User's Manual

CubeSuite+ V1.00.00

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 マイクロコントローラ

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

ルネサス エレクトロニクス
www.renesas.com

Rev.1.00　2011.04

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

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

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

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

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

このマニュアルの使い方

このマニュアルは，V850 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る CubeSuite+について説明します。

CubeSuite+は，V850 マイクロコントローラの統合開発環境（ソフトウエア開発における，設計，実装，デバッグなどの各開発フェーズに必要なツールをプラットフォームである IDE に統合）です。統合することで，さまざまなツールを使い分ける必要がなく，本製品のみを使用して開発のすべてを行うことができます。

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

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

構　　成　このマニュアルは，大きく分けて次の内容で構成しています。

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

凡　　例

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

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

資　料　名		資料番号	
		和　文	英　文
CubeSuite+ 統合開発環境 ユーザーズ・マニュアル	起動編	R20UT0545J	R20UT0545E
	78K0 設計編	R20UT0546J	R20UT0546E
	78K0R 設計編	R20UT0547J	R20UT0547E
	RL78 設計編	R20UT0548J	R20UT0548E
	V850 設計編	このマニュアル	R20UT0549E
	R8C 設計編	R20UT0550J	R20UT0550E
	78K0 コーディング編	R20UT0551J	R20UT0551E
	RL78,78K0R コーディング編	R20UT0552J	R20UT0552E
	V850 コーディング編	R20UT0553J	R20UT0553E
	コーディング編（CX コンパイラ）	R20UT0554J	R20UT0554E
	R8C コーディング編	R20UT0576J	R20UT0576E
	78K0 ビルド編	R20UT0555J	R20UT0555E
	RL78,78K0R ビルド編	R20UT0556J	R20UT0556E
	V850 ビルド編	R20UT0557J	R20UT0557E
	ビルド編（CX コンパイラ）	R20UT0558J	R20UT0558E
	R8C ビルド編	R20UT0575J	R20UT0575E
	78K0 デバッグ編	R20UT0559J	R20UT0559E
	78K0R デバッグ編	R20UT0560J	R20UT0560E
	RL78 デバッグ編	R20UT0561J	R20UT0561E
	V850 デバッグ編	R20UT0562J	R20UT0562E
	R8C デバッグ編	R20UT0574J	R20UT0574E
	解析編	R20UT0563J	R20UT0563E
	メッセージ編	R20UT0407J	R20UT0407E

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

〔メ　モ〕

〔メ　モ〕

〔メ　モ〕

目　　次

第 1 章 概 説

CubeSuite+ は，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，外部バス・インタフェース，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。

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

- レポート機能

  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操作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，端子配置の操作手順は，以下のとおりです。

（1）CubeSuite+ の起動

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

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

（2）プロジェクトの作成／読み込み

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

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

（3）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+　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］を選択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“［端子配置表］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。

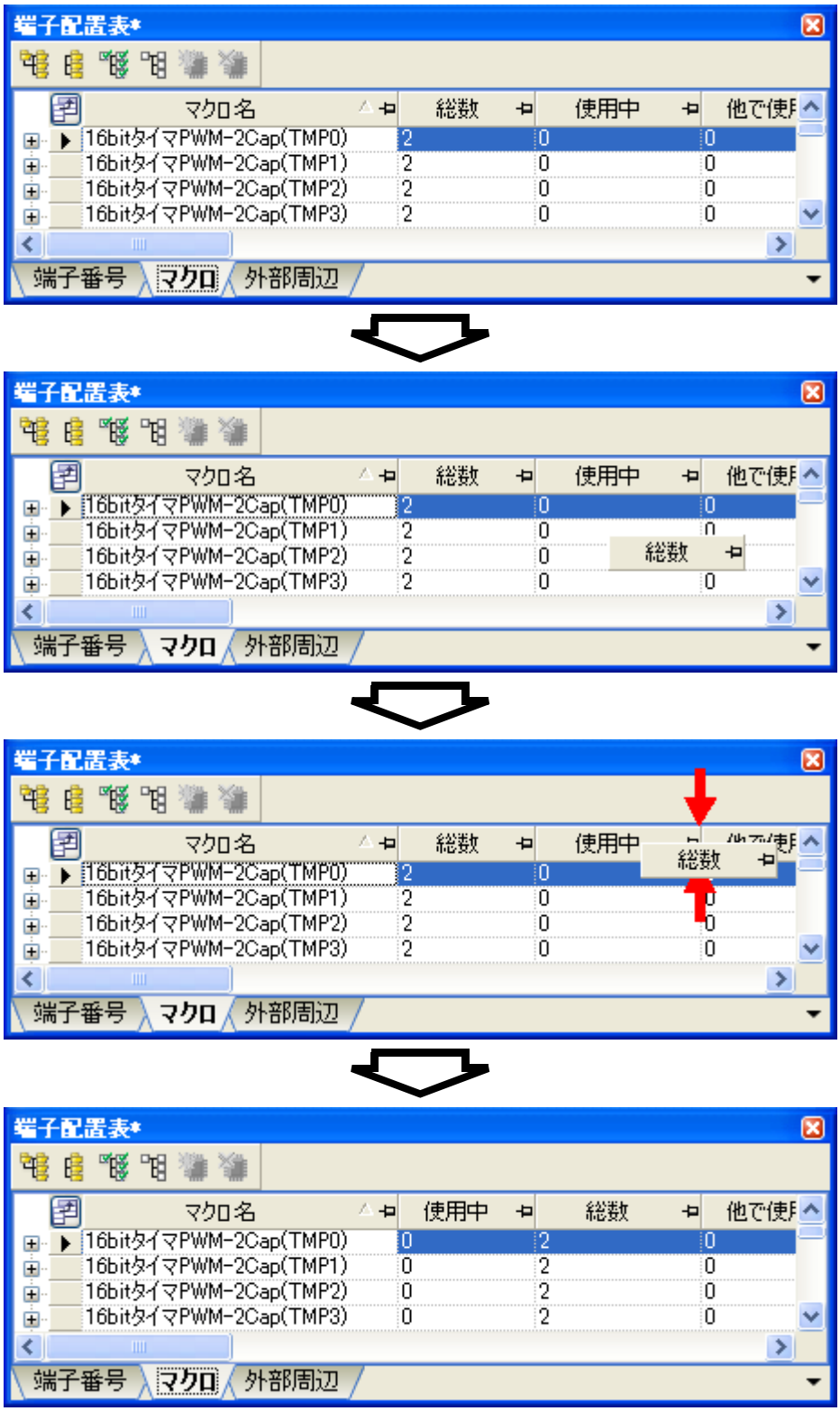
表 2—1 表示項目の選択

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



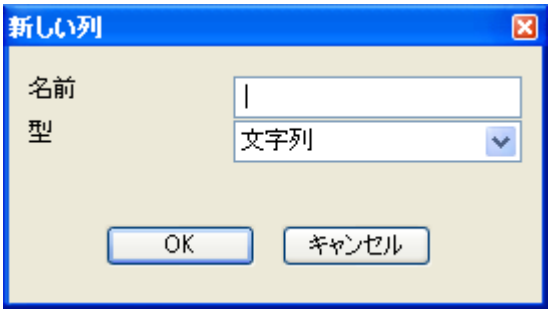
備考 表示順序の変更は，端子配置表の左上に設けられたボタンをクリックすることによりオープンする列の選択 ダイアログの表示項目選択エリアに表示されている項目をドラッグしたのち，端子配置表の移動先にドロップすることでも，表示項目の表示順序を変更することができます。

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

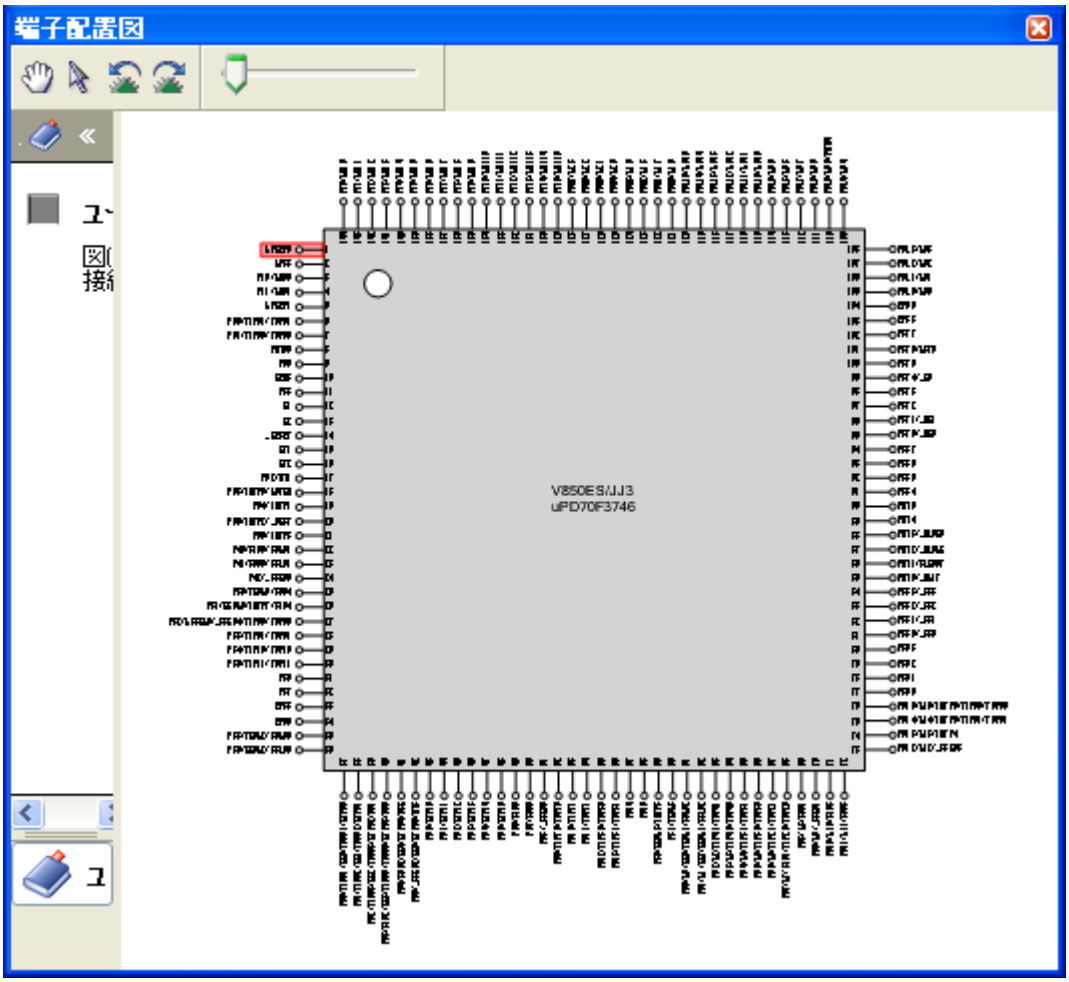
備考 削除可能な列は，新しい列 ダイアログでユーザが独自に追加した列に限られます。

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］を選択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



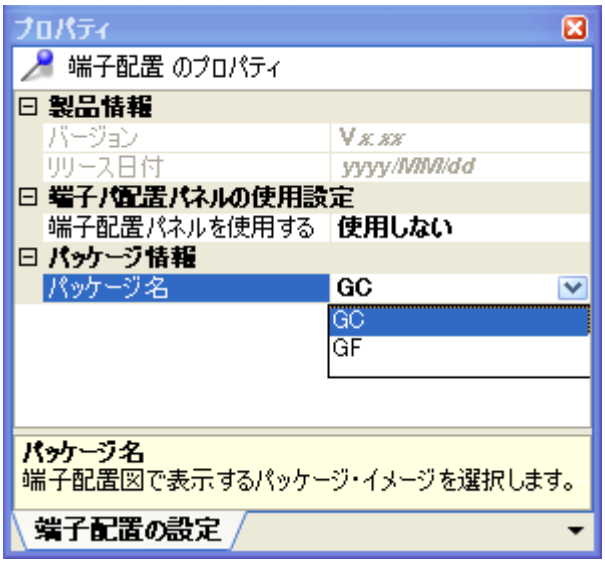
備考　プロパティ パネルの［端子配置の設定］タブでパッケージ名に“BGA”を選択している場合，端子配置図 パネルをオープンすることができません。

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。
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。

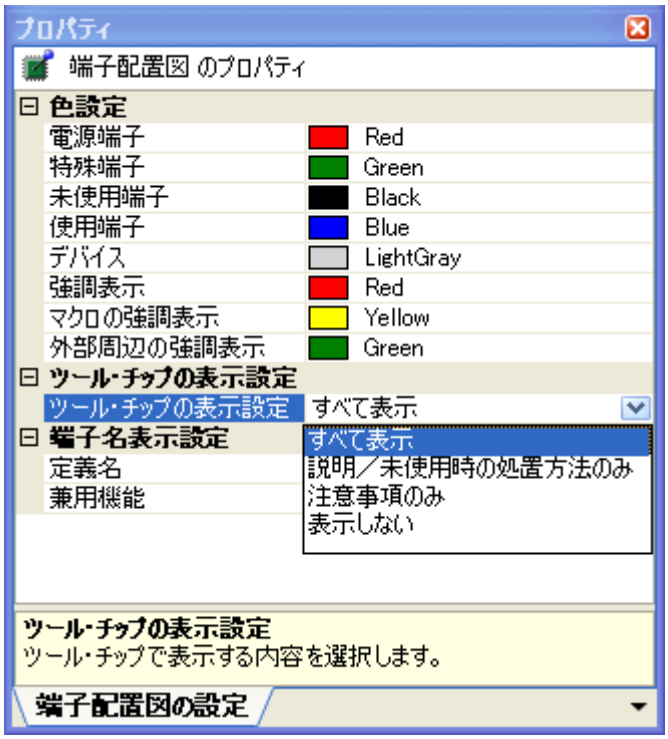
設定対象	概要
マクロの強調表示	端子配置表 パネルの［マクロ］タブで選択された項目に対応した端子の背景色を選択します。
外部周辺の強調表示	端子配置表 パネルの［外部周辺］タブで選択された項目に対応した端子の背景色を選択します。

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カッコ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。

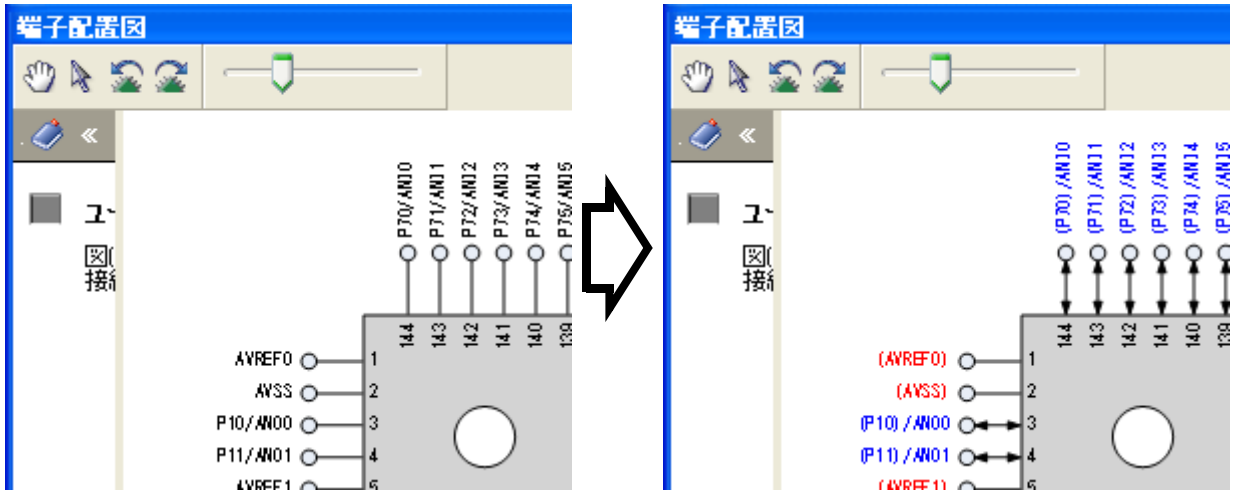
2.4 情報の記述

「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操作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（システム，ポート，外部バスなど）を制御する際に必要な情報を CubeSuite+ のパネル上で選択／入力することにより，対応するソース・コード（デバイス・ドライバ・プログラム）を出力します。

なお，コード生成の操作手順は，以下のとおりです。

（1）CubeSuite+ の起動

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

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

（2）プロジェクトの作成／読み込み

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

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

（3）コード生成 パネルのオープン

周辺機能（クロック発生回路，外部バス・インタフェース，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。

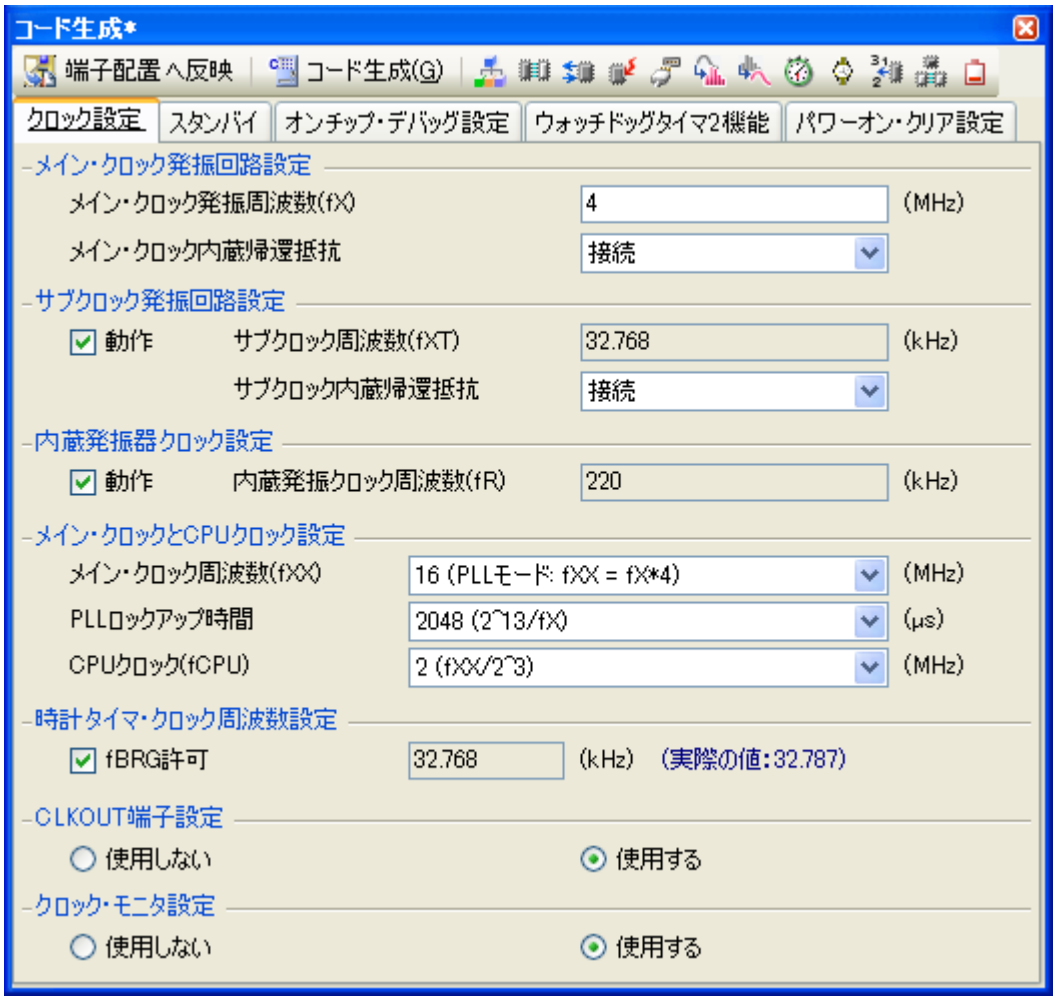
備考　“プロジェクトの保存”についての詳細は，「CubeSuite+　起動編」を参照してください。

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，外部バス・インタフェース，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］，［外部バス］，［ポート］など）を選択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

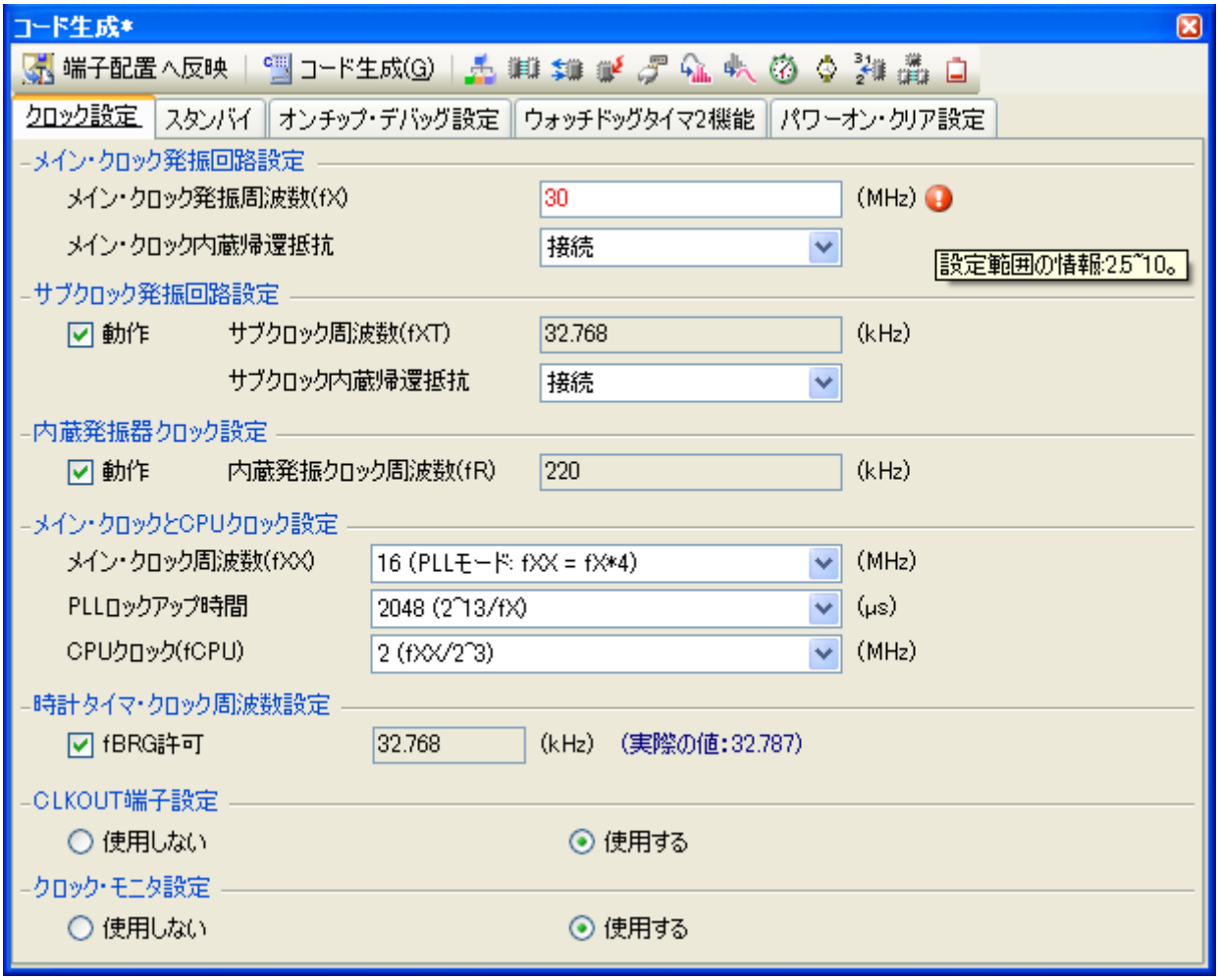
進数表記	概要
10 進数	1 ～ 9 の数字で始まり 0 ～ 9 の数字が続く数値，および 0
16 進数	0x で始まり 0 ～ 9 の数字，および a ～ f の英字が続く数値 （英字の大文字／小文字については，不問）

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，および入力が必須な箇所に値が未入力の際，設定すべき情報として誤っていることを示すアイコンを該当箇所に表示するとともに，文字列を赤色表示し，入力の不備を警告します。

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

図 3—2　入力不備箇所に対するアイコン表示

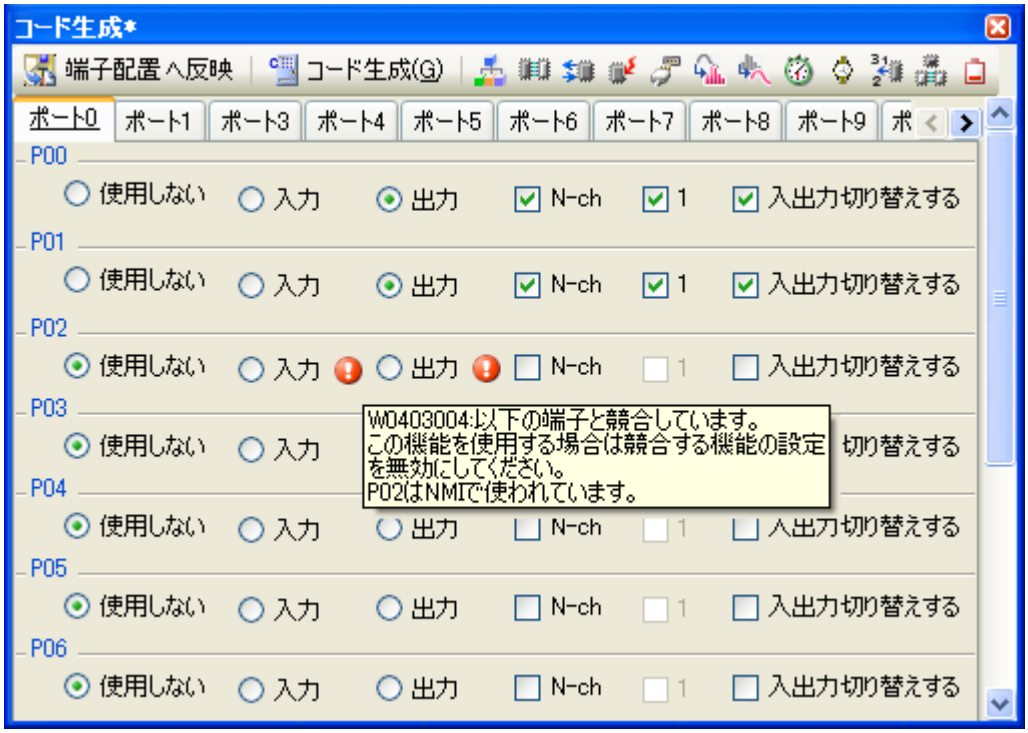


3. 3. 3　端子の競合に対するアイコン表示

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

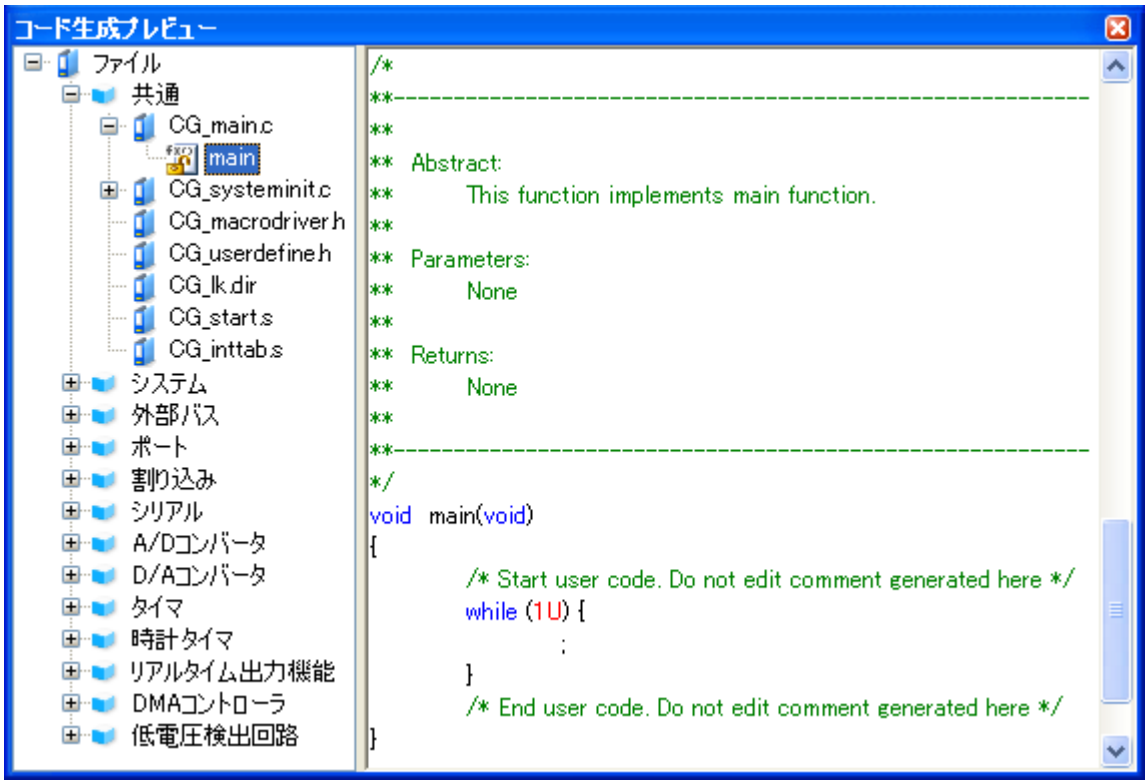


3.4 ソース・コードの確認

「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。

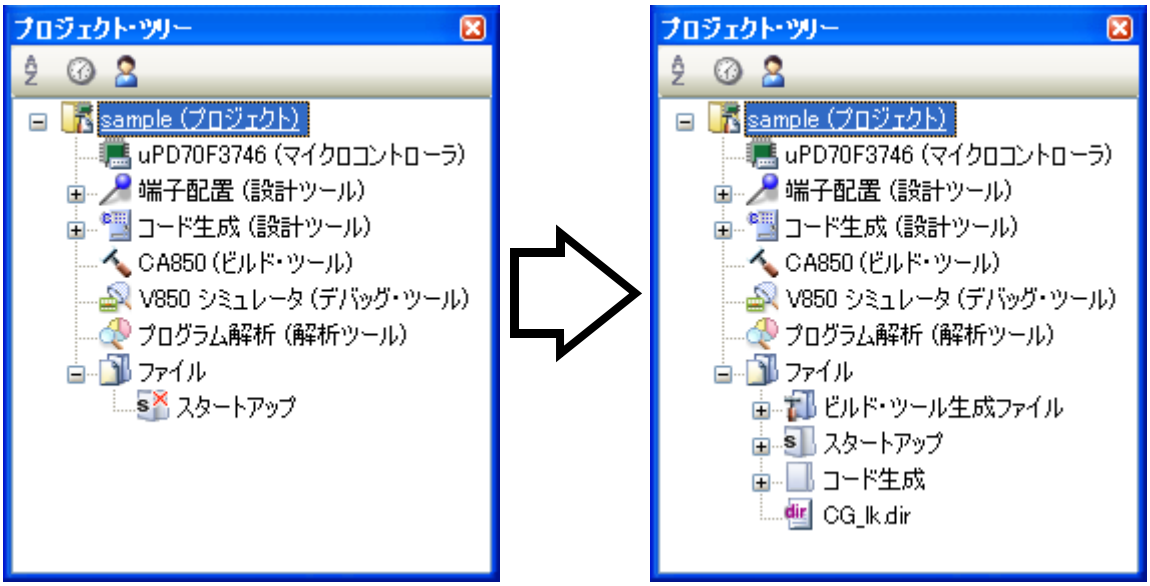
4. 一部の API 関数（シリアル・アレイ・ユニット用 API 関数など）については，ソース・コードの出力時（コード生成 パネルの コード生成(G) ボタンをクリックした際）にレジスタ値 SFR などが計算され確定するものがあります。このため，コード生成プレビュー パネルに表示されるソース・コードは，実際に出力されるソース・コードと一致しない場合があります。

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。

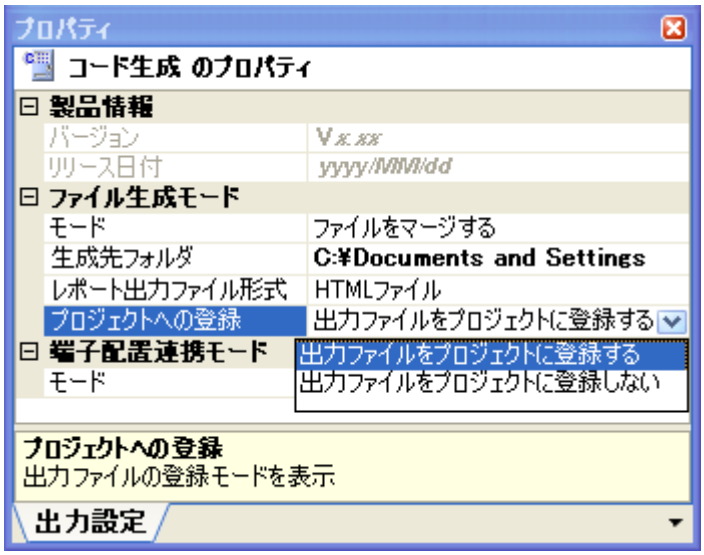
なお，ソース・コードの出力先は，プロパティ パネルの［出力設定］タブ→［生成先フォルダ］で指定されたフォルダとなります。

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。

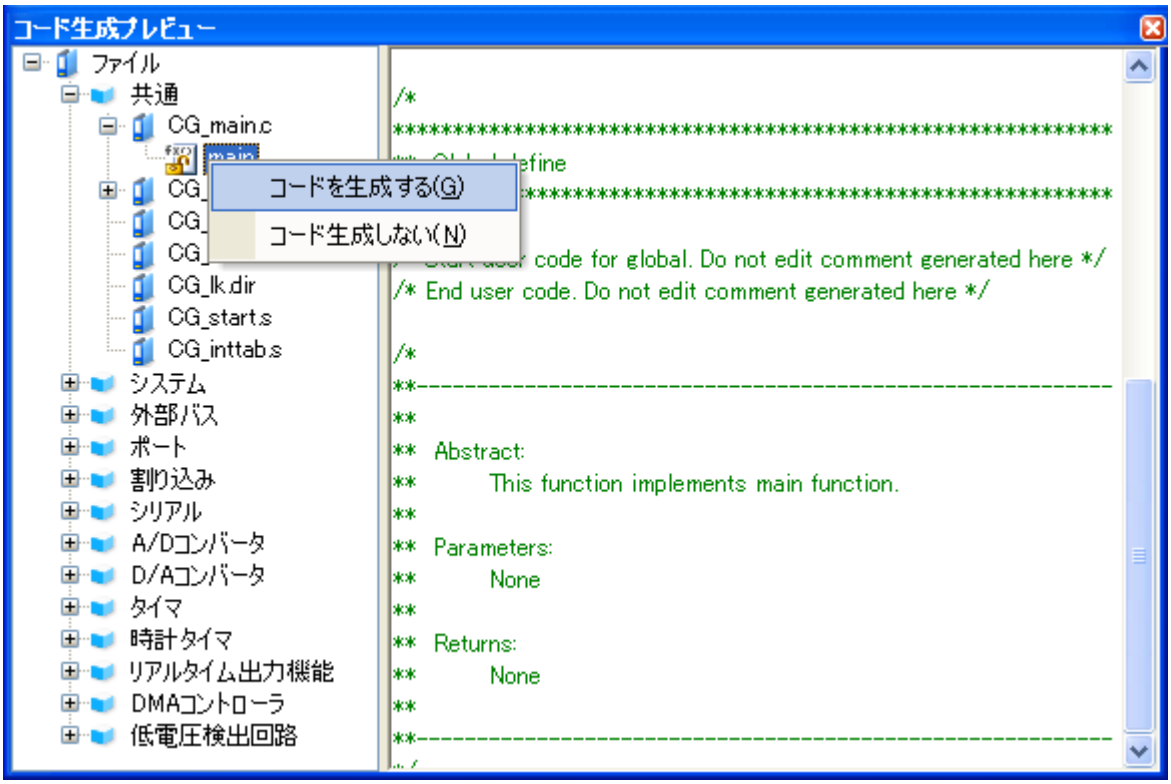
図 3—6　登録有無の設定


プロパティ
コード生成 のプロパティ
製品情報
バージョン
V.x.xx
リリース日付
yyyy/MM/dd
ファイル生成モード
モード
ファイルをマージする
生成先フォルダ
C:¥Documents and Settings
レポート出力ファイル形式
HTMLファイル
プロジェクトへの登録
出力ファイルをプロジェクトに登録する
端子配置連携モード
モード
出力ファイルをプロジェクトに登録する
出力ファイルをプロジェクトに登録しない
プロジェクトへの登録
出力ファイルの登録モードを表示
出力設定

3. 5. 1　出力有無の設定

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

図 3—7　出力有無の設定



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

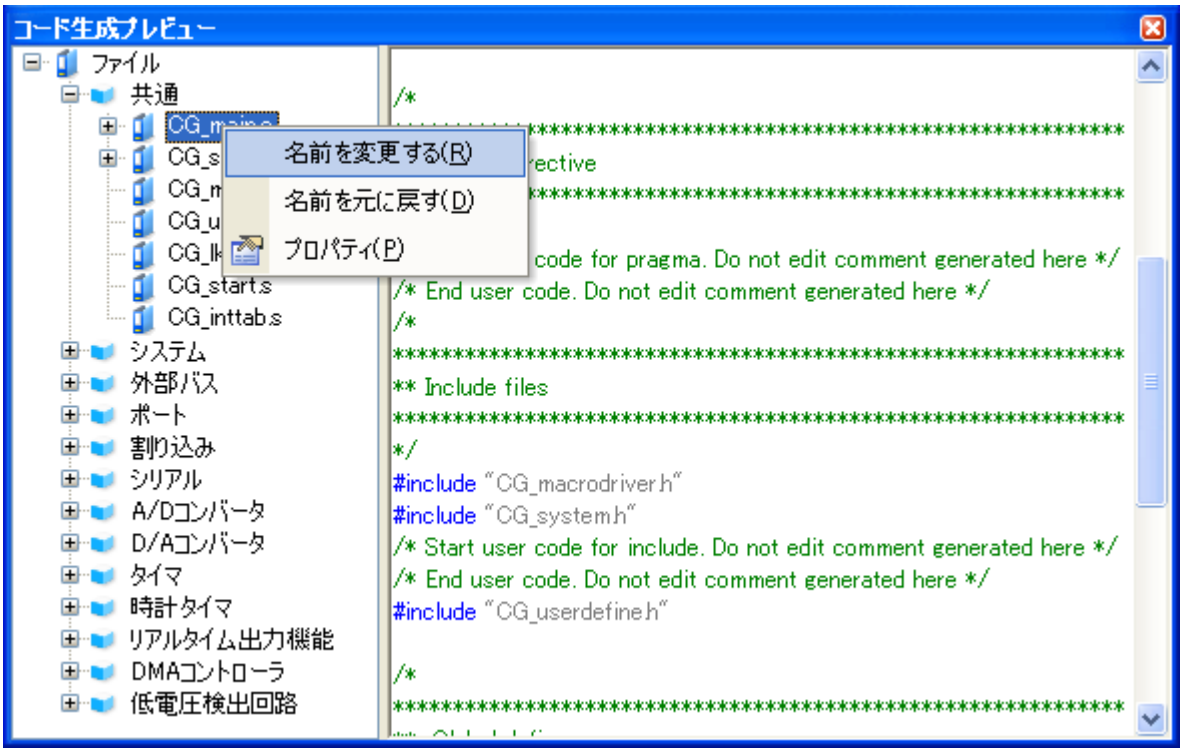
表 3—4　ソース・コードの出力有無

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—8 ファイル名の変更

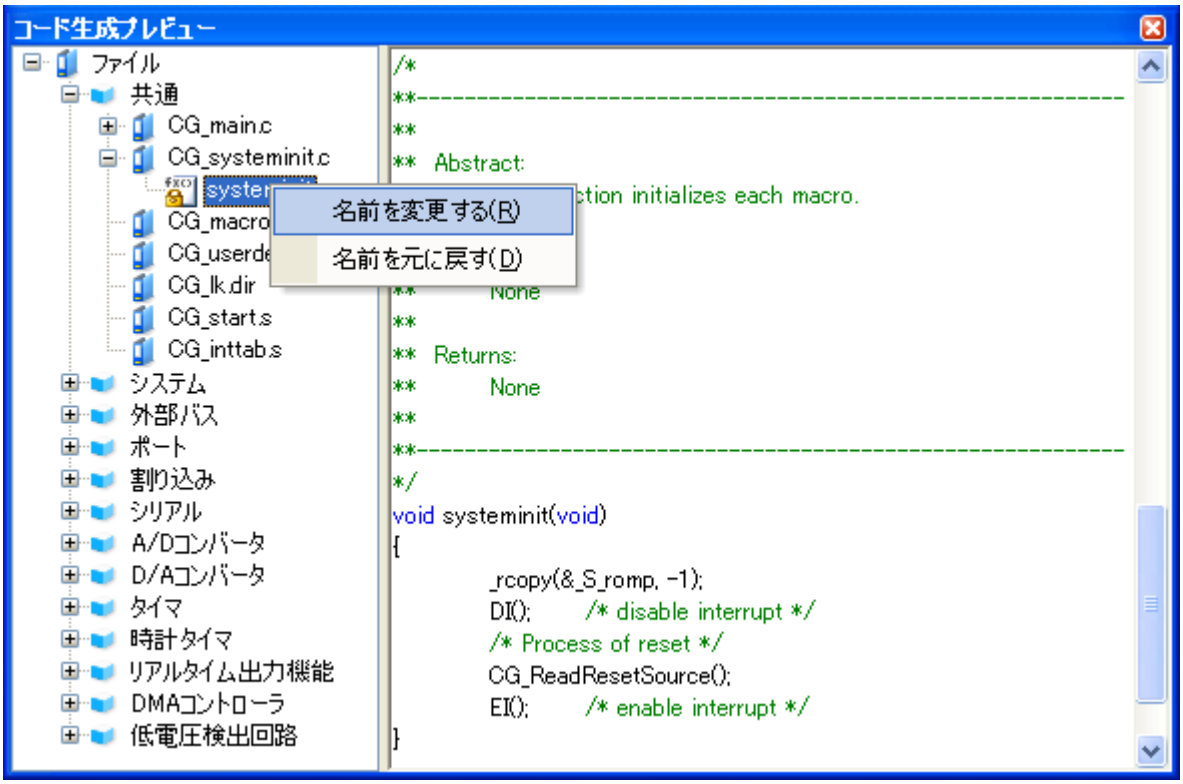


備考 コード生成が規定しているデフォルト・ファイル名に戻す際には，コンテキスト・メニューから“名前を元に戻す”を選択します。

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—9 API 関数名の変更

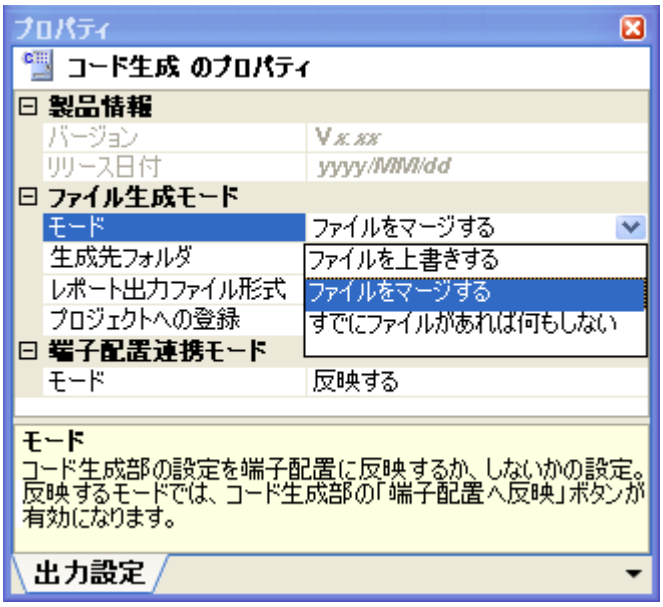


備考 コード生成が規定しているデフォルト API 関数名に戻す際には，コンテキスト・メニューから“名前を元に戻す”を選択します。

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

図 3—10　出力モードの変更



備考　出力モードの選択は，以下の 3 種類から行います。

表 3—5　ソース・コードの出力モード

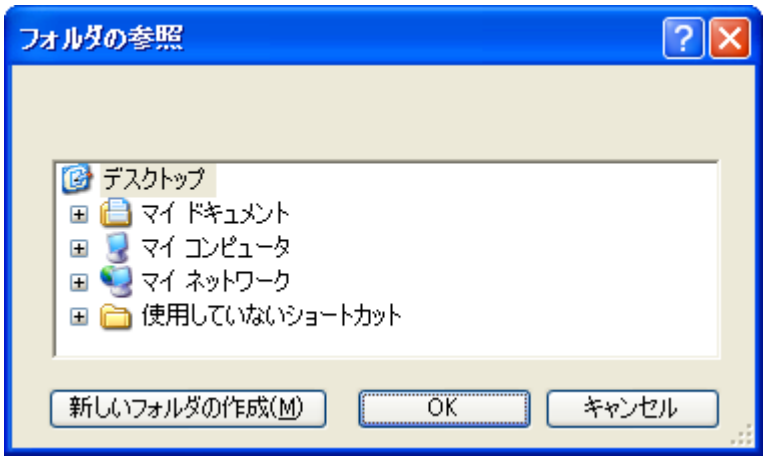
出力モード	概要
ファイルを上書きする	同一のファイル名を有するファイルが既存していた場合，該当ファイルを上書きします。
ファイルをマージする	同一のファイル名を有するファイルが既存していた場合，該当ファイルをマージします。 なお，マージする部位については， /* Start user code …. Do not edit comment generated here */ から /* End user code. Do not edit comment generated here */ で囲まれた部位に限られます。
すでにファイルがあれば何もしない	同一のファイル名を有するファイルが既存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

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—11　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—12 レポート・ファイル macro の出力例

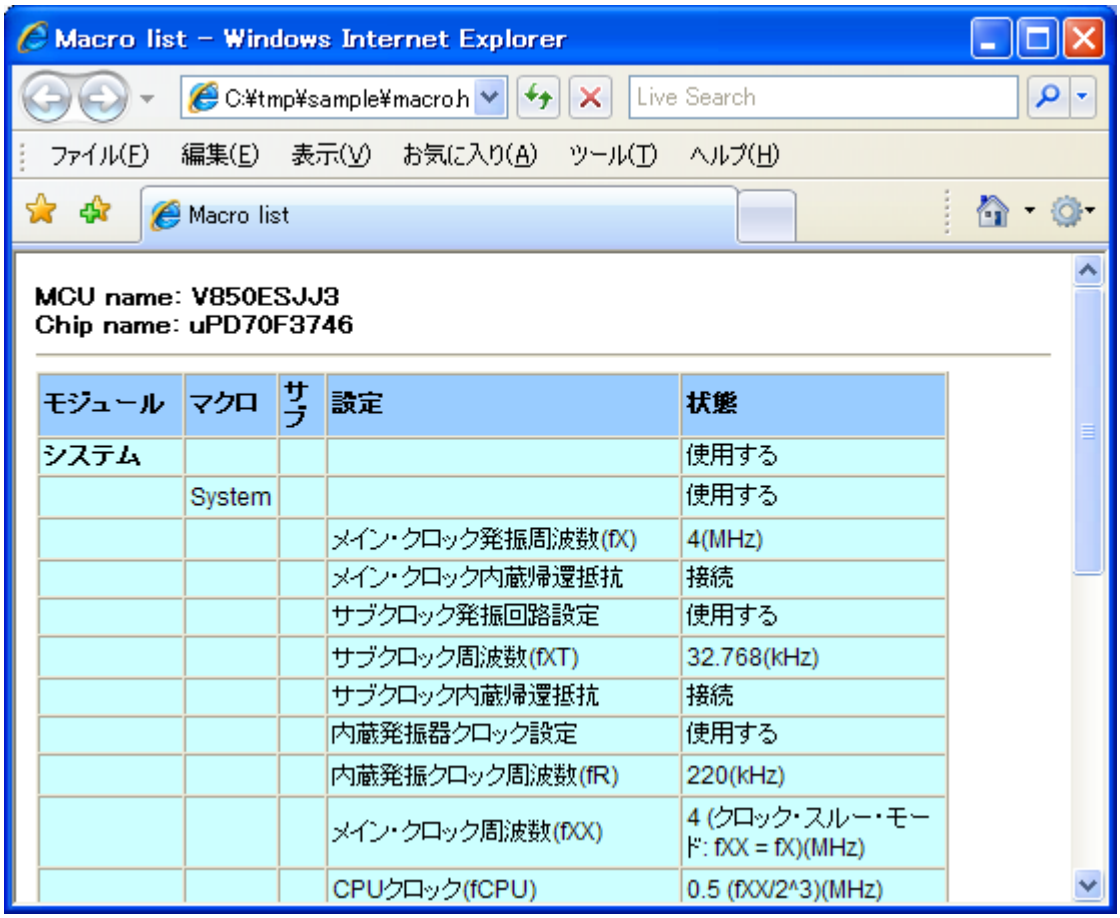


図 3—13 レポート・ファイル function の出力例

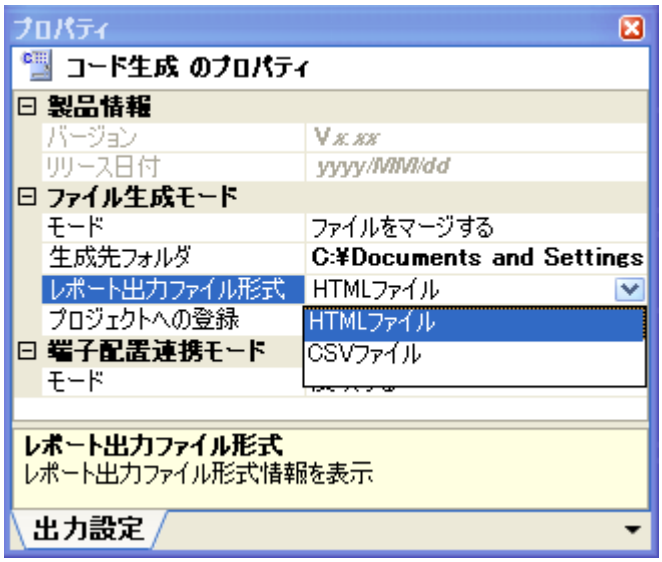
MCU name: V850ESJJ3
Chip name: uPD70F3746

モジュール	ファイル	マクロ	機能	デフォルト
共通				
	CG_main.c			CG_main.c
			void main(void)	main
	CG_systeminit.c			CG_systeminit.c
			void systeminit(void)	systeminit
	CG_macrodriver.h			CG_macrodriver.h
	CG_userdefine.h			CG_userdefine.h
	CG_lk.dir			CG_lk.dir
	CG_start.s			CG_start.s

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—14 出力形式の変更



備考 出力形式の選択は，以下の 2 種類から行います。

表 3—7 ソース・コードの出力モード

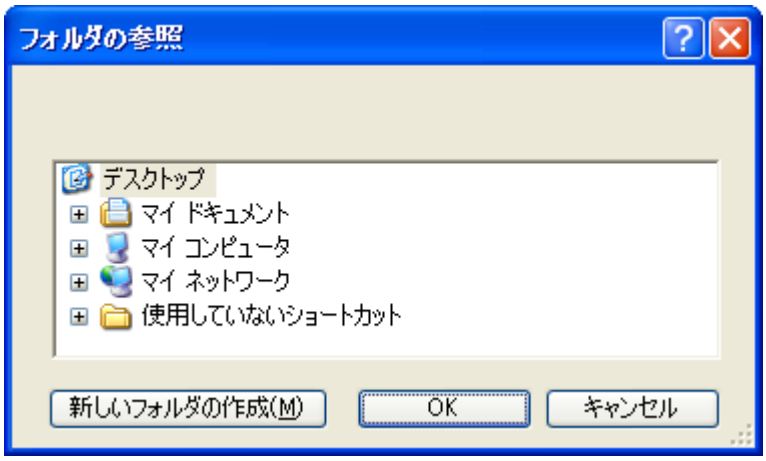
出力形式	概要
HTML ファイル	HTML 形式でレポート・ファイルを出力します。
CSV ファイル	CSV 形式でレポート・ファイルを出力します。

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—15　出力先の変更



付録 A　ウインドウ・リファレンス

本付録では，設計ツールのウインドウ／パネル／ダイアログについて説明します。

A. 1　説　　明

以下に，設計ツールのウインドウ／パネル／ダイアログの一覧を示します。

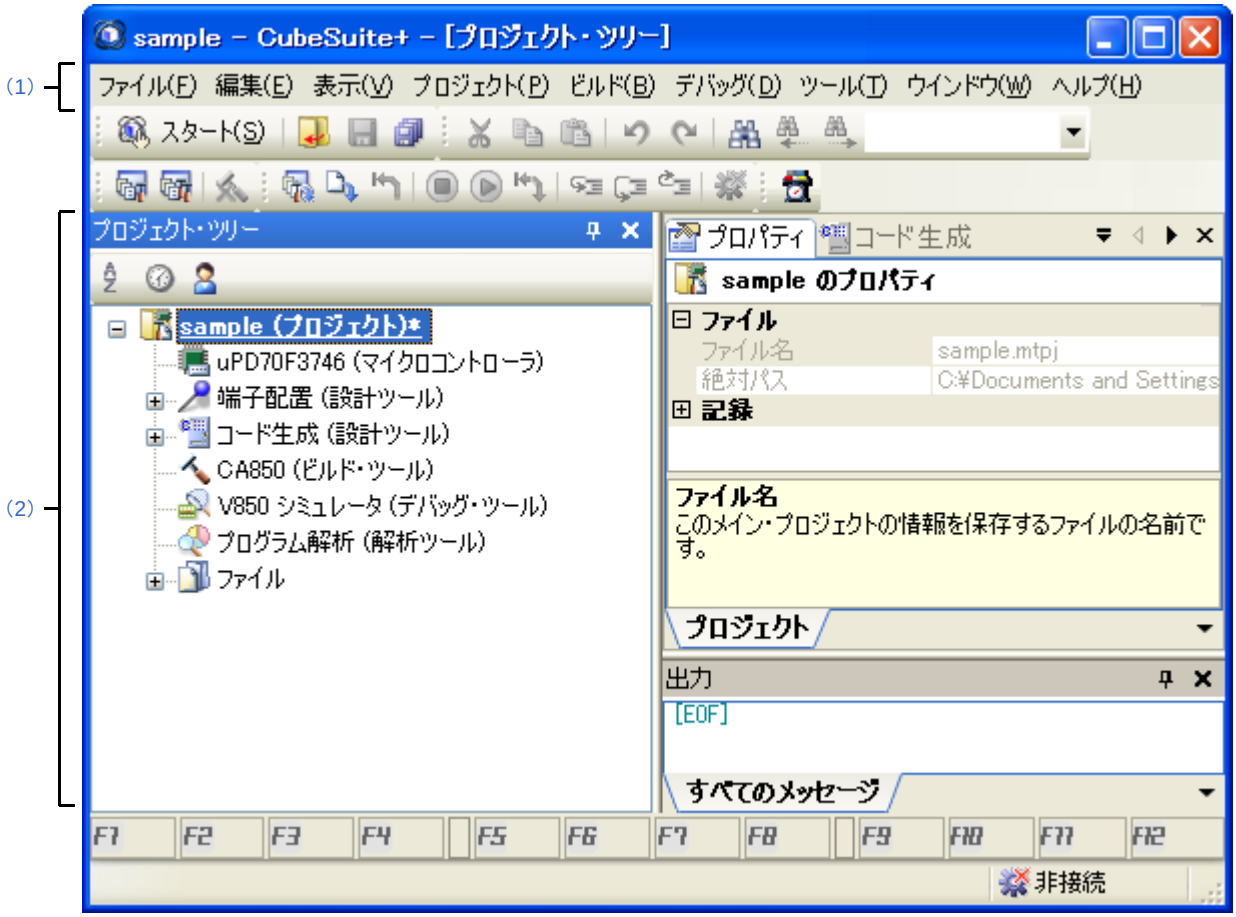
表 A—1　ウインドウ／パネル／ダイアログの一覧

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

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

図 A―1　メイン・ウインドウ



ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］

［オープン方法］

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

［各エリアの説明］

(1) メニューバー

本エリアは，以下に示したメニュー群から構成されています。

(a)［ファイル］メニュー

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。

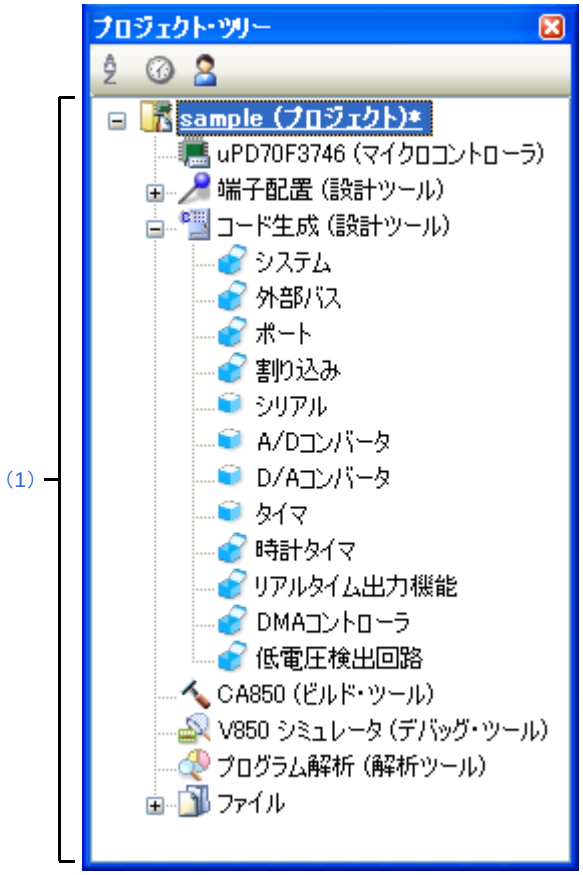
本エリアについての詳細は，以下を参照してください。

- 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端子配置表パネルをオープンします。
端子配置図	端子配置表 パネルにおける情報の記述状況を表示するための端子配置図パネルをオープンします。

(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［システム］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［外部バス］をオープンします。
ポート	マイクロコントローラが提供しているポートの機能を制御するうえで必要な情報を設定するための［ポート］をオープンします。
割り込み	マイクロコントローラが提供している割り込み／キー割り込みの機能を制御するうえで必要な情報を設定するための［割り込み］をオープンします。
シリアル	マイクロコントローラが提供しているシリアル・アレイ・ユニット，およびシリアル・インタフェースの機能を制御するうえで必要な情報を設定するための［シリアル］をオープンします。
A/D コンバータ	マイクロコントローラが提供している A/D コンバータの機能を制御するうえで必要な情報を設定するための［A/D コンバータ］をオープンします。
D/A コンバータ	マイクロコントローラが提供している D/A コンバータの機能を制御するうえで必要な情報を設定するための［D/A コンバータ］をオープンします。
タイマ	マイクロコントローラが提供しているタイマ・アレイ・ユニットの機能を制御するうえで必要な情報を設定するための［タイマ］をオープンします。

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]

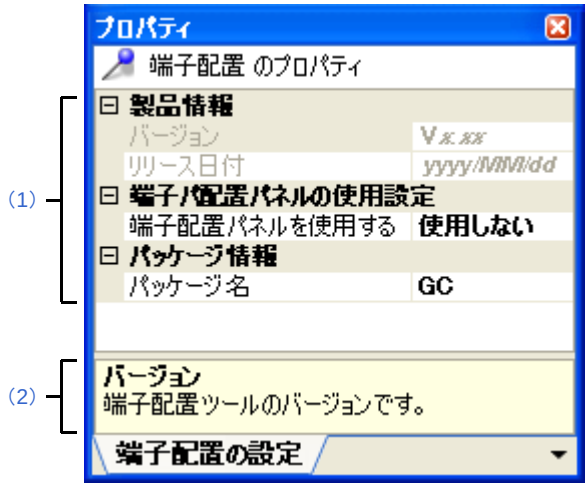
マウスを右クリックすることにより表示されるコンテキスト・メニューは，以下のとおりです。

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)]，［コード生成（設計ツール)]）に対応した情報を保持したプロパティ パネルをオープンします。

プロパティ パネル

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

図 A―3　プロパティ パネル（[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)]
- [[ヘルプ] メニュー（プロパティ パネル専用部分)]
- [コンテキスト・メニュー]

[オープン方法]

- プロジェクト・ツリー パネルにおいて，ノード（[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]，[外部バス]，[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]，[外部バス]，[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]，[外部

バス]，[ポート] など”）を選択することにより，詳細情報表示／変更エリア，および説明エリアの表示内容が該当ノードに対応したものへと切り替わります。

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

[各エリアの説明]

(1) 詳細情報表示／変更エリア

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

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

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

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

備考 1. 本エリアの表示内容についての詳細は，[端子配置の設定] タブ，[端子配置の情報] 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] タブを参照してください。

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

(2) 説明エリア

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

[[編集] メニュー（プロパティ パネル専用部分)]

元に戻す	直前に行った編集作業を取り消します。
切り取り	選択している文字列を切り取り，クリップ・ボードに保存します。
コピー	選択している文字列をクリップ・ボードに保存します。
貼り付け	指定された箇所に，クリップ・ボードの内容を挿入します。

削除	選択している文字列を削除します。
すべて選択	編集中の項目に表示されている全文字列を選択します。

[[ヘルプ]メニュー（プロパティ パネル専用部分）]

プロパティ パネルのヘルプを開く	本パネルのヘルプを表示します。

[コンテキスト・メニュー]

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

（1）項目を編集中の場合

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

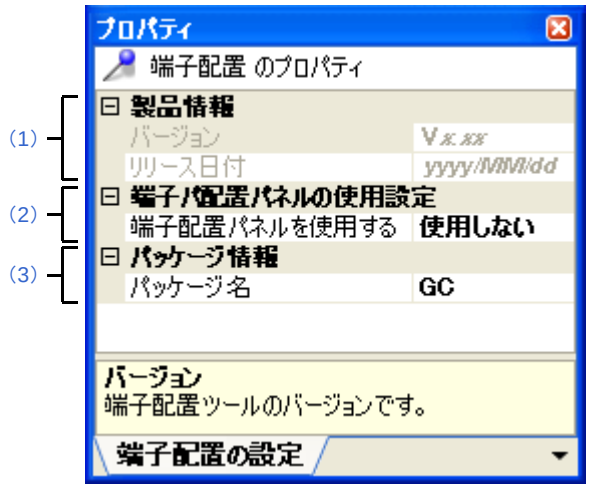
（2）項目を編集中以外の場合

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

［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，端子配置パネルの使用設定，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	端子配置のバージョンを表示します。
リリース日付	端子配置のリリース日付を表示します。

(2)［端子配置パネルの使用設定］カテゴリ

端子配置表 パネル，および端子配置図 パネルの表示有無を選択します。

端子配置パネルを使用する	次に本プロジェクトをオープンした際，メイン・ウインドウに端子配置表 パネル，および端子配置図 パネルを表示させるか否かを選択します。	
	使用する	端子配置表 パネル，および端子配置図 パネルを表示します。
	使用しない	端子配置表 パネル，および端子配置図 パネルを表示しません。

(3)［パッケージ情報］カテゴリ

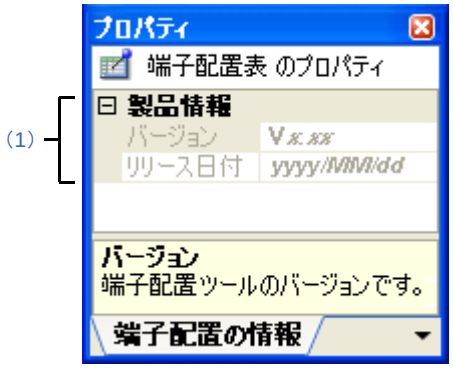
端子配置図 パネルに端子配置図として表示するマイクロコントローラの形状（パッケージ名）を選択します。

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	端子配置のバージョンを表示します。
リリース日付	端子配置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，未使用端子など）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。
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

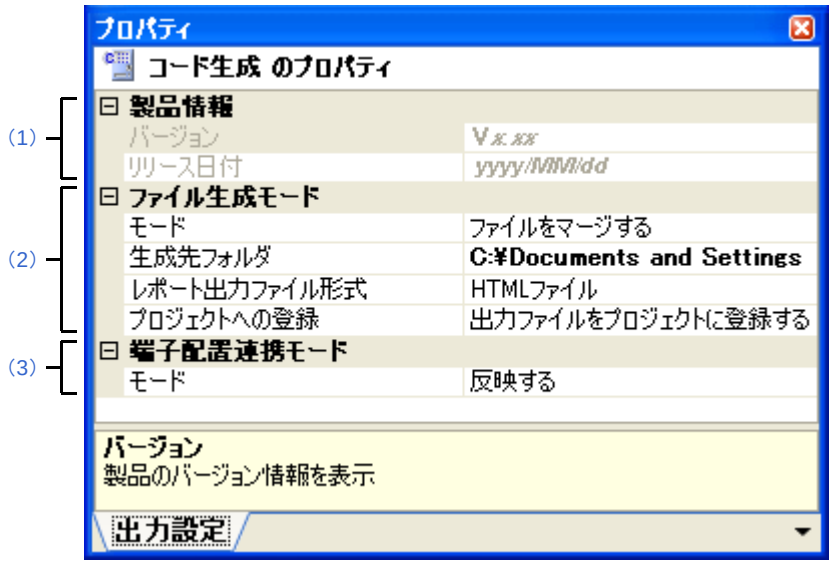
端子の付加情報を端子配置図に表示するか否かを選択します。

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カッコ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）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	コード生成のバージョンを表示します。

リリース日付	コード生成のリリース日付を表示します。

(2)［ファイル生成モード］カテゴリ

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

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

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

(3) [端子配置連携モード] カテゴリ

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。

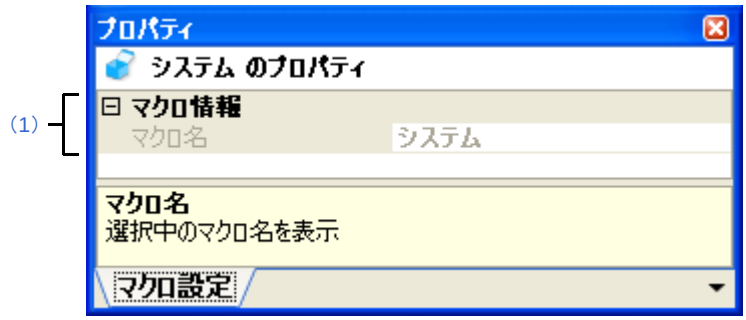
モード	 ボタンをクリックした際，コード生成 パネルで設定した各種情報を端子配置表 パネルに反映するか否かを選択します。	
	反映する	コード生成 パネルで設定した各種情報を端子配置表 パネルに反映します。
	反映しない	コード生成 パネルで設定した各種情報を端子配置表 パネルに反映しません。

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]，[外部バス]，[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]，[外部バス]，[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]，[外部バス]，[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]，[外部バス]，[ポート]など”を選択することにより，表示内容が該当ノードに対応したものへと切り替わります。

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

［各エリアの説明］

(1)［マクロ情報］カテゴリ

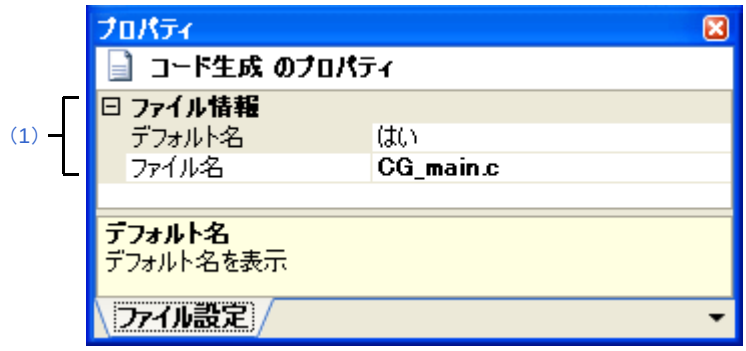
プロジェクト・ツリー パネルで選択した周辺機能ノード（システム，外部バス，ポートなど），コード生成 パネルでクリックした周辺機能ボタンに関する情報（マクロ名）の表示，および設定の変更を行います。

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へと切り替わります。

［各エリアの説明］

(1)［ファイル情報］カテゴリ

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

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

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。

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］を選択
- ［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］を選択
- ［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。
端子名	該当端子の端子名を表示します。

列の見出し	概要
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。
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。

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。

［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］を選択
- ［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。

列の見出し	概要
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。
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］を選択
- ［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。
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。

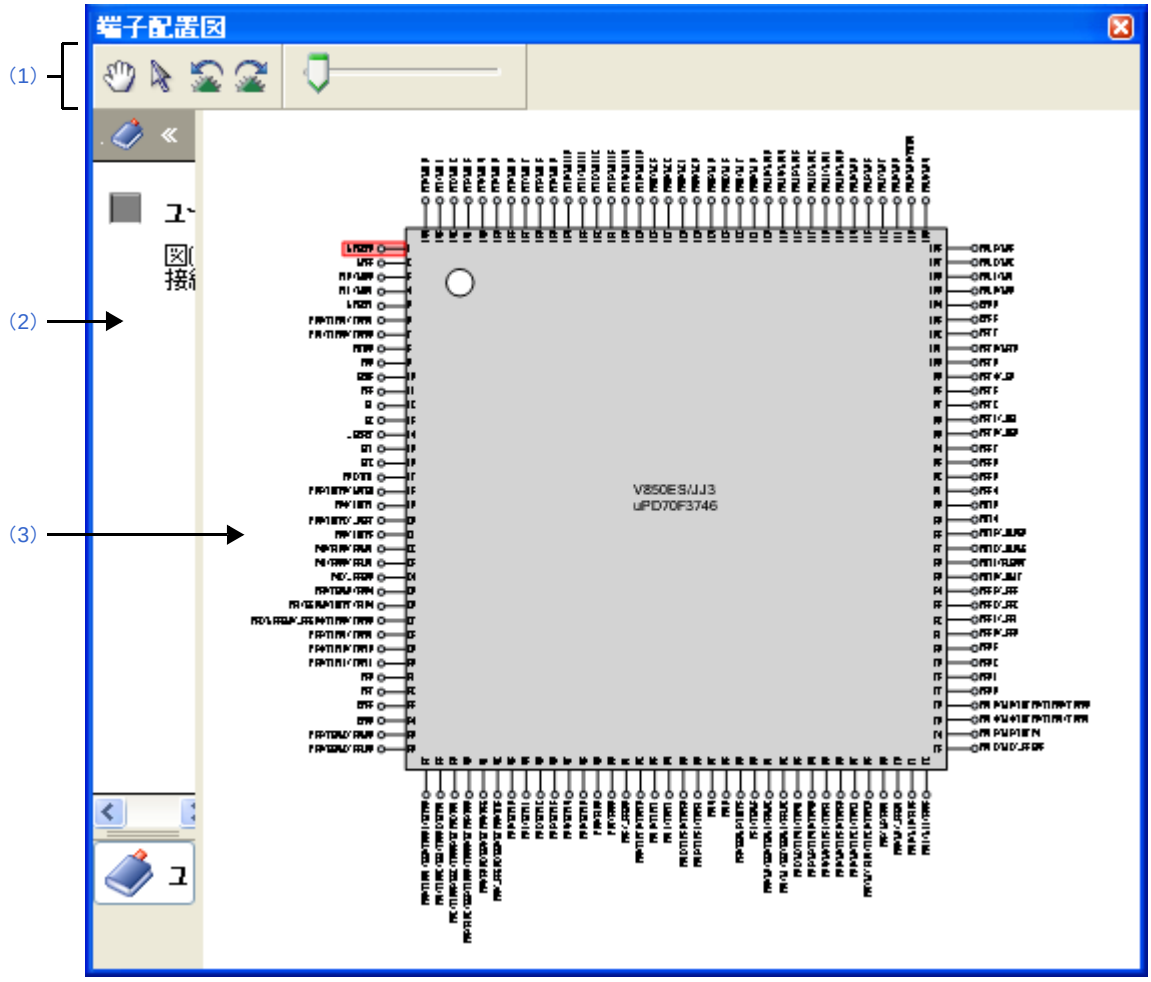
3. 列の移動（表示順序の変更）は，端子配置表の該当列をドラッグしたのち，移動先にドロップすることにより行います。

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。

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］を選択
- ［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。

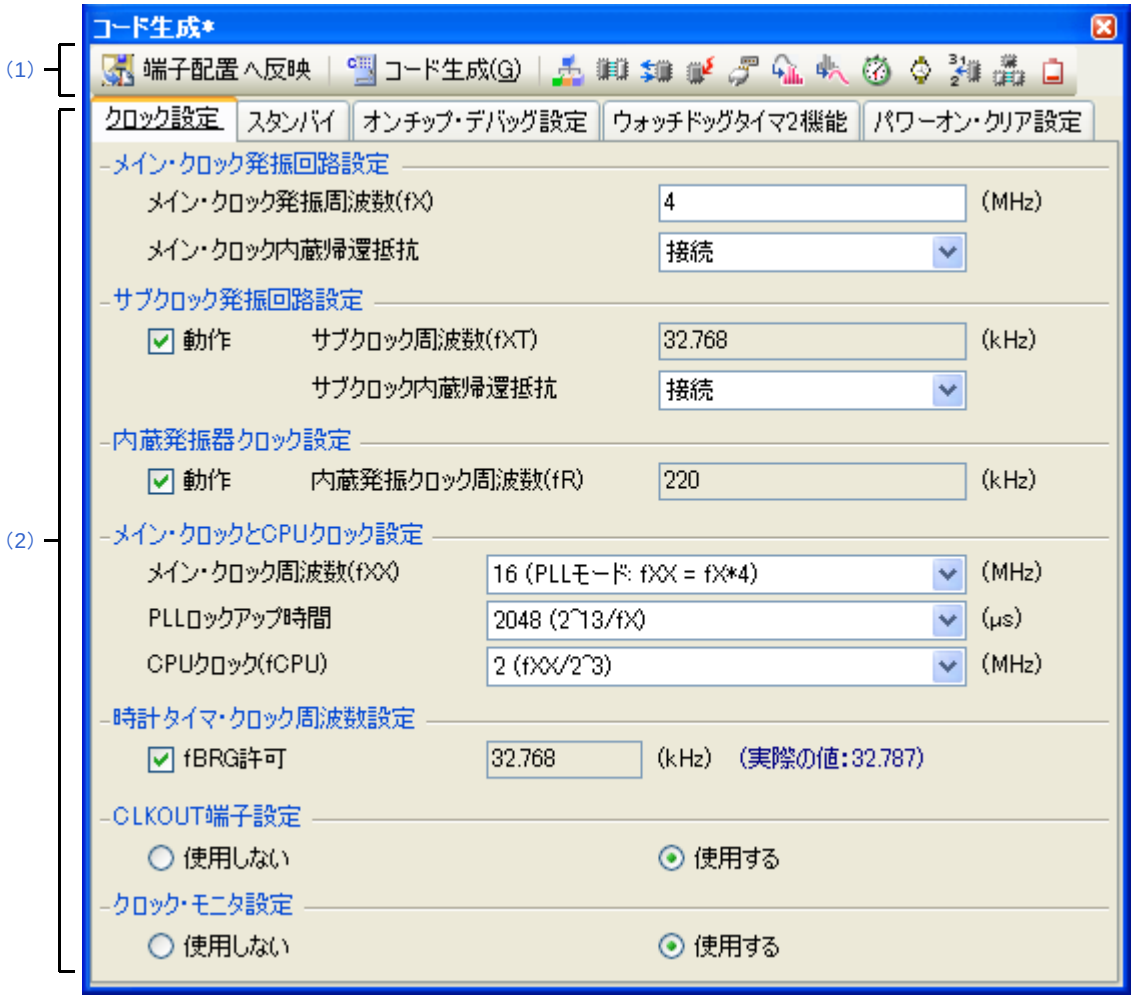
(2) 外部周辺コントローラ上で右クリックした場合

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］，［外部バス］，［ポート］など）を選択

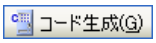
備考　すでに本パネルがオープンしていた場合，周辺機能ボタン（  ，  ，  など）をクリックすることにより，情報設定エリアの表示内容が該当ボタンに対応したものへと切り替わります。

［各エリアの説明］

(1) ツールバー

本エリアは，以下に示したボタン群“周辺機能ボタン”から構成されています。

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

	本パネルで設定した各種情報を端子配置表 パネルに反映します。 なお，本ボタンは，［出力設定］タブの［端子配置連携モード］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。
	プロパティ パネルの［出力設定］タブ→［生成先フォルダ］で指定されたフォルダにソース・コード（デバイス・ドライバ・プログラム）を出力します。
	情報設定エリアの表示内容を“マイクロコントローラが提供しているクロック発生機能，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［システム］”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［外部バス］”へと切り替えます。
	情報設定エリアの表示内容を“マイクロコントローラが提供しているポートの機能を制御するうえで必要な情報を設定するための［ポート］”へと切り替えます。
	情報設定エリアの表示内容を“マイクロコントローラが提供している外部割り込み／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［割り込み］”へと切り替えます。
	情報設定エリアの表示内容を“マイクロコントローラが提供しているアシンクロナス・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［シリアル］”へと切り替えます。
	情報設定エリアの表示内容を“マイクロコントローラが提供している A/D コンバータの機能を制御するうえで必要な情報を設定するための［A/D コンバータ］”へと切り替えます。
	情報設定エリアの表示内容を“マイクロコントローラが提供している D/A コンバータの機能を制御するうえで必要な情報を設定するための［D/A コンバータ］”へと切り替えます。
	情報設定エリアの表示内容を“マイクロコントローラが提供している 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［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。

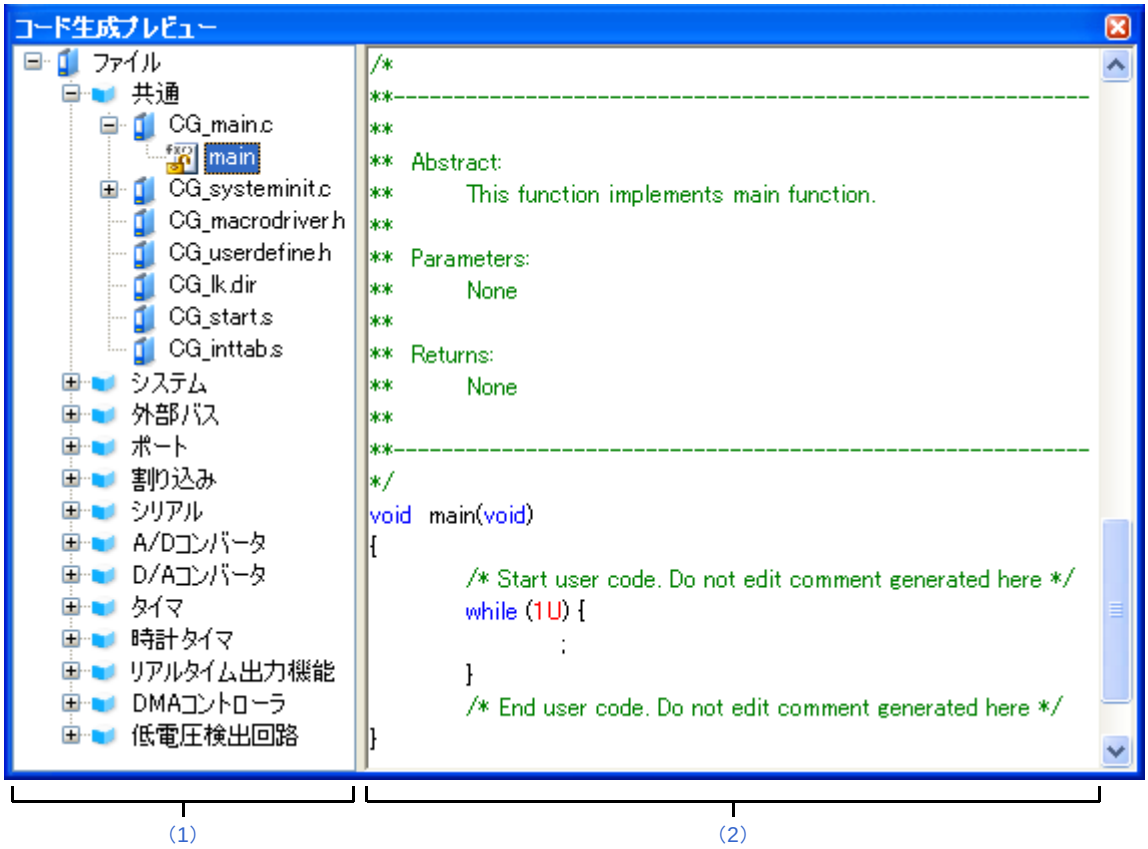
[[ヘルプ］メニュー（コード生成 パネル専用部分)]

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

図 A―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。

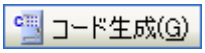
- ［オープン方法］
- ［各エリアの説明］
- ［［ファイル］メニュー（コード生成プレビュー パネル専用部分）］
- ［［ヘルプ］メニュー（コード生成プレビュー パネル専用部分）］
- ［コンテキスト・メニュー］

［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

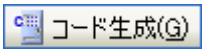
コード生成 パネルで設定した情報に応じたソース・コード（デバイス・ドライバ・プログラム）の確認を行います。

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 

ボタンをクリックした際）にレジスタ値 SFR などが計算され確定するものがあります。このため，本パネルに表示されるソース・コードは，実際に出力されるソース・コードと一致しない場合があります。

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

[[ファイル]メニュー（コード生成プレビュー パネル専用部分）]

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

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

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

[[ヘルプ]メニュー（コード生成プレビュー パネル専用部分）]

コード生成プレビュー パネルのヘルプを開く	本パネルのヘルプを表示します。

[コンテキスト・メニュー]

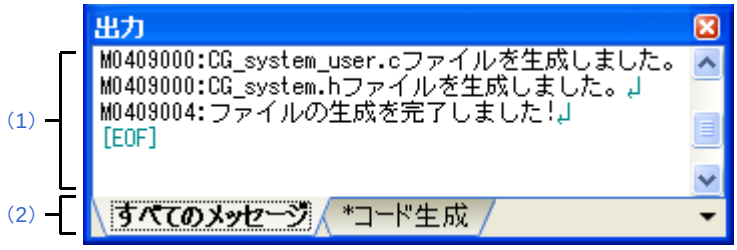
マウスを右クリックすることにより表示されるコンテキスト・メニューは，以下のとおりです。

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

出力 パネル

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

図 A—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）］
- ［［ヘルプ］メニュー（出力 パネル専用部分）］
- ［コンテキスト・メニュー］

［オープン方法］

- ［表示］メニュー→［出力］を選択

［各エリアの説明］

（1）メッセージ・エリア

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

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

表 A—4　メッセージの文字色／背景色

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

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

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

(2) タブ選択エリア

メッセージの出力元を選択します。

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

［［ファイル］メニュー（出力 パネル専用部分）］

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

［［編集］メニュー（出力 パネル専用部分）］

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

［［ヘルプ］メニュー（出力 パネル専用部分）］

出力 パネルのヘルプを開く	本パネルのヘルプを表示します。

［コンテキスト・メニュー］

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

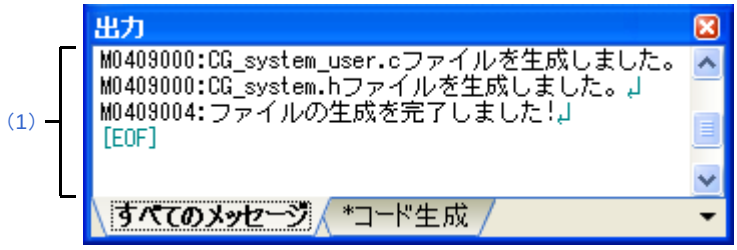
コピー	選択している文字列をクリップ・ボードに保存します。
すべて選択	メッセージ・エリアに表示されている全文字列を選択します。
クリア	メッセージ・エリアに表示されている全文字列を消去します。
検索の中止	実行中の文字列検索を中止します。 文字列検索を非実行中の場合，本項目は無効となります。

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

［すべてのメッセージ］タブ

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

図 A—19　［すべてのメッセージ］タブ



ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］

［オープン方法］

- ［表示］メニュー→［出力］を選択

［各エリアの説明］

（1）メッセージ・エリア

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

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

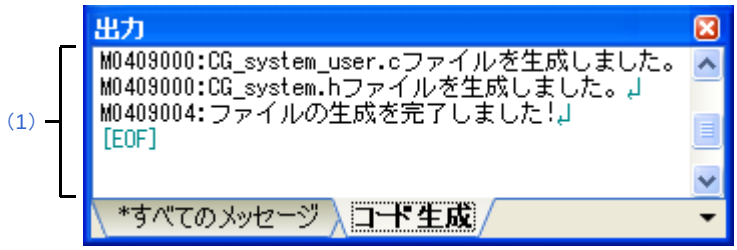
表 A—5　メッセージの文字色／背景色

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

［コード生成］タブ

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

図 A—20 ［コード生成］タブ



ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］

［オープン方法］

- ［表示］メニュー→［出力］を選択

［各エリアの説明］

(1) メッセージ・エリア

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

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

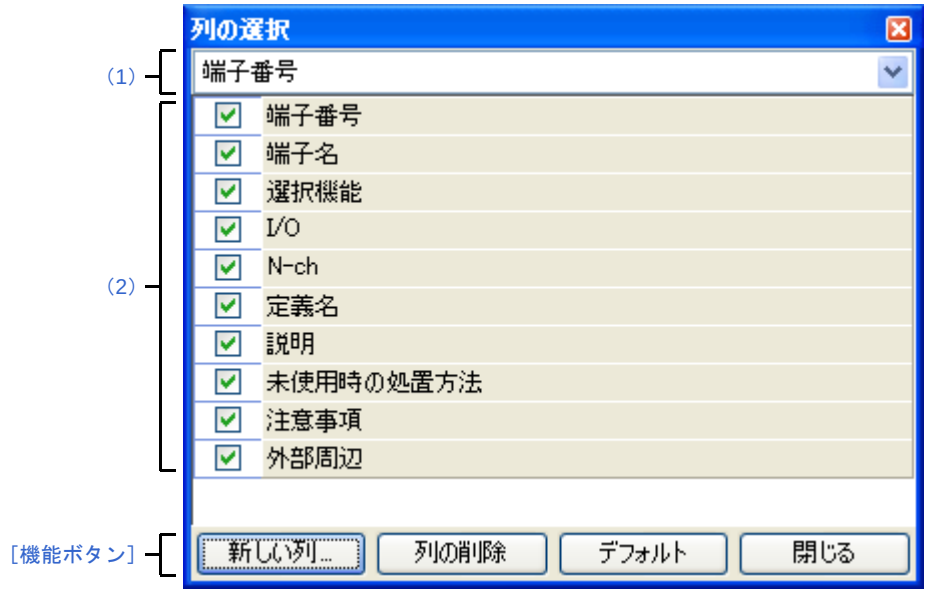
表 A—6　メッセージの文字色／背景色

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

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—21　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—22　操作対象（[端子番号] タブ）

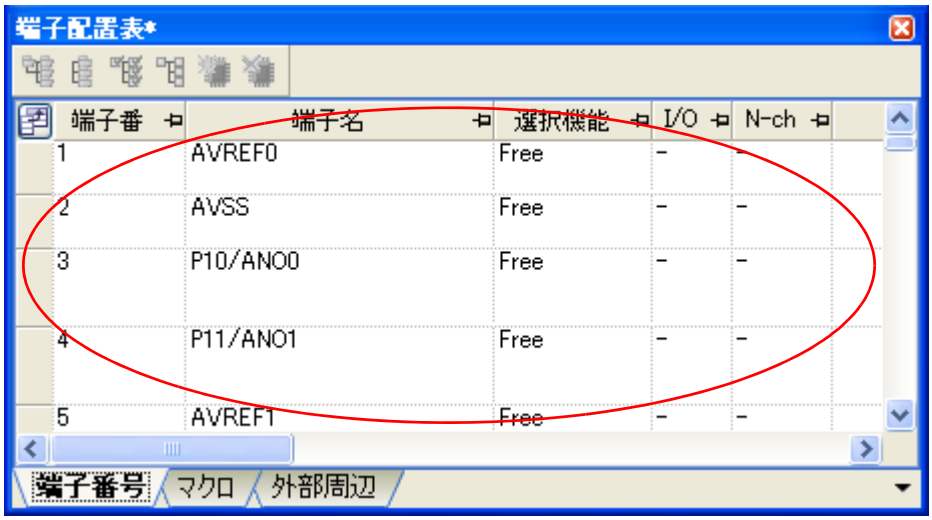


図 A—23　操作対象（[マクロ] タブ：第 1 階層）

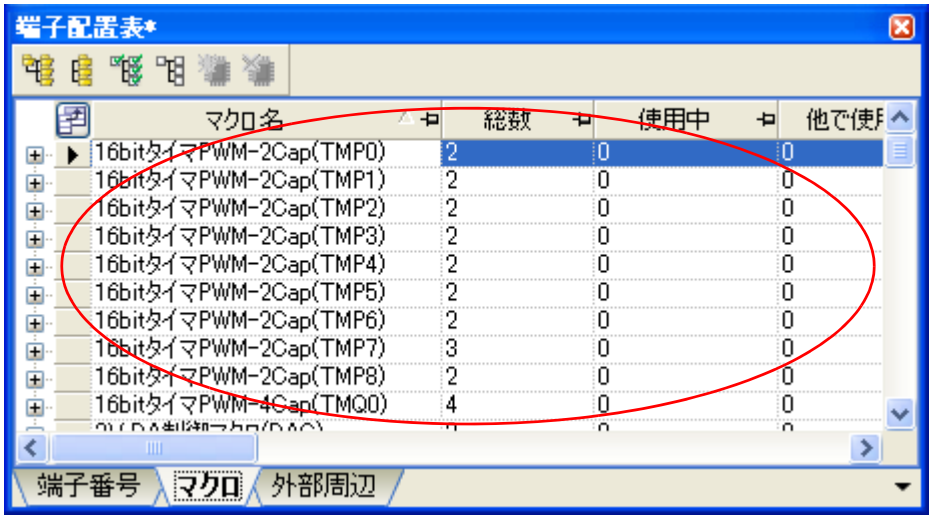


図 A―24　操作対象（[マクロ] タブ：第 2 階層）

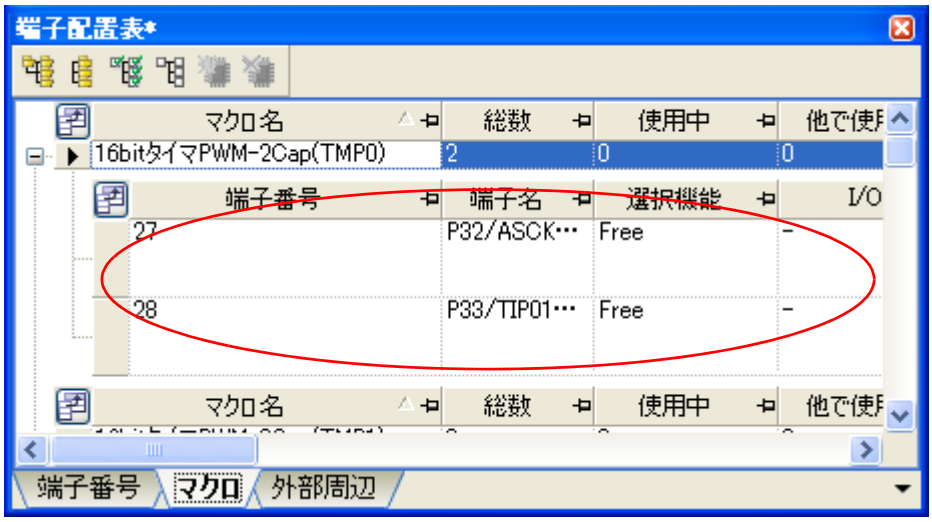


図 A―25　操作対象（[外部周辺] タブ：第 1 階層）

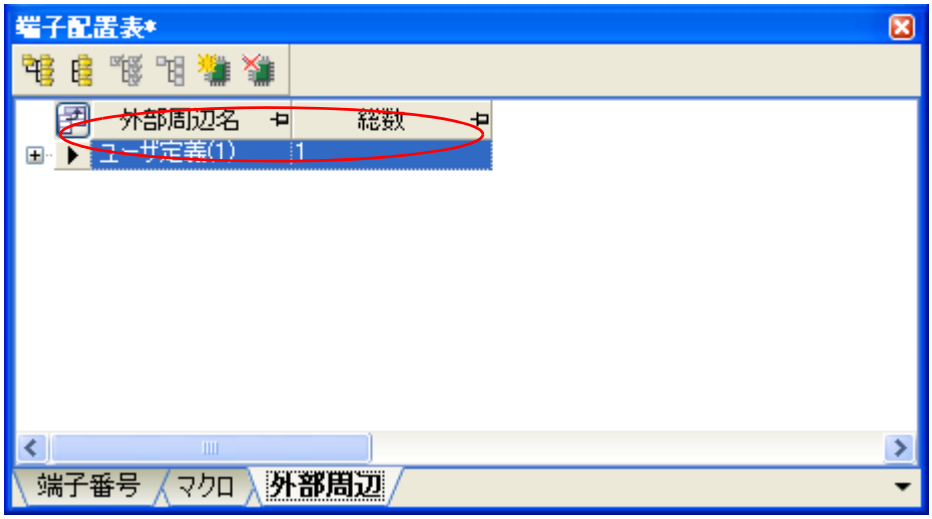
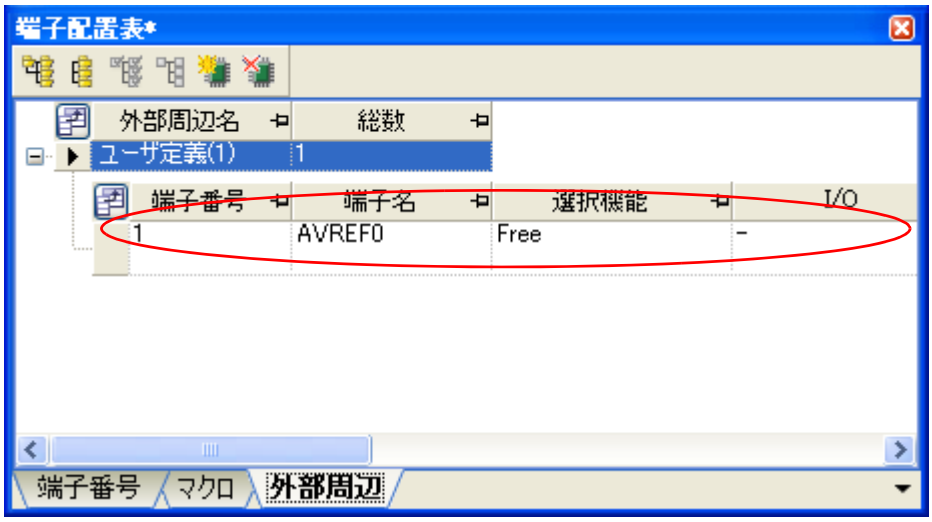


図 A―26　操作対象（[外部周辺] タブ：第 2 階層）



(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。

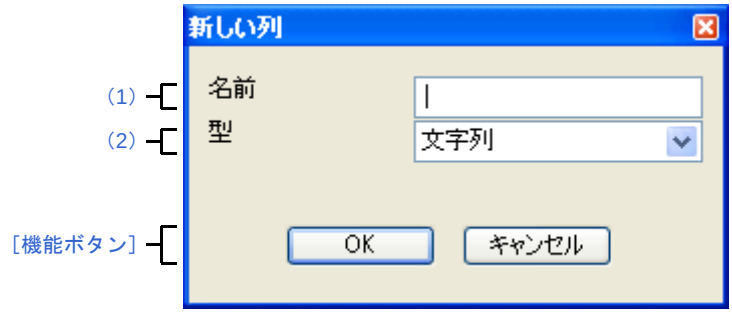
[機能ボタン]

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―27　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。

(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。

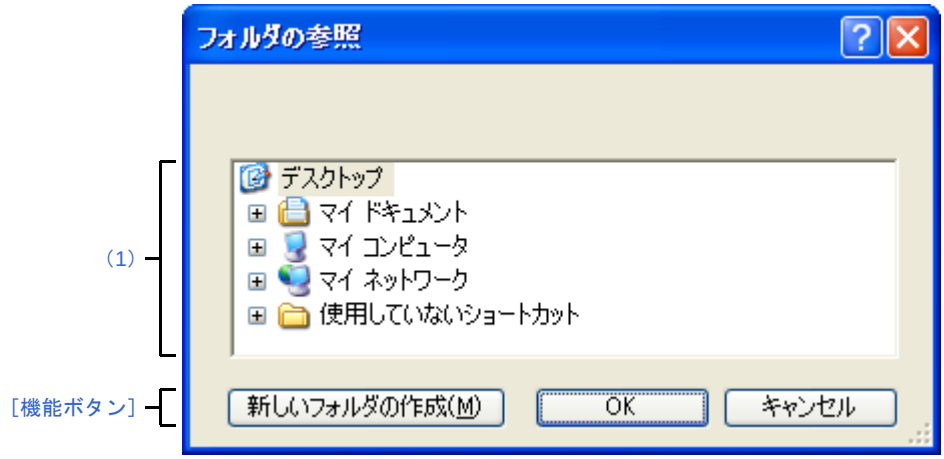
［機能ボタン］

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—28　フォルダの参照 ダイアログ



ここでは，次の項目について説明します。

- ［オープン方法］
- ［各エリアの説明］
- ［機能ボタン］

［オープン方法］

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

［各エリアの説明］

（1）保存する場所エリア

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

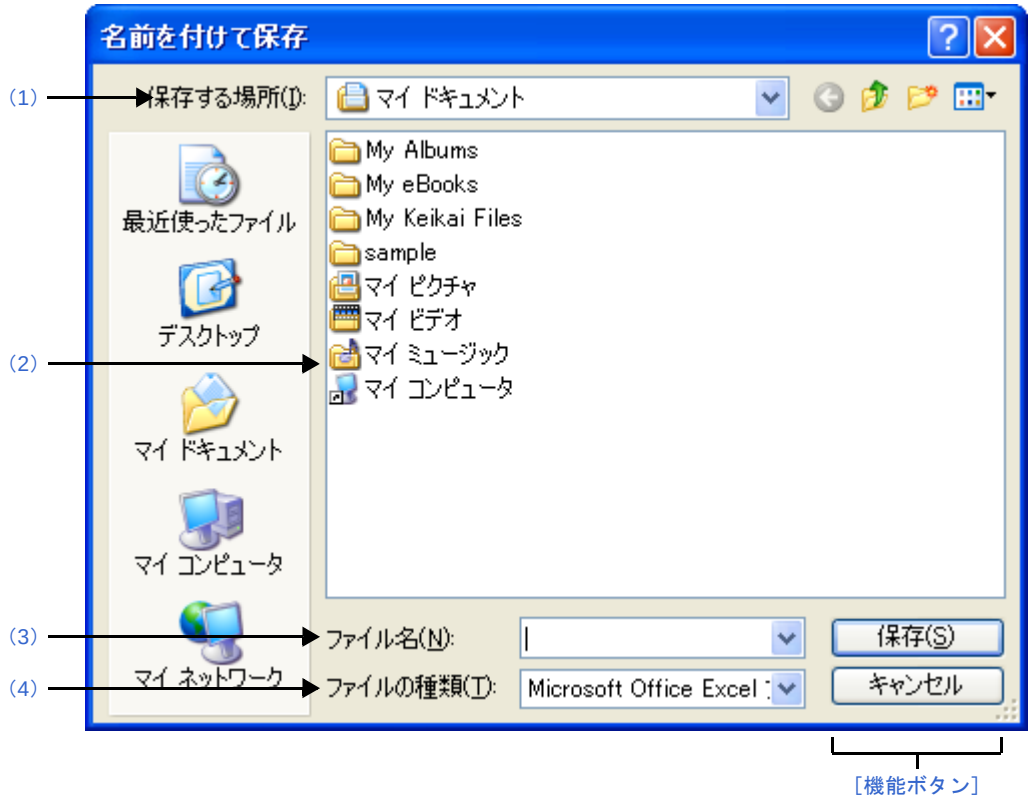
［機能ボタン］

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

名前を付けて保存 ダイアログ

ファイル（レポート・ファイルなど）に名前を付けて保存します。

図 A—29　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，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	option.asm	オプション・バイト，MINICUBE2 用 ROM 確保
	option.inc	オプション・バイトへの設定値マクロを定義
	systeminit.c	各周辺機能の初期化関数コール CG_ReadResetSource のコール
	main.c	main 関数
	macrodriver.h	全ソース・ファイルで共通使用するマクロを定義
	user_define.h	空ファイル（ユーザ定義用）
	lk.dir	リンク・ディレクティブ

B. 2　出力ファイル

以下に，コード生成が出力するファイル（各周辺機能）を示します。

表 B—2　出力ファイル（各周辺機能）

周辺機能	ファイル名	含まれる API 関数名
システム	CG_system.c	CLOCK_Init CG_ChangeClockMode CG_ChangeFrequency CG_SelectPowerSaveMode CG_SelectStabTime CG_SelectPllMode 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	MD_INTWDT2 CLOCK_UserInit CG_ReadResetSource
	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	MD_INTNMI MD_INTP*n* MD_INTKR INTP_UserInit KEY_UserInit
	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 CSIEn_Init CSIEn_Start CSIEn_Stop CSIEn_SendData CSIEn_ReceiveData CSIEn_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	MD_INTUA*n*T MD_INTUA*n*R MD_INTUB*n*TIT MD_INTUB*n*TIF MD_INTUB*n*TIR MD_INTUB*n*TIRE MD_INTUB*n*TITO MD_INTCB*n*T MD_INTCB*n*R MD_INTUC*n*T MD_INTUC*n*R MD_INTCE*n*T MD_INTCE*n*TIOF MD_INTCF*n*T MD_INTCF*n*R MD_INTIIC*n* UARTAn_UserInit UARTAn_SendEndCallback UARTAn_ReceiveEndCallback UARTAn_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	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 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
	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	MD_INTAD AD_UserInit
	CG_ad.h	—

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_Init 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 TAAn_Init TAAn_Start TAAn_Stop TAAn_ChangeTimerCondition TAAn_ControlOutputToggle TAAn_GetPulseWidth TAAn_GetFreeRunningValue TAAn_ChangeDuty TAAn_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

周辺機能	ファイル名	含まれる API 関数名
タイマ	CG_timer.c	TMT0_GetPulseWidth TMT0_GetFreeRunningValue TMT0_ChangeDuty TMT0_SoftwareTriggerOn TMT0_EnableHold TMT0_DisableHold TMT0_ChangeCountValue TMMn_Init TMMn_Start TMMn_Stop TMMn_ChangeTimerCondition
	CG_timer_user.c	MD_INTTP*n*OV MD_INTTP*n*CC*m* MD_INTTQ0OV MD_INTTQ0CC*m* MD_INTTAA*n*OV MD_INTTAA*n*CC*m* MD_INTTAB*n*OV MD_INTTAB*n*CC*m* MD_INTTT0EC MD_INTTT0OV MD_INTTT0CC*m* MD_INTTM*n*EQ0 TMPn_UserInit TMQ0_UserInit TAAn_UserInit TABn_UserInit TMT0_UserInit TMMn_UserInit
	CG_timer.h	—
時計タイマ	CG_wt.c	WT_Init WT_Start WT_Stop
	CG_wt_user.c	MD_INTWT MD_INTWTI WT_UserInit
	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

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_ConstPeriodInterruptEnable 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	MD_INTRTC*n* RTC_UserInit
	CG_rtc.h	―
リアルタイム出力機能	CG_rto.c	RTOn_Init RTOn_Enable RTOn_Disable RTOn_Set2BitsData RTOn_Set4BitsData RTOn_Set6BitsData RTOn_Set8BitsData RTOn_SetHigh2BitsData RTOn_SetLow2BitsData RTOn_SetHigh4BitsData RTOn_SetLow4BitsData RTOn_GetValue
	CG_rto_user.c	RTOn_UserInit
	CG_rto.h	―
DMA コントローラ	CG_dma.c	DMAn_Init DMAn_Enable DMAn_Disable

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	DMAn_CheckStatus DMAn_SetData DMAn_SoftwareTriggerOn
	CG_dma_user.c	MD_INTDMA*n* DMAn_UserInit
	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	MD_INTLVI LVI_UserInit
	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。

- ローカル変数へのポインタ名

  先頭に“p”を付与し，すべて小文字。

- グローバル変数へのポインタ名

  先頭に“gp”を付与し，構成単語の先頭のみ大文字。

- 列挙指定子 enum の要素名

  すべて大文字。

C. 2　出力関数

以下に，コード生成が出力する API 関数の一覧を示します。

表 C—1　API 関数一覧

周辺機能	API 関数名	機能概要
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。

周辺機能	API 関数名	機能概要
システム	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_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。
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*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	マスカブル割り込みの受け付けを禁止／許可します。
	INTPn_Disable	マスカブル割り込み（外部割込み要求）INTP*n* の受け付けを禁止します。

周辺機能	API 関数名	機能概要
割り込み	INTPn_Enable	マスカブル割り込み（外部割込み要求）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。
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 UARTA*n* 送信を開始します。
	UARTAn_ReceiveData	データの UARTA*n* 受信を開始します。
	UARTAn_SendEndCallback	UARTA*n* の連続送信許可割り込み INTUA*n*T の発生に伴う処理を行います。
	UARTAn_ReceiveEndCallback	UARTA*n* の受信終了割り込み INTUA*n*R の発生に伴う処理を行います。
	UARTAn_ErrorCallback	UARTA*n* 受信エラー割り込み INTUA*n*R 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 UARTB*n* 送信を開始します。
	UARTBn_ReceiveData	データの UARTB*n* 受信を開始します。
	UARTBn_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。
	UARTBn_ReceiveEndCallback	UARTB*n* の受信完了割り込み INTUB*n*TIR 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_SingleErrorCallback	受信エラー割り込み INTUB*n*TIRE（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 INTUB*n*TIRE（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 INTUB*n*TITO 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 UARTC*n* 送信を開始します。
	UARTCn_ReceiveData	データの UARTC*n* 受信を開始します。
	UARTCn_SendEndCallback	UARTC*n* の連続送信許可割り込み INTUC*n*T の発生に伴う処理を行います。
	UARTCn_ReceiveEndCallback	UARTC*n* の受信終了割り込み INTUC*n*R の発生に伴う処理を行います。
	UARTCn_ErrorCallback	UARTC*n* 受信エラー割り込み INTUC*n*R 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。

周辺機能	API 関数名	機能概要
シリアル	CSIBn_SendEndCallback	CSIB*n* の受信終了割り込み INTCB*n*R，および CSIB*n* の連続送信書き込み許可割り込み INTCB*n*T の発生に伴う処理を行います。
	CSIBn_ReceiveEndCallback	CSIB*n* の受信終了割り込み INTCB*n*R の発生に伴う処理を行います。
	CSIBn_ErrorCallback	CSIB*n* の受信エラー割り込み INTCB*n*R（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	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。
	CSIEn_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。
	CSIEn_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。
	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	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。
	CSIFn_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。

周辺機能	API 関数名	機能概要
シリアル	CSIFn_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。
	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。
	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	IIC0*n* 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 IIC0*n* マスタ送信を開始します。
	IIC0n_MasterReceiveStart	データの IIC0*n* マスタ受信を開始します。
	IIC0n_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。
	IIC0n_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。
	IIC0n_MasterErrorCallback	IIC0*n* 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 IIC0*n* スレーブ送信を開始します。
	IIC0n_SlaveReceiveStart	データの IIC0*n* スレーブ受信を開始します。
	IIC0n_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。
	IIC0n_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。
	IIC0n_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。
	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。

周辺機能	API 関数名	機能概要
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。
	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（ソフトウエア・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。

周辺機能	API 関数名	機能概要
タイマ	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（ソフトウエア・トリガ）を発生させます。
	TAAn_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。
	TAAn_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。
	TAAn_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。
	TAAn_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。
	TAAn_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。
	TAAn_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。
	TAAn_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。
	TAAn_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。
	TAAn_ChangeDuty	PWM 信号のデューティ比を変更します。
	TAAn_SoftwareTriggerOn	タイマ出力のためのトリガ（ソフトウエア・トリガ）を発生させます。
	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。

周辺機能	API 関数名	機能概要
タイマ	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（ソフトウエア・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（ソフトウエア・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。

周辺機能	API 関数名	機能概要
タイマ	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。
	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。

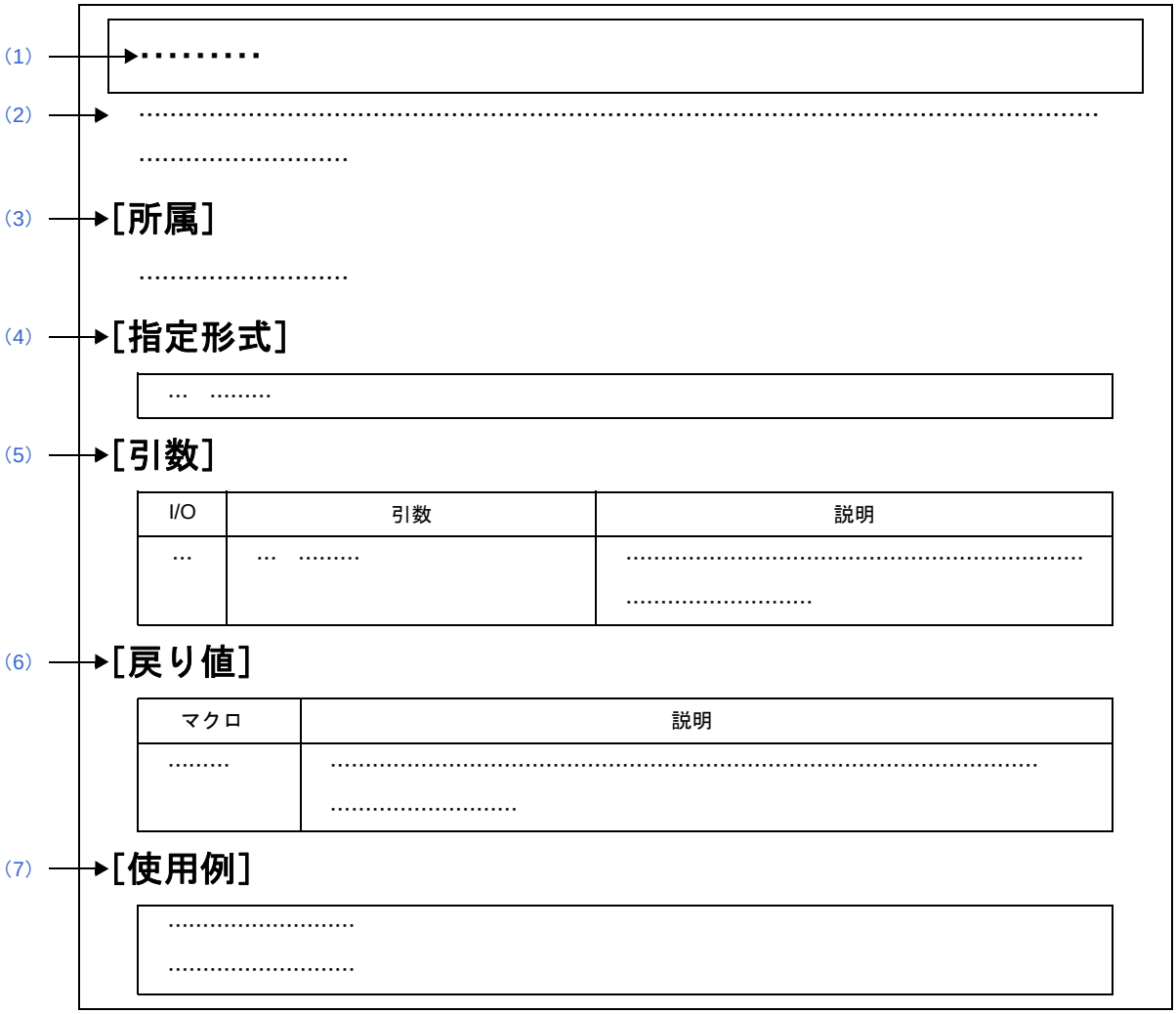
周辺機能	API 関数名	機能概要
リアルタイム・カウンタ	RTC_AlarmGet	アラームの発生条件（曜日，時，分）を読み出します。
	RTC_IntervalStart	インターバル割り込み機能を開始します。
	RTC_IntervalStop	インターバル割り込み機能を終了します。
	RTC_IntervalInterruptEnable	割り込み INTRTC2 の発生周期を設定したのち，インターバル割り込み機能を開始します。
	RTC_IntervalInterruptDisable	インターバル割り込み機能を終了します。
	RTC_RC1CK1HZ_OutputEnable	RC1CK1HZ 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。
	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	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。

周辺機能	API 関数名	機能概要
リアルタイム出力機能	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。
DMA コントローラ	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	チャネル *n* を動作許可状態に設定します。
	DMAn_Disable	チャネル *n* 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，ソフトウエア・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_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 *clock*;	分周比の種類 SYSTEMCLOCK : fxx SYSONEHALF : fxx/2 SYSONEFOURTH : fxx/4 SYSONEEIGHTH : 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： 2^10/fx STLEVEL1： 2^11/fx STLEVEL2： 2^12/fx STLEVEL3： 2^13/fx STLEVEL4： 2^14/fx STLEVEL5： 2^15/fx STLEVEL6： 2^16/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 逓倍（PLL 機能使用時） SYS8PLL： 8 逓倍（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 逓倍への変更はできません。
MD_ERROR3	異常終了【E/Sx3-H】 - 8 逓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 *sscgmode*;	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_ChangeP*mn*Input

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_ChangeP*mn*Input ( 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_ChangeP*mn*Output

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・ドレー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・ドレーン出力：なし】

```
#include    "CG_macrodriver.h"
void    PORT_ChangeP*mn*Output ( BOOL initialvalue );
```

- 【N-ch オープン・ドレーン出力：あり】

```
#include    "CG_macrodriver.h"
void    PORT_ChangeP*mn*Output ( 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 *enablench*;	出力モードの種類 MD_TRUE： N-ch オープン・ドレーン出力（V_{DD} 耐圧）モード MD_FALSE： 通常出力モード
I	BOOL *initialvalue*;	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・ドレーン出力：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・ドレーン出力：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・ドレーン出力（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	マスカブル割り込みの受け付けを禁止／許可します。
INTPn_Disable	マスカブル割り込み（外部割込み要求）INTP*n* の受け付けを禁止します。
INTPn_Enable	マスカブル割り込み（外部割込み要求）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

マスカブル割り込み（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    INTPn_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

マスカブル割り込み（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    INTPn_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 UARTA*n* 送信を開始します。
UARTAn_ReceiveData	データの UARTA*n* 受信を開始します。
UARTAn_SendEndCallback	UARTA*n* の連続送信許可割り込み INTUA*n*T の発生に伴う処理を行います。
UARTAn_ReceiveEndCallback	UARTA*n* の受信終了割り込み INTUA*n*R の発生に伴う処理を行います。
UARTAn_ErrorCallback	UARTA*n* 受信エラー割り込み INTUA*n*R（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 UARTB*n* 送信を開始します。
UARTBn_ReceiveData	データの UARTB*n* 受信を開始します。
UARTBn_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。
UARTBn_ReceiveEndCallback	UARTB*n* の受信完了割り込み INTUB*n*TIR の発生に伴う処理を行います。
UARTBn_SingleErrorCallback	受信エラー割り込み INTUB*n*TIRE（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 INTUB*n*TIRE（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 INTUB*n*TITO 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 UARTC*n* 送信を開始します。
UARTCn_ReceiveData	データの UARTC*n* 受信を開始します。
UARTCn_SendEndCallback	UARTC*n* の連続送信許可割り込み INTUC*n*T の発生に伴う処理を行います。
UARTCn_ReceiveEndCallback	UARTC*n* の受信終了割り込み INTUC*n*R の発生に伴う処理を行います。
UARTCn_ErrorCallback	UARTC*n* 受信エラー割り込み INTUC*n*R（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	CSIB*n* の受信終了割り込み INTCB*n*R，および CSIB*n* の連続送信書き込み許可割り込み INTCB*n*T の発生に伴う処理を行います。
CSIBn_ReceiveEndCallback	CSIB*n* の受信終了割り込み INTCB*n*R の発生に伴う処理を行います。
CSIBn_ErrorCallback	CSIB*n* の受信エラー割り込み INTCB*n*R（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	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。

API 関数名	機能概要
CSIEn_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。
CSIEn_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。
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	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。
CSIFn_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。
CSIFn_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。
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。
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	IIC0*n* 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 IIC0*n* マスタ送信を開始します。
IIC0n_MasterReceiveStart	データの IIC0*n* マスタ受信を開始します。
IIC0n_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。
IIC0n_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。
IIC0n_MasterErrorCallback	IIC0*n* 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 IIC0*n* スレーブ送信を開始します。
IIC0n_SlaveReceiveStart	データの IIC0*n* スレーブ受信を開始します。
IIC0n_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。
IIC0n_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。
IIC0n_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。
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。

UARTA*n*_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，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

UARTA*n*_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，チャネル番号を意味します。

［引数］

なし

［戻り値］

なし

UARTA*n*_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，チャネル番号を意味します。

［引数］

なし

［戻り値］

なし

UARTA*n*_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，チャネル番号を意味します。

［引数］

なし

［戻り値］

なし

UARTA*n*_SendData

データの UARTA*n* 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 UARTA*n* 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. UARTA*n* 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，チャネル番号を意味します。

[引数]

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

[戻り値]

マクロ	説明
MD_OK	正常終了
MD_ARGERROR	引数の指定が不正
MD_DATAEXISTS	異常終了 - UA*n*TX 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）

UARTA*n*_ReceiveData

データの UARTA*n* 受信を開始します。

備考 1. 本 API 関数では，1 バイト単位の UARTA*n* 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 UARTA*n* 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，チャネル番号を意味します。

[引数]

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

[戻り値]

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

UARTA*n*_SendEndCallback

UARTA*n* の連続送信許可割り込み INTUA*n*T の発生に伴う処理を行います。

備考 本 API 関数は，UARTA*n* の連続送信許可割り込み INTUA*n*T に対応した割り込み処理 MD_INTUA*n*T 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 UARTA*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    UARTAn_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

UARTA*n*_ReceiveEndCallback

UARTA*n* の受信終了割り込み INTUA*n*R の発生に伴う処理を行います。

備考 本 API 関数は，UARTA*n* の受信終了割り込み INTUA*n*R に対応した割り込み処理 MD_INTUA*n*R 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 UARTA*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    UARTAn_ReceiveEndCallback ( 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

UARTA*n*_ErrorCallback

UARTA*n* 受信エラー割り込み INTUA*n*R（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，UARTA*n* 受信エラー割り込み INTUA*n*R に対応した割り込み処理 MD_INTUA*n*R 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 *err_type*;	エラー割り込みの発生要因 00000*xx*1B：オーバラン・エラー 00000*x*1*x*B：フレーミング・エラー 000001*xx*B：パリティ・エラー

[戻り値]

なし

UARTA*n*_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，UARTA*n* の受信エラー割り込み INTUA*n*R に対応した割り込み処理 MD_INTUA*n*R のコールバック・ルーチン（UARTAn_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    UARTAn_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 *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    UARTBn_Init ( 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

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，UARTBn_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    UARTBn_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

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，チャネル番号を意味します。

［引数］

なし

［戻り値］

なし

UARTB*n*_SendData

データの UARTB*n* 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 UARTB*n* 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. UARTB*n* 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. UARTB*n* 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，チャネル番号を意味します。

[引数]

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

[戻り値]

マクロ	説明
MD_OK	正常終了
MD_ARGERROR	引数の指定が不正 - *txnum* が送信 FIFO トリガ数の倍数でない
MD_DATAEXISTS	異常終了 - UB*n*TX 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）

UARTB*n*_ReceiveData

データの UARTB*n* 受信を開始します。

備考 1. 本 API 関数では，1 バイト単位の UARTB*n* 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 UARTB*n* 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，チャネル番号を意味します。

[引数]

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

[戻り値]

マクロ	説明
MD_OK	正常終了
MD_ARGERROR	引数の指定が不正 - *rxnum* 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（UARTBn_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    UARTBn_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

UARTB*n*_ReceiveEndCallback

UARTB*n* の受信完了割り込み INTUB*n*TIR の発生に伴う処理を行います。

備考 本 API 関数は，UARTB*n* の受信完了割り込み INTUB*n*TIR に対応した割り込み処理 MD_INTUB*n*TIR 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 UARTB*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    UARTBn_ReceiveEndCallback ( 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

UARTB*n*_SingleErrorCallback

受信エラー割り込み INTUB*n*TIRE（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，UARTB*n*（シングル・モード）の受信エラー割り込み INTUBA*n*TIRE に対応した割り込み処理 MD_INTUB*n*TIRE 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，チャネル番号を意味します。

[引数]

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

[戻り値]

なし

UARTB*n*_FIFOErrorCallback

受信エラー割り込み INTUB*n*TIRE（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，UARTB*n*（FIFO モード）の受信エラー割り込み INTUBA*n*TIRE に対応した割り込み処理 MD_INTUB*n*TIRE 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，チャネル番号を意味します。

[引数]

I/O	引数	説明
O	UCHAR *err_type1*;	受信エラー割り込みの発生要因 00001000B：オーバラン・エラー
O	UCHAR *err_type2*;	受信エラー割り込みの発生要因 000000*x*1B：フレーミング・エラー 0000001*x*B：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    UARTBn_TimeoutErrorCallback ( 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

UARTB*n*_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，UARTB*n* の受信エラー割り込み INTUB*n*TIRE に対応した割り込み処理 MD_INTUB*n*TIRE 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

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    UARTCn_Init ( 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

UARTC*n*_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，チャネル番号を意味します。

［引数］

なし

［戻り値］

なし

UARTC*n*_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，チャネル番号を意味します。

［引数］

なし

［戻り値］

なし

UARTC*n*_SendData

データの UARTC*n* 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 UARTC*n* 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. UARTC*n* 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，チャネル番号を意味します。

[引数]

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

[戻り値]

マクロ	説明
MD_OK	正常終了
MD_ARGERROR	引数の指定が不正
MD_DATAEXISTS	異常終了 - UC*n*TX 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）

UARTC*n*_ReceiveData

データの UARTC*n* 受信を開始します。

備考 1. 本 API 関数では，1 バイト単位の UARTC*n* 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 UARTC*n* 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，チャネル番号を意味します。

[引数]

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

[戻り値]

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

UARTC*n*_SendEndCallback

UARTC*n* の連続送信許可割り込み INTUC*n*T の発生に伴う処理を行います。

備考 本 API 関数は，UARTC*n* の連続送信許可割り込み INTUC*n*T に対応した割り込み処理 MD_INTUC*n*T 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 UARTC*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    UARTCn_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

UARTC*n*_ReceiveEndCallback

UARTC*n* の受信終了割り込み INTUC*n*R の発生に伴う処理を行います。

備考 本 API 関数は，UARTC*n* の受信終了割り込み INTUC*n*R に対応した割り込み処理 MD_INTUC*n*R 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 UARTC*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    UARTCn_ReceiveEndCallback ( 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

UARTC*n*_ErrorCallback

UARTC*n* 受信エラー割り込み INTUC*n*R（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，UARTC*n* 受信エラー割り込み INTUC*n*R に対応した割り込み処理 MD_INTUC*n*R 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 *err_type*;	エラー割り込みの発生要因 00000*xx*1B：オーバラン・エラー 00000*x*1*x*B：フレーミング・エラー 000001*xx*B：パリティ・エラー

[戻り値]

なし

UARTC*n*_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，UARTC*n* の受信エラー割り込み INTUC*n*R に対応した割り込み処理 MD_INTUC*n*R 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 *rx_data*;	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    CSIBn_Init ( 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

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，CSIBn_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    CSIBn_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

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    CSIBn_Stop ( void );
```

備考 *n* は，チャネル番号を意味します。

［引数］

なし

［戻り値］

なし

CSIB*n*_SendData

データの CSIB*n* 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 CSIB*n* 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. CSIB*n* 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，チャネル番号を意味します。

[引数]

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

[戻り値]

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

CSIB*n*_ReceiveData

データの CSIB*n* 受信を開始します。

備考 1. 本 API 関数では，1 バイト単位の CSIB*n* 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. CSIB*n* 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，チャネル番号を意味します。

[引数]

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

[戻り値]

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

CSIB*n*_SendReceiveData

データの CSIB*n* 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 CSIB*n* 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 CSIB*n* 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. CSIB*n* 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，チャネル番号を意味します。

[引数]

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

CSIB*n*_SendEndCallback

CSIB*n* の受信終了割り込み INTCB*n*R，および CSIB*n* の連続書き込み許可割り込み INTCB*n*T の発生に伴う処理を行います。

備考 本 API 関数は，CSIB*n* の受信終了割り込み INTCB*n*R に対応した割り込み処理 MD_INTCB*n*R，および CSIB*n* の連続書き込み許可割り込み INTCB*n*T に対応した割り込み処理 MD_INTCB*n*T 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 CSIB*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    CSIBn_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

CSIB*n*_ReceiveEndCallback

CSIB*n* の受信終了割り込み INTCB*n*R の発生に伴う処理を行います。

備考 本 API 関数は，CSIB*n* の受信終了割り込み INTCB*n*R に対応した割り込み処理 MD_INTCB*n*R 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 CSIB*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    CSIBn_ReceiveEndCallback ( 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

CSIB*n*_ErrorCallback

CSIB*n* の受信エラー割り込み INTCB*n*R（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，CSIB*n* の受信エラー割り込み INTCB*n*R に対応した割り込み処理 MD_INTCB*n*R 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    CSIEn_Init ( 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

CSIE*n*_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    CSIEn_Start ( void );
```

備考 *n* は，チャネル番号を意味します。

［引数］

なし

［戻り値］

なし

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，チャネル番号を意味します。

［引数］

なし

［戻り値］

なし

CSIE*n*_SendData

データの CSIE*n* 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*n* 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*n* 送信を行う際には，本 API 関数の呼び出し以前に CSIEn_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，チャネル番号を意味します。

[引数]

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

[戻り値]

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

CSIE*n*_ReceiveData

データの CSIE*n* 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*n* 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*n* 受信を行う際には，本 API 関数の呼び出し以前に CSIEn_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，チャネル番号を意味します。

[引数]

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

[戻り値]

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

CSIE*n*_SendReceiveData

データの CSIE*n* 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*n* 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*n* 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*n* 送受信を行う際には，本 API 関数の呼び出し以前に CSIEn_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，チャネル番号を意味します。

[引数]

I/O	引数	説明
O	UCHAR **txbuf*;	受信したデータを格納するバッファへのポインタ
I	USHORT *txnum*;	送受信するデータの総数
I	UCHAR **rxbuf*;	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（CSIEn_SendData，または CSIEn_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    CSIEn_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

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（CSIEn_ReceiveData，または CSIEn_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    CSIEn_ErrorCallback ( 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

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，CSIFn_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    CSIFn_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

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，チャネル番号を意味します。

［引数］

なし

［戻り値］

なし

CSIF*n*_SendData

データの CSIF*n* 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 CSIF*n* 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. CSIF*n* 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，チャネル番号を意味します。

[引数]

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

[戻り値]

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

CSIF*n*_ReceiveData

データの CSIF*n* 受信を開始します。

備考 1. 本 API 関数では，1 バイト単位の CSIF*n* 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. CSIF*n* 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，チャネル番号を意味します。

[引数]

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

[戻り値]

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

CSIF*n*_SendReceiveData

データの CSIF*n* 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 CSIF*n* 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 CSIF*n* 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. CSIF*n* 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，チャネル番号を意味します。

[引数]

I/O	引数	説明
O	UCHAR *txbuf;	受信したデータを格納するバッファへのポインタ
I	USHORT txnum;	送受信するデータの総数
I	UCHAR *rxbuf;	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（CSIFn_SendData，または CSIFn_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    CSIFn_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

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。

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（CSIFn_ReceiveData，または CSIFn_SendReceiveData の引数 *rxnum* で指定された総数の CSIF*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    CSIFn_ReceiveEndCallback ( 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

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。

備考 本 API 関数は，CSIF*n* の受信エラー割り込み INTCF*n*R に対応した割り込み処理 MD_INTCF*n*R 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，チャネル番号を意味します。

［引数］

なし

［戻り値］

なし

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。

[所属]

CG_serial.c

[指定形式]

```
void    IIC0n_Init ( 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

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。

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，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

IIC0*n*_Stop

IIC0*n* 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，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

IIC0*n*_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，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

IIC0*n*_MasterSendStart

データの IIC0*n* 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 IIC0*n* 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]

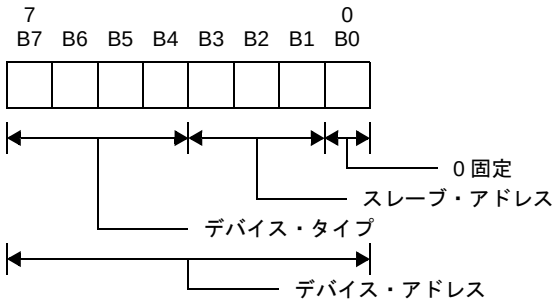
```
#include    "CG_macrodriver.h"
MD_STATUS   IIC0n_MasterSendStart ( UCHAR adr, UCHAR *txbuf, USHORT txnum, UCHAR wait );
```

備考 *n* は，チャネル番号を意味します。

[引数]

I/O	引数	説明
I	UCHAR *adr*;	デバイス・アドレス
I	UCHAR **txbuf*;	送信するデータを格納したバッファへのポインタ
I	USHORT *txnum*;	送信するデータの総数
I	UCHAR *wait*;	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

IIC0*n*_MasterReceiveStart

データの IIC0*n* 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 IIC0*n* 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]

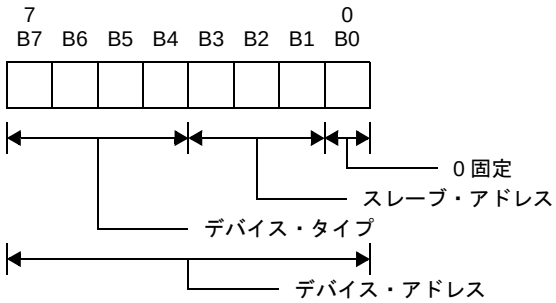
```
#include    "CG_macrodriver.h"
MD_STATUS   IIC0n_MasterReceiveStart ( UCHAR adr, UCHAR *rxbuf, USHORT rxnum, UCHAR wait );
```

備考 *n* は，チャネル番号を意味します。

[引数]

I/O	引数	説明
I	UCHAR *adr*;	デバイス・アドレス
O	UCHAR **rxbuf*;	受信したデータを格納するバッファへのポインタ
I	USHORT *rxnum*;	受信するデータの総数
I	UCHAR *wait*;	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

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。

備考 本 API 関数は，IIC0*n* マスタ送信の転送終了割り込み INTIIC*n* に対応した割り込み処理 MD_INTIIC*n* 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 IIC0*n* 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，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

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。

備考 本 API 関数は，IIC0*n* マスタ受信の転送終了割り込み INTIIC*n* に対応した割り込み処理 MD_INTIIC*n* 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 IIC0*n* 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，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

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。

備考 本 API 関数は，IIC0*n* の転送終了割り込み INTIIC*n* に対応した割り込み処理 MD_INTIIC*n* のコールバック・ルーチン（IIC0*n* 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，チャネル番号を意味します。

[引数]

I/O	引数	説明
O	MD_STATUS *flag*;	IIC0*n* マスタ通信エラーの発生要因 MD_SPT： IIC0*n* 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

IIC0*n*_SlaveSendStart

データの IIC0*n* 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 IIC0*n* 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，チャネル番号を意味します。

[引数]

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

[戻り値]

なし

IIC0*n*_SlaveReceiveStart

データの IIC0*n* 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 IIC0*n* 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，チャネル番号を意味します。

[引数]

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

[戻り値]

なし

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。

備考 本 API 関数は，IIC0*n* スレーブ送信の転送終了割り込み INTIIC*n* に対応した割り込み処理 MD_INTIIC*n* 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 IIC0*n* 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，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

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。

備考 本 API 関数は，IIC0*n* スレーブ受信の転送終了割り込み INTIIC*n* に対応した割り込み処理 MD_INTIIC*n* 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 IIC0*n* 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，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

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。

備考 本 API 関数は，IIC0*n* の転送終了割り込み INTIIC*n* に対応した割り込み処理 MD_INTIIC*n* のコールバック・ルーチン（IIC0*n* 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，チャネル番号を意味します。

[引数]

I/O	引数	説明
O	MD_STATUS *flag*;	IIC0*n* 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

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。

備考 本 API 関数は，IIC0*n* の転送終了割り込み INTIIC*n* に対応した割り込み処理 MD_INTIIC*n* 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 *channel*;	アナログ電圧の入力端子 ADCHANNEL*n*： 入力端子

備考 アナログ電圧の入力端子 ADCHANNEL*n* についての詳細は，ヘッダ・ファイル 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 *pftvalue*;	比較値
I	enum ADPFTMode *mode*;	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： ADA0CR*n*H ≧ ADA0PFT の際に INTAD を発生 PFTLOWER： ADA0CR*n*H ＜ 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，または 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 *n*H（ADA0CR*n*H）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

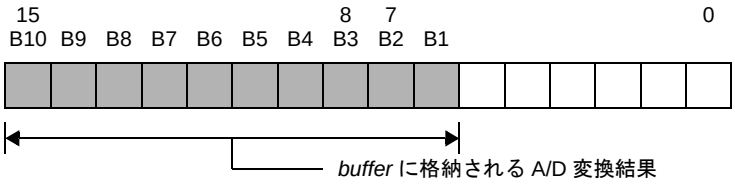
［指定形式］

```
#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

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。

［所属］

CG_ad.c

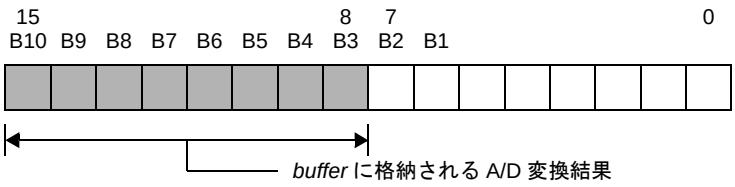
［指定形式］

```
#include    "CG_macrodriver.h"
MD_STATUS   AD_ReadByte ( UCHAR *buffer );
```

［引数］

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

備考 以下に，*buffer* に格納される A/D 変換結果を示します。



［戻り値］

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	ANO*n* 端子に出力するアナログ電圧値を設定します。

DA*n*_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，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

DA*n*_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，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

DA*n*_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，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

DA*n*_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，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

DA*n*_SetValue

ANO*n* 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，チャネル 0，およびチャネル 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_INTTP2CC0 ( void ) {    /* 割り込み INTTP2CC0 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（ソフトウエア・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（ソフトウエア・トリガ）を発生させます。
TAAn_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。
TAAn_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
TAAn_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。
TAAn_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。
TAAn_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。
TAAn_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。
TAAn_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。
TAAn_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。
TAAn_ChangeDuty	PWM 信号のデューティ比を変更します。
TAAn_SoftwareTriggerOn	タイマ出力のためのトリガ（ソフトウエア・トリガ）を発生させます。
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（ソフトウエア・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（ソフトウエア・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    TMPn_Init ( 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

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，TMPn_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    TMPn_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

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    TMPn_Stop ( void );
```

備考 *n* は，チャネル番号を意味します。

［引数］

なし

［戻り値］

なし

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 *arrar_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*（TP*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   TMPn_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： TP*n*CCR0 2： TP*n*CCR0，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

［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_TP0CCR0_VALUE >> 1; /* TMP_TP0CCR0_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_ChanneTimerCondition ( &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，チャネル番号を意味します。

[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

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*;	読み出し対象チャネル TMCHANNEL0： チャネル 0（TP*n*CCR0） TMCHANNEL1： チャネル 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（ソフトウエア・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    TMPn_SoftwareTriggerOn ( void );
```

備考 *n* は，チャネル番号を意味します。

［引数］

なし

［戻り値］

なし

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 *arrar_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_ChanneTimerCondition ( &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 *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

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（ソフトウエア・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    TAAn_Init ( 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

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，TAAn_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    TAAn_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

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    TAAn_Start ( void );
```

備考 *n* は，チャネル番号を意味します。

［引数］

なし

［戻り値］

なし

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    TAAn_Stop ( void );
```

備考 *n* は，チャネル番号を意味します。

［引数］

なし

［戻り値］

なし

TAA*n*_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 *arrar_reg* に指定された値は，TAA*n* キャプチャ／コンペア・レジスタ *m*（TAA*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   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*n*CCR0 2： TAA*n*CCR0，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*_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 TMOutput toggle, enum TMChannel channel );
```

備考 *n* は，チャネル番号を意味します。

［引数］

I/O	引数	説明
I	enum TMOutput *toggle*;	トグル制御 STANDARD： 通常のトグル動作 INVACTIVE： リセット要求 ACTIVE： セット要求 FREEZE： キープ要求
I	enum TMChannel *channel*;	対象端子 TMCHANNEL0： TOAA*n*0 端子 TMCHANNEL1： TOAA*n*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*n*_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，チャネル番号を意味します。

［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

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*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   TAAn_GetFreeRunningValue ( ULONG *count, enum TMChannel channel );
```

備考 *n* は，チャネル番号を意味します。

［引数］

I/O	引数	説明
O	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（TAA*n*CCR0） TMCHANNEL1： チャネル 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    TAAn_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（ソフトウエア・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    TABn_Init ( 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*_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，TABn_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    TABn_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    TABn_Start ( void );
```

備考 *n* は，チャネル番号を意味します。

［引数］

なし

［戻り値］

なし

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    TABn_Stop ( void );
```

備考 *n* は，チャネル番号を意味します。

［引数］

なし

［戻り値］

なし

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 *arrar_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*n*_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 TMOutput toggle, enum TMChannel channel );
```

備考 *n* は，チャネル番号を意味します。

［引数］

I/O	引数	説明
I	enum TMOutput *toggle*;	トグル制御 STANDARD： 通常のトグル動作 INVACTIVE： リセット要求 ACTIVE： セット要求 FREEZE： キープ要求
I	enum TMChannel *channel*;	対象端子 TMCHANNEL0： TOAB*n*0 端子 TMCHANNEL1： TOAB*n*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*n*_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 *activewith, ULONG *inactivewidth );
```

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	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

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*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   TABn_GetFreeRunningValue ( ULONG *count, enum TMChannel channel );
```

備考 *n* は，チャネル番号を意味します。

［引数］

I/O	引数	説明
O	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（TAB*n*CCR0） TMCHANNEL1： チャネル 1（TAB*n*CCR1） TMCHANNEL2： チャネル 2（TAB*n*CCR2) TMCHANNEL3： チャネル 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*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 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：%）
I	UCHAR *array_num*;	変更対象レジスタ 1： TAB*n*CCR1 2： TAB*n*CCR1，TAB*n*CCR2 3： TAB*n*CCR1，TAB*n*CCR2，TAB*n*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（ソフトウエア・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    TABn_SoftwareTriggerOn ( void );
```

備考 *n* は，チャネル番号を意味します。

［引数］

なし

［戻り値］

なし

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 *arrar_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*（TT0CCR*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   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 *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］

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（ソフトウエア・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]

```
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]

```
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 *regvalue*;	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    TMMn_Init ( 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

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，TMMn_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    TMMn_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

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    TMMn_Start ( void );
```

備考 *n* は，チャネル番号を意味します。

［引数］

なし

［戻り値］

なし

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    TMMn_Stop ( void );
```

備考 *n* は，チャネル番号を意味します。

［引数］

なし

［戻り値］

なし

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    TMMn_ChangeTimerCondition ( USHORT regvalue );
```

備考 *n* は，チャネル番号を意味します。

［引数］

I/O	引数	説明
I	USHORT *regvalue*;	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，および補正値を変更します。

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 *period*;	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ヵ月

［戻り値］

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

［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 RTCAlarmValue alarmval );
```

［引数］

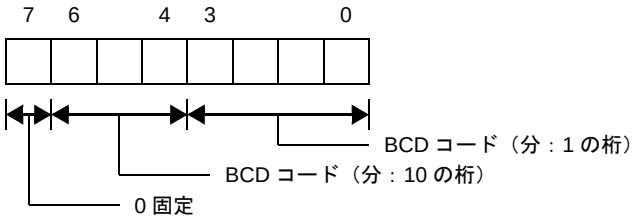
I/O	引数	説明
I	struct RTCAlarmValue *alarmval*;	アラームの発生条件（曜日，時，分）

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

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

- Alarmwm（分）

以下に，構成メンバ Alarmwm の各ビットに対する意味を示します。



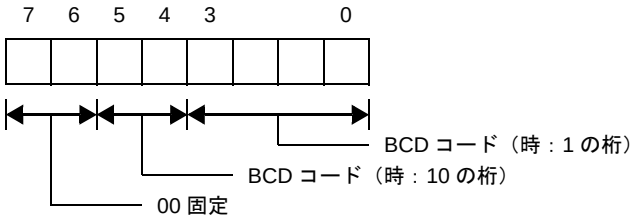
- Alarmwh（時）

以下に，構成メンバ Alarmwh の各ビットに対する意味を示します。

なお，ビット 5 は，リアルタイム・カウンタが 12 時間制の場合，以下の意味となります。

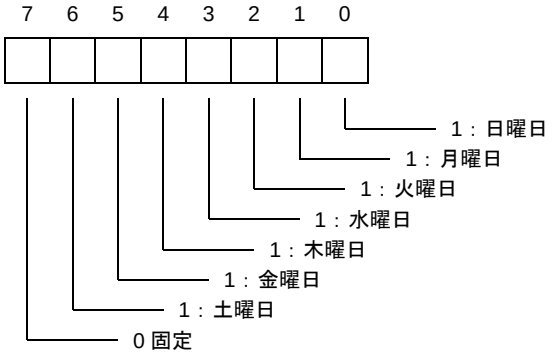
0： 午前

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  RTCAlarmValue   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  RTCAlarmValue   alarmval;
    .........
    RTC_AlarmEnable ();             /* アラーム割り込み機能の開始 */
    .........
    alarmval.Alarmww = 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 RTCAlarmValue *alarmval );
```

備考 アラームの発生条件 RTCAlarmValue 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 RTCAlarmValue *alarmval;	読み出した発生条件を格納する構造体へのポインタ

［戻り値］

なし

［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  RTCAlarmValue   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 *interval*;	割り込み INTRTC2 の発生周期 INTERVAL0： 2^6/fRTC INTERVAL1： 2^7/fRTC INTERVAL2： 2^8/fRTC INTERVAL3： 2^9/fRTC INTERVAL4： 2^10/fRTC INTERVAL5： 2^11/fRTC INTERVAL6： 2^12/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，および補正値を変更します。

［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*;	時計誤差の補正値

備考 本 API 関数では，補正値 *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    RTOn_Init ( 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

RTO*n*_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    RTOn_Enable ( 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

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    RTOn_Disable ( 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

RTO*n*_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]

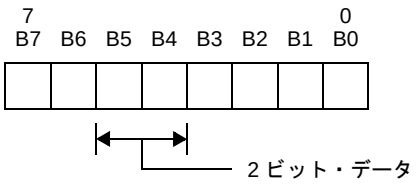
```
#include    "CG_macrodriver.h"
void    RTOn_Set2BitsData ( UCHAR data );
```

備考 *n* は，チャネル番号を意味します。

[引数]

I/O	引数	説明
I	UCHAR *data*;	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

RTO*n*_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]

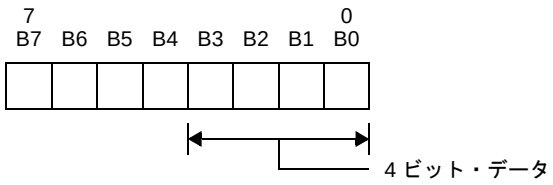
```
#include    "CG_macrodriver.h"
void    RTOn_Set4BitsData ( UCHAR data );
```

備考 *n* は，チャネル番号を意味します。

[引数]

I/O	引数	説明
I	UCHAR *data*;	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

RTO*n*_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]

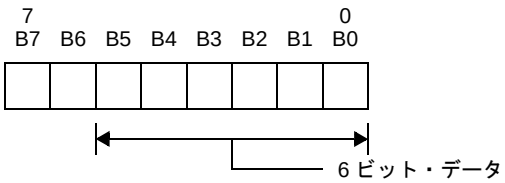
```
#include    "CG_macrodriver.h"
void    RTOn_Set6BitsData ( UCHAR data );
```

備考 *n* は，チャネル番号を意味します。

[引数]

I/O	引数	説明
I	UCHAR *data*;	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

RTO*n*_Set8BitsData

リアルタイム出力する 8 ビット・データを設定します。

[所属]

CG_rto.c

[指定形式]

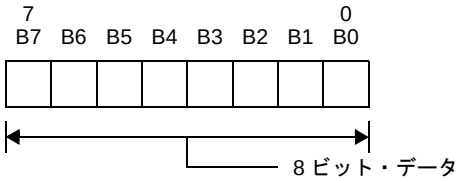
```
#include    "CG_macrodriver.h"
void    RTOn_Set8BitsData ( UCHAR data );
```

備考 *n* は，チャネル番号を意味します。

[引数]

I/O	引数	説明
I	UCHAR *data*;	8 ビット・データ

備考 本 API 関数では，0 ～ 7 ビットに設定された値を 8 ビット・データとして扱います。



[戻り値]

なし

RTO*n*_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]

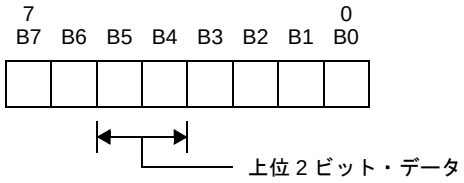
```
#include    "CG_macrodriver.h"
void    RTOn_SetHigh2BitsData ( UCHAR data );
```

備考 *n* は，チャネル番号を意味します。

[引数]

I/O	引数	説明
I	UCHAR *data*;	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

RTO*n*_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]

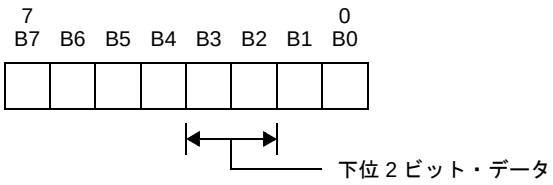
```
#include    "CG_macrodriver.h"
void    RTOn_SetLow2BitsData ( UCHAR data );
```

備考 *n* は，チャネル番号を意味します。

[引数]

I/O	引数	説明
I	UCHAR *data*;	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

RTO*n*_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]

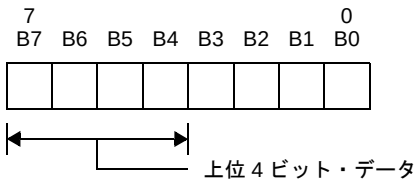
```
#include    "CG_macrodriver.h"
void    RTOn_SetHigh4BitsData ( UCHAR data );
```

備考 *n* は，チャネル番号を意味します。

[引数]

I/O	引数	説明
I	UCHAR *data*;	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

RTO*n*_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]

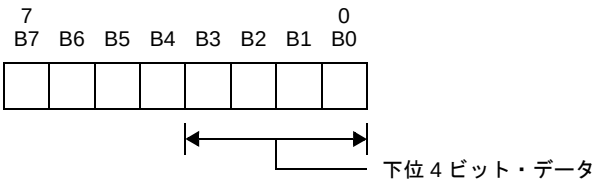
```
#include    "CG_macrodriver.h"
void    RTOn_SetLow4BitsData ( UCHAR data );
```

備考 *n* は，チャネル番号を意味します。

[引数]

I/O	引数	説明
I	UCHAR *data*;	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

RTO*n*_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 ) {
    RTO0_Set2BitData ( 0x30 );          /* 出力データの設定 */
    RTO0_Enable ();                     /* リアルタイム出力の許可 */
    ……
    RTO0_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;
    RTO0_GetValue ( &value );             /* 出力データの読み出し */
    value = ~value;
    RTO0_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	チャネル *n* を動作許可状態に設定します。
DMAn_Disable	チャネル *n* 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，ソフトウエア・トリガを使用します。

DMA*n*_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，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

DMA*n*_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，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

DMA*n*_Enable

チャネル *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，チャネル番号を意味します。

[引数]

なし

[戻り値]

なし

DMA*n*_Disable

チャネル *n* を動作停止状態に設定します。

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

2. 本 API 関数は，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 */
    ……
}
```

DMA*n*_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

DMA*n*_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，チャネル番号を意味します。

［引数］

I/O	引数	説明
I	UINT *srcaddr*;	転送元の RAM アドレス
I	UINT *dstaddr*;	転送先の RAM アドレス
I	UINT *count*;	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，ソフトウエア・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    DMAn_SoftwareTriggerOn ( void );
```

備考 *n* は，チャネル番号を意味します。

［引数］

なし

［戻り値］

なし

［使用例］

以下に，DMA 転送の起動要因をソフトウエア・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_{DD}）の際，電源電圧（V_{DD}）≦ 検出電圧（V_{LVI}） - 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_{EXLVI}）

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_{DD}）の際，電源電圧（V_{DD}）≦ 検出電圧（V_{LVI}） - 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_{EXLVI}）

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

【T】

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2011.04.01	―	初版発行

CubeSuite+ V1.00.00 ユーザーズマニュアル
V850 設計編

発行年月日　2011 年 4 月 1 日　Rev.1.00
発行　ルネサス エレクトロニクス株式会社
〒211-8668 神奈川県川崎市中原区下沼部 1753

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口

http://www.renesas.com

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

ルネサス エレクトロニクス販売株式会社　〒100-0004　千代田区大手町2-6-2（日本ビル）　(03)5201-5307

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

CubeSuite+ V1.00.00

ルネサス エレクトロニクス株式会社

R20UT0549JJ0100