .....
    RTC_IntervalStop ();          /* インターバル割り込み機能の終了 */
    .....
    RTC_IntervalInterruptEnable ( INTERVAL6 ); /* インターバル割り込み機能の開始 */
    .....
}
```

## RTC\_IntervalInterruptDisable

インターバル割り込み機能を終了します。

### [所属]

CG\_rtc.c

### [指定形式]

```
void RTC_IntervalInterruptDisable ( void );
```

### [引数]

なし

### [戻り値]

なし

## RTC\_RC1CK1HZ\_OutputEnable

RC1CK1HZ 端子に対するリアルタイム・カウンタ補正クロック（1 Hz）の出力を許可します。

### [所属]

CG\_rtc.c

### [指定形式]

```
void    RCT_RC1CK1HZ_OutputEnable ( void );
```

### [引数]

なし

### [戻り値]

なし

## RTC\_RC1CK1HZ\_OutputDisable

RC1CK1HZ 端子に対するリアルタイム・カウンタ補正クロック (1 Hz) の出力を禁止します。

### [所属]

CG\_rtc.c

### [指定形式]

```
void RTC_RC1CK1HZ_OutputDisable ( void );
```

### [引数]

なし

### [戻り値]

なし

## RTC\_RC1CKO\_OutputEnable

RC1CKO 端子に対するリアルタイム・カウンタ・クロック（32 kHz 原発）の出力を許可します。

### [所属]

CG\_rtc.c

### [指定形式]

```
void RTC_RC1CKO_OutputEnable ( void );
```

### [引数]

なし

### [戻り値]

なし

## RTC\_RC1CKO\_OutputDisable

RC1CKO 端子に対するリアルタイム・カウンタ・クロック（32 kHz 原発）の出力を禁止します。

### [所属]

CG\_rtc.c

### [指定形式]

```
void    RTC_RC1CKO_OutputDisable ( void );
```

### [引数]

なし

### [戻り値]

なし

## RTC\_RC1CKDIV\_OutputEnable

RC1CKDIV 端子に対するリアルタイム・カウンタ・クロック（32 kHz 分周）の出力を許可します。

### [所属]

CG\_rtc.c

### [指定形式]

```
void RTC_RC1CKDIV_OutputEnable ( void );
```

### [引数]

なし

### [戻り値]

なし

## RTC\_RC1CKDIV\_OutputDisable

RC1CKDIV 端子に対するリアルタイム・カウンタ・クロック（32 kHz 分周）の出力を禁止します。

### [所属]

CG\_rtc.c

### [指定形式]

```
void RTC_RC1CKDIV_OutputDisable ( void );
```

### [引数]

なし

### [戻り値]

なし

## RTC\_RTC1HZ\_OutputEnable

RTC1HZ 端子に対するリアルタイム・カウンタ補正クロック（1 Hz）の出力を許可します。

### [所属]

CG\_rtc.c

### [指定形式]

```
void    RCT_RTC1HZ_OutputEnable ( void );
```

### [引数]

なし

### [戻り値]

なし

## RTC\_RTC1HZ\_OutputDisable

RTC1HZ 端子に対するリアルタイム・カウンタ補正クロック（1 Hz）の出力を禁止します。

### [所属]

CG\_rtc.c

### [指定形式]

```
void    RTC_RTC1HZ_OutputDisable ( void );
```

### [引数]

なし

### [戻り値]

なし

## RTC\_RTCCL\_OutputEnable

RTCCL 端子に対するリアルタイム・カウンタ・クロック（32 kHz 原発）の出力を許可します。

### [所属]

CG\_rtc.c

### [指定形式]

```
void RTC_RTCCL_OutputEnable ( void );
```

### [引数]

なし

### [戻り値]

なし

## RTC\_RTCCL\_OutputDisable

RTCCL 端子に対するリアルタイム・カウンタ・クロック（32 kHz 原発）の出力を禁止します。

### [所属]

CG\_rtc.c

### [指定形式]

```
void RTC_RTCCL_OutputDisable ( void );
```

### [引数]

なし

### [戻り値]

なし

## RTC\_RTCDIV\_OutputEnable

RTCDIV 端子に対するリアルタイム・カウンタ・クロック（32 kHz 分周）の出力を許可します。

### [所属]

CG\_rtc.c

### [指定形式]

```
void    RTC_RTCDIV_OutputEnable ( void );
```

### [引数]

なし

### [戻り値]

なし

## RTC\_RTCDIV\_OutputDisable

RTCDIV 端子に対するリアルタイム・カウンタ・クロック（32 kHz 分周）の出力を禁止します。

### [所属]

CG\_rtc.c

### [指定形式]

```
void    RTC_RTCDIV_OutputDisable ( void );
```

### [引数]

なし

### [戻り値]

なし

## RTC\_ChangeCorrectionValue

時計誤差を補正するタイミング、および補正值を変更します。

### [所属]

CG\_rtc.c

### [指定形式]

```
#include "CG_macrodriver.h"
#include "CG_rtc.h"
MD_STATUS RTC_ChangeCorrectionValue ( enum RTCCorectionTiming timing, UCHAR corectval );
```

### [引数]

I/O	引数	説明
I	enum RTCCorectionTiming timing;	時計誤差の補正タイミング EVERY20S: 秒桁が 00, 20, 40 の時 EVERY60S: 秒桁が 00 の時
I	UCHAR corectval;	時計誤差の補正值

**備考** 本 API 関数では、補正值 *corectVal* に 0x0, 0x1, 0x40, または 0x41 が指定された際、時計誤差の補正処理を行いません。

### [戻り値]

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

### C. 3. 11 リアルタイム出力機能

以下に、コード生成がリアルタイム出力機能用として出力する API 関数の一覧を示します。

表 C—12 リアルタイム出力機能用 API 関数

API 関数名	機能概要
RTOn_Init	リアルタイム出力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。
RTOn_UserInit	リアルタイム出力に関するユーザ独自の初期化処理を行います。
RTOn_Enable	リアルタイム出力を許可します。
RTOn_Disable	リアルタイム出力を禁止します。
RTOn_Set2BitsData	リアルタイム出力する 2 ビット・データを設定します。
RTOn_Set4BitsData	リアルタイム出力する 4 ビット・データを設定します。
RTOn_Set6BitsData	リアルタイム出力する 6 ビット・データを設定します。
RTOn_Set8BitsData	リアルタイム出力する 8 ビット・データを設定します。
RTOn_SetHigh2BitsData	リアルタイム出力する上位 2 ビット・データを設定します。
RTOn_SetLow2BitsData	リアルタイム出力する下位 2 ビット・データを設定します。
RTOn_SetHigh4BitsData	リアルタイム出力する上位 4 ビット・データを設定します。
RTOn_SetLow4BitsData	リアルタイム出力する下位 4 ビット・データを設定します。
RTOn_GetValue	リアルタイム出力しているデータを読み出します。

## RTO $n$ \_Init

リアルタイム出力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]

CG\_rto.c

### [指定形式]

```
void RTO $n$ _Init ( void );
```

**備考**  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## RTOn\_UserInit

リアルタイム出力に関するユーザ独自の初期化処理を行います。

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

### [所属]

CG\_rto\_user.c

### [指定形式]

```
void RTOn_UserInit ( void );
```

**備考**  $n$  は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## RTO $n$ \_Enable

リアルタイム出力を許可します。

### [所属]

CG\_rto.c

### [指定形式]

```
void RTO $n$ _Enable ( void );
```

**備考**  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## RTO $n$ \_Disable

リアルタイム出力を禁止します。

### [所属]

CG\_rto.c

### [指定形式]

```
void RTO $n$ _Disable ( void );
```

**備考**  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## RTOn\_Set2BitsData

リアルタイム出力する2ビット・データを設定します。

### [所属]

CG\_rto.c

### [指定形式]

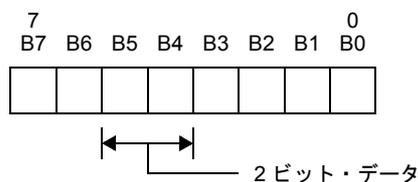
```
#include "CG_macrodriver.h"
void RTOn_Set2BitsData ( UCHAR data );
```

**備考** *n*は、チャンネル番号を意味します。

### [引数]

I/O	引数	説明
I	UCHAR <i>data</i> ;	2ビット・データ

**備考** 本API関数では、4～5ビットに設定された値を2ビット・データとして扱います。



### [戻り値]

なし

## RTOn\_Set4BitsData

リアルタイム出力する4ビット・データを設定します。

### [所属]

CG\_rto.c

### [指定形式]

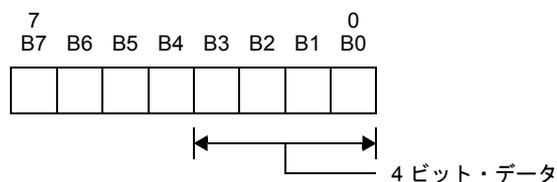
```
#include "CG_macrodriver.h"
void RTOn_Set4BitsData ( UCHAR data );
```

**備考** *n*は、チャンネル番号を意味します。

### [引数]

I/O	引数	説明
I	UCHAR <i>data</i> ;	4ビット・データ

**備考** 本API関数では、0～3ビットに設定された値を4ビット・データとして扱います。



### [戻り値]

なし

## RTOn\_Set6BitsData

リアルタイム出力する6ビット・データを設定します。

### [所属]

CG\_rto.c

### [指定形式]

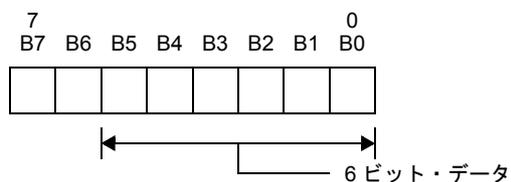
```
#include "CG_macrodriver.h"
void RTOn_Set6BitsData ( UCHAR data );
```

**備考** *n*は、チャンネル番号を意味します。

### [引数]

I/O	引数	説明
I	UCHAR <i>data</i> ;	6ビット・データ

**備考** 本API関数では、0～5ビットに設定された値を6ビット・データとして扱います。



### [戻り値]

なし



## RTOn\_SetHigh2BitsData

リアルタイム出力する上位2ビット・データを設定します。

### [所属]

CG\_rto.c

### [指定形式]

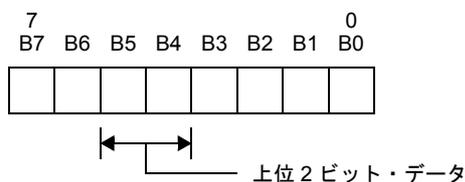
```
#include "CG_macrodriver.h"
void RTOn_SetHigh2BitsData ( UCHAR data );
```

**備考** *n*は、チャンネル番号を意味します。

### [引数]

I/O	引数	説明
I	UCHAR <i>data</i> ;	上位2ビット・データ

**備考** 本API関数では、4～5ビットに設定された値を上位2ビット・データとして扱います。



### [戻り値]

なし

## RTOn\_SetLow2BitsData

リアルタイム出力する下位 2 ビット・データを設定します。

### [所属]

CG\_rto.c

### [指定形式]

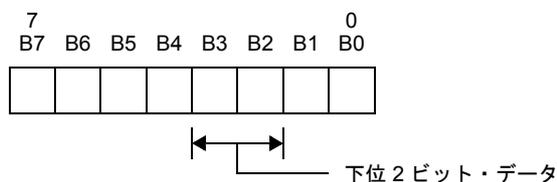
```
#include "CG_macrodriver.h"
void RTOn_SetLow2BitsData ( UCHAR data );
```

**備考**  $n$  は、チャンネル番号を意味します。

### [引数]

I/O	引数	説明
I	UCHAR <i>data</i> ;	下位 2 ビット・データ

**備考** 本 API 関数では、2 ~ 3 ビットに設定された値を下位 2 ビット・データとして扱います。



### [戻り値]

なし

## RTOn\_SetHigh4BitsData

リアルタイム出力する上位4ビット・データを設定します。

### [所属]

CG\_rto.c

### [指定形式]

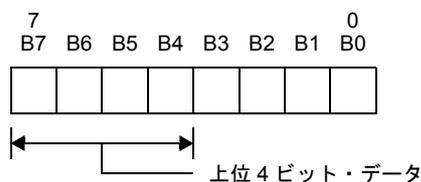
```
#include "CG_macrodriver.h"
void RTOn_SetHigh4BitsData ( UCHAR data );
```

**備考** *n*は、チャンネル番号を意味します。

### [引数]

I/O	引数	説明
I	UCHAR <i>data</i> ;	上位4ビット・データ

**備考** 本API関数では、4～7ビットに設定された値を上位4ビット・データとして扱います。



### [戻り値]

なし

## RTOn\_SetLow4BitsData

リアルタイム出力する下位4ビット・データを設定します。

### [所属]

CG\_rto.c

### [指定形式]

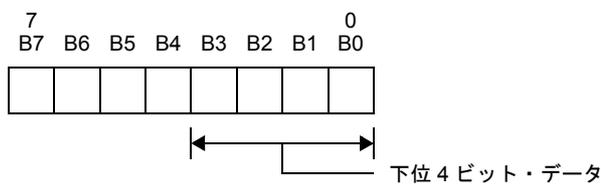
```
#include "CG_macrodriver.h"
void RTOn_SetLow4BitsData ( UCHAR data );
```

**備考** *n*は、チャンネル番号を意味します。

### [引数]

I/O	引数	説明
I	UCHAR <i>data</i> ;	下位4ビット・データ

**備考** 本API関数では、0～3ビットに設定された値を下位4ビット・データとして扱います。



### [戻り値]

なし

## RTOn\_GetValue

リアルタイム出力している値を読み出します。

### [所属]

CG\_rto.c

### [指定形式]

```
#include "CG_macrodriver.h"
void RTOn_GetValue ( UCHAR *value );
```

**備考** *n*は、チャンネル番号を意味します。

### [引数]

I/O	引数	説明
O	UCHAR *value;	読み出した値を格納する領域へのポインタ

### [戻り値]

なし

### [使用例]

以下に、リアルタイム・カウンタのカウント値を読み出す際の例を示します。

#### 【CG\_main.c】

```
#include "CG_macrodriver.h"
void main ( void ) {
    RT00_Set2BitData ( 0x30 );          /* 出力データの設定 */
    RT00_Enable ();                    /* リアルタイム出力の許可 */
    .....
    RT00_Disable ();                   /* リアルタイム出力の禁止 */
    .....
}
```

#### 【CG\_timer\_user.c】

```
#include "CG_macrodriver.h"
```

```
__interrupt void MD_INTTP4CC0 ( void ) { /* 割り込み INTTP4CC0 発生時の割り込み処理 */
    UCHAR    value = 0;
    RT00_GetValue ( &value );          /* 出力データの読み出し */
    value = ~value;
    RT00_Set2BitData ( value );       /* 出力データの設定 */
}
```

### C. 3.12 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—13 DMA コントローラ用 API 関数

API 関数名	機能概要
DMAn_Init	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。
DMAn_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。
DMAn_Enable	チャンネル <i>n</i> を動作許可状態に設定します。
DMAn_Disable	チャンネル <i>n</i> を動作停止状態に設定します。
DMAn_CheckStatus	転送状態（転送終了，転送中）を読み出します。
DMAn_SetData	転送先／転送元の RAM アドレス，およびデータの転送回数を設定します。
DMAn_SoftwareTriggerOn	DMA 転送の起動要因として，ソフトウェア・トリガを使用します。

## DMAn\_Init

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]

CG\_dma.c

### [指定形式]

```
void    DMAn_Init ( void );
```

**備考** *n*は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## DMAn\_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、[DMAn\\_Init](#) のコールバック・ルーチンとして呼び出されます。

### [所属]

CG\_dma\_user.c

### [指定形式]

```
void    DMAn_UserInit ( void );
```

**備考** *n* は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## DMAn\_Enable

チャンネル  $n$  を動作許可状態に設定します。

### [所属]

CG\_dma.c

### [指定形式]

```
void    DMAn_Enable ( void );
```

**備考**  $n$  は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

## DMAn\_Disable

チャンネル  $n$  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、`DMAn_CheckStatus` による“転送終了”の確認後に呼び出す必要があります。

### [所属]

CG\_dma.c

### [指定形式]

```
void DMAn_Disable ( void );
```

備考  $n$  は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

### [使用例]

以下に、チャンネル 0 の動作モードを“動作停止状態”へと移行させる際の例を示します。

【CG\_main.c】

```
#include "CG_macrodriver.h"
void main ( void ) {
    .....
    while ( MD_COMPLETED == DMA0_CheckStatus ( ) ); /* 転送状態の確認 */
    DMA0_Disable ( ); /* 動作停止状態への移行 */
    .....
}
```

## DMAn\_CheckStatus

転送状態（転送終了，転送中）を読み出します。

### [所属]

CG\_dma.c

### [指定形式]

```
#include "CG_macrodriver.h"
MD_STATUS DMAn_CheckStatus ( void );
```

備考 *n*は，チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

マクロ	説明
MD_UNDEREXEC	転送中
MD_COMPLETED	転送終了

## DMAn\_SetData

転送先／転送元の RAM アドレス， およびデータの転送回数を設定します。

**備考** 本 API 関数を転送中に呼び出した場合， 転送は強制終了します。

### [所属]

CG\_dma.c

### [指定形式]

```
#include "CG_macrodriver.h"
MD_STATUS DMAn_SetData ( UINT srcaddr, UINT dstaddr, UINT count );
```

**備考** *n* は， チャンネル番号を意味します。

### [引数]

I/O	引数	説明
I	UINT <i>srcaddr</i> ;	転送元の RAM アドレス
I	UINT <i>dstaddr</i> ;	転送先の RAM アドレス
I	UINT <i>count</i> ;	データの転送回数 (1 ~ 1024)

### [戻り値]

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

## DMA $n$ \_SoftwareTriggerOn

DMA 転送の起動要因として、ソフトウェア・トリガ（本 API 関数の呼び出し）を使用します。

### [所属]

CG\_dma.c

### [指定形式]

```
void DMA $n$ _SoftwareTriggerOn ( void );
```

**備考**  $n$ は、チャンネル番号を意味します。

### [引数]

なし

### [戻り値]

なし

### [使用例]

以下に、DMA 転送の起動要因をソフトウェア・トリガとした場合の例を示します。

【CG\_main.c】

```
void main ( void ) {  
    .....  
    DMA0_Enable ();          /* 動作許可状態への移行 */  
    DMA0_SoftwareTriggerOn (); /* DMA 転送の開始 */  
    .....  
}
```

### C. 3. 13 低電圧検出回路

以下に、コード生成が低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—14 低電圧検出回路用 API 関数

API 関数名	機能概要
LVI_Init	低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。
LVI_UserInit	低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。
LVI_InterruptModeStart	低電圧検出動作を開始します（割り込み発生モード時）。
LVI_ResetModeStart	低電圧検出動作を開始します（内部リセット・モード時）。
LVI_Start	低電圧検出動作を開始します。
LVI_Stop	低電圧検出動作を停止します。

## LVI\_Init

低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]

CG\_lvi.c

### [指定形式]

```
void LVI_Init ( void );
```

### [引数]

なし

### [戻り値]

なし

## LVI\_UserInit

低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、[LVI\\_Init](#)のコールバック・ルーチンとして呼び出されます。

### [所属]

CG\_lvi\_user.c

### [指定形式]

```
void LVI_UserInit ( void );
```

### [引数]

なし

### [戻り値]

なし

## LVI\_InterruptModeStart

低電圧検出動作を開始します（割り込み発生モード時）。

### [所属]

CG\_lvi.c

### [指定形式]

```
void LVI_InterruptModeStart ( void );
```

### [引数]

なし

### [戻り値]

なし

### [使用例]

以下に、低電圧を検出した際の動作モードが割り込み発生モード（割り込み INTLVI を発生させる）における例を示します。

#### 【CG\_main.c】

```
void main ( void ) {  
    .....  
    LVI_InterruptModeStart ( );          /* 低電圧検出動作の開始 */  
    .....  
}
```

#### 【CG\_lvi\_user.c】

```
__interrupt void MD_INTLVI ( void ) { /* 割り込み INTLVI 発生時の割り込み処理 */  
    if ( LVIF == 1 ) {                /* 発生要因の判別：LVIF フラグのチェック */  
        ..... /* “電源電圧 (VDD) < 検出電圧 (VLVI)” を検出した際の処理 */  
    } else {  
        ..... /* “電源電圧 (VDD) ≥ 検出電圧 (VLVI)” を検出した際の処理 */  
    }  
}
```

## LVI\_ResetModeStart

低電圧検出動作を開始します（内部リセット・モード時）。

### [所属]

CG\_lvi.c

### [指定形式]

```
#include "CG_macrodriver.h"
MD_STATUS LVI_ResetModeStart ( void );
```

### [引数]

なし

### [戻り値]

マクロ	説明
MD_OK	正常終了
MD_ERROR	異常終了 - 低電圧検出回路の機能を使用しない設定が行われている - 低電圧検出対象が外部電圧 (V <sub>DD</sub> ) の際、電源電圧 (V <sub>DD</sub> ) ≤ 検出電圧 (V <sub>LVI</sub> ) - 低電圧検出対象が外部入力電圧 (EXLVI) の際、外部入力電圧 (EXLVI) ≤ 検出電圧 (V <sub>EXLVI</sub> )

## LVI\_Start

低電圧検出動作を開始します。

### [所属]

CG\_lvi.c

### [指定形式]

```
#include "CG_macrodriver.h"
MD_STATUS LVI_Start ( void );
```

### [引数]

なし

### [戻り値]

マクロ	説明
MD_OK	正常終了
MD_ARGERROR	異常終了 - 低電圧検出回路の機能を使用しない設定が行われている - 低電圧検出対象が外部電圧 (V <sub>DD</sub> ) の際, 電源電圧 (V <sub>DD</sub> ) ≤ 検出電圧 (V <sub>LVI</sub> ) - 低電圧検出対象が外部入力電圧 (EXLVI) の際, 外部入力電圧 (EXLVI) ≤ 検出電圧 (V <sub>EXLVI</sub> )

## LVI\_Stop

低電圧検出動作を停止します。

### [所属]

CG\_lvi.c

### [指定形式]

```
void LVI_Stop ( void );
```

### [引数]

なし

### [戻り値]

なし

## 付録D 索引

## 【A】

AD\_Init ... 229  
 AD\_Read ... 236  
 AD\_ReadByte ... 237  
 AD\_SelectADChannel ... 234  
 AD\_SetPFTCondition ... 235  
 AD\_Start ... 231  
 AD\_Stop ... 233  
 AD\_UserInit ... 230  
 A/D コンバータ ... 228  
   AD\_Init ... 229  
   AD\_Read ... 236  
   AD\_ReadByte ... 237  
   AD\_SelectADChannel ... 234  
   AD\_SetPFTCondition ... 235  
   AD\_Start ... 231  
   AD\_Stop ... 233  
   AD\_UserInit ... 230  
 API 関数 ... 97  
   A/D コンバータ ... 228  
   D/A コンバータ ... 238  
   DMA コントローラ ... 366  
   外部バス ... 126  
   システム ... 111  
   シリアル ... 146  
   タイマ ... 245  
   低電圧検出回路 ... 374  
   時計タイマ ... 307  
   ポート ... 129  
   リアルタイム・カウンタ ... 312  
   リアルタイム出力機能 ... 351  
   割り込み ... 135

## 【B】

BUS\_Init ... 127  
 BUS\_UserInit ... 128

## 【C】

CG\_ChangeClockMode ... 116  
 CG\_ChangeFrequency ... 117  
 CG\_ReadResetSource ... 114  
 CG\_SelectPIIMode ... 120  
 CG\_SelectPowerSaveMode ... 118  
 CG\_SelectSSCGMode ... 121  
 CG\_SelectStabTime ... 119  
 CLOCK\_Init ... 112  
 CLOCK\_UserInit ... 113  
 CRC\_GetResult ... 125  
 CRC\_SetData ... 124  
 CRC\_Start ... 123  
 CSIBn\_ErrorCallback ... 190  
 CSIBn\_Init ... 181  
 CSIBn\_ReceiveData ... 186  
 CSIBn\_ReceiveEndCallback ... 189  
 CSIBn\_SendData ... 185  
 CSIBn\_SendReceiveData ... 187  
 CSIBn\_SendEndCallback ... 188  
 CSIBn\_Start ... 183  
 CSIBn\_Stop ... 184  
 CSIBn\_UserInit ... 182  
 CSIEn\_ErrorCallback ... 200  
 CSIEn\_Init ... 191  
 CSIEn\_ReceiveData ... 196  
 CSIEn\_ReceiveEndCallback ... 199  
 CSIEn\_SendData ... 195  
 CSIEn\_SendReceiveData ... 197  
 CSIEn\_SendEndCallback ... 198  
 CSIEn\_Start ... 193  
 CSIEn\_Stop ... 194  
 CSIEn\_UserInit ... 192  
 CSIFn\_ErrorCallback ... 210  
 CSIFn\_ReceiveData ... 206  
 CSIFn\_ReceiveEndCallback ... 209  
 CSIFn\_SendData ... 205  
 CSIFn\_SendReceiveData ... 207

CSIFn\_SendEndCallback ... 208  
 CSIFn\_Start ... 203  
 CSIFn\_Stop ... 204  
 CSIFn\_UserInit ... 202  
 CSIFn\_Init ... 201

**[D]**

DAn\_Init ... 239  
 DAn\_SetValue ... 243  
 DAn\_Start ... 241  
 DAn\_Stop ... 242  
 DAn\_UserInit ... 240  
 D/A コンバータ ... 238  
   DAn\_Init ... 239  
   DAn\_SetValue ... 243  
   DAn\_Start ... 241  
   DAn\_Stop ... 242  
   DAn\_UserInit ... 240  
 DMAAn\_CheckStatus ... 371  
 DMAAn\_Disable ... 370  
 DMAAn\_Enable ... 369  
 DMAAn\_Init ... 367  
 DMAAn\_SetData ... 372  
 DMAAn\_SoftwareTriggerOn ... 373  
 DMAAn\_UserInit ... 368  
 DMA コントローラ ... 366  
   DMAAn\_CheckStatus ... 371  
   DMAAn\_Disable ... 370  
   DMAAn\_Enable ... 369  
   DMAAn\_Init ... 367  
   DMAAn\_SetData ... 372  
   DMAAn\_SoftwareTriggerOn ... 373  
   DMAAn\_UserInit ... 368

**[I]**

IIC0n\_GetStopConditionCallback ... 227  
 IIC0n\_Init ... 211  
 IIC0n\_MasterErrorCallback ... 221  
 IIC0n\_MasterReceiveEndCallback ... 220  
 IIC0n\_MasterReceiveStart ... 217  
 IIC0n\_MasterSendEndCallback ... 219

IIC0n\_MasterSendStart ... 215  
 IIC0n\_SlaveErrorCallback ... 226  
 IIC0n\_SlaveReceiveEndCallback ... 225  
 IIC0n\_SlaveSendEndCallback ... 224  
 IIC0n\_SlaveSendStart ... 222  
 IIC0n\_SlaveReceiveStart ... 223  
 IIC0n\_Stop ... 213  
 IIC0n\_StopCondition ... 214  
 IIC0n\_UserInit ... 212  
 INT\_MaskableInterruptEnable ... 140  
 INTP\_Init ... 136  
 INTPn\_Disable ... 142  
 INTPn\_Enable ... 143  
 INTP\_UserInit ... 137

**[K]**

KEY\_Disable ... 144  
 KEY\_Enable ... 145  
 KEY\_Init ... 138  
 KEY\_UserInit ... 139

**[L]**

LVI\_Init ... 375  
 LVI\_InterruptModeStart ... 377  
 LVI\_ResetModeStart ... 378  
 LVI\_Start ... 379  
 LVI\_Stop ... 380  
 LVI\_UserInit ... 376

**[P]**

PORT\_ChangePmnInput ... 132  
 PORT\_ChangePmnOutput ... 133  
 PORT\_Init ... 130  
 PORT\_UserInit ... 131

**[R]**

RTC\_AlarmDisable ... 328  
 RTC\_AlarmEnable ... 327  
 RTC\_AlarmGet ... 332  
 RTC\_AlarmSet ... 329  
 RTC\_ConstPeriodInterruptDisable ... 326  
 RTC\_ConstPeriodInterruptEnable ... 324

RTC_CounterDisable	...	317	TAAAn_GetFreeRunningValue	...	277
RTC_CounterEnable	...	316	TAAAn_GetPulseWidth	...	276
RTC_CounterGet	...	322	TAAAn_Init	...	270
RTC_CounterSet	...	320	TAAAn_SoftwareTriggerOn	...	279
RTC_Init	...	314	TAAAn_Start	...	272
RTC_IntervallInterruptDisable	...	337	TAAAn_Stop	...	273
RTC_IntervallInterruptEnable	...	335	TAAAn_UserInit	...	271
RTC_IntervalStart	...	333	TABn_ChangeDuty	...	288
RTC_IntervalStop	...	334	TABn_ChangeTimerCondition	...	284
RTC_RC1CK1HZ_OutputDisable	...	339	TABn_ControlOutputToggle	...	285
RTC_RC1CK1HZ_OutputEnable	...	338	TABn_GetFreeRunningValue	...	287
RTC_RC1CKDIV_OutputDisable	...	343	TABn_GetPulseWidth	...	286
RTC_RC1CKDIV_OutputEnable	...	342	TABn_Init	...	280
RTC_RC1CKO_OutputDisable	...	341	TABn_SoftwareTriggerOn	...	289
RTC_RC1CKO_OutputEnable	...	340	TABn_Start	...	282
RTC_RTC1HZ_OutputDisable	...	345	TABn_Stop	...	283
RTC_RTC1HZ_OutputEnable	...	344	TABn_UserInit	...	281
RTC_RTCCL_OutputDisable	...	347	TMMn_Init	...	302
RTC_RTCCL_OutputEnable	...	346	TMMn_Start	...	304
RTC_RTCDIV_OutputDisable	...	349	TMMn_Stop	...	305
RTC_RTCDIV_OutputEnable	...	348	TMMn_UserInit	...	303
RTC_SetHourSystem	...	318	TMPn_ChangeDuty	...	256
RTC_UserInit	...	315	TMPn_ChangeTimerCondition	...	252
RTOOn_Disable	...	355	TMPn_GetFreeRunningValue	...	255
RTOOn_Enable	...	354	TMPn_GetPulseWidth	...	254
RTOOn_GetValue	...	364	TMPn_Init	...	248
RTOOn_Init	...	352	TMPn_SoftwareTriggerOn	...	258
RTOOn_Set2BitsData	...	356	TMPn_Start	...	250
RTOOn_Set4BitsData	...	357	TMPn_Stop	...	251
RTOOn_Set6BitsData	...	358	TMPn_UserInit	...	249
RTOOn_Set8BitsData	...	359	TMQ0_ChangeDuty	...	267
RTOOn_SetHigh2BitsData	...	360	TMQ0_ChangeTimerCondition	...	263
RTOOn_SetHigh4BitsData	...	362	TMQ0_GetFreeRunningValue	...	266
RTOOn_SetLow2BitsData	...	361	TMQ0_GetPulseWidth	...	265
RTOOn_SetLow4BitsData	...	363	TMQ0_Init	...	259
RTOOn_UserInit	...	353	TMQ0_SoftwareTriggerOn	...	269
			TMQ0_Start	...	261
			TMQ0_Stop	...	262
			TMQ0_UserInit	...	260
			TMT0_ChangeCountValue	...	301
			TMT0_ChangeDuty	...	297

**[T]**

TAAAn_ChangeDuty	...	278
TAAAn_ChangeTimerCondition	...	274
TAAAn_ControlOutputToggle	...	275

TMT0\_ChangeTimerCondition ... 294  
 TMT0\_DisableHold ... 300  
 TMT0\_EnableHold ... 299  
 TMT0\_GetFreeRunningValue ... 296  
 TMT0\_GetPulseWidth ... 295  
 TMT0\_Init ... 290  
 TMT0\_SoftwareTriggerOn ... 298  
 TMT0\_Start ... 292  
 TMT0\_Stop ... 293  
 TMT0\_UserInit ... 291

**【U】**

UARTAn\_ErrorCallback ... 157  
 UARTAn\_Init ... 149  
 UARTAn\_ReceiveData ... 154  
 UARTAn\_ReceiveEndCallback ... 156  
 UARTAn\_SendData ... 153  
 UARTAn\_SendEndCallback ... 155  
 UARTAn\_SoftOverRunCallback ... 158  
 UARTAn\_Start ... 151  
 UARTAn\_Stop ... 152  
 UARTAn\_UserInit ... 150  
 UARTBn\_FIFOErrorCallback ... 168  
 UARTBn\_Init ... 159  
 UARTBn\_ReceiveData ... 164  
 UARTBn\_ReceiveEndCallback ... 166  
 UARTBn\_SendData ... 163  
 UARTBn\_SendEndCallback ... 165  
 UARTBn\_SingleErrorCallback ... 167  
 UARTBn\_SoftOverRunCallback ... 170  
 UARTBn\_Start ... 161  
 UARTBn\_Stop ... 162  
 UARTBn\_TimeoutErrorCallback ... 169  
 UARTBn\_UserInit ... 160  
 UARTCn\_ErrorCallback ... 179  
 UARTCn\_Init ... 171  
 UARTCn\_ReceiveData ... 176  
 UARTCn\_ReceiveEndCallback ... 178  
 UARTCn\_SendData ... 175  
 UARTCn\_SendEndCallback ... 177  
 UARTCn\_SoftOverRunCallback ... 180

UARTCn\_Start ... 173  
 UARTCn\_Stop ... 174  
 UARTCn\_UserInit ... 172

**【W】**

WDT2\_Restart ... 122  
 WT\_Init ... 308  
 WT\_Start ... 310  
 WT\_Stop ... 311  
 WT\_UserInit ... 309

**【あ行】**

新しい列 ダイアログ ... 84  
 ウィンドウ・リファレンス ... 38

**【か行】**

[外部周辺] タブ ... 66  
 外部バス ... 126  
     BUS\_Init ... 127  
     BUS\_UserInit ... 128  
 機能 (コード生成) ... 22  
 機能 (端子配置) ... 8  
 コード生成 パネル ... 71  
 コード生成プレビュー パネル ... 74

**【さ行】**

システム ... 111  
     CG\_ChangeClockMode ... 116  
     CG\_ChangeFrequency ... 117  
     CG\_ReadResetSource ... 114  
     CG\_SelectPIIMode ... 120  
     CG\_SelectPowerSaveMode ... 118  
     CG\_SelectSSCGMode ... 121  
     CG\_SelectStabTime ... 119  
 CLOCK\_Init ... 112  
 CLOCK\_UserInit ... 113  
 CRC\_GetResult ... 125  
 CRC\_SetData ... 124  
 CRC\_Start ... 123  
 WDT2\_Restart ... 122  
 [出力設定] タブ ... 54  
 出力パネル ... 77

シリアル … 146	IIC0n_SlaveSendStart … 222
CSIBn_ErrorCallback … 190	IIC0n_SlaveReceiveStart … 223
CSIBn_Init … 181	IIC0n_Stop … 213
CSIBn_ReceiveData … 186	IIC0n_StopCondition … 214
CSIBn_ReceiveEndCallback … 189	IIC0n_UserInit … 212
CSIBn_SendData … 185	UARTAn_ErrorCallback … 157
CSIBn_SendReceiveData … 187	UARTAn_Init … 149
CSIBn_SendEndCallback … 188	UARTAn_ReceiveData … 154
CSIBn_Start … 183	UARTAn_ReceiveEndCallback … 156
CSIBn_Stop … 184	UARTAn_SendData … 153
CSIBn_UserInit … 182	UARTAn_SendEndCallback … 155
CSIEn_ErrorCallback … 200	UARTAn_SoftOverRunCallback … 158
CSIEn_Init … 191	UARTAn_Start … 151
CSIEn_ReceiveData … 196	UARTAn_Stop … 152
CSIEn_ReceiveEndCallback … 199	UARTAn_UserInit … 150
CSIEn_SendData … 195	UARTBn_FIFOErrorCallback … 168
CSIEn_SendReceiveData … 197	UARTBn_Init … 159
CSIEn_SendEndCallback … 198	UARTBn_ReceiveData … 164
CSIEn_Start … 193	UARTBn_ReceiveEndCallback … 166
CSIEn_Stop … 194	UARTBn_SendData … 163
CSIEn_UserInit … 192	UARTBn_SendEndCallback … 165
CSIFn_ErrorCallback … 210	UARTBn_SingleErrorCallback … 167
CSIFn_Init … 201	UARTBn_SoftOverRunCallback … 170
CSIFn_ReceiveData … 206	UARTBn_Start … 161
CSIFn_ReceiveEndCallback … 209	UARTBn_Stop … 162
CSIFn_SendData … 205	UARTBn_TimeoutErrorCallback … 169
CSIFn_SendReceiveData … 207	UARTBn_UserInit … 160
CSIFn_SendEndCallback … 208	UARTCn_ErrorCallback … 179
CSIFn_Start … 203	UARTCn_Init … 171
CSIFn_Stop … 204	UARTCn_ReceiveData … 176
CSIFn_UserInit … 202	UARTCn_ReceiveEndCallback … 178
IIC0n_GetStopConditionCallback … 227	UARTCn_SendData … 175
IIC0n_Init … 211	UARTCn_SendEndCallback … 177
IIC0n_MasterErrorCallback … 221	UARTCn_SoftOverRunCallback … 180
IIC0n_MasterReceiveEndCallback … 220	UARTCn_Start … 173
IIC0n_MasterReceiveStart … 217	UARTCn_Stop … 174
IIC0n_MasterSendEndCallback … 219	UARTCn_UserInit … 172
IIC0n_MasterSendStart … 215	
IIC0n_SlaveErrorCallback … 226	<b>【た行】</b>
IIC0n_SlaveReceiveEndCallback … 225	タイマ … 245
IIC0n_SlaveSendEndCallback … 224	TAA_n_ChangeDuty … 278

- TAA<sub>n</sub>\_ChangeTimerCondition ... 274  
 TAA<sub>n</sub>\_ControlOutputToggle ... 275  
 TAA<sub>n</sub>\_GetFreeRunningValue ... 277  
 TAA<sub>n</sub>\_GetPulseWidth ... 276  
 TAA<sub>n</sub>\_Init ... 270  
 TAA<sub>n</sub>\_SoftwareTriggerOn ... 279  
 TAA<sub>n</sub>\_Start ... 272  
 TAA<sub>n</sub>\_Stop ... 273  
 TAA<sub>n</sub>\_UserInit ... 271  
 TAB<sub>n</sub>\_ChangeDuty ... 288  
 TAB<sub>n</sub>\_ChangeTimerCondition ... 284  
 TAB<sub>n</sub>\_ControlOutputToggle ... 285  
 TAB<sub>n</sub>\_GetFreeRunningValue ... 287  
 TAB<sub>n</sub>\_GetPulseWidth ... 286  
 TAB<sub>n</sub>\_Init ... 280  
 TAB<sub>n</sub>\_SoftwareTriggerOn ... 289  
 TAB<sub>n</sub>\_Start ... 282  
 TAB<sub>n</sub>\_Stop ... 283  
 TAB<sub>n</sub>\_UserInit ... 281  
 TMM<sub>n</sub>\_Init ... 302  
 TMM<sub>n</sub>\_Start ... 304  
 TMM<sub>n</sub>\_Stop ... 305  
 TMM<sub>n</sub>\_UserInit ... 303  
 TMP<sub>n</sub>\_ChangeDuty ... 256  
 TMP<sub>n</sub>\_ChangeTimerCondition ... 252  
 TMP<sub>n</sub>\_GetFreeRunningValue ... 255  
 TMP<sub>n</sub>\_GetPulseWidth ... 254  
 TMP<sub>n</sub>\_Init ... 248  
 TMP<sub>n</sub>\_SoftwareTriggerOn ... 258  
 TMP<sub>n</sub>\_Start ... 250  
 TMP<sub>n</sub>\_Stop ... 251  
 TMP<sub>n</sub>\_UserInit ... 249  
 TMQ<sub>0</sub>\_ChangeDuty ... 267  
 TMQ<sub>0</sub>\_ChangeTimerCondition ... 263  
 TMQ<sub>0</sub>\_GetFreeRunningValue ... 266  
 TMQ<sub>0</sub>\_GetPulseWidth ... 265  
 TMQ<sub>0</sub>\_Init ... 259  
 TMQ<sub>0</sub>\_SoftwareTriggerOn ... 269  
 TMQ<sub>0</sub>\_Start ... 261  
 TMQ<sub>0</sub>\_Stop ... 262  
 TMQ<sub>0</sub>\_UserInit ... 260  
 TMT<sub>0</sub>\_ChangeCountValue ... 301  
 TMT<sub>0</sub>\_ChangeDuty ... 297  
 TMT<sub>0</sub>\_ChangeTimerCondition ... 294  
 TMT<sub>0</sub>\_DisableHold ... 300  
 TMT<sub>0</sub>\_EnableHold ... 299  
 TMT<sub>0</sub>\_GetFreeRunningValue ... 296  
 TMT<sub>0</sub>\_GetPulseWidth ... 295  
 TMT<sub>0</sub>\_Init ... 290  
 TMT<sub>0</sub>\_SoftwareTriggerOn ... 298  
 TMT<sub>0</sub>\_Start ... 292  
 TMT<sub>0</sub>\_Stop ... 293  
 TMT<sub>0</sub>\_UserInit ... 291  
 [端子配置図の設定] タブ ... 51  
 端子配置図 パネル ... 68  
 [端子配置の設定] タブ ... 48  
 [端子配置表の情報] タブ ... 50  
 端子配置表 パネル ... 60  
     [外部周辺] タブ ... 66  
     [端子番号] タブ ... 62  
     [マクロ] タブ ... 64  
 [端子番号] タブ ... 62  
 低電圧検出回路 ... 374  
     LVI\_Init ... 375  
     LVI\_InterruptModeStart ... 377  
     LVI\_ResetModeStart ... 378  
     LVI\_Start ... 379  
     LVI\_Stop ... 380  
     LVI\_UserInit ... 376  
 時計タイマ ... 307  
     WT\_Init ... 308  
     WT\_Start ... 310  
     WT\_Stop ... 311  
     WT\_UserInit ... 309  
**【な行】**  
 名前を付けて保存 ダイアログ ... 87  
**【は行】**  
 [ファイル設定] タブ ... 58  
 フォルダの参照 ダイアログ ... 86  
 プロジェクト・ツリー パネル ... 42

- プロパティ パネル … 45
  - [出力設定] タブ … 54
  - [端子配置図の設定] タブ … 51
  - [端子配置の設定] タブ … 48
  - [端子配置表の情報] タブ … 50
  - [ファイル設定] タブ … 58
  - [マクロ設定] タブ … 57
- ポート … 129
  - PORT\_ChangePmnInput … 132
  - PORT\_ChangePmnOutput … 133
  - PORT\_Init … 130
  - PORT\_UserInit … 131
- 【ま行】**
  - [マクロ設定] タブ … 57
  - [マクロ] タブ … 64
  - メイン・ウインドウ … 39
- 【ら行】**
  - リアルタイム・カウンタ … 312
    - RTC\_AlarmDisable … 328
    - RTC\_AlarmEnable … 327
    - RTC\_AlarmGet … 332
    - RTC\_AlarmSet … 329
    - RTC\_ConstPeriodInterruptDisable … 326
    - RTC\_ConstPeriodInterruptEnable … 324
    - RTC\_CounterEnable … 316
    - RTC\_CounterGet … 322
    - RTC\_Disable … 317
    - RTC\_Init … 314
    - RTC\_IntervalInterruptDisable … 337
    - RTC\_IntervalInterruptEnable … 335
    - RTC\_IntervalStart … 333
    - RTC\_IntervalStop … 334
    - RTC\_RC1CK1HZ\_OutputDisable … 339
    - RTC\_RC1CK1HZ\_OutputEnable … 338
    - RTC\_RC1CKDIV\_OutputDisable … 343
    - RTC\_RC1CKDIV\_OutputEnable … 342
    - RTC\_RC1CKO\_OutputDisable … 341
    - RTC\_RC1CKO\_OutputEnable … 340
    - RTC\_RTC1HZ\_OutputDisable … 345
    - RTC\_RTC1HZ\_OutputEnable … 344
    - RTC\_RTCCL\_OutputDisable … 347
    - RTC\_RTCCL\_OutputEnable … 346
    - RTC\_RTCDIV\_OutputDisable … 349
    - RTC\_RTCDIV\_OutputEnable … 348
    - RTC\_SetHourSystem … 318
    - RTC\_UserInit … 315
    - RTC\_CounterSet … 320
  - リアルタイム出力機能 … 351
    - RTOOn\_Disable … 355
    - RTOOn\_Enable … 354
    - RTOOn\_GetValue … 364
    - RTOOn\_Init … 352
    - RTOOn\_Set2BitsData … 356
    - RTOOn\_Set4BitsData … 357
    - RTOOn\_Set6BitsData … 358
    - RTOOn\_Set8BitsData … 359
    - RTOOn\_SetHigh2BitsData … 360
    - RTOOn\_SetHigh4BitsData … 362
    - RTOOn\_SetLow2BitsData … 361
    - RTOOn\_SetLow4BitsData … 363
    - RTOOn\_UserInit … 353
  - 列の選択 ダイアログ … 80
- 【わ行】**
  - 割り込み … 135
    - INT\_MaskableInterruptEnable … 140
    - INTP\_Init … 136
    - INTPn\_Disable … 142
    - INTPn\_Enable … 143
    - INTP\_UserInit … 137
    - KEY\_Disable … 144
    - KEY\_Enable … 145
    - KEY\_Init … 138
    - KEY\_UserInit … 139

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2012.09.01	－	初版発行

---

CubeSuite+ V1.03.00 ユーザーズマニュアル  
V850 設計編

発行年月日 2012年9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/contact/>

CubeSuite+ V1.03.00