

CubeSuite+ V1.03.00

統合開発環境

ユーザーズマニュアル 78K0 設計編

対象デバイス

78K0 マイクロコントローラ

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

ご注意書き

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

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

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

このマニュアルの使い方

このマニュアルは、78K0 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、78K0 マイクロコントローラの統合開発環境（ソフトウェア開発における、設計、実装、デバッグなどの各開発フェーズに必要なツールをプラットフォームである IDE に統合）です。統合することで、さまざまなツールを使い分ける必要がなく、本製品のみを使用して開発のすべてを行うことができます。

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

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

構成 このマニュアルは、大きく分けて次の内容で構成しています。

第 1 章 概 説

第 2 章 機能（端子配置）

第 3 章 機能（コード生成）

付録 A ウィンドウ・リファレンス

付録 B 出力ファイル

付録 C API 関数

付録 D 索 引

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

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

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

資料名	資料番号		
	和文	英文	
CubeSuite+ 統合開発環境 ユーザーズ・マニュアル	起動編	R20UT2133J	R20UT2133E
	V850 設計編	R20UT2134J	R20UT2134E
	RL78 設計編	R20UT2136J	R20UT2136E
	78K0R 設計編	R20UT2137J	R20UT2137E
	78K0 設計編	このマニュアル	R20UT2138E
	RX コーディング編	R20UT0767J	R20UT0767E
	V850 コーディング編	R20UT0553J	R20UT0553E
	コーディング編 (CX コンパイラ)	R20UT2139J	R20UT2139E
	RL78, 78K0R コーディング編	R20UT2140J	R20UT2140E
	78K0 コーディング編	R20UT2141J	R20UT2141E
	RX ビルド編	R20UT0768J	R20UT0768E
	V850 ビルド編	R20UT0557J	R20UT0557E
	ビルド編 (CX コンパイラ)	R20UT2142J	R20UT2142E
	RL78, 78K0R ビルド編	R20UT2143J	R20UT2143E
	78K0 ビルド編	R20UT0783J	R20UT0783E
	RX デバッグ編	R20UT2175J	R20UT2175E
	V850 デバッグ編	R20UT2144J	R20UT2144E
	RL78 デバッグ編	R20UT2145J	R20UT2145E
	78K0R デバッグ編	R20UT0732J	R20UT0732E
	78K0 デバッグ編	R20UT0731J	R20UT0731E
解析編	R20UT2146J	R20UT2146E	
メッセージ編	R20UT2147J	R20UT2147E	

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

目 次

第 1 章 概 説 … 7

- 1.1 概 要 … 7
- 1.2 特 長 … 7

第 2 章 機能（端子配置） … 8

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

第 3 章 機能（コード生成） … 22

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

付録 A ウィンドウ・リファレンス … 38

A.1 説 明 … 38

付録 B 出力ファイル … 89

B.1 概 要 … 89

B.2 出力ファイル … 89

付録 C API 関数 … 94

C.1 概 要 … 94

C.2 出力関数 … 94

C.3 関数リファレンス … 101

C.3.1 システム … 103

C.3.2 ポート … 114

C.3.3 割り込み … 122

C.3.4 シリアル … 133

C.3.5 オペアンプ … 173

C.3.6 コンパレータ … 183

C.3.7 A/D コンバータ … 188

C.3.8 タイマ … 198

C.3.9 ウォッチドッグ・タイマ … 233

C.3.10 リアルタイム・カウンタ … 235

C.3.11 クロック出力 … 271

C.3.12 低電圧検出回路 … 277

付録 D 索 引 … 285

第1章 概 説

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

本章では、設計ツール（端子配置／コード生成）の概要について説明します。

1.1 概 要

設計ツールは、CubeSuite+ が提供しているコンポーネントの1種であり、GUI ベースで各種情報を設定することにより、マイクロコントローラの端子配置状況（端子配置表、端子配置図）／マイクロコントローラが提供している周辺機能（クロック発生回路、I/O ポートなど）を制御するうえで必要なソース・コード（デバイス・ドライバ・プログラム：C ソース・ファイル、ヘッダ・ファイル）を出力することができます。

1.2 特 長

以下に、設計ツール（端子配置／コード生成）の特長を示します。

- コード生成機能

コード生成では、GUI ベースで設定した情報に応じたデバイス・ドライバ・プログラムを出力するだけでなく、main 関数を含んだサンプル・プログラム、リンク・ディレクティブ・ファイルなどといったビルド環境一式を出力することもできます。

- レポート機能

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

- リネーム機能

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

第2章 機能（端子配置）

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

2.1 概要

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

なお、端子配置の実践手順は、以下のとおりです。

(1) CubeSuite+ の起動

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

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

(2) プロジェクトの作成／読み込み

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

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

(3) 端子配置表 パネルのオープン

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

(a) 表示項目の選択

端子配置表に表示する項目を選択します。

(b) 表示順序の変更

端子配置表に表示する項目の順序を変更します。

(c) 列の追加

端子配置表に対する列の追加を行います。

(d) 列の削除

端子配置表に対する列の削除を行います。

(4) 端子配置図 パネルのオープン

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

(a) マイクロコントローラの形状選択

端子配置図 パネルに表示するマイクロコントローラの形状を選択します。

(b) 表示色の選択

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

(c) ポップアップ情報の選択

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

(d) 付加情報の選択

端子配置図 パネルの端子部分に表示させる情報の種類を選択します。

(5) 情報の記述

端子配置表 パネルでマイクロコントローラの各端子に関する情報を記述します。

(6) レポート・ファイルの出力

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

(a) 端子配置表の出力

端子配置表を出力します。

(b) 端子配置図の出力

端子配置図を出力します。

(7) プロジェクトの保存

プロジェクトの保存を行います。

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

2.2 端子配置表 パネルのオープン

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


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

図 2—1 端子配置表 パネルのオープン



- 備考 1.** 端子配置が未対応のマイクロコントローラがプロジェクトで定義された場合、プロジェクト・ツリー パネルの [Project name (プロジェクト)] に “[端子配置 (設計ツール)] ノード” は表示されません。
- 2.** 端子配置表 パネルは3個のタブから構成され、タブを選択することにより、“マイクロコントローラの各端子に関する情報” の表示順序が切り替わります。
- [端子番号] タブ
マイクロコントローラの各端子に関する情報を端子番号順で表示
 - [マクロ] タブ
マイクロコントローラの各端子に関する情報を周辺機能単位にグルーピングされた順序で表示
 - [外部周辺] タブ
外部周辺に接続された端子に関する情報を外部周辺部品単位にグルーピングされた順序で表示

2.2.1 表示項目の選択

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


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

図 2—2 表示項目の選択



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

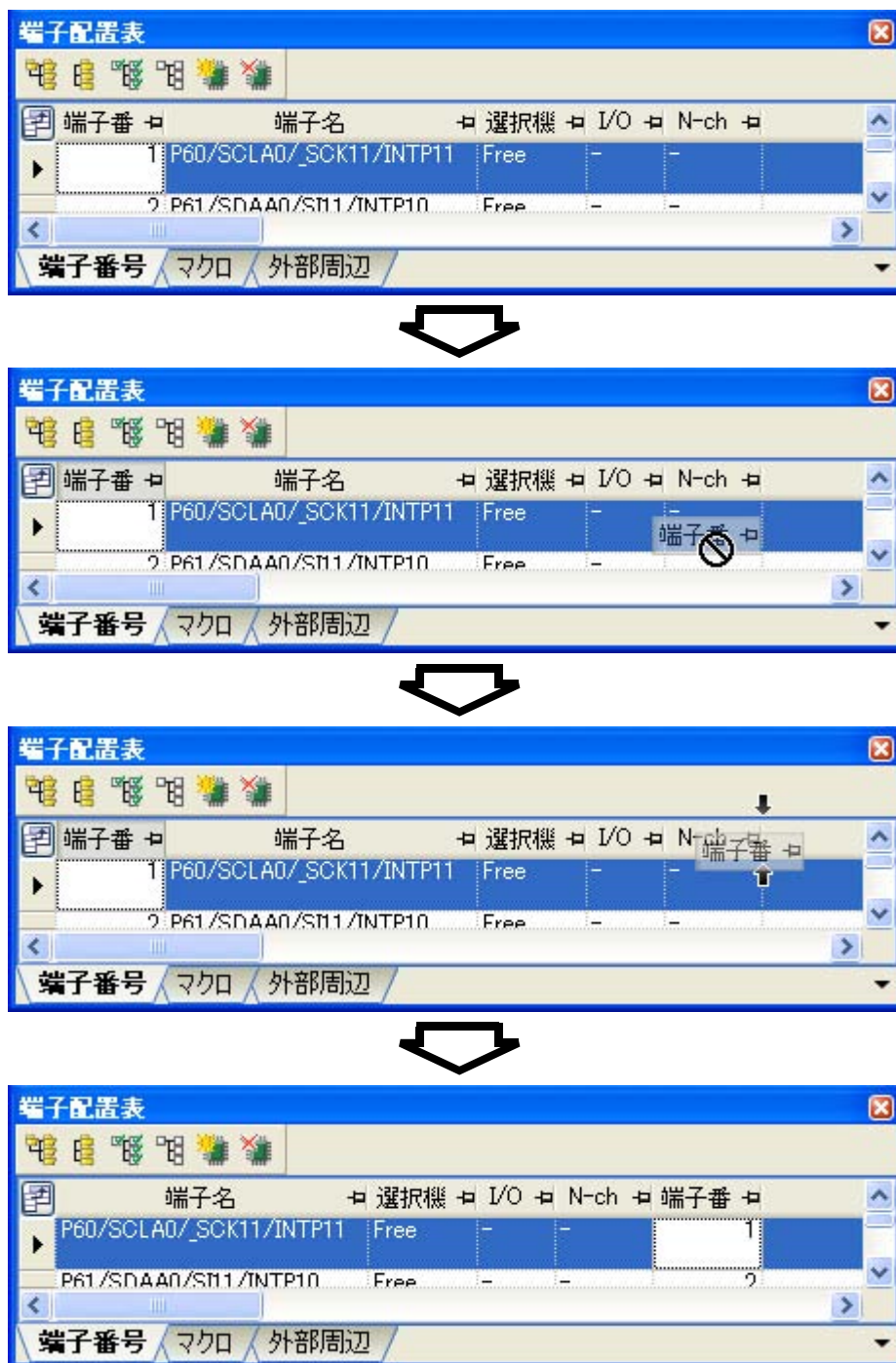
表 2—1 表示項目の選択


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

2.2.2 表示順序の変更


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

図 2—3 表示順序の変更



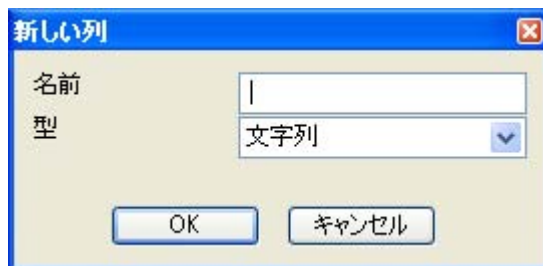
備考 表示順序の変更は、端子配置表の左上に設けられた  ボタンをクリックすることによりオープンする列の選択ダイアログの表示項目選択エリアに表示されている項目をドラッグしたのち、端子配置表の移動先にドロップすることでも、表示項目の表示順序を変更することができます。

2.2.3 列の追加

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


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

図 2—4 列の追加



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

2.2.4 列の削除

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

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

図 2—5 列の削除



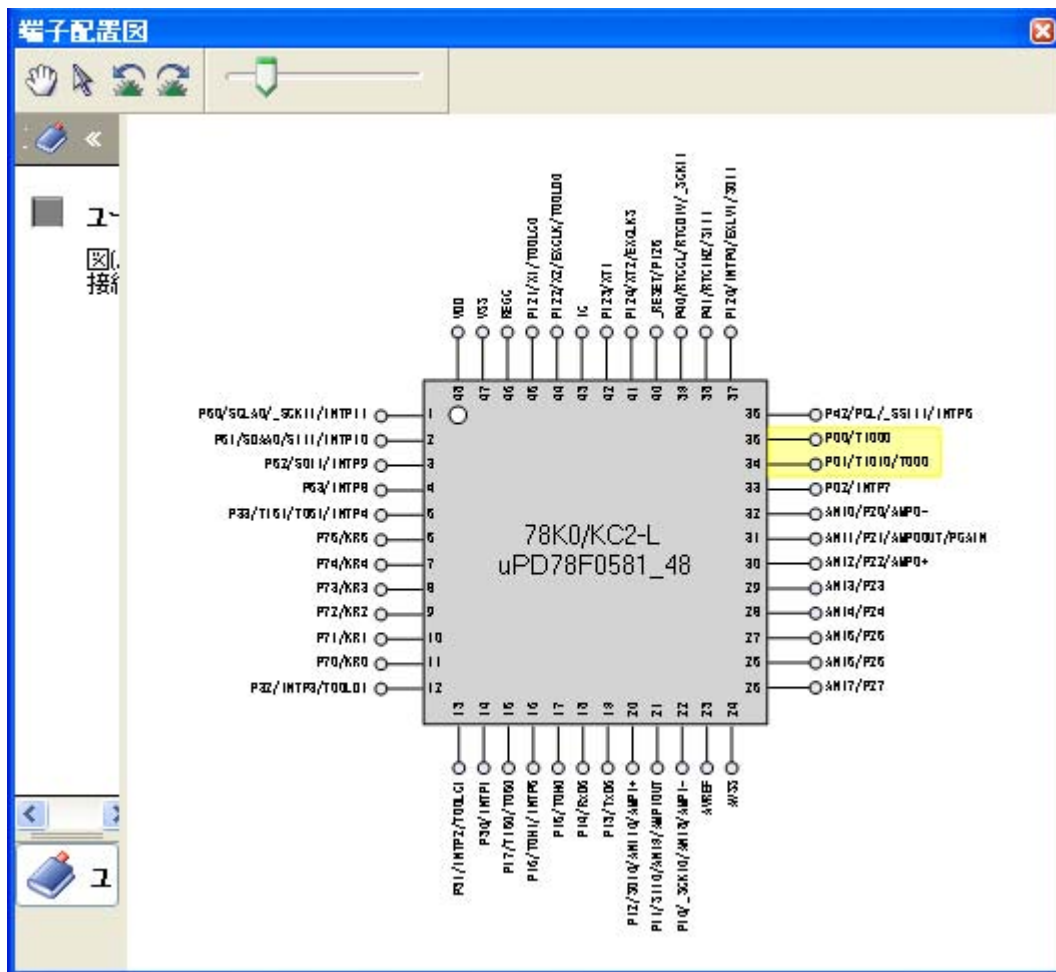
備考 削除可能な列は、**新しい列 ダイアログ** でユーザが独自に追加した列に限られます。

2.3 端子配置図 パネルのオープン

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

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

図 2—6 端子配置図 パネルのオープン



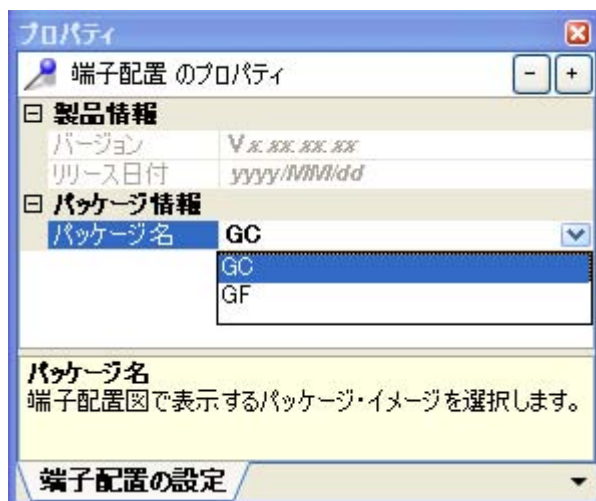
備考 プロパティパネルの [端子配置の設定] タブでパッケージ名に“BGA”を選択している場合、端子配置図パネルをオープンすることができません。

2.3.1 マイクロコントローラの形状選択

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

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

図 2-7 マイクロコントローラの形状選択



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

2.3.2 表示色の選択

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

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

図 2—8 表示色の選択



備考 表示色の選択は，以下の8種類に対して行います。

表 2—2 表示色の選択

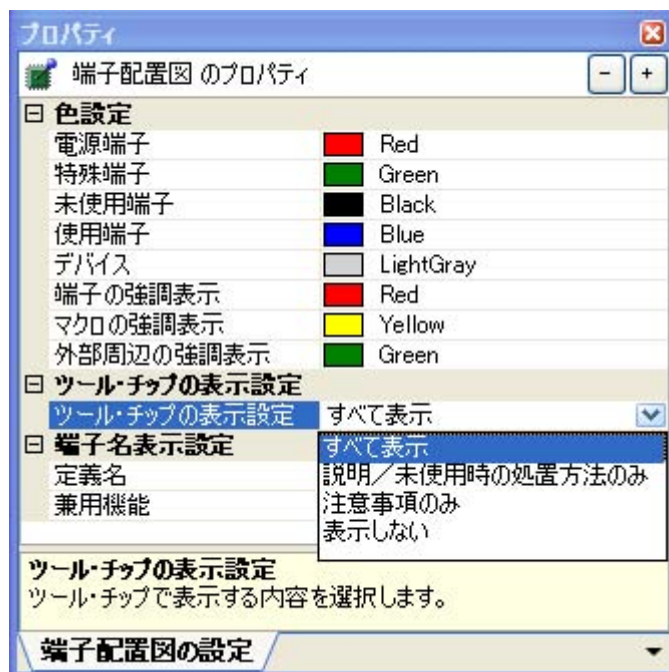
設定対象	概要
電源端子	電源端子（用途が電源に限定されている端子）の表示色を選択します。
特殊端子	特殊端子（用途が規定されている端子）の表示色を選択します。
未使用端子	未使用端子（端子配置表 パネルにおいて，用途が未設定の兼用端子）の表示色を選択します。
使用端子	使用端子（端子配置表 パネルにおいて，用途が設定済みの兼用端子）の表示色を選択します。
デバイス	マイクロコントローラ本体部の表示色を選択します。
端子の強調表示	端子配置表 パネルの [端子番号] タブで選択された項目に対応した端子の背景色を選択します。
マクロの強調表示	端子配置表 パネルの [マクロ] タブで選択された項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

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	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。

2.3.4 付加情報の選択

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

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

図 2—10 付加情報の選択



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

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

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

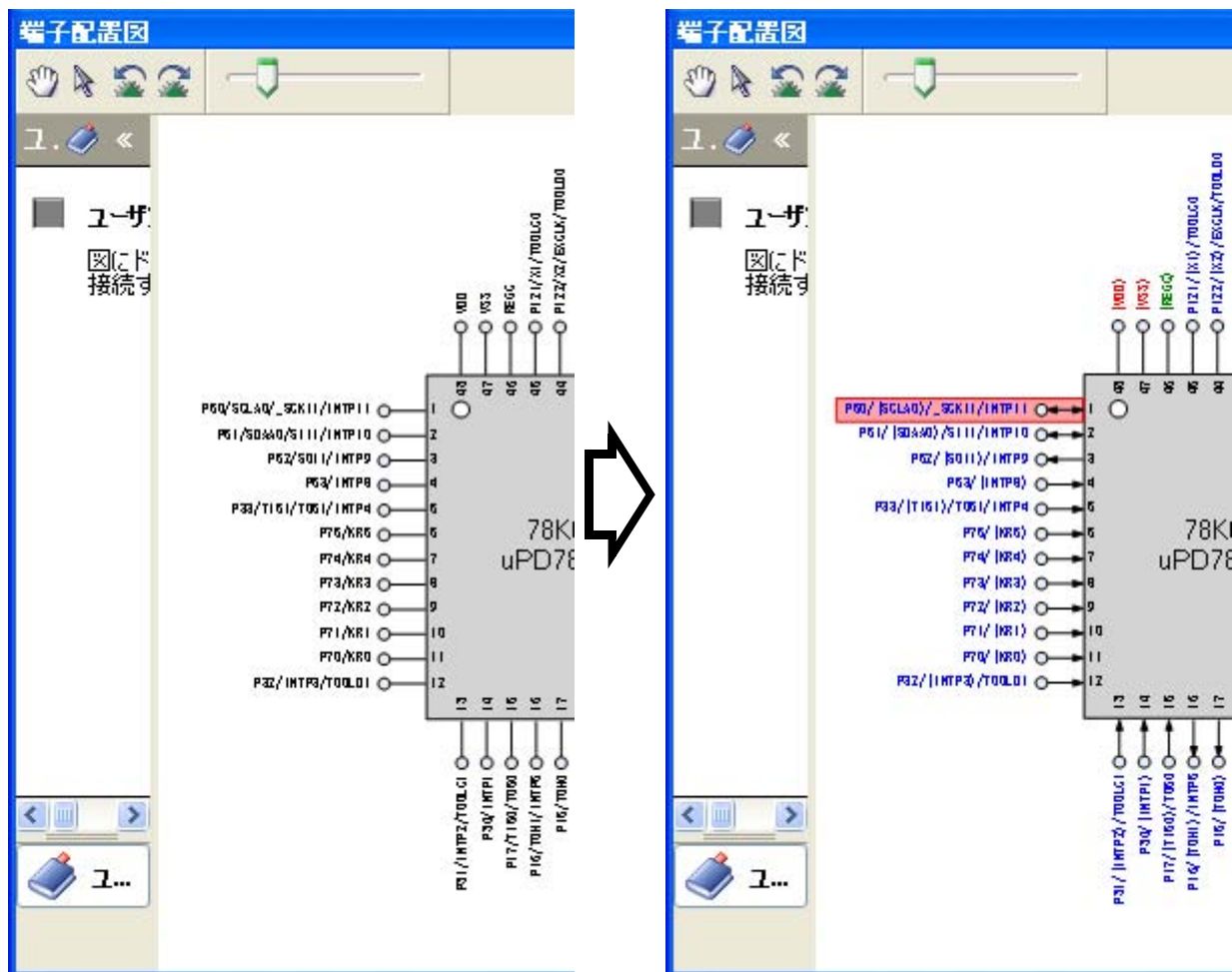
すべて	端子配置表の“選択機能”で選択された機能をかっこで括った形式で表示します。
選択機能のみ	端子配置表の“選択機能”で選択された機能のみを端子配置図に表示します。

2.4 情報の記述

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

- 備考 1.** 端子配置表の“端子番号”，“端子名”，“説明”，“未使用時の処置方法”，“注意事項”については，固定化された情報のため，該当欄に情報を追記することはできません。
- 2.** “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。

図 2—11 表示色の変化



2.5 レポート・ファイルの出力

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

2.5.1 端子配置表の出力

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

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

図 2—12 端子配置表の出力



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

2.5.2 端子配置図の出力

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

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

図 2—13 端子配置図の出力



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

第3章 機能（コード生成）

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

3.1 概要

コード生成は、マイクロコントローラが提供している周辺機能（クロック発生回路、I/Oポートなど）を制御する際に必要な情報を CubeSuite+ のパネル上で選択／入力することにより、対応するソース・コード（デバイス・ドライバ・プログラム）を出力します。

なお、コード生成の実践手順は、以下のとおりです。

(1) CubeSuite+ の起動

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

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

(2) プロジェクトの作成／読み込み

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

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

(3) コード生成 パネルのオープン

周辺機能（クロック発生回路、I/Oポートなど）を制御するうえで必要な情報を設定するための**コード生成パネル**をオープンします。

(4) 情報の設定

コード生成パネルで周辺機能を制御するうえで必要な情報を設定します。

(5) ソース・コードの確認

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

(6) ソース・コードの出力

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

(7) レポート・ファイルの出力

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

(8) プロジェクトの保存

プロジェクトの保存を行います。

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

3.2 コード生成 パネルのオープン

マイクロコントローラが提供している周辺機能（クロック発生回路、I/Oポートなど）を制御するうえで必要な情報を設定するための**コード生成 パネル**をオープンします。

なお、**コード生成 パネル**のオープンは、**プロジェクト・ツリー パネル**の [Project name (プロジェクト)] → [コード生成 (設計ツール)] → 周辺機能ノード “[システム], [ポート] など” をダブルクリックすることにより行います。

図 3—1 コード生成 パネルのオープン



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

3.3 情報の設定

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

備考 複数の周辺機能を制御する場合は、「3.2 コード生成パネルのオープン」から「3.3 情報の設定」の操作を繰り返し行うことになります。

3.3.1 入力規約

以下に、コード生成パネルに各種情報を設定する際の入力規約を示します。

(1) 文字セット

以下に、コード生成が入力を許可している文字セットを示します。

表 3—1 文字セットの一覧

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


(2) 数値

以下に、コード生成が入力を許可している進数を示します。

表 3—2 進数の一覧

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

3.3.2 入力不備箇所に対するアイコン表示

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


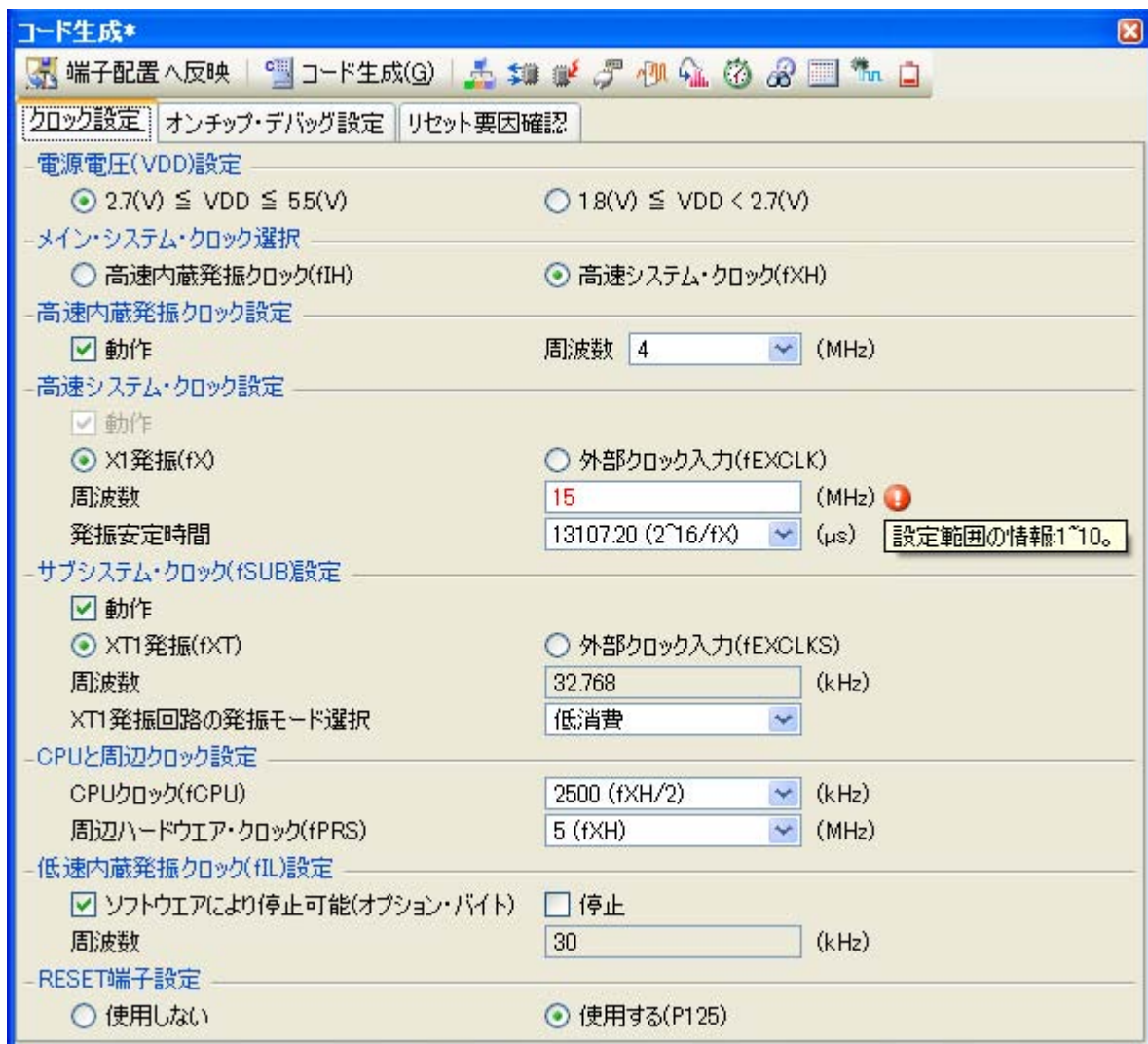

備考  アイコン上にマウス・カーソルを移動した際には、入力すべき文字列に関する情報（入力の不備を解決するためのヒント）がポップアップ表示されます。

図 3—2 入力不備箇所に対するアイコン表示



3.3.3 端子の競合に対するアイコン表示

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


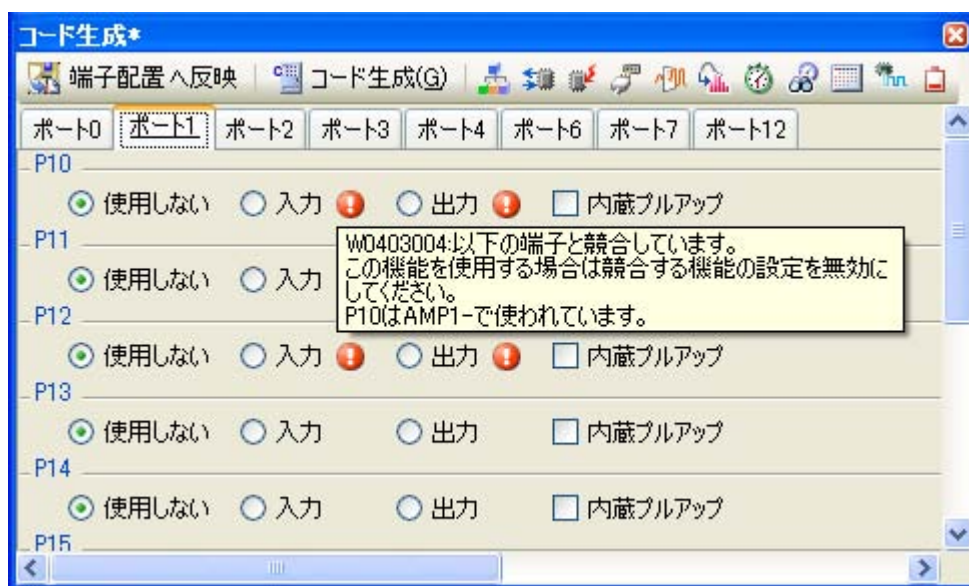
備考  アイコン上にマウス・カーソルを移動した際には、端子の競合に関する情報（競合を回避するためのヒント）がポップアップ表示されます。

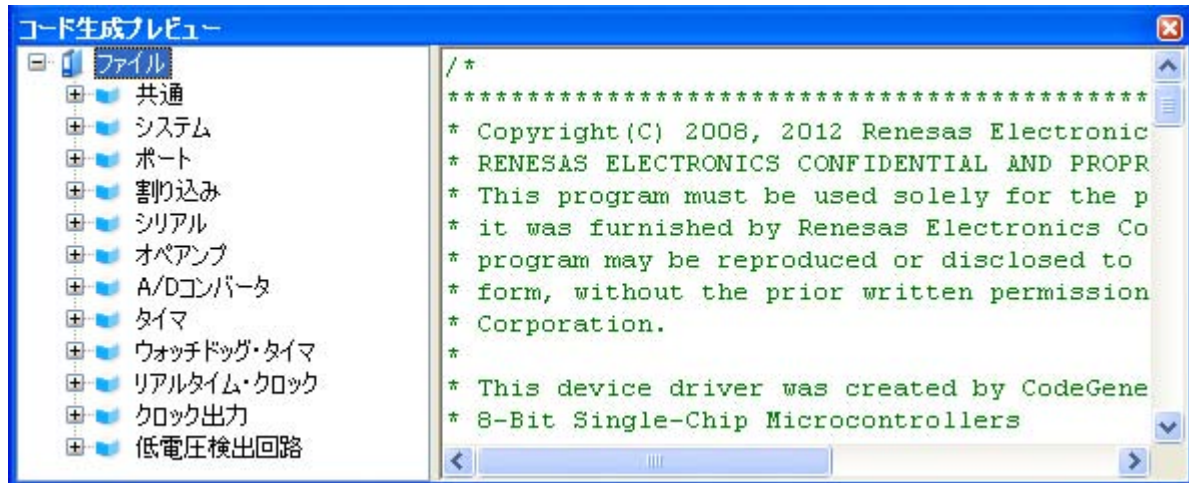
図 3—3 端子の競合に対するアイコン表示



3.4 ソース・コードの確認

「3.3 情報の設定」で設定した情報に応じたソース・コード（デバイス・ドライバ・プログラム）を確認します。
 なお、ソース・コードの確認は、[表示]メニュー→[コード生成プレビュー]を選択することによりオープンする
 コード生成プレビューパネルで行います。


図3—4 ソース・コードの確認




- 備考 1. コード生成プレビューパネルのソース・ファイル名、またはAPI関数名を選択することにより、ソース・コードの表示を切り替えることができます。
2. コード生成プレビューパネルに表示されるソース・コードの文字色は、以下の意味を持ちます。

表3—3 ソース・コードの文字色

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

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

3.5 ソース・コードの出力

コード生成パネルの  コード生成(G) ボタンをクリックし、ソース・コード（デバイス・ドライバ・プログラム）を出力します。

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

図 3—5 ソース・コードの出力




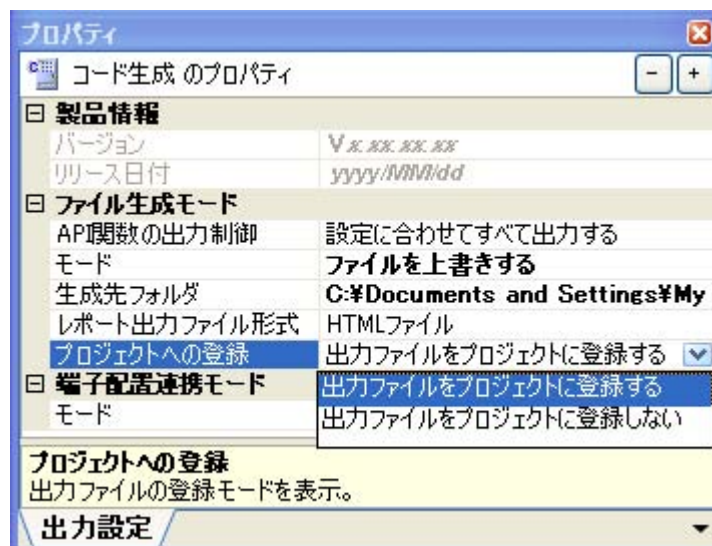
備考  コード生成(G) ボタンをクリックした際、ソース・コードを出力するとともに、該当ファイル群をプロジェクトに登録（プロジェクト・ツリーパネルに対する該当ソース・ファイル名の表示）する場合は、プロパティパネルの [出力設定] タブ→ [プロジェクトへの登録] で“出力ファイルをプロジェクトに登録する”を指定する必要があります。

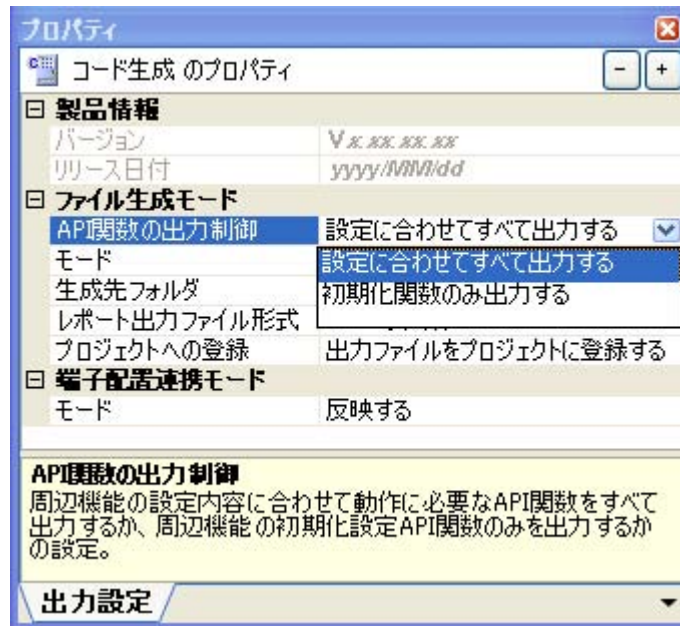
図 3—6 登録有無の設定



3.5.1 出力有無の設定

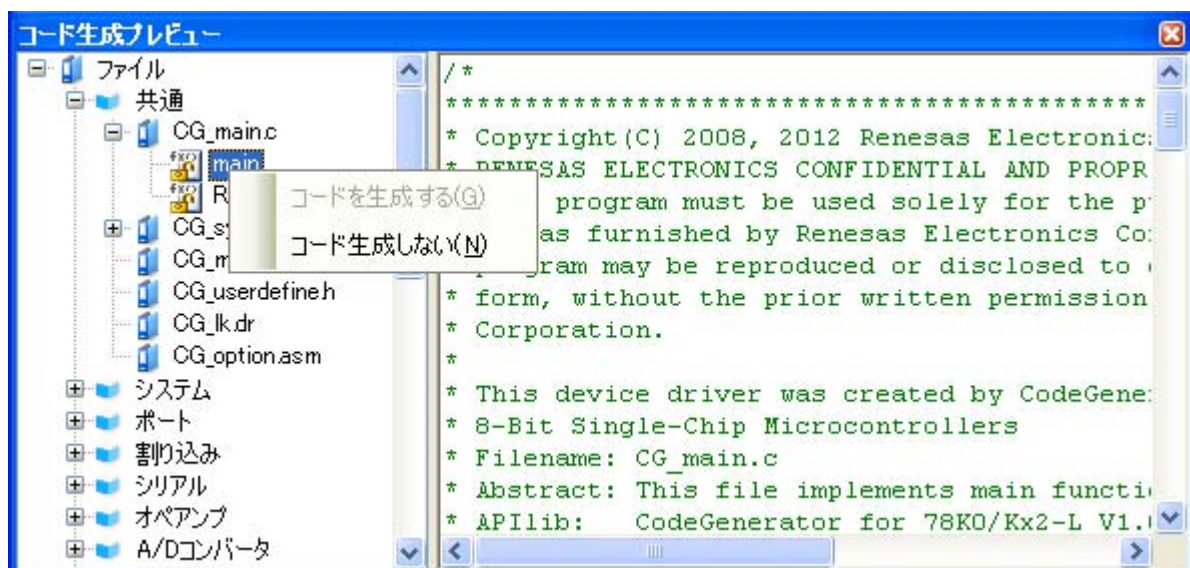
コード生成では、**プロパティパネル**の**[出力設定]**タブ→**[API関数の出力制御]**で“設定に合わせてすべて出力する／初期化関数のみ出力する”を選択することにより、出力するAPI関数の種類（全API関数、初期化用API関数のみ）を設定することができます。

図 3—7 出力有無の設定







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

図 3—8 出力有無の設定



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

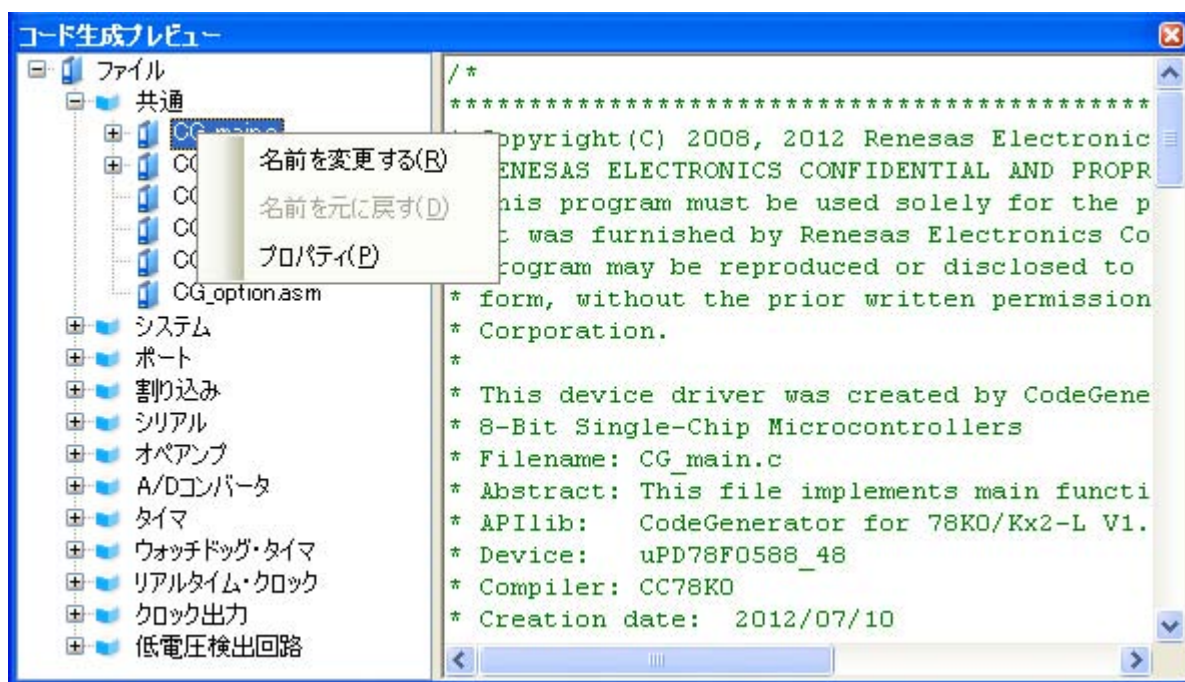
表 3—4 ソース・コードの出力有無

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

3.5.2 ファイル名の変更

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

図 3—9 ファイル名の変更

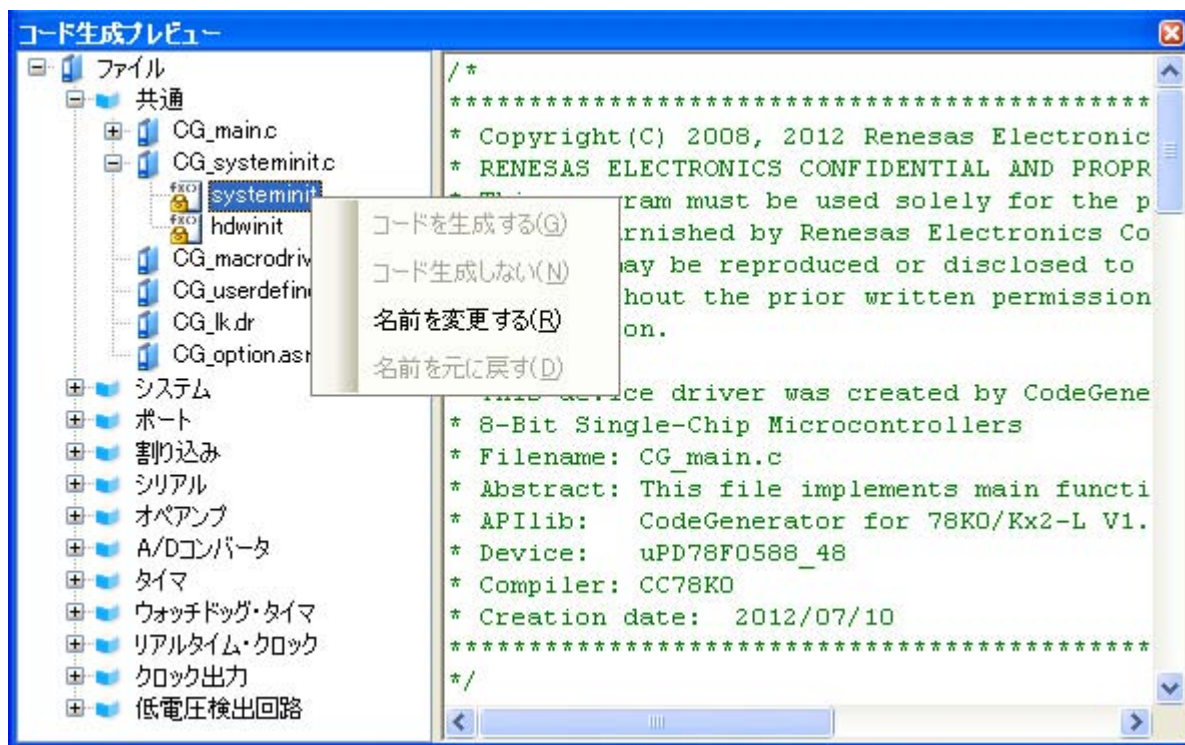


備考 コード生成が規定しているデフォルト・ファイル名に戻す際には、コンテキスト・メニューから“名前を元に戻す”を選択します。

3.5.3 API 関数名の変更

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

図 3—10 API 関数名の変更

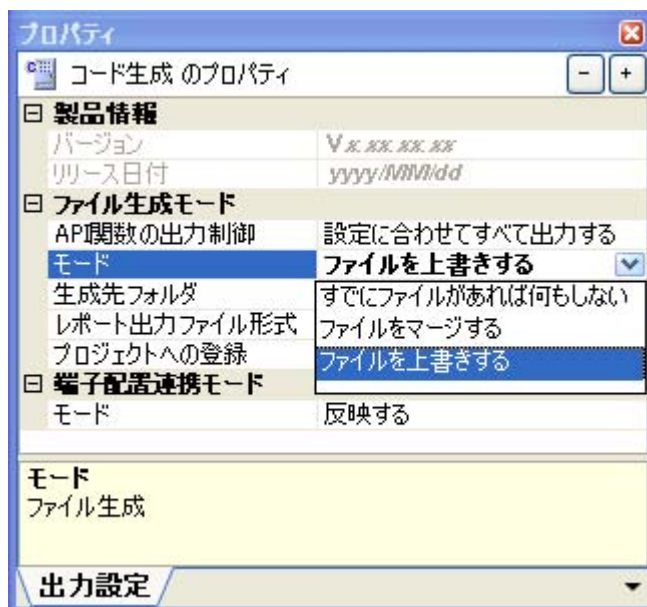


備考 コード生成が規定しているデフォルトAPI関数名に戻す際には、コンテキスト・メニューから“名前を元に戻す”を選択します。

3.5.4 出力モードの変更

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

図 3—11 出力モードの変更



備考 出力モードの選択は、以下の3種類から行います。

表 3—5 ソース・コードの出力モード

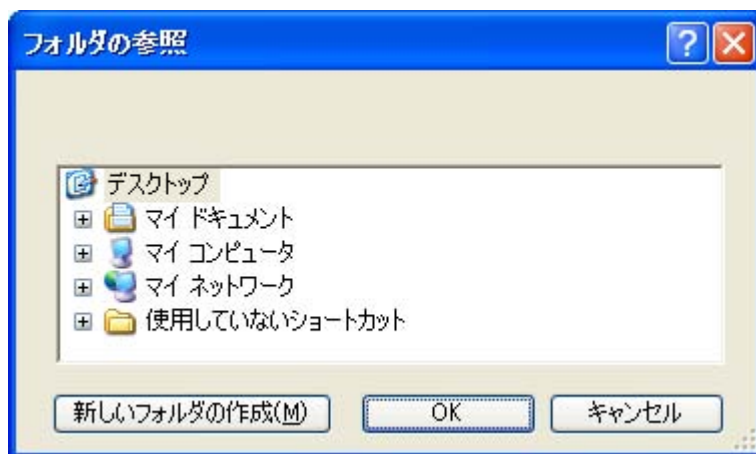
出力モード	概要
すでにファイルがあれば何もしない	同一のファイル名を有するファイルが既存していた場合、該当ファイルの出力を行いません。
ファイルをマージする	同一のファイル名を有するファイルが既存していた場合、該当ファイルをマージします。 なお、マージする部位については、 /* Start user code ... Do not edit comment generated here */ から /* End user code. Do not edit comment generated here */ で囲まれた部位に限られます。
ファイルを上書きする	同一のファイル名を有するファイル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

3.5.5 出力先の変更

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

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

図 3—12 出力先の変更



3.6 レポート・ファイルの出力

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

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

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

表 3—6 レポート・ファイルの出力

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

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

図 3—13 レポート・ファイル macro の出力例

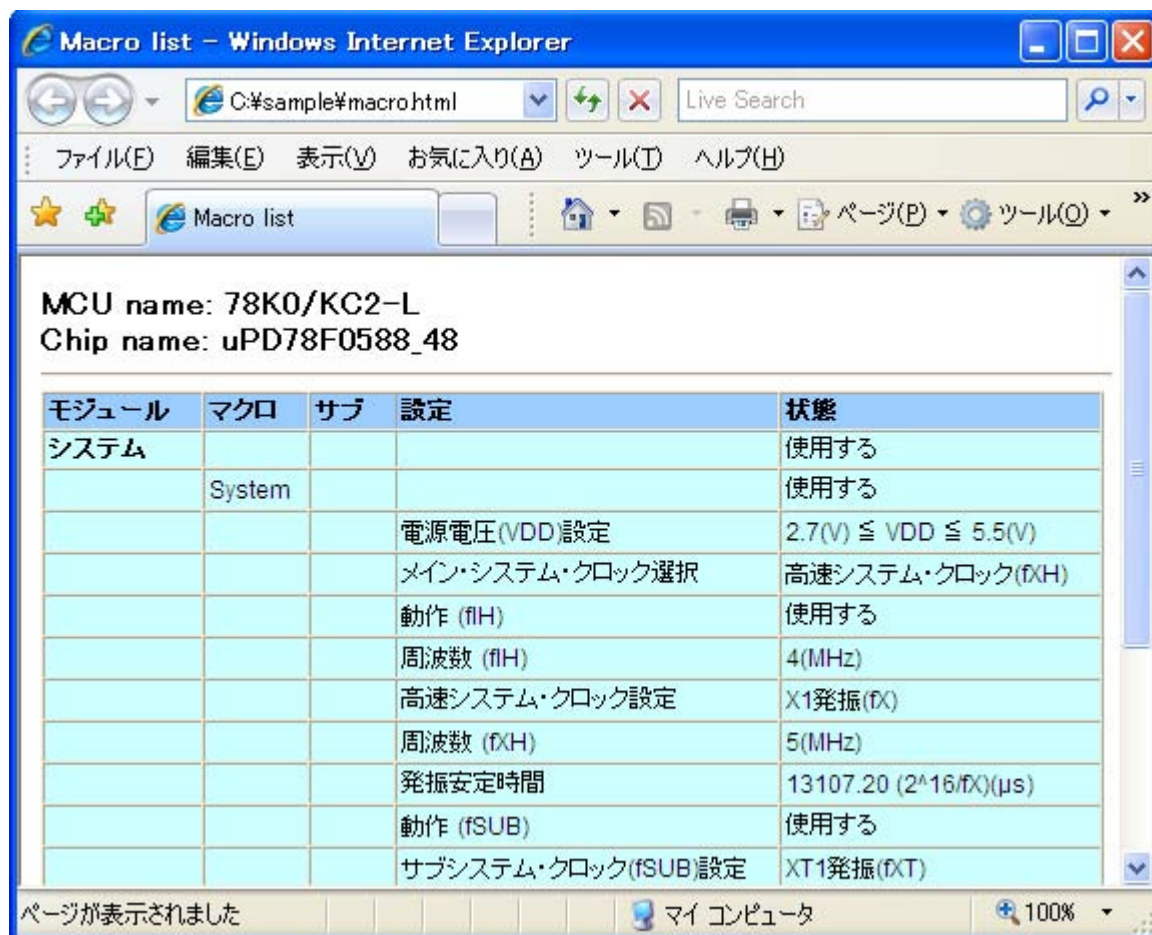
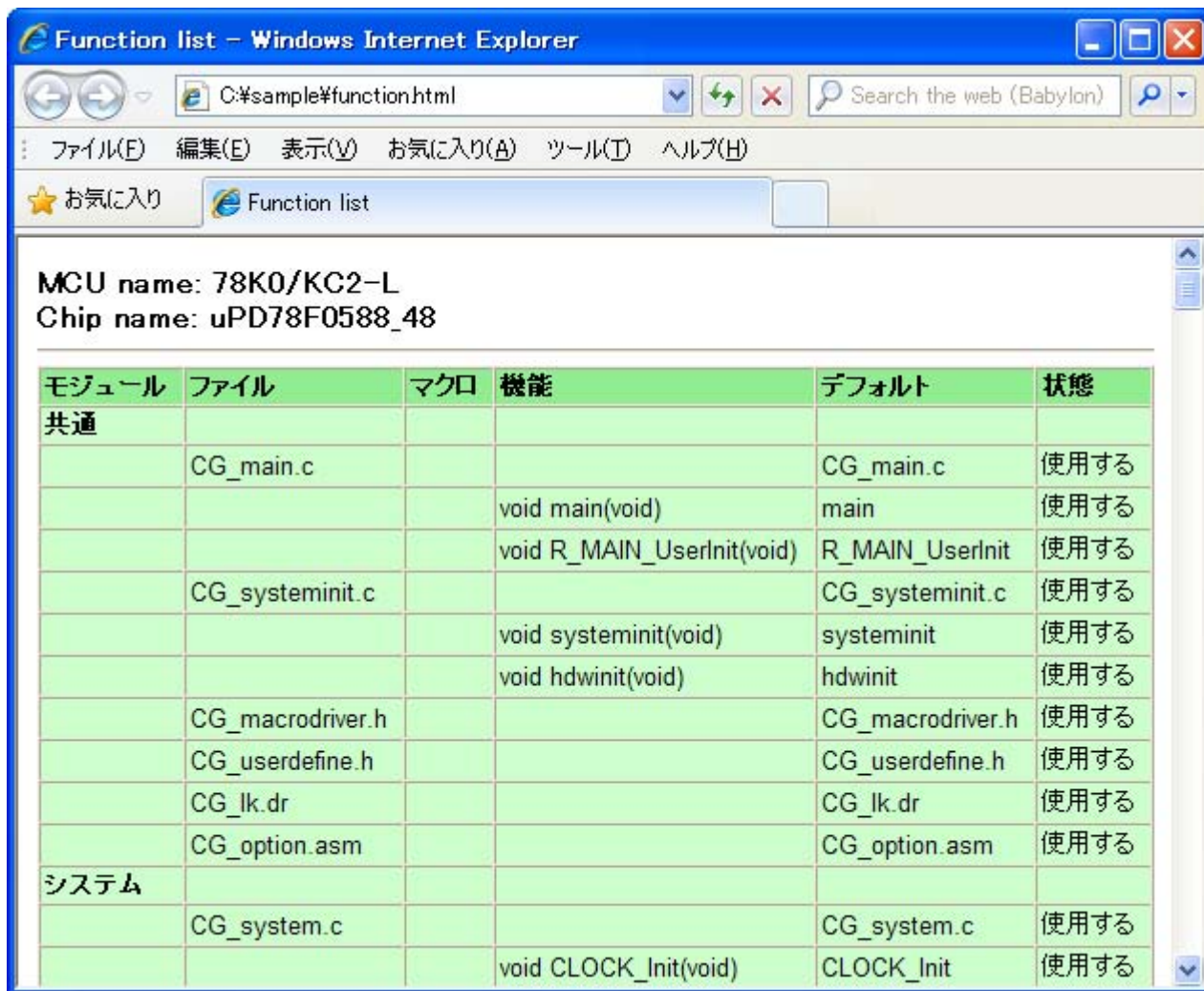


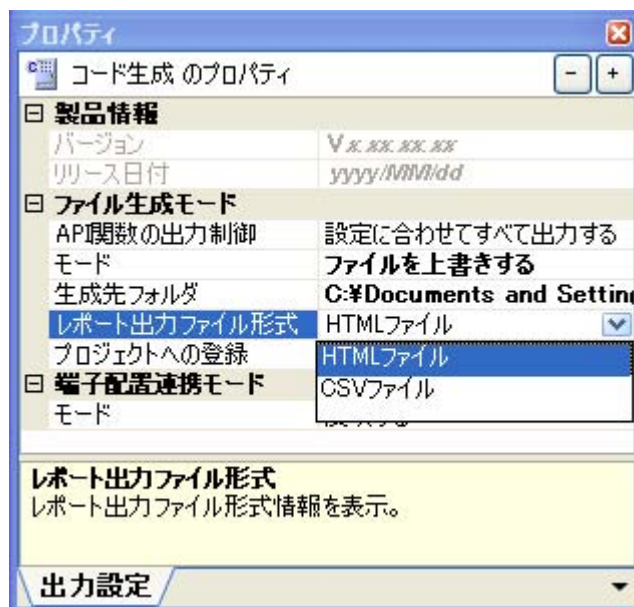
図 3—14 レポート・ファイル function の出力例



3.6.1 出力形式の変更

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

図 3—15 出力形式の変更



備考 出力形式の選択は、以下の2種類から行います。

表 3—7 ソース・コードの出力モード

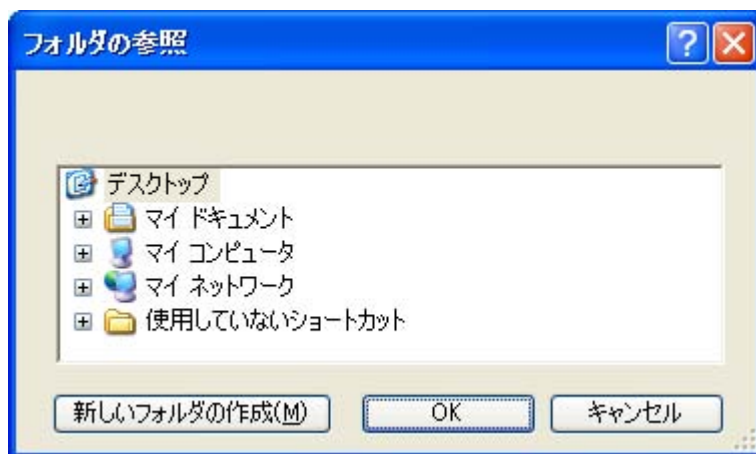
出力形式	概要
HTML ファイル	HTML 形式でレポート・ファイルを出力します。
CSV ファイル	CSV 形式でレポート・ファイルを出力します。

3.6.2 出力先の変更

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

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

図 3—16 出力先の変更




付録 A ウィンドウ・リファレンス

本付録では、設計ツールのウィンドウ／パネル／ダイアログについて説明します。

A.1 説明

以下に、設計ツールのウィンドウ／パネル／ダイアログの一覧を示します。

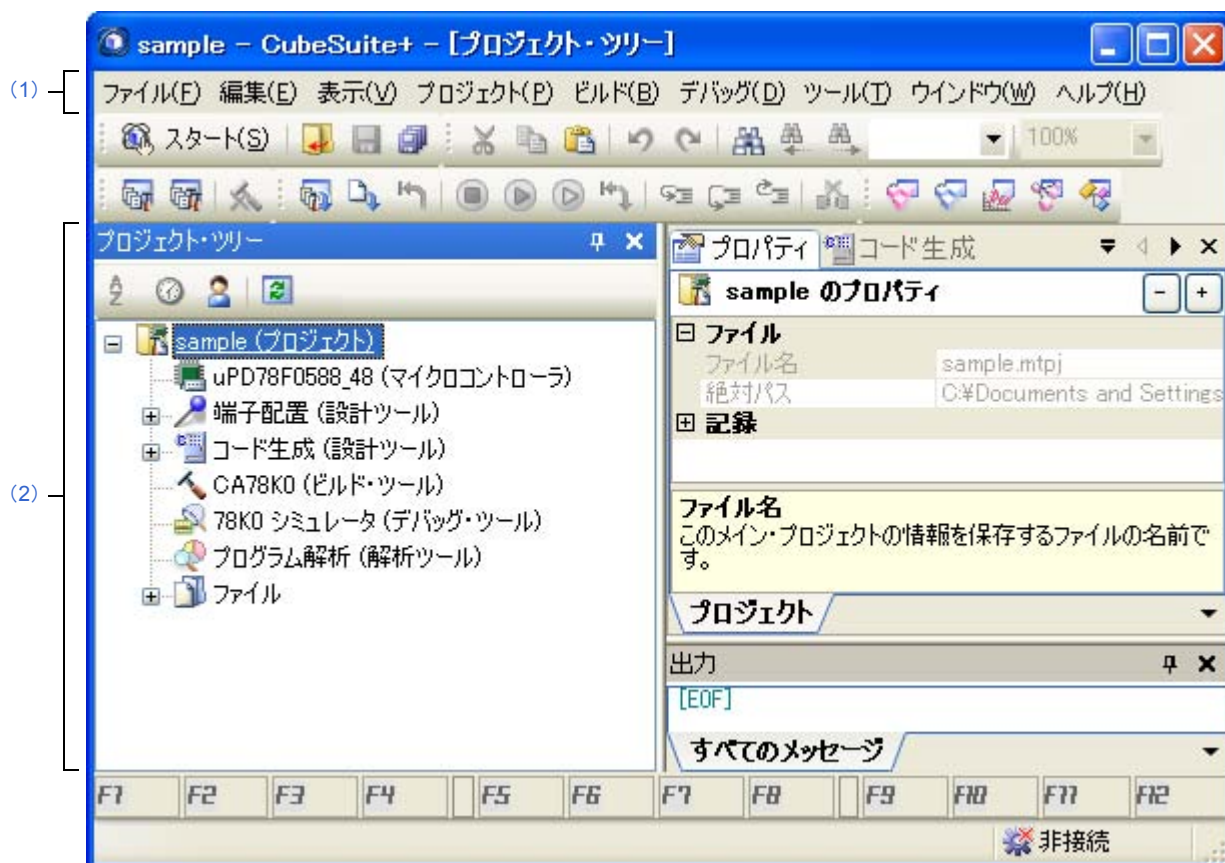
表 A—1 ウィンドウ／パネル／ダイアログの一覧

ウィンドウ／パネル／ダイアログ名	機能概要
メイン・ウィンドウ	CubeSuite+ を起動した際、最初にオープンするウィンドウであり、本ウィンドウから CubeSuite+ が提供している各種コンポーネント（設計ツール、ビルド・ツールなど）に対する操作を行います。
プロジェクト・ツリー パネル	プロジェクトの構成要素（マイクロコントローラ、設計ツール、ビルド・ツールなど）をツリー形式で表示します。
プロパティ パネル	プロジェクト・ツリー パネルで選択したノード、コード生成 パネルでクリックした周辺機能ボタン、コード生成プレビュー パネルで選択したファイルの種類に対応した情報の表示、および設定の変更を行います。
端子配置表 パネル	マイクロコントローラの各端子に関する情報を記述します。
端子配置図 パネル	端子配置表 パネルにおける情報の記述状況を表示します。
コード生成 パネル	マイクロコントローラが提供している周辺機能を制御するうえで必要な情報を設定します。
コード生成プレビュー パネル	コード生成 パネルの  ボタンをクリックした際に出力されるソース・コード（デバイス・ドライバ・プログラム）の出力有無を API 関数単位で確認／設定するとともに、コード生成 パネルで設定した情報に応じたソース・コードの確認を行います。
出力 パネル	CubeSuite+ が提供している各種コンポーネント（設計ツール、ビルド・ツールなど）の操作ログを表示します。
列の選択 ダイアログ	本ダイアログに表示されている項目を端子配置表に表示するか否かの選択、および端子配置表に対する列の追加／削除を行います。
新しい列 ダイアログ	端子配置表に列を追加します。
フォルダの参照 ダイアログ	ファイル（ソース・コード、レポート・ファイルなど）の出力先を設定します。
名前を付けて保存 ダイアログ	ファイル（レポート・ファイルなど）に名前を付けて保存します。

メイン・ウィンドウ

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

図 A-1 メイン・ウィンドウ



ここでは、次の項目について説明します。

- [オープン方法]
- [各エリアの説明]

[オープン方法]

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

[各エリアの説明]

(1) メニューバー

本エリアは、以下に示したメニュー群から構成されています。

(a) [ファイル] メニュー

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

(b) [編集] メニュー

元に戻す	プロパティ パネル専用部分 直前に行った編集作業を取り消します。
切り取り	プロパティ パネル専用部分 選択している文字列を切り取り、クリップ・ボードに保存します。
コピー	プロパティ パネル／出力 パネル専用部分 選択している文字列をクリップ・ボードに保存します。
貼り付け	プロパティ パネル専用部分 指定された箇所に、クリップ・ボードの内容を挿入します。
削除	プロパティ パネル専用部分 選択している文字列を削除します。

すべて選択	プロパティ パネル／出力 パネル専用部分 編集中の項目に表示されている全文字列、またはメッセージ・エリアに表示されている全文字列を選択します。
検索 ...	端子配置表 パネル／コード生成プレビュー パネル／出力 パネル専用部分 文字列検索を行うための検索・置換 ダイアログを [クイック検索] タブが選択された状態でオープンします。
置換 ...	出力 パネル専用部分 文字列置換を行うための検索・置換 ダイアログを [一括置換] タブが選択された状態でオープンします。

(c) [ヘルプ] メニュー

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

(2) パネル表示エリア

本エリアは、用途別に用意された各種パネルから構成されています。

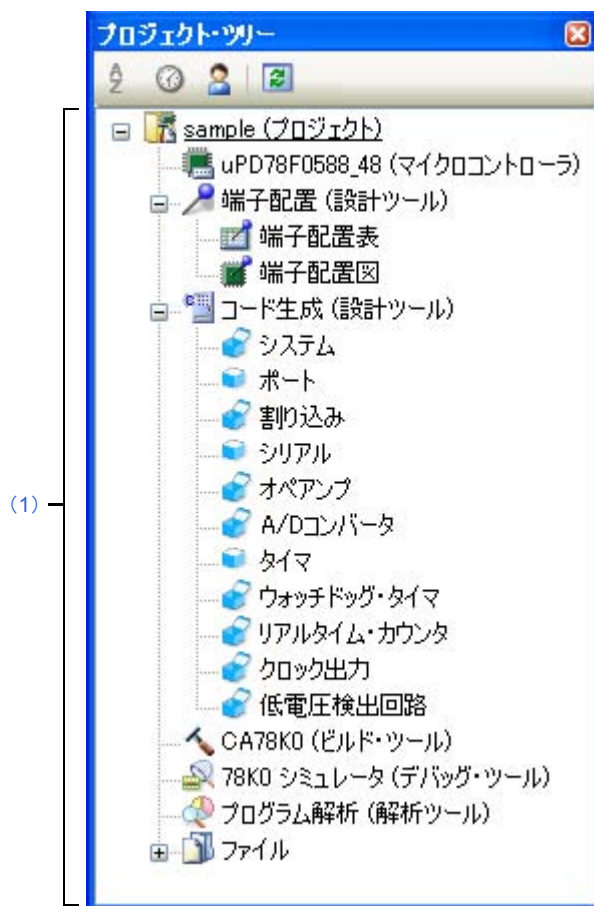
本エリアについての詳細は、以下を参照してください。

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

プロジェクト・ツリーパネル

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

図 A—2 プロジェクト・ツリー パネル



ここでは、次の項目について説明します。

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

[オープン方法]

- [表示] メニュー → [プロジェクト・ツリー] を選択

[各エリアの説明]

(1) プロジェクト・ツリー・エリア

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

(a) 端子配置（設計ツール）

本ノードは、以下に示した端子ノードから構成されています。

端子配置表	マイクロコントローラの各端子に関する情報を記述するための 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] 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。
オペアンプ	マイクロコントローラが提供しているコンパレータ／プログラマブル・ゲイン・アンプの機能を制御するうえで必要な情報を設定するための [オペアンプ] をオープンします。
コンパレータ	マイクロコントローラが提供しているコンパレータの機能を制御するうえで必要な情報を設定するための [コンパレータ] をオープンします。
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。
タイマ	マイクロコントローラが提供しているタイマ・アレイ・ユニットの機能を制御するうえで必要な情報を設定するための [タイマ] をオープンします。

ウォッチドッグ・タイマ	マイクロコントローラが提供しているウォッチドッグ・タイマの機能を制御するうえで必要な情報を設定するための「ウォッチドッグ・タイマ」をオープンします。
リアルタイム・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。
クロック出力	マイクロコントローラが提供しているクロック出力制御回路の機能を制御するうえで必要な情報を設定するための「クロック出力」をオープンします。
低電圧検出回路	マイクロコントローラが提供している低電圧検出回路の機能を制御するうえで必要な情報を設定するための「低電圧検出回路」をオープンします。

(c) アイコン

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

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

[[ヘルプ] メニュー (プロジェクト・ツリーパネル専用部分)]

プロジェクト・ツリーパネルのヘルプを開く	本パネルのヘルプを表示します。
----------------------	-----------------

[コンテキスト・メニュー]

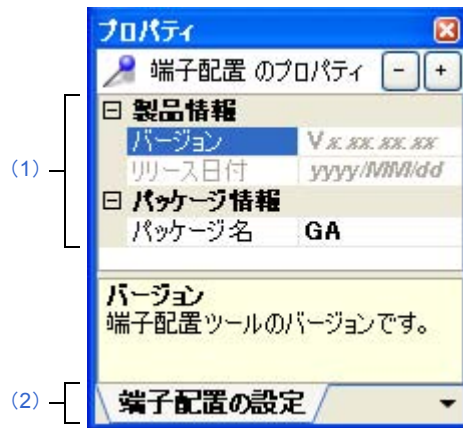
マウスを右クリックすることにより表示されるコンテキスト・メニューは、以下のとおりです。

リセット時の設定に戻す	選択された周辺機能ノードに対応した情報をデフォルトの状態に戻します。
プロパティ	選択されたノード ([端子配置 (設計ツール)], [端子配置表], [端子配置図], [コード生成 (設計ツール)], 周辺機能ノード “[システム], [ポート] など”) に対応した情報を保持した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)]
- [コンテキスト・メニュー]

[オープン方法]



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

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

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と切り替わります。



[各エリアの説明]



(1) 詳細情報表示／変更エリア

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

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

各カテゴリの直前に表示されている 、および  は、以下の意味を持ちます。

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

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

(2) タブ選択エリア

タブを選択することにより、詳細情報を表示するカテゴリが切り替わります。

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

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

[[編集] メニュー（プロパティ パネル専用部分）]

元に戻す	直前に行った編集作業を取り消します。
------	--------------------

切り取り	選択している文字列を切り取り、クリップ・ボードに保存します。
コピー	選択している文字列をクリップ・ボードに保存します。
貼り付け	指定された箇所に、クリップ・ボードの内容を挿入します。
削除	選択している文字列を削除します。
すべて選択	編集中の項目に表示されている全文字列を選択します。

[コンテキスト・メニュー]

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

(1) 項目を編集中の場合

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

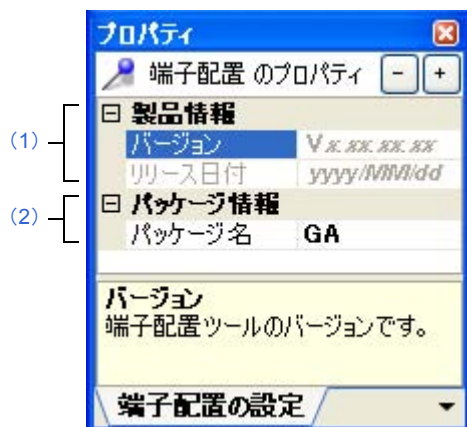
(2) 項目を編集中外の場合

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

[端子配置の設定] タブ

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

図 A—4 [端子配置の設定] タブ



ここでは、次の項目について説明します。

- [オープン方法]
- [各エリアの説明]

[オープン方法]

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

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

[各エリアの説明]

(1) [製品情報] カテゴリ

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

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

(2) [パッケージ情報] カテゴリ

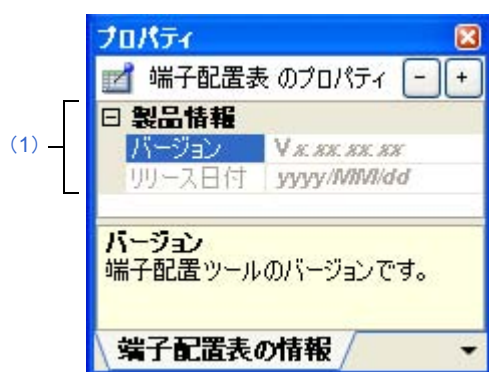
端子配置図 [パネル](#)に端子配置図として表示するマイクロコントローラの形状（パッケージ名）を選択します。

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

[端子配置表の情報] タブ

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

図 A—5 [端子配置表の情報] タブ



ここでは、次の項目について説明します。

- [オープン方法]
- [各エリアの説明]

[オープン方法]

- プロジェクト・ツリーパネルにおいて、[Project name (プロジェクト)] → [端子配置 (設計ツール)] → [端子配置表] を選択したのち、[表示] メニュー → [プロパティ] を選択
- プロジェクト・ツリーパネルにおいて、[Project name (プロジェクト)] → [端子配置 (設計ツール)] → [端子配置表] を選択したのち、コンテキスト・メニューから [プロパティ] を選択

備考 すでに本パネルがオープンしていた場合、プロジェクト・ツリーパネルの [端子配置表] を選択することにより、表示内容が切り替わります。

[各エリアの説明]

(1) [製品情報] カテゴリ

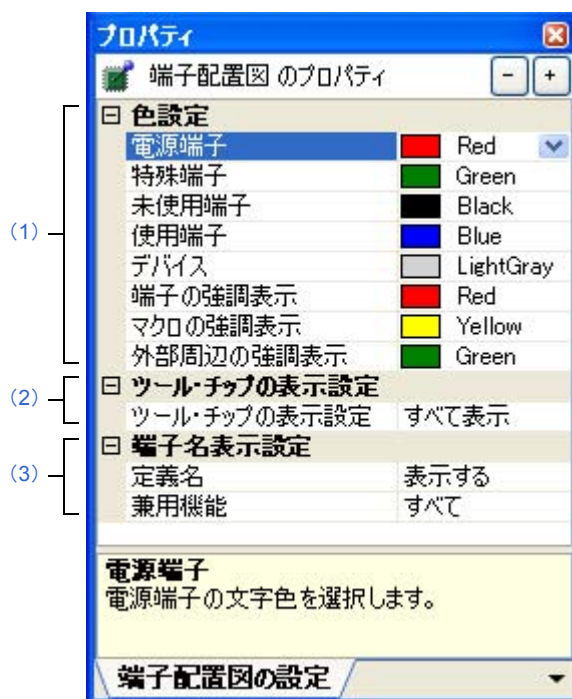
端子配置に関する製品情報（バージョン、リリース日付）の表示を行います。

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

[端子配置図の設定] タブ

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

図 A—6 [端子配置図の設定] タブ



ここでは、次の項目について説明します。

- [オープン方法]
- [各エリアの説明]

[オープン方法]

- プロジェクト・ツリーパネルにおいて、[Project name (プロジェクト)] → [端子配置 (設計ツール)] → [端子配置図] を選択したのち、[表示] メニュー → [プロパティ] を選択
- プロジェクト・ツリーパネルにおいて、[Project name (プロジェクト)] → [端子配置 (設計ツール)] → [端子配置図] を選択したのち、コンテキスト・メニューから [プロパティ] を選択

備考 すでに本パネルがオープンしていた場合、プロジェクト・ツリーパネルの [端子配置図] を選択することにより、表示内容が切り替わります。

[各エリアの説明]

(1) [色設定] カテゴリ

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

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

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

図 A-7 カラー・パレット



(2) [ツール・チップの表示設定] カテゴリ

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

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

(3) [端子名表示設定] カテゴリ

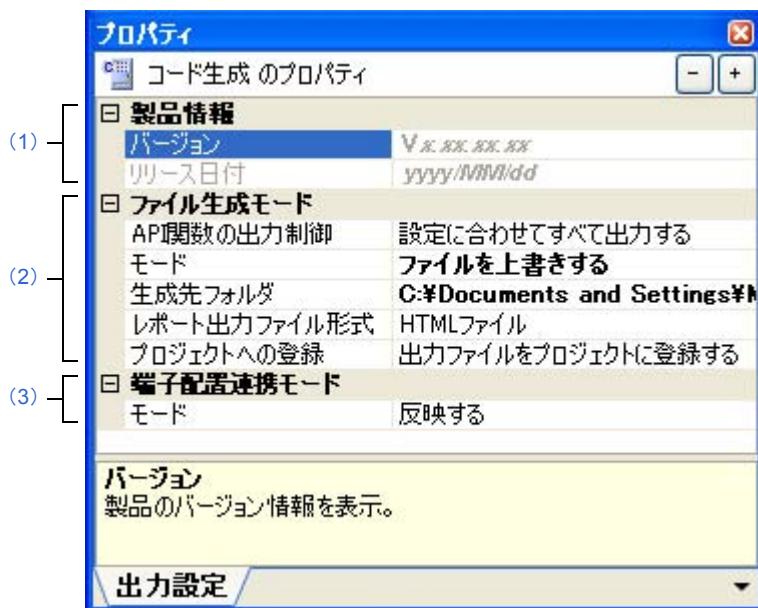
端子の付加情報を端子配置図に表示するか否かを選択します。

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

[出力設定] タブ

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

図 A—8 [出力設定] タブ



ここでは、次の項目について説明します。

- [オープン方法]
- [各エリアの説明]

[オープン方法]

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

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

[各エリアの説明]

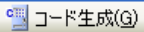
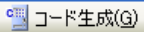
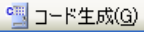
(1) [製品情報] カテゴリ

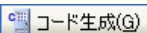
コード生成に関する製品情報 (バージョン, リリース日付) の表示を行います。

バージョン	コード生成（コードライブラリ）のバージョンを表示します。
リリース日付	コード生成（コードライブラリ）のリリース日付を表示します。

(2) [ファイル生成モード] カテゴリ

コード生成のファイル生成モード（API 関数の出力制御、モードなど）の表示、および設定の変更を行います。


API 関数の出力制御	 ボタンをクリックした際に出力する API 関数の種類（全 API 関数、初期化用 API 関数のみ）を表示／選択します。	
	設定に合わせてすべて出力する	全 API 関数の出力となります。
	初期化関数のみ出力する	初期化用 API 関数のみの出力となります。
モード	 ボタンをクリックした際の動作モードを表示／選択します。 なお、[ファイル] メニュー→[コード生成レポートを保存] を選択した際の動作モードは、“ファイルを上書きする”となります。	
	すでにファイルがあれば何もしない	同一のファイル名を有するファイルが既存していた場合、該当ファイルの出力を行いません。
	ファイルをマージする	同一のファイル名を有するファイルが既存していた場合、該当ファイルをマージします。 なお、マージする部位については、 /* Start user code ... Do not edit comment generated here */ から /* End user code. Do not edit comment generated here */ で囲まれた部位に限られます。
	ファイルを上書きする	同一のファイル名を有するファイルが既存していた場合、該当ファイルを上書きします。
生成先フォルダ	 ボタンをクリックした際、[ファイル] メニュー→[コード生成レポートを保存] を選択した際に出力する各種ファイル（ソース・コード、レポート・ファイル）の出力先を表示／選択します。	
レポート出力ファイル形式	[ファイル] メニュー→[コード生成レポートを保存] を選択した際に出力するレポート・ファイル（コード生成を用いて設定した情報を保持したファイル、ソース・コードに関する情報を保持したファイル）の形式を表示／選択します。	
	HTML ファイル	HTML 形式でレポート・ファイルを出力します。
	CSV ファイル	CSV 形式でレポート・ファイルを出力します。

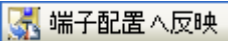
プロジェクトへの登録	 ボタンをクリックした際に出力するソース・コードをプロジェクトに登録するか否かを選択します。	
	出力ファイルをプロジェクトに登録する	出力したソース・コードをプロジェクトに登録します。 なお、登録されたソース・コードは、 プロジェクト・ツリーパネル の [ファイル] - [コード生成] ノードの直下に表示されます。
	出力ファイルをプロジェクトに登録しない	出力したソース・コードをプロジェクトに登録しません。

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

(3) [端子配置連携モード] カテゴリ

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



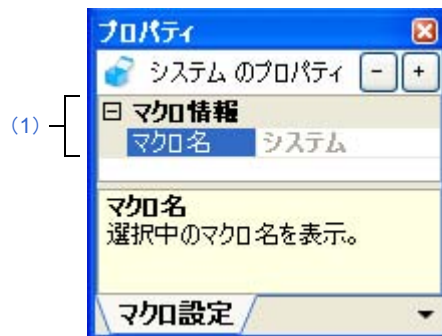
プロジェクト・ツリー パネルで選択した周辺機能ノード “[システム], [ポート] など”, コード生成 パネルでクリックした周辺機能ボタンの種類 “,  など” に対応した情報（マクロ情報）の表示、および設定の変更を行います。

図 A—9 [マクロ設定] タブ





ここでは、次の項目について説明します。

- [オープン方法]
- [各エリアの説明]

[オープン方法]

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

- 備考 1.** すでに本パネルがオープンしていた場合、プロジェクト・ツリー パネルの周辺機能ノード “[システム], [ポート] など” を選択することにより、表示内容が該当ノードに対応したものと切り替わります。
- 2.** すでに本パネルがオープンしていた場合、コード生成 パネルの周辺機能ボタン “,  など” をクリックすることにより、表示内容が該当ボタンに対応したものと切り替わります。

[各エリアの説明]

(1) [マクロ情報] カテゴリ

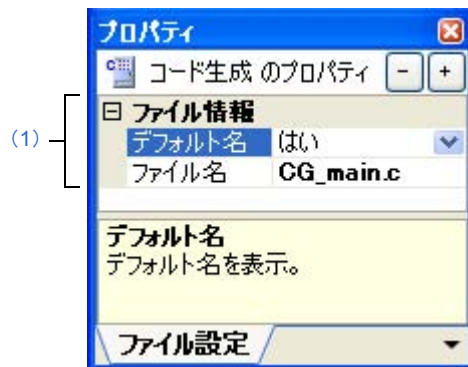
プロジェクト・ツリー パネルで選択した周辺機能ノード “[システム], [ポート] など”, コード生成 パネルでクリックした周辺機能ボタンに関する情報（マクロ名）の表示、および設定の変更を行います。

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

[ファイル設定] タブ

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

図 A—10 [ファイル設定] タブ



ここでは、次の項目について説明します。

- [オープン方法]
- [各エリアの説明]

[オープン方法]

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

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

[各エリアの説明]

(1) [ファイル情報] カテゴリ

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

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

端子配置表 パネル

マイクロコントローラの各端子に関する情報を記述します。

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

図 A—11 端子配置表 パネル



ここでは、次の項目について説明します。

- [\[オープン方法\]](#)
- [\[各エリアの説明\]](#)
- [\[\[ファイル\] メニュー \(端子配置表 パネル専用部分\)\]](#)
- [\[\[ヘルプ\] メニュー \(端子配置表 パネル専用部分\)\]](#)

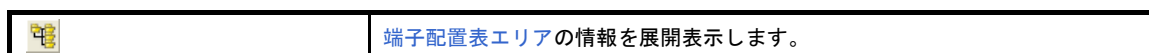
[オープン方法]






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



[各エリアの説明]

(1) ツールバー

本エリアは、以下に示したボタン群から構成されています。



	端子配置表エリアの情報を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] タブに外部周辺コントローラに関する情報が、端子配置図パネルに外部周辺コントローラが作成表示されます。
	[外部周辺] タブの第 1 階層に表示されている外部周辺コントローラを選択したのち、本ボタンをクリックすることにより、該当情報が削除されます。

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

(2) 端子配置表エリア

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

(3) タブ選択エリア

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

- [端子番号] タブ

マイクロコントローラの各端子に関する情報を端子番号順で表示

- [マクロ] タブ

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

- [外部周辺] タブ

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

[[ファイル] メニュー (端子配置表 パネル専用部分)]

端子配置表 を保存	レポート・ファイル (端子配置を用いて設定した情報を保持したファイル: 端子配置表) を既存のファイルに上書き保存します。
名前を付けて 端子配置表 を保存 ...	レポート・ファイル (端子配置を用いて設定した情報を保持したファイル: 端子配置表) に名前を付けて保存するための名前を付けて保存 ダイアログをオープンします。

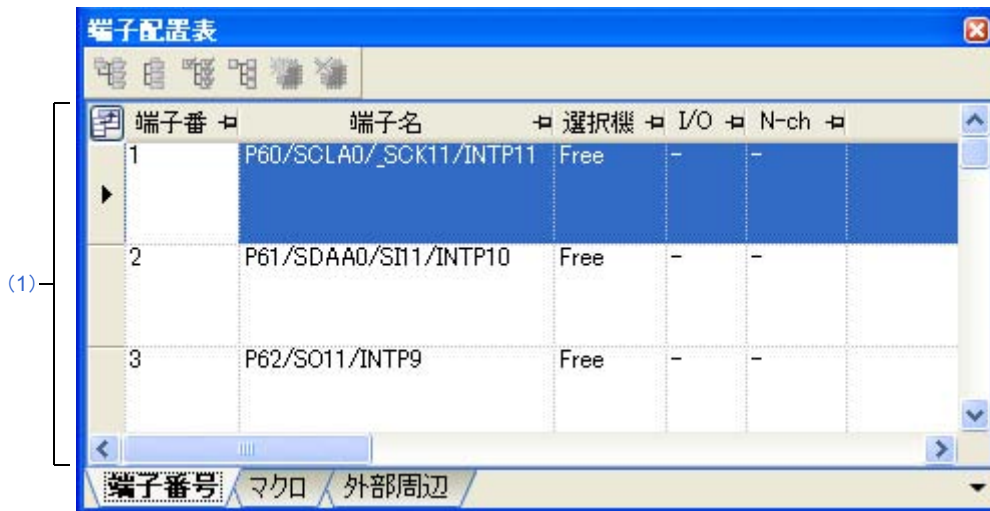
[[ヘルプ] メニュー (端子配置表 パネル専用部分)]

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

[端子番号] タブ

マイクロコントローラの各端子に関する情報を端子番号順で表示します。

図 A—12 [端子番号] タブ



ここでは、次の項目について説明します。

- [\[オープン方法\]](#)
- [\[各エリアの説明\]](#)

[オープン方法]

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

[各エリアの説明]

(1) 端子配置表エリア


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

なお、本エリアの端子配置表は、端子番号順となっています。

以下に、端子配置表を構成する列を示します。

列の見出し	概要
端子番号	該当端子の端子番号を表示します。

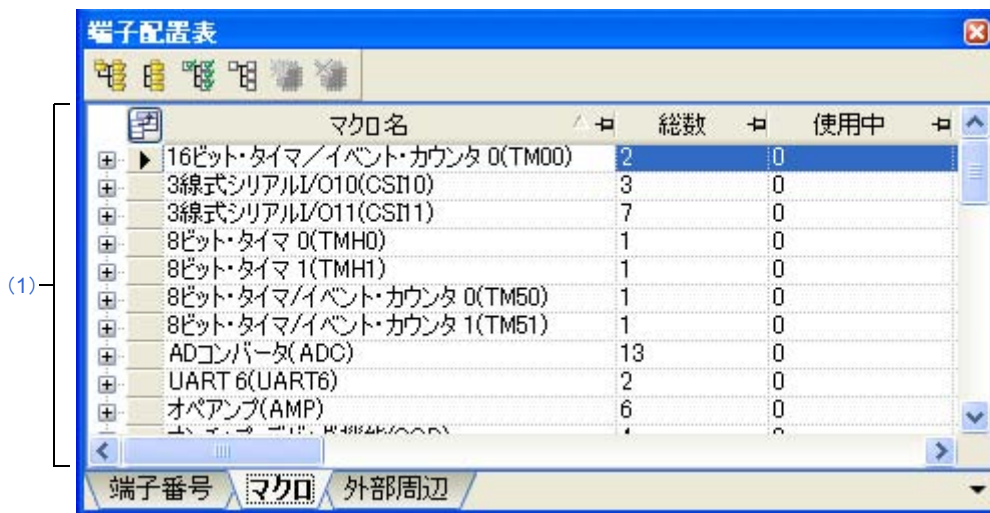
列の見出し	概要
端子名	該当端子の端子名を表示します。
選択機能	該当端子が複数の機能を有している際、“どのような機能で利用するのか”を選択するための領域です。
I/O	該当端子の入出力モードを選択するための領域です。
N-ch	該当端子を出力モードで使用する際、“どのような出力モードで利用するのか”を選択するための領域です。
定義名	該当端子に“ユーザ独自の端子名”を付与するための領域です。 なお、定義名として入力可能な文字数は、256文字までに限られます。
説明	該当端子の機能概要を表示します。
未使用時の処置方法	該当端子を使用しない場合の処置方法を表示します。 なお、本欄は、“選択機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

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

[マクロ] タブ

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

図 A—13 [マクロ] タブ



ここでは、次の項目について説明します。

- [オープン方法]
- [各エリアの説明]

[オープン方法]

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

[各エリアの説明]

(1) 端子配置表エリア

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


(a) 第1階層

以下に、端子配置表を構成する列を示します。

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

(b) 第2階層

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

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

[外部周辺] タブ

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

図 A—14 [外部周辺] タブ



ここでは、次の項目について説明します。

- [\[オープン方法\]](#)
- [\[各エリアの説明\]](#)

[オープン方法]

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

[各エリアの説明]

(1) 端子配置表エリア

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

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


(a) 第1階層

以下に、端子配置表を構成する列を示します。

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

(b) 第2階層

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

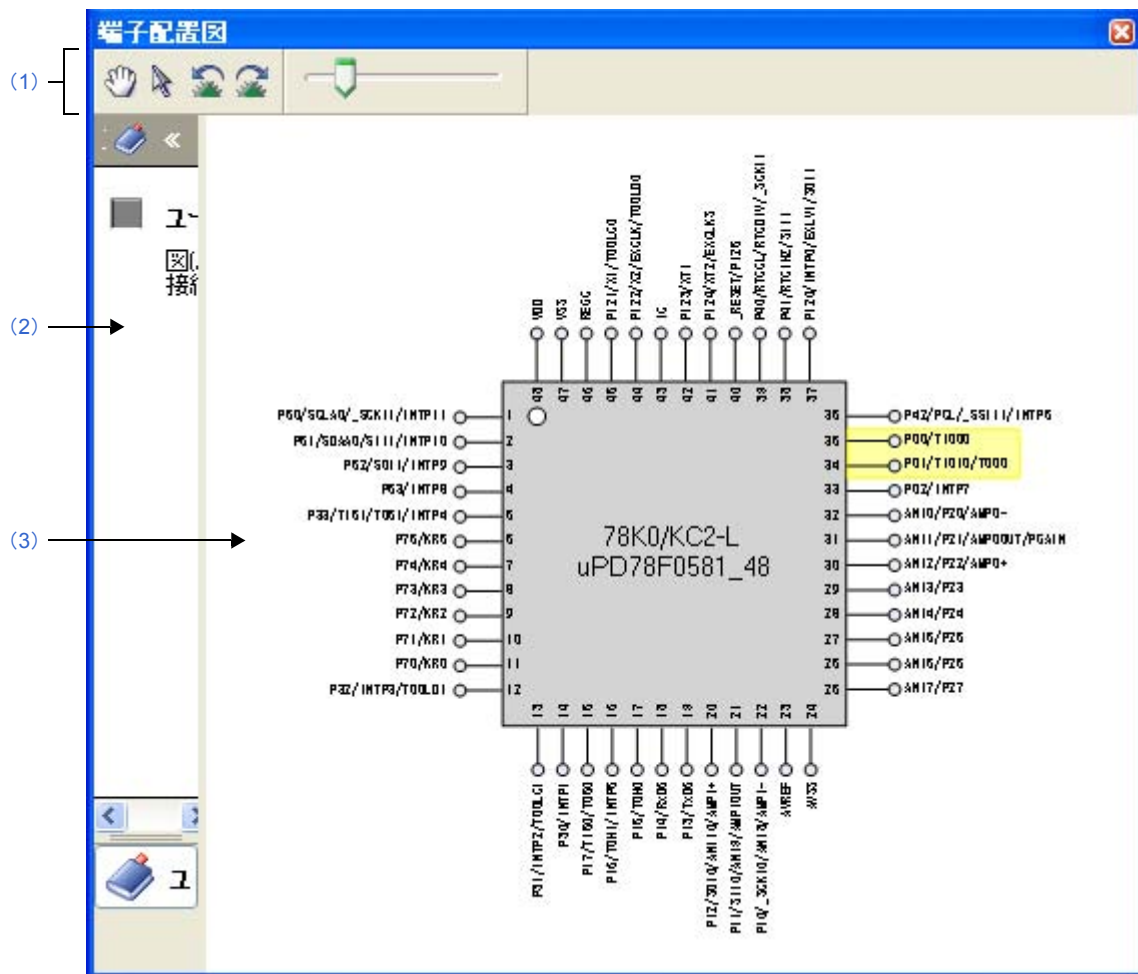
- 備考 1.** “接続数”，“端子番号”，“端子名”，“説明”，“未使用時の処置方法”，“注意事項”については、固定化された情報のため、該当欄に情報を追記することはできません。
- 2.** “選択機能”欄の Free を固有端子名に変更した場合、[端子配置図パネル](#)の該当端子色が[プロパティパネル](#)の[\[端子配置図の設定\] タブ](#)→[\[色設定\]](#)で選択された“未使用端子の表示色”から“使用端子の表示”へと変化します。
- 3.** 列の移動（表示順序の変更）は、端子配置表の該当列をドラッグしたのち、移動先にドロップすることにより行います。
- 4.** “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

端子配置図 パネル

端子配置表 パネルにおける情報の記述状況を表示します。

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

図 A—15 端子配置図 パネル



ここでは、次の項目について説明します。

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

[オープン方法]







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

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


[各エリアの説明]

(1) ツールバー

本エリアは, 以下に示したボタン群から構成されています。

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

(2) [ユーザ定義] エリア

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

(3) 端子配置図エリア

マイクロコントローラの端子配置状況を表示します。

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

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

[[ファイル] メニュー (端子配置図 パネル専用部分)]

端子配置図 を保存	レポート・ファイル (端子配置を用いて設定した情報を保持したファイル: 端子配置図) を既存のファイルに上書き保存します。
名前を付けて 端子配置図 を保存 ...	レポート・ファイル (端子配置を用いて設定した情報を保持したファイル: 端子配置図) に名前を付けて保存するための名前を付けて保存 ダイアログをオープンします。

[[ヘルプ] メニュー (端子配置図 パネル専用部分)]

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

[コンテキスト・メニュー]

端子配置図エリアの端子上、または外部周辺コントローラ上でマウスを右クリックすることにより表示されるコンテキスト・メニューは、以下のとおりです。

(1) 端子上で右クリックした場合

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

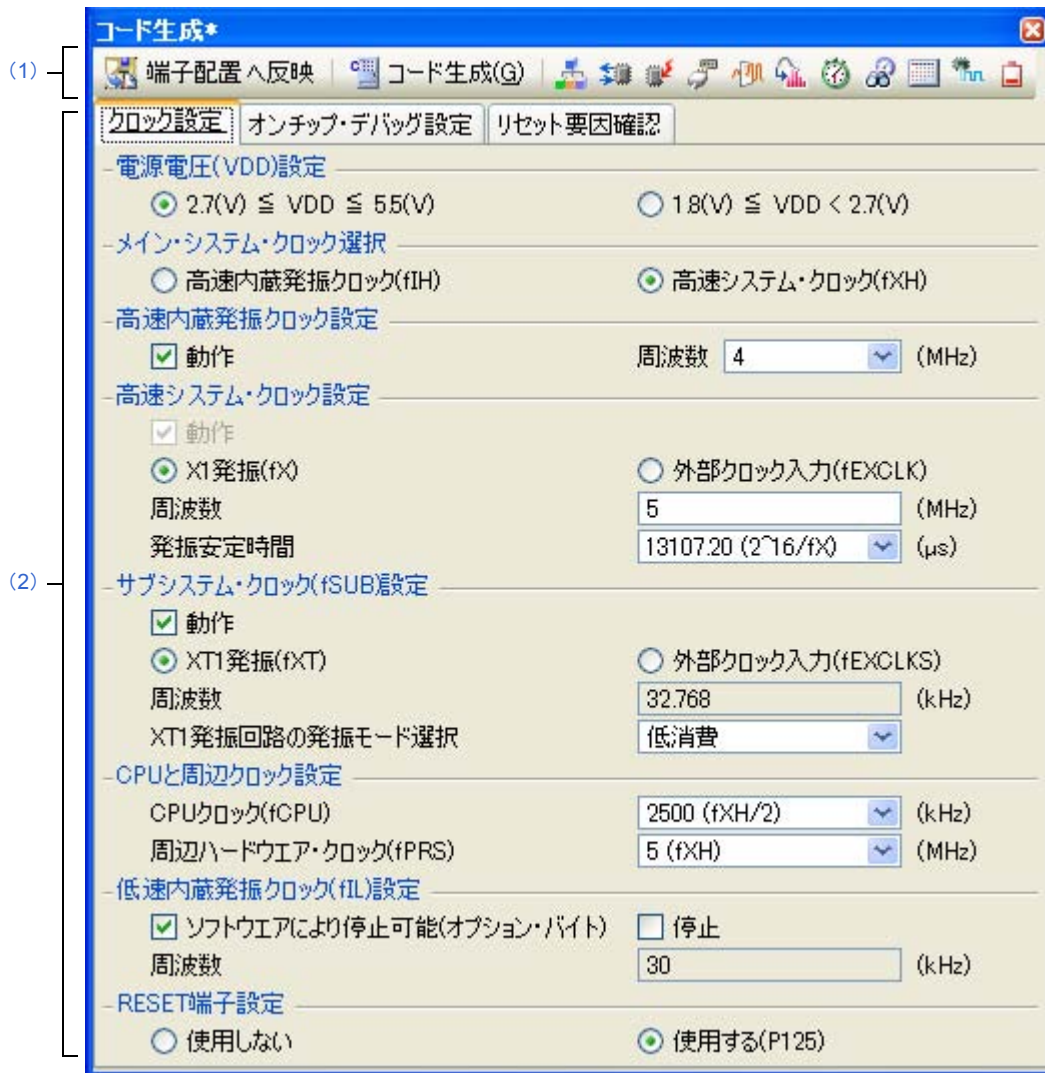
(2) 外部周辺コントローラ上で右クリックした場合

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

コード生成 パネル

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

図 A—16 コード生成 パネル : [システム]





ここでは、次の項目について説明します。

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

[オープン方法]

- プロジェクト・ツリーパネルの [Project name (プロジェクト)] → [コード生成 (設計ツール)] → 周辺機能ノード “[システム], [ポート] など” をダブルクリック

- プロジェクト・ツリーパネルの [Project name (プロジェクト)] → [コード生成 (設計ツール)] → 周辺機能
ノード “[システム], [ポート] など” を選択したのち, [Enter] キーを押下






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






[各エリアの説明]

(1) ツールバー

本エリアは、以下に示したボタン群“周辺機能ボタン”から構成されています。

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

	<p>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するとともに、出力パネルに変更内容を出力します。</p> <p>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。</p>
	<p>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。</p>
	<p>情報設定エリアの表示内容を “マイクロコントローラが提供しているクロック発生回路の機能、オンチップ・デバッグ機能などを制御するうえで必要な情報を設定するための [システム]” へと切り替えます。</p>
	<p>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。</p>
	<p>情報設定エリアの表示内容を “マイクロコントローラが提供している割り込み / キー割り込みの機能を制御するうえで必要な情報を設定するための [割り込み]” へと切り替えます。</p>
	<p>情報設定エリアの表示内容を “マイクロコントローラが提供しているシリアル・アレイ・ユニット、およびシリアル・インタフェースの機能を制御するうえで必要な情報を設定するための [シリアル]” へと切り替えます。</p>
	<p>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 [オペアンプ]” へと切り替えます。</p>
	<p>情報設定エリアの表示内容を “マイクロコントローラが提供しているコンパレータの機能を制御するうえで必要な情報を設定するための [コンパレータ]” へと切り替えます。</p>
	<p>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。</p>

	情報設定エリアの表示内容を“マイクロコントローラが提供しているタイマ・アレイ・ユニットの機能を制御するうえで必要な情報を設定するための [タイマ]”へと切り替えます。
	情報設定エリアの表示内容を“マイクロコントローラが提供しているウォッチドッグ・タイマの機能を制御するうえで必要な情報を設定するための [ウォッチドッグ・タイマ]”へと切り替えます。
	情報設定エリアの表示内容を“マイクロコントローラが提供しているリアルタイム・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クロック出力制御回路の機能を制御するうえで必要な情報を設定するための [クロック出力]”へと切り替えます。
	情報設定エリアの表示内容を“マイクロコントローラが提供している低電圧検出回路の機能を制御するうえで必要な情報を設定するための [低電圧検出回路]”へと切り替えます。

(2) 情報設定エリア

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

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

[[ファイル] メニュー (コード生成 パネル専用部分)]

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

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

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

[[ヘルプ] メニュー (コード生成 パネル専用部分)]

コード生成パネルのヘルプを開く	本パネルのヘルプを表示します。
-----------------	-----------------

コード生成プレビュー パネル

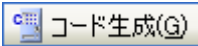
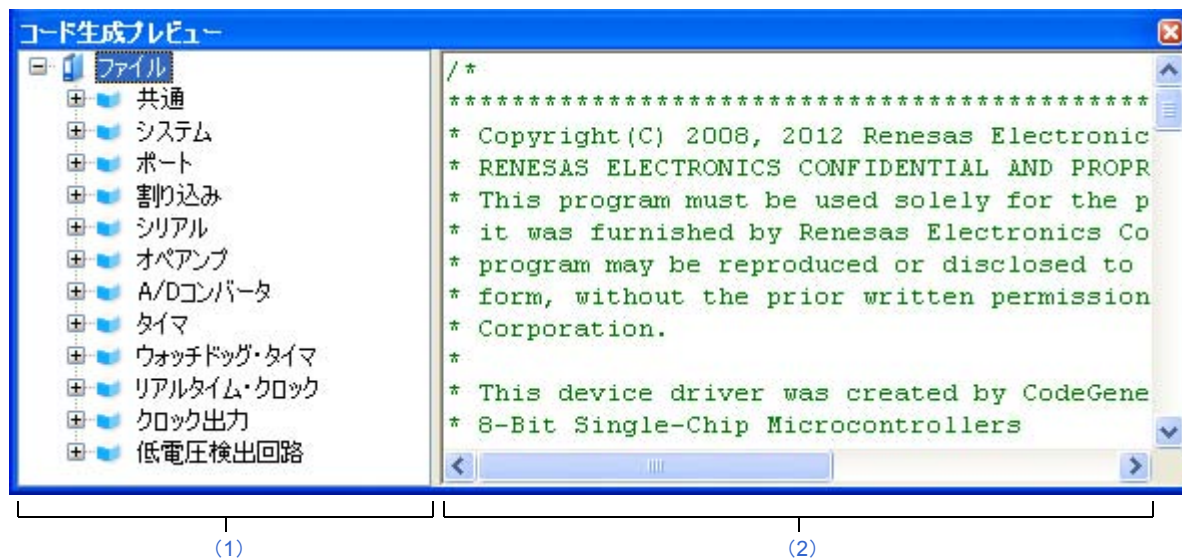
コード生成 パネルの  ボタンをクリックした際に出力されるソース・コード（デバイス・ドライバ・プログラム）の出力有無を API 関数単位で確認／設定するとともに、コード生成 パネルで設定した情報に応じたソース・コードの確認を行います。

図 A—17 コード生成プレビュー パネル



ここでは、次の項目について説明します。

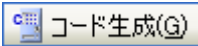
- [オープン方法]
- [各エリアの説明]
- [[ファイル] メニュー（コード生成プレビュー パネル専用部分）]
- [[ヘルプ] メニュー（コード生成プレビュー パネル専用部分）]
- [コンテキスト・メニュー]

[オープン方法]

- [表示] メニュー→ [コード生成プレビュー] を選択

[各エリアの説明]





(1) プレビュー・ツリー

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

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

2. 出力有無の設定は、ツリー内のアイコン上にマウス・カーソルを移動した際、マウスを右クリックすることにより表示されるコンテキスト・メニュー（コードを生成する、コードを生成しない）から行います。
3. 出力有無の設定状況については、アイコン種別により確認することができます。

表 A—2 ソース・コードの出力有無

アイコン種別	概要
	該当 API 関数のソース・コードは、出力されません。 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、出力されます。

(2) ソース・コード表示エリア


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

なお、本エリアに表示されるソース・コードの文字色は、以下の意味を持ちます。

表 A—3 ソース・コードの文字色

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

備考 1. 本パネル内でソース・コードを編集することはできません。

2. 一部の API 関数（シリアル・アレイ・ユニット用 API 関数など）については、ソース・コードの出力時（コード生成パネルの  コード生成(G) ボタンをクリックした際）にレジスタ値 SFR などが計算され確定するものがあります。このため、本パネルに表示されるソース・コードは、実際に出力されるソース・コードと一致しない場合があります。
3. プレビュー・ツリー内のソース・ファイル名、または API 関数名を選択することにより、ソース・コードの表示を切り替えることができます。

[[ファイル] メニュー（コード生成プレビューパネル専用部分）]

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






- 備考 1. レポート・ファイルの出力形式は**プロパティ パネル**の **[出力設定] タブ**→ [レポート出力ファイル形式] で選択された形式（HTML 形式、または CSV 形式）となります。
2. レポート・ファイルの出力先は、**プロパティ パネル**の **[出力設定] タブ**→ [生成先フォルダ] で指定されたフォルダとなります。

[[ヘルプ] メニュー（コード生成プレビュー パネル専用部分）]

コード生成プレビュー パネルのヘルプを開く	本パネルのヘルプを表示します。
-----------------------	-----------------

[コンテキスト・メニュー]

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

コードを生成する	<p>選択された API 関数のソース・コードをプロパティ 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。</p> <p>なお、本コンテキスト・メニューをクリックすることにより、該当 API 関数のアイコンは、 から  へと変化します。</p> <p>本項目は、選択された API 関数が初期化用 API 関数以外であり、プロパティ パネルの [出力設定] タブ→ [API 関数の出力制御] で“初期化関数のみ出力する”が選択されている際には、グレー表記（非選択状態）となります。</p>
コードを生成しない	<p>選択された API 関数のソース・コードをコード生成 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。</p> <p>なお、本コンテキスト・メニューをクリックすることにより、該当 API 関数のアイコンは、 から  へと変化します。</p>
名前を変更する	<p>選択されたファイル名／API 関数名の部位が該当名称を編集するためにエディット・ボックス化されます。</p> <p>該当エディット・ボックスの文字列を編集することにより、ファイル名／API 関数名が変更されます。</p>
名前を元に戻す	<p>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。</p>
プロパティ	<p>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。</p>

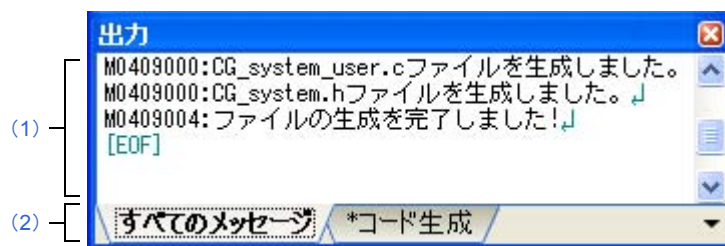
出力パネル

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

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

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

図 A—18 出力パネル



ここでは、次の項目について説明します。

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

[オープン方法]

- [表示] メニュー→ [出力] を選択

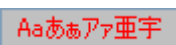
[各エリアの説明]

(1) メッセージ・エリア

各ツールから出力されたメッセージを表示します。

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

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

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

(2) タブ選択エリア

メッセージの出力元を示すタブを選択します。

設計ツールでは，次のタブを使用します。

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

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

[[ファイル] メニュー（出力パネル専用部分）]

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

[[編集] メニュー（出力パネル専用部分）]

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

[コンテキスト・メニュー]

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

コピー	選択している文字列をクリップ・ボードに保存します。
すべて選択	メッセージ・エリアに表示されている全文字列を選択します。
クリア	メッセージ・エリアに表示されている全文字列を消去します。

検索の中止	実行中の文字列検索を中止します。 文字列検索を非実行中の場合、本項目は無効となります。
メッセージに関するヘルプ	メッセージに対応したヘルプを表示します。 ただし、本項目の選択は、カーレットが警告メッセージ/エラー・メッセージの表示行にある場合に限られます。

列の選択 ダイアログ

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




図 A—19 列の選択 ダイアログ



ここでは、次の項目について説明します。

- [オープン方法]
- [各エリアの説明]
- [機能ボタン]

[オープン方法]

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

[各エリアの説明]

(1) 操作対象選択エリア

本ダイアログの操作対象となる端子配置表を選択します。

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

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

図 A—20 操作対象 ([端子番号] タブ)

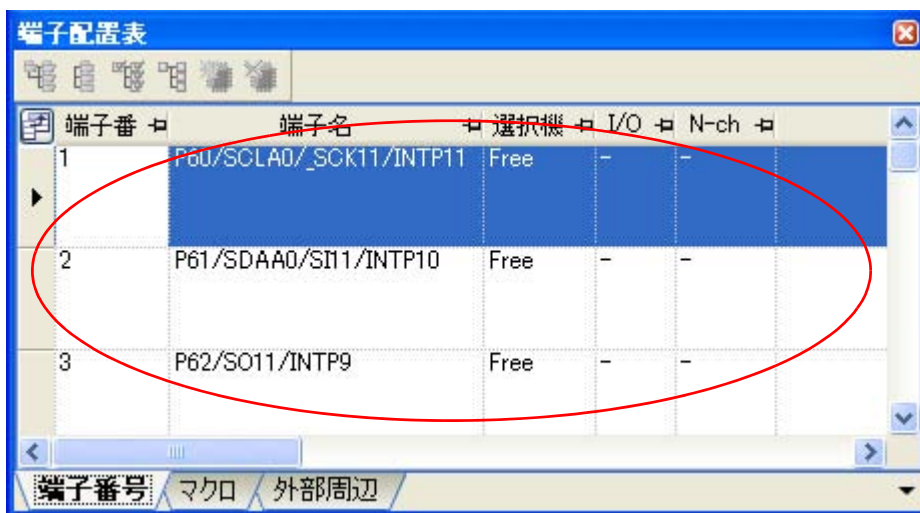


図 A—21 操作対象 ([マクロ] タブ: 第 1 階層)

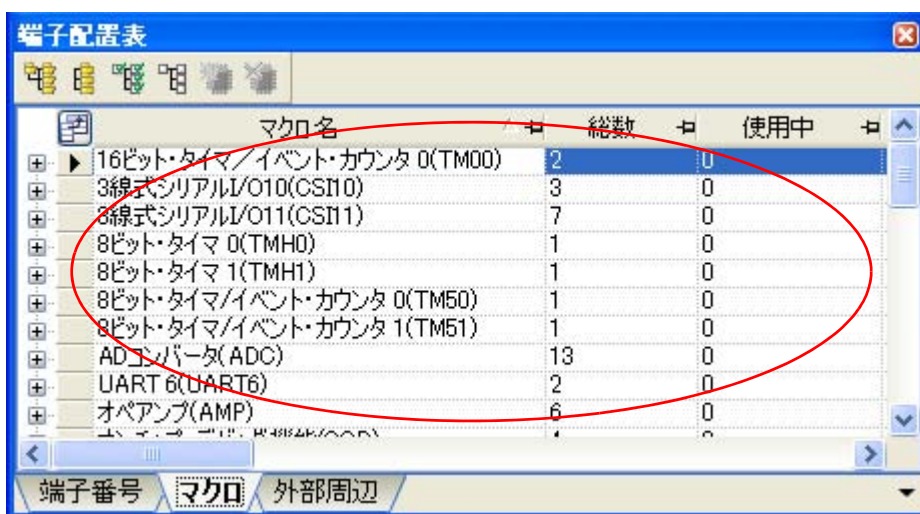


図 A—22 操作対象 ([マクロ] タブ: 第 2 階層)

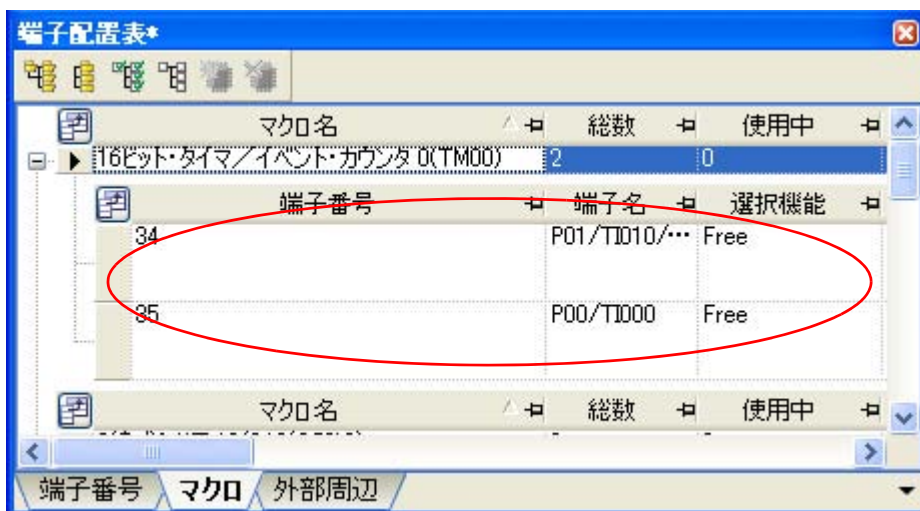


図 A—23 操作対象 ([外部周辺] タブ: 第 1 階層)

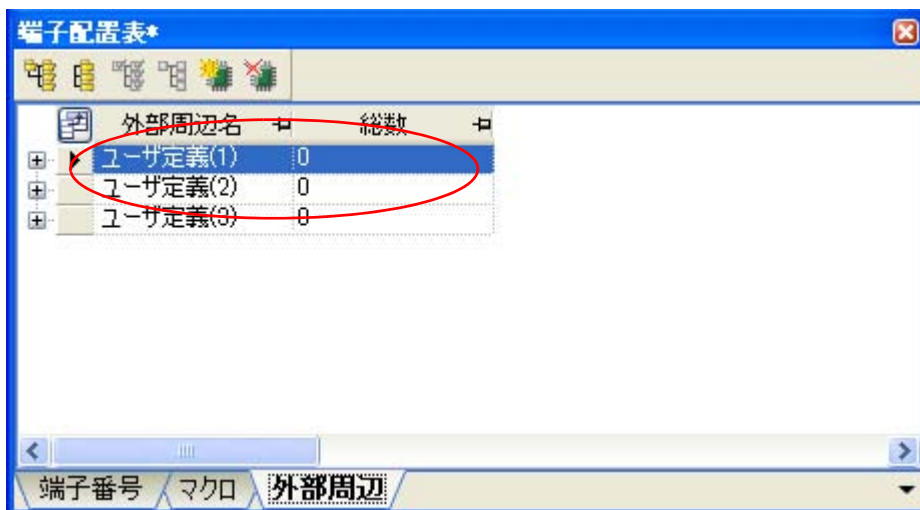
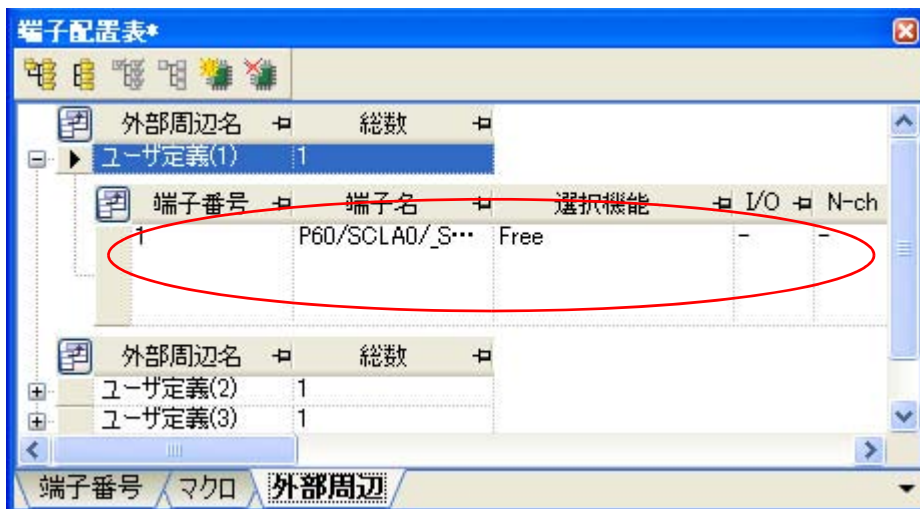


図 A—24 操作対象 ([外部周辺] タブ: 第2階層)



(2) 表示項目選択エリア

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

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

[機能ボタン]

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

新しい列 ダイアログ

端子配置表に列を追加します。

図 A—25 新しい列 ダイアログ



ここでは、次の項目について説明します。

- [オープン方法]
- [各エリアの説明]
- [機能ボタン]

[オープン方法]

- 列の選択 ダイアログの [新しい列 ...] ボタンをクリック

[各エリアの説明]

(1) [名前]

端子配置表に追加する列の見出しを入力します。

なお、名前として入力可能な文字数は、256文字までに限られます。

(2) [型]

端子配置表に追加する列の入力フォームを選択します。

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

[機能ボタン]

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

フォルダの参照 ダイアログ

ファイル（ソース・コード，レポート・ファイルなど）の出力先を設定します。

図 A-26 フォルダの参照 ダイアログ



ここでは，次の項目について説明します。

- [オープン方法]
- [各エリアの説明]
- [機能ボタン]

[オープン方法]

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

[各エリアの説明]

(1) 保存する場所エリア

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

[機能ボタン]

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

名前を付けて保存 ダイアログ

ファイル（レポート・ファイルなど）に名前を付けて保存します。

図 A—27 名前を付けて保存 ダイアログ



ここでは、次の項目について説明します。

- [オープン方法]
- [各エリアの説明]
- [機能ボタン]

[オープン方法]

- [ファイル] メニュー → [名前を付けて <対象> を保存] を選択

[各エリアの説明]

(1) [保存する場所]

ファイル（レポート・ファイルなど）を出力するフォルダを選択します。

(2) ファイルの一覧エリア

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

(3) [ファイル名]

出力するファイルのファイル名を指定します。

(4) [ファイルの種類]

出力するファイルの種類（ファイル・タイプ）を選択します。

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

[機能ボタン]

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

付録B 出力ファイル

本付録では、コード生成が出力するファイルについて説明します。

B.1 概要

以下に、コード生成が出力するファイルの一覧を示します。

表 B—1 出力ファイル

出力単位	ファイル名	出力内容
各周辺機能	CG_周辺機能名.c	初期化関数, API 関数
	CG_周辺機能名_user.c	割り込み関数 (MD_INTxxx), コールバック関数
	CG_周辺機能名.h	レジスタへの代入値マクロを定義
プロジェクト	CG_main.c	main 関数, R_MAIN_UserInit 関数
	CG_systeminit.c	各周辺機能の初期化関数コール CG_ReadResetSource のコール
	CG_macrodriver.h	全ソース・ファイルで共通使用するマクロを定義
	CG_userdefine.h	空ファイル (ユーザ定義用)
	CG_lk.dr	リンク・ディレクティブ
	CG_option.asm	オプション・バイト, MINICUBE2 用 ROM 確保

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_ChagngePIIMode
	CG_system_user.c	CLOCK_UserInit CG_ReadResetSource
	CG_system.h	—
ポート	CG_port.c	PORT_Init PORT_ChangePmnInput

周辺機能	ファイル名	含まれる API 関数名
ポート	CG_port.c	PORT_ChangePmnOutput
	CG_port_user.c	PORT_UserInit
	CG_port.h	—
割り込み	CG_int.c	INTP_Init KEY_Init INT_MaskableInterruptEnable INTPn_Disable INTPn_Enable KEY_Disable KEY_Enable
	CG_int_user.c	INTP_UserInit KEY_UserInit MD_INTPn MD_INTKR
	CG_int.h	—
シリアル	CG_serial.c	UART6_Init UART6_Start UART6_Stop UART6_SendData UART6_ReceiveData CSI1n_Init CSI1n_Start CSI1n_Stop CSI1n_ReceiveData CSI1n_SendReceiveData IICA_Init IICA_Stop IICA_MasterSendStart IICA_MasterReceiveStart IICA_StopCondition IICA_SlaveSendStart IICA_SlaveReceiveStart
	CG_serial_user.c	UART6_UserInit UART6_SendEndCallback UART6_ReceiveEndCallback UART6_SoftOverRunCallback UART6_ErrorCallback CSI1n_UserInit CSI1n_SendEndCallback CSI1n_ReceiveEndCallback IICA_UserInit IICA_MasterSendEndCallback IICA_MasterReceiveEndCallback

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	IICA_MasterErrorCallback IICA_SlaveSendEndCallback IICA_SlaveReceiveEndCallback IICA_SlaveErrorCallback IICA_GetStopConditionCallback MD_INTSR6 MD_INTSRE6 MD_INTST6 MD_INTCSI1n MD_INTIICA0
	CG_serial.h	—
オペアンプ	CG_opamp.c	OPAMP_Init PGA_Start PGA_Stop PGA_ChangePGAFactor AMP_Start AMP_Stop AMPn_Start AMPn_Stop
	CG_opamp_user.c	OPAMP_UserInit
	CG_opamp.h	—
コンパレータ	CG_comparator.c	Comparator_Init Comparatorm_Start Comparatorm_Stop
	CG_comparator_user.c	Comparator_UserInit MD_INTCMPn
	CG_comparator.h	—
A/D コンバータ	CG_ad.c	AD_Init AD_ComparatorOn AD_ComparatorOff AD_Start AD_Stop AD_SelectADChannel AD_Read AD_ReadByte
	CG_ad_user.c	AD_UserInit MD_INTAD
	CG_ad.h	—
タイマ	CG_timer.c	TMX_Init TMXn_Start TMXn_Stop TMXn_ChangeDuty

周辺機能	ファイル名	含まれる API 関数名
タイマ	CG_timer.c	TMXn_ChangeDualDuty TMX_EnableHighImpedanceState TMX_DisableHighImpedanceState TM00_Init TM00_Start TM00_Stop TM00_ChangeTimerCondition TM00_GetFreeRunningValue TM00_SoftwareTriggerOn TM00_ChangeDuty TM00_GetPulseWidth TM5n_Init TM5n_Start TM5n_Stop TM5n_ChangeTimerCondition TM5n_ChangeDuty TMHn_Init TMHn_Start TMHn_Stop TMHn_ChangeTimerCondition TMHn_ChangeDuty TMH1_CarrierOutputEnable TMH1_CarrierOutputDisable
	CG_timer_user.c	TM00_UserInit TM5n_UserInit TMHn_UserInit MD_INTTMXn MD_INTTM0n0 MD_INTTM5n MD_INTMHn
	CG_timer.h	—
ウォッチドッグ・タイマ	CG_wdt.c	WDT_Restart
	CG_wdt.h	—
リアルタイム・カウンタ	CG_rtc.c	RTC_Init RTC_PowerOff RTC_CounterEnable RTC_CounterDisable RTC_SetHourSystem RTC_CounterSet RTC_CounterGet RTC_ConstPeriodInterruptEnable RTC_ConstPeriodInterruptDisable RTC_AlarmEnable RTC_AlarmDisable

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_AlarmSet RTC_AlarmGet RTC_IntervalStart RTC_IntervalStop RTC_IntervalInterruptEnable RTC_IntervalInterruptDisable RTC_RTC1HZ_OutputEnable RTC_RTC1HZ_OutputDisable RTC_RTCCCL_OutputEnable RTC_RTCCCL_OutputDisable RTC_RTCDIV_OutputEnable RTC_RTCDIV_OutputDisable RTC_ChangeCorrectionValue
	CG_rtc_user.c	RTC_UserInit RTC_ConstPeriodInterruptCallback RTC_AlarmInterruptCallback MD_INTRTC MD_INTRTCI
	CG_rtc.h	—
クロック出力	CG_pcl.c	PCL_Init PCL_Start PCL_Stop PCL_ChangeFreq
	CG_pcl_user.c	PCL_UserInit
	CG_pcl.h	—
低電圧検出回路	CG_lvi.c	LVI_Init LVI_InterruptModeStart LVI_ResetModeStart LVI_Stop LVI_SetLVILevel
	CG_lvi_user.c	LVI_UserInit MD_INTLVI
	CG_lvi.h	—

付録C API関数

本付録では、コード生成が出力するAPI関数について説明します。

C.1 概要

以下に、コード生成がAPI関数を出力する際の命名規則を示します。

- マクロ名

すべて大文字。

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

- ローカル変数名

すべて小文字。

- グローバル変数名

先頭に“g”を付与し、構成単語の先頭のみ大文字。

- グローバル変数へのポインタ名

先頭に“gp”を付与し、構成単語の先頭のみ大文字。

- 列挙指定子 enum の要素名

すべて大文字。

C.2 出力関数

以下に、コード生成が出力するAPI関数の一覧を示します。

表 C—1 API関数一覧

周辺機能	API関数名	機能概要
システム	CLOCK_Init	クロック発生回路の機能、オンチップ・デバッグ機能などを制御するうえで必要となる初期化処理を行います。
	CLOCK_UserInit	クロック発生回路、オンチップ・デバッグなどに関するユーザ独自の初期化処理を行います。
	CG_ReadResetSource	内部リセットの発生に伴う処理を行います。
	CG_ChangeClockMode	CPUクロック／周辺ハードウェア・クロックを変更します。
	CG_ChangeFrequency	CPUクロック／周辺ハードウェア・クロックの分周比を変更します。

周辺機能	API 関数名	機能概要
システム	CG_SelectPowerSaveMode	CPU のスタンバイ・モードを設定します。
	CG_SelectStabTime	X1 クロックの発振安定時間を設定します。
	CG_ChagngnPllMode	PLL 機能の動作を制御します。
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（外部割り込み要求）INTP _n の受け付けを禁止します。
	INTPn_Enable	マスカブル割り込み（外部割り込み要求）INTP _n の受け付けを許可します。
	KEY_Disable	キー割り込み INTKR の受け付けを禁止します。
	KEY_Enable	キー割り込み INTKR の受け付けを許可します。
シリアル	UART6_Init	シリアル・インタフェース（UART6）の初期化処理を行います。
	UART6_UserInit	シリアル・インタフェース（UART6）に関するユーザ独自の初期化処理を行います。
	UART6_Start	UART 通信を待機状態にします。
	UART6_Stop	UART 通信を終了します。
	UART6_SendData	データの UART 送信を開始します。
	UART6_ReceiveData	データの UART 受信を開始します。
	UART6_SendEndCallback	UART 送信完了割り込み INTST6 の発生に伴う処理を行います。
	UART6_ReceiveEndCallback	UART 受信完了割り込み INTSR6 の発生に伴う処理を行います。
	UART6_SoftOverRunCallback	UART 受信完了割り込み INTSR6 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	UART6_ErrorCallback	UART 通信におけるエラー割り込み INTSRE6 の発生に伴う処理を行います。
	CSI1n_Init	シリアル・インタフェース (CSI1n) の初期化処理を行います。
	CSI1n_UserInit	シリアル・インタフェース (CSI1n) に関するユーザ独自の初期化処理を行います。
	CSI1n_Start	CSI1n 通信を待機状態にします。
	CSI1n_Stop	CSI1n 通信を終了します。
	CSI1n_ReceiveData	データの CSI1n 受信を開始します。
	CSI1n_SendReceiveData	データの CSI1n 送受信を開始します。
	CSI1n_SendEndCallback	CSI1n 通信完了割り込み INTCSI1n の発生に伴う処理を行います。
	CSI1n_ReceiveEndCallback	CSI1n 通信完了割り込み INTCSI1n の発生に伴う処理を行います。
	IICA_Init	シリアル・インタフェース (IICA) の初期化処理を行います。
	IICA_UserInit	シリアル・インタフェース (IICA) に関するユーザ独自の初期化処理を行います。
	IICA_Stop	IICA 通信を終了します。
	IICA_MasterSendStart	IICA マスタ送信を開始します。
	IICA_MasterReceiveStart	IICA マスタ受信を開始します。
	IICA_StopCondition	ストップ・コンディションを発生します。
	IICA_MasterSendEndCallback	IICA 通信完了割り込み INTIICA0 の発生に伴う処理を行います。
	IICA_MasterReceiveEndCallback	IICA 通信完了割り込み INTIICA0 の発生に伴う処理を行います。
	IICA_MasterErrorCallback	IICA マスタ通信におけるエラーの検出に伴う処理を行います。
	IICA_SlaveSendStart	IICA スレーブ送信を開始します。
	IICA_SlaveReceiveStart	IICA スレーブ受信を開始します。
	IICA_SlaveSendEndCallback	IICA 通信完了割り込み INTIICA0 の発生に伴う処理を行います。
	IICA_SlaveReceiveEndCallback	IICA 通信完了割り込み INTIICA0 の発生に伴う処理を行います。
	IICA_SlaveErrorCallback	IICA スレーブ通信におけるエラーの検出に伴う処理を行います。
IICA_GetStopConditionCallback	IICA スレーブ通信におけるストップ・コンディションの検出に伴う処理を行います。	
オペアンプ	OPAMP_Init	オペアンプの機能を制御するうえで必要となる初期化処理を行います。

周辺機能	API 関数名	機能概要
オペアンプ	OPAMP_UserInit	オペアンプに関するユーザ独自の初期化処理を行います。
	PGA_Start	オペアンプ (PGA モード) の動作を開始します。
	PGA_Stop	オペアンプ (PGA モード) の動作を停止します。
	PGA_ChangePGAFactor	オペアンプ (PGA モード) における入力電圧の増幅率を設定します。
	AMP_Start	オペアンプ (シングル・アンプ・モード) の動作を開始します。
	AMP_Stop	オペアンプ (シングル・アンプ・モード) の動作を停止します。
	AMPn_Start	オペアンプ n (シングル・アンプ・モード) の動作を開始します。
	AMPn_Stop	オペアンプ n (シングル・アンプ・モード) の動作を停止します。
コンパレータ	Comparator_Init	コンパレータの機能を制御するうえで必要となる初期化処理を行います。
	Comparator_UserInit	コンパレータに関するユーザ独自の初期化処理を行います。
	Comparatorm_Start	コンパレータ n の動作を開始します。
	Comparatorm_Stop	コンパレータ n の動作を停止します。
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_ComparatorOn	電圧コンパレータを動作許可状態に設定します。
	AD_ComparatorOff	電圧コンパレータを動作停止状態に設定します。
	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_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。
タイマ	TMX_Init	16 ビット・タイマ Xn の機能を制御するうえで必要となる初期化処理を行います。
	TMXn_Start	16 ビット・タイマ Xn のカウントを開始します。
	TMXn_Stop	16 ビット・タイマ Xn のカウントを終了します。
	TMXn_ChangeDuty	TOX0 n 端子にシングル出力する PWM 信号のデューティ比を変更します。
	TMXn_ChangeDualDuty	TOX0 n 端子にデュアル出力する PWM 信号のデューティ比を変更します。

周辺機能	API 関数名	機能概要
タイマ	TMX_EnableHighImpedanceState	16 ビット・タイマ Xn のハイ・インピーダンス出力を開始します。
	TMX_DisableHighImpedanceState	16 ビット・タイマ Xn のハイ・インピーダンス出力を終了します。
	TM00_Init	16 ビット・タイマ/イベント・カウンタ 00 の機能を制御するうえで必要となる初期化処理を行います。
	TM00_UserInit	16 ビット・タイマ/イベント・カウンタ 00 に関するユーザ独自の初期化処理を行います。
	TM00_Start	16 ビット・タイマ/イベント・カウンタ 00 のカウントを開始します。
	TM00_Stop	16 ビット・タイマ/イベント・カウンタ 00 のカウントを終了します。
	TM00_ChangeTimerCondition	キャプチャ/コンペア・コントロール・レジスタ (CRC00) の内容を変更します。
	TM00_GetFreeRunningValue	キャプチャ・レジスタ (CR0 n) の内容を獲得します。
	TM00_SoftwareTriggerOn	ワンショット・パルス出力のためのトリガ (ソフトウェア・トリガ) を発生させます。
	TM00_ChangeDuty	TO00 端子に出力する信号のデューティ比を変更します。
	TM00_GetPulseWidth	TIO n 0 端子に対する入力信号 (入力パルス) のハイ/ロウ・レベルの測定幅を獲得します。
	TM5n_Init	8 ビット・タイマ/イベント・カウンタ 5 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。
	TM5n_UserInit	8 ビット・タイマ/イベント・カウンタ 5 n に関するユーザ独自の初期化処理を行います。
	TM5n_Start	8 ビット・タイマ/イベント・カウンタ 5 n のカウントを開始します。
	TM5n_Stop	8 ビット・タイマ/イベント・カウンタ 5 n のカウントを終了します。
	TM5n_ChangeTimerCondition	8 ビット・タイマ・コンペア・レジスタ 5 n (CR5 n) の内容を変更します。
	TM5n_ChangeDuty	TO5 n 端子に出力する PWM 信号のデューティ比を変更します。
	TMHn_Init	8 ビット・タイマ H 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。
	TMHn_UserInit	8 ビット・タイマ H n に関するユーザ独自の初期化処理を行います。
	TMHn_Start	8 ビット・タイマ H n のカウントを開始します。
TMHn_Stop	8 ビット・タイマ H n のカウントを終了します。	

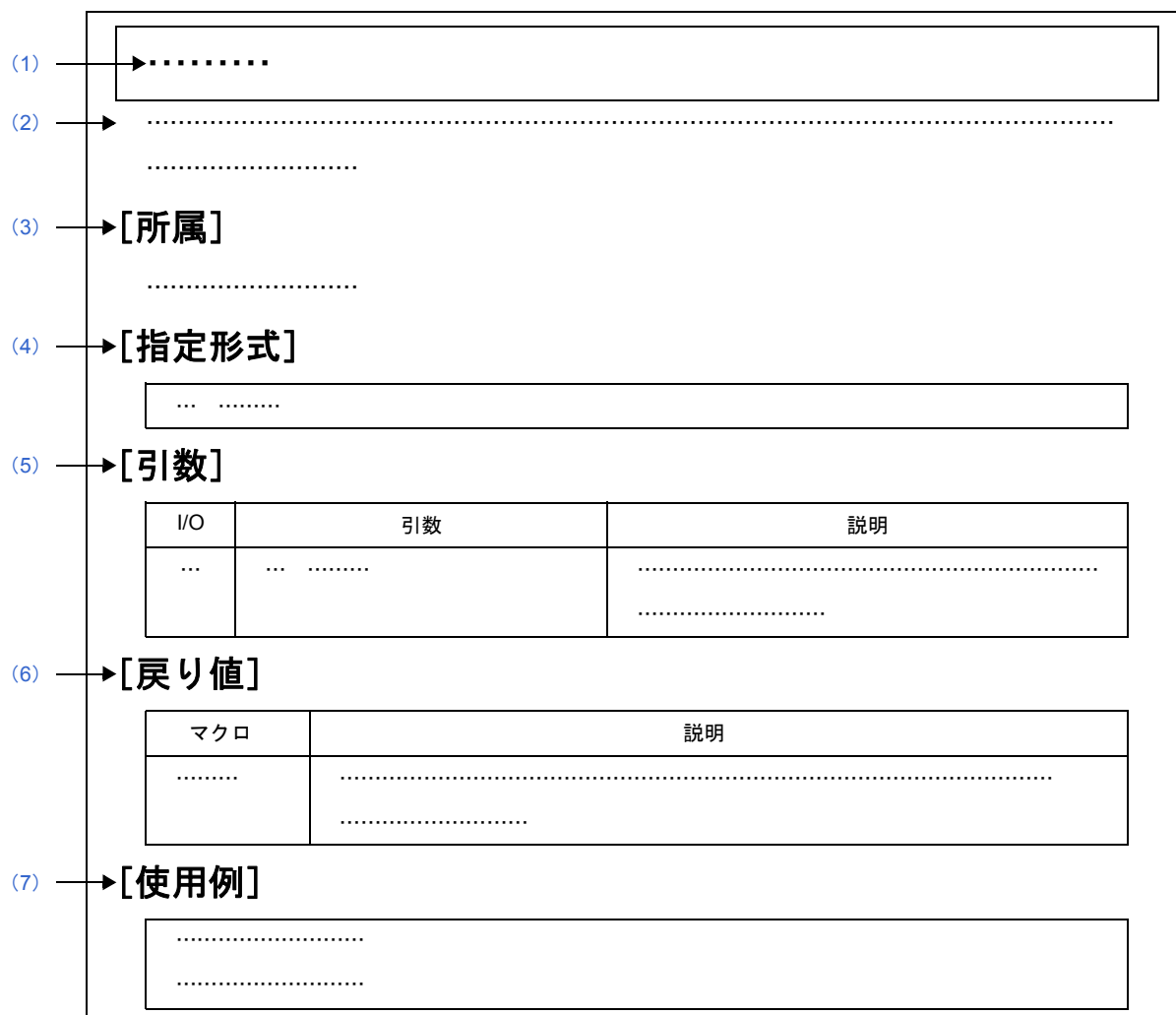
周辺機能	API 関数名	機能概要
タイマ	TMHn_ChangeTimerCondition	8ビット・タイマHコンペア・レジスタ0n / 1n (CMP0n / CMP1n) の内容を変更します。
	TMHn_ChangeDuty	TOHn 端子に出力する PWM 信号のデューティ比を変更します。
	TMH1_CarrierOutputEnable	8ビット・タイマH1 (キャリア・ジェネレータ・モード) のキャリア・パルス出力を開始します。
	TMH1_CarrierOutputDisable	8ビット・タイマH1 (キャリア・ジェネレータ・モード) のキャリア・パルス出力を終了します。
ウォッチドッグ・タイマ	WDT_Restart	ウォッチドッグ・タイマのカウンタをクリアしたのち、カウント処理を再開します。
リアルタイム・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_PowerOff	リアルタイム・カウンタに対するクロック供給を停止します。
	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 INTRTC の発生周期を設定したのち、定周期割り込み機能を開始します。
	RTC_ConstPeriodInterruptDisable	定周期割り込み機能を終了します。
	RTC_ConstPeriodInterruptCallback	定周期割り込み INTRTC の発生に伴う処理を行います。
	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_AlarmInterruptCallback	アラーム割り込み INTRTC の発生に伴う処理を行います。
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。	

周辺機能	API 関数名	機能概要
リアルタイム・カウンタ	RTC_IntervallInterruptEnable	割り込み INTRTCI の発生周期を設定したのち、インターバル割り込み機能を開始します。
	RTC_IntervallInterruptDisable	インターバル割り込み機能を終了します。
	RTC_RTC1HZ_OutputEnable	RTC1HZ 端子に対するリアルタイム・カウンタ補正クロック（1 Hz）の出力を許可します。
	RTC_RTC1HZ_OutputDisable	RTC1HZ 端子に対するリアルタイム・カウンタ補正クロック（1 Hz）の出力を禁止します。
	RTC_RTCCL_OutputEnable	RTCCL 端子に対するリアルタイム・カウンタ・クロック（32 kHz 原発）の出力を許可します。
	RTC_RTCCL_OutputDisable	RTCCL 端子に対するリアルタイム・カウンタ・クロック（32 kHz 原発）の出力を禁止します。
	RTC_RTCDIV_OutputEnable	RTCDIV 端子に対するリアルタイム・カウンタ・クロック（32 kHz 分周）の出力を許可します。
	RTC_RTCDIV_OutputDisable	RTCDIV 端子に対するリアルタイム・カウンタ・クロック（32 kHz 分周）の出力を禁止します。
	RTC_ChangeCorrectionValue	時計誤差を補正するタイミング、および補正值を変更します。
クロック出力	PCL_Init	クロック出力制御回路の機能を制御するうえで必要となる初期化処理を行います。
	PCL_UserInit	クロック出力制御回路に関するユーザ独自の初期化処理を行います。
	PCL_Start	クロック出力を開始します。
	PCL_Stop	クロック出力を停止します。
	PCL_ChangeFreq	PCL 端子への出力クロックを変更します。
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_Stop	低電圧検出動作を停止します。
	LVI_SetLVIlevel	低電圧検出レベルを設定します。

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	クロック発生回路の機能、オンチップ・デバッグ機能などを制御するうえで必要となる初期化処理を行います。
CLOCK_UserInit	クロック発生回路、オンチップ・デバッグなどに関するユーザ独自の初期化処理を行います。
CG_ReadResetSource	内部リセットの発生に伴う処理を行います。
CG_ChangeClockMode	CPU クロック／周辺ハードウェア・クロックを変更します。
CG_ChangeFrequency	CPU クロック／周辺ハードウェア・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	X1 クロックの発振安定時間を設定します。
CG_ChangePIIMode	PLL 機能の動作を制御します。

CLOCK_Init

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

[所属]

CG_system.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

CLOCK_UserInit

クロック発生回路、オンチップ・デバッグなどに関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、[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

内部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、リセット信号 RESET 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 ();      /* リセット信号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 flag = 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 ( flag & 0x1 ) {         /* 発生要因の判別 : LVIRF フラグのチェック */
        ..... /* 低電圧検出回路による内部リセット要求に対応した処理 */
    } else if ( flag & 0x10 ) { /* 発生要因の判別 : WDTRF フラグのチェック */
        ..... /* ウォッチドッグ・タイマによる内部リセット要求に対応した処理 */
    }
    .....
}
```

CG_ChangeClockMode

CPUクロック／周辺ハードウェア・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;	<p>CPUクロック (fCPU) / 周辺ハードウェア・クロック (fPRS) の種類</p> <p>HIOCLK : fCPU / fPRS → 高速内蔵発振クロック</p> <p>HIOSYSCLK : fCPU → 高速内蔵発振クロック, fPRS → 高速システム・クロック</p> <p>SYSX1CLK : fCPU / fPRS → X1クロック</p> <p>SYSEXTCLK : fCPU / fPRS → 外部メイン・システム・クロック</p> <p>SUBXT1CLK : fCPU → XT1クロック</p> <p>SUBEXTCLK : fCPU → 外部サブシステム・クロック</p>

備考 SUBXT1CLK, SUBEXTCLK の指定は、対象デバイスが 78K0/KC2-L の場合に限られます。

[戻り値]

マクロ	説明
MD_OK	正常終了
MD_ERROR1	異常終了 - fCPU / fPRS を高速内蔵発振クロックへ変更することはできません。
MD_ERROR2	異常終了 - fCPU を高速内蔵発振クロックへ変更することはできません。 - fPRS を高速システム・クロックへ変更することはできません。
MD_ERROR3	異常終了 - fCPU / fPRS を X1クロックへ変更することはできません。

マクロ	説明
MD_ERROR4	異常終了 - fCPU / fPRS を外部メイン・システム・クロックへ変更することはできません。
MD_ERROR5	異常終了【Kx2-L】 - fCPU を XT1 クロックへ変更することはできません。
MD_ERROR6	異常終了【Kx2-L】 - fCPU を外部サブシステム・クロックへ変更することはできません。
MD_ARGERROR	引数の指定が不正

備考 戻り値 MD_ERROR5, MD_ERROR6 は, 対象デバイスが 78K0/KC2-L の場合に限られます。

CG_ChangeFrequency

CPUクロック／周辺ハードウェア・クロックの分周比を変更します。

[所属]

CG_system.c

[指定形式]

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

[引数]

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

備考 fMAIN は、メイン・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

CG_SelectPowerSaveMode

CPUのスタンバイ・モードを設定します。

[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 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_ERROR	異常終了【Kx2-L】 - CPU が XT1 クロックで動作している場合、STOP モードを指定することはできません。
MD_ARGERROR	引数の指定が不正

備考 戻り値 MD_ERROR は、対象デバイスが 78K0/KC2-L の場合に限られます。

[使用例]

以下に、スタンバイ・モードを“STOP モード”へと移行させる際の例を示します。

【CG_main.c】

```
#include "CG_macrodriver.h"
#include "CG_system.h"
void main ( void ) {
    MD_STATUS ret;
```

```
.....
TM00_Stop ();                                /* カウントの終了 */
ret = CG_SelectPowerSaveMode ( PSSTOP );     /* STOP モードへの移行 */
if ( ret != MD_OK ) {
    while ( 1 );
}
TM00_Init ();                                /* TM00 の初期化 */
TM00_Start ();                               /* カウントの開始 */
.....
}
```

CG_SelectStabTime

X1クロックの発振安定時間を設定します。

[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 ¹¹ /fx STLEVEL1 : 2 ¹³ /fx STLEVEL2 : 2 ¹⁴ /fx STLEVEL3 : 2 ¹⁵ /fx STLEVEL4 : 2 ¹⁶ /fx

備考 fx は、X1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

CG_ChagngePllMode

PLL 機能の動作を制御します。

[所属]

CG_system.c

[指定形式]

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

[引数]

I/O	引数	説明
I	enum PllMode <i>pllmode</i> ;	動作の制御 SYSPLLON : 動作許可 SYSPLLOFF : 停止

[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.3.2 ポート

以下に、コード生成がポート用として出力するAPI関数の一覧を示します。

表 C—3 ポート用API関数

API関数名	機能概要
PORT_Init	ポートの機能を制御するうえで必要となる初期化処理を行います。
PORT_UserInit	ポートに関するユーザ独自の初期化処理を行います。
PORT_ChangePmnInput	端子の入出力モードを出力モードから入力モードへと切り替えます。
PORT_ChangePmnOutput	端子の入出力モードを入力モードから出力モードへと切り替えます。

PORT_Init

ポートの機能を制御するうえで必要となる初期化処理を行います。

[所属]

CG_port.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

PORT_UserInit

ポートに関するユーザ独自の初期化処理を行います。

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

[所属]

CG_port_user.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

PORT_ChangePmnInput

端子の入出力モードを出力モードから入力モードへと切り替えます。

[所属]

CG_port.c

[指定形式]

本 API 関数の指定形式は、対象端子に内蔵プルアップ抵抗／SMBus 入力バッファが存在するか否かにより異なります。

- 内蔵プルアップ抵抗：なし，SMBus 入力バッファ：なし

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

- 内蔵プルアップ抵抗：あり，SMBus 入力バッファ：なし

```
#include "CG_macrodriver.h"
void PORT_ChangePmnInput ( BOOL enablepu );
```

- 内蔵プルアップ抵抗：あり，SMBus 入力バッファ：あり

```
#include "CG_macrodriver.h"
void PORT_ChangePmnInput ( BOOL enablepu, BOOL enablesmbus );
```

備考 *mn* は、ポート番号を意味します。

[引数]

I/O	引数	説明
I	BOOL <i>enablepu</i> ;	内蔵プルアップ抵抗の使用有無 MD_TRUE： 使用する MD_FALSE： 使用しない
I	BOOL <i>enablesmbus</i> ;	入力バッファの種類 MD_TRUE： SMBus 入力バッファ MD_FALSE： 通常入力バッファ

[戻り値]

なし

[使用例 1]

以下に、P00 端子（内蔵プルアップ抵抗：あり，SMBus 入力バッファ：なし）を

入出力モードの種類： 入力モード

内蔵プルアップ抵抗の使用有無： 使用する

に変更する際の例を示します。

【CG_main.c】

```
#include "CG_macrodriver.h"
void main ( void ) {
    .....
    PORT_ChangeP00Input ( MD_TRUE ); /* 入出力モードの切り替え */
    .....
}
```

[使用例 2]

以下に、P00 端子（内蔵プルアップ抵抗：あり，SMBus 入力バッファ：なし）を

入出力モードの種類： 入力モード

内蔵プルアップ抵抗の使用有無： 使用しない

に変更する際の例を示します。

【CG_main.c】

```
#include "CG_macrodriver.h"
void main ( void ) {
    .....
    PORT_ChangeP00Input ( MD_FALSE ); /* 入出力モードの切り替え */
    .....
}
```

[使用例 3]

以下に、P04 端子（内蔵プルアップ抵抗：あり，SMBus 入力バッファ：あり）を

入出力モードの種類： 入力モード

内蔵プルアップ抵抗の使用有無： 使用しない

入力バッファの種類： SMBus 入力バッファ

に変更する際の例を示します。

【CG_main.c】

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

```
.....  
PORT_ChangeP04Input ( MD_FALSE, MD_TRUE ); /* 入出力モードの切り替え */  
.....  
}
```

PORT_ChangePmnOutput

端子の入出力モードを入力モードから出力モードへと切り替えます。

[所属]

CG_port.c

[指定形式]

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

- N-ch オープン・ドレイン出力：なし

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

- N-ch オープン・ドレイン出力：あり

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

備考 *nm* は、ポート番号を意味します。

[引数]

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

[戻り値]

なし

[使用例 1]

以下に、P00 端子 (N-ch オープン・ドレイン出力：なし) を

入出力モードの種類： 出力モード

初期出力値： High レベル “1” を出力

に変更する際の例を示します。

【CG_main.c】

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

[使用例 2]

以下に、P04 端子 (N-ch オープン・ドレイン出力：あり) を

入出力モードの種類： 出力モード

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

初期出力値： Low レベル “0” を出力

に変更する際の例を示します。

【CG_main.c】

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

C. 3.3 割り込み

以下に、コード生成が割り込み用として出力する API 関数の一覧を示します。

表 C—4 割り込み用 API 関数

API 関数名	機能概要
INTP_Init	外部割り込み INTP n の機能を制御するうえで必要となる初期化処理を行います。
INTP_UserInit	外部割り込み INTP n に関するユーザ独自の初期化処理を行います。
KEY_Init	キー割り込み INTKRの機能を制御するうえで必要となる初期化処理を行います。
KEY_UserInit	キー割り込み INTKRに関するユーザ独自の初期化処理を行います。
INT_MaskableInterruptEnable	マスクブル割り込みの受け付けを禁止／許可します。
INTPn_Disable	マスクブル割り込み（外部割り込み要求）INTP n の受け付けを禁止します。
INTPn_Enable	マスクブル割り込み（外部割り込み要求）INTP n の受け付けを許可します。
KEY_Disable	キー割り込み INTKRの受け付けを禁止します。
KEY_Enable	キー割り込み INTKRの受け付けを許可します。

INTP_Init

外部割り込み INTP n の機能を制御するうえで必要となる初期化処理を行います。

[所属]

CG_int.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

INTP_UserInit

外部割り込み INTP n に関するユーザ独自の初期化処理を行います。

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

[所属]

CG_int_user.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

KEY_Init

キー割り込み INTKR の機能を制御するうえで必要となる初期化処理を行います。

[所属]

CG_int.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

KEY_UserInit

キー割り込み INTKR に関するユーザ独自の初期化処理を行います。

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

[所属]

CG_int_user.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

INT_MaskableInterruptEnable

マスカブル割り込みの受け付けを禁止／許可します。

[所属]

CG_int.c

[指定形式]

```
#include "CG_macrodriver.h"
#include "CG_int.h"
MD_STATUS 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]

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

[使用例 1]

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

【CG_main.c】

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

```
*/ INT_MaskableInterruptEnable ( INT_INTP0, MD_FALSE ); /* マスカブル割り込み INTP0 の受け付け禁止
.....
}
```

[使用例 2]

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

【CG_main.c】

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


INTP n _Disable

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

[所属]

CG_int.c

[指定形式]

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

備考 n は、割り込み要因番号を意味します。

[引数]

なし

[戻り値]

なし

INTP n _Enable

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

[所属]

CG_int.c

[指定形式]

```
void    INTP $n$ _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.3.4 シリアル

以下に、コード生成がシリアル用として出力する API 関数の一覧を示します。

表 C—5 シリアル用 API 関数

API 関数名	機能概要
UART6_Init	シリアル・インタフェース (UART6) の初期化処理を行います。
UART6_UserInit	シリアル・インタフェース (UART6) に関するユーザ独自の初期化処理を行います。
UART6_Start	UART 通信を待機状態にします。
UART6_Stop	UART 通信を終了します。
UART6_SendData	データの UART 送信を開始します。
UART6_ReceiveData	データの UART 受信を開始します。
UART6_SendEndCallback	UART 送信完了割り込み INTST6 の発生に伴う処理を行います。
UART6_ReceiveEndCallback	UART 受信完了割り込み INTSR6 の発生に伴う処理を行います。
UART6_SoftOverRunCallback	UART 受信完了割り込み INTSR6 の発生に伴う処理を行います。
UART6_ErrorCallback	UART 通信におけるエラー割り込み INTSRE6 の発生に伴う処理を行います。
CSI1n_Init	シリアル・インタフェース (CSI1n) の初期化処理を行います。
CSI1n_UserInit	シリアル・インタフェース (CSI1n) に関するユーザ独自の初期化処理を行います。
CSI1n_Start	CSI1n 通信を待機状態にします。
CSI1n_Stop	CSI1n 通信を終了します。
CSI1n_ReceiveData	データの CSI1n 受信を開始します。
CSI1n_SendReceiveData	データの CSI1n 送受信を開始します。
CSI1n_SendEndCallback	CSI1n 通信完了割り込み INTCSI1n の発生に伴う処理を行います。
CSI1n_ReceiveEndCallback	CSI1n 通信完了割り込み INTCSI1n の発生に伴う処理を行います。
IICA_Init	シリアル・インタフェース (IICA) の初期化処理を行います。
IICA_UserInit	シリアル・インタフェース (IICA) に関するユーザ独自の初期化処理を行います。
IICA_Stop	IICA 通信を終了します。
IICA_MasterSendStart	IICA マスタ送信を開始します。
IICA_MasterReceiveStart	IICA マスタ受信を開始します。
IICA_StopCondition	ストップ・コンディションを発生します。
IICA_MasterSendEndCallback	IICA 通信完了割り込み INTIICA0 の発生に伴う処理を行います。
IICA_MasterReceiveEndCallback	IICA 通信完了割り込み INTIICA0 の発生に伴う処理を行います。
IICA_MasterErrorCallback	IICA マスタ通信におけるエラーの検出に伴う処理を行います。
IICA_SlaveSendStart	IICA スレーブ送信を開始します。
IICA_SlaveReceiveStart	IICA スレーブ受信を開始します。
IICA_SlaveSendEndCallback	IICA 通信完了割り込み INTIICA0 の発生に伴う処理を行います。

API 関数名	機能概要
IICA_SlaveReceiveEndCallback	IICA 通信完了割り込み INTIICA0 の発生に伴う処理を行います。
IICA_SlaveErrorCallback	IICA スレーブ通信におけるエラーの検出に伴う処理を行います。
IICA_GetStopConditionCallback	IICA スレーブ通信におけるストップ・コンディションの検出に伴う処理を行います。

UART6_Init

シリアル・インタフェース (UART6) の初期化処理を行います。

[所属]

CG_serial.c

[指定形式]

```
void    UART6_Init ( void );
```

[引数]

なし

[戻り値]

なし

UART6_UserInit

シリアル・インタフェース (UART6) 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、[UART6_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    UART6_UserInit ( void );
```

[引数]

なし

[戻り値]

なし

UART6_Start

UART 通信を待機状態にします。

[所属]

CG_serial.c

[指定形式]

```
void UART6_Start ( void );
```

[引数]

なし

[戻り値]

なし

UART6_Stop

UART 通信を終了します。

[所属]

CG_serial.c

[指定形式]

```
void    UART6_Stop ( void );
```

[引数]

なし

[戻り値]

なし

UART6_SendData

データの UART 送信を開始します。

- 備考 1.** 本 API 関数では、引数 *txbuf* で指定されたバッファから 1 バイト単位の UART 送信を引数 *txnum* で指定された回数だけ繰り返し行います。
- 2.** UART 送信を行う際には、本 API 関数の呼び出し以前に [UART6_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 UART6_ReceiveData ( UCHAR *rxbuf, USHORT rxnum );
```

[引数]

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

備考 シリアル・インタフェース (UART6) が DALI モードで動作している場合、送信するデータの総数 *txnum* として設定可能な値は、1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、4 バイトの固定長データを 1 回だけ UART 送信する際の例を示します。

【CG_main.c】

```
#include "CG_macrodriver.h"
BOOL gFlag; /* 送信完了フラグ */
void main ( void ) {
```

```
    UCHAR    txbuf[] = "ABCD";
    USHORT   txnum = 4;

    gFlag = 1;                                /* 送信完了フラグの初期化 */
    .....

    UART6_Start ();                            /* UART 通信の開始 */
    UART6_SendData ( &txbuf, txnum );         /* UART 送信の開始 */
    while ( gFlag );                            /* txnum 個の送信待ち */
    .....
}
```

【CG_serial_user.c】

```
#include    "CG_macrodriver.h"
extern  BOOL    gFlag;                        /* 送信完了フラグ */
__interrupt void MD_INTST6 ( void ) {        /* 割り込み INTST6 発生時の割り込み処理 */
    if ( gUart6TxCnt > 0 ) {
        .....
    } else {
        UART6_SendEndCallback ();           /* コールバック・ルーチンの呼び出し */
    }
}

void UART6_SendEndCallback ( void ) {        /* 割り込み INTST6 発生時のコールバック・ルーチン */
    gFlag = 0;                              /* 送信完了フラグの設定 */
}
```

UART6_ReceiveData

データの UART 受信を開始します。

- 備考 1.** 本 API 関数では、1 バイト単位の UART 受信を引数 *rxnum* で指定された回数だけ繰り返し行い、引数 *rxbuf* で指定されたバッファに格納します。
- 2.** 実際の UART 受信は、本 API 関数の呼び出し後、[UART6_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 UART6_ReceiveData ( UCHAR *rxbuf, USHORT rxnum );
```

[引数]

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

備考 シリアル・インタフェース (UART6) が DALI モードで動作している場合、受信するデータの総数 *rxnum* として設定可能な値は、2 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、4 バイトの固定長データを 1 回だけ UART 受信する際の例を示します。

【CG_main.c】

```
#include "CG_macrodriver.h"
BOOL gFlag; /* 受信完了フラグ */
void main ( void ) {
```

```

    UCHAR   rxbuf[10];
    USHORT  rxnum = 4;

    gFlag = 1;                               /* 受信完了フラグの初期化 */
    .....

    UART6_ReceiveData ( &rxbuf, rxnum );    /* UART 受信の開始 */
    UART6_Start ();                          /* UART 通信の開始 */
    while ( gFlag );                          /* rxnum 個の受信待ち */
    .....
}

```

【CG_serial_user.c】

```

#include "CG_macrodriver.h"

extern BOOL   gFlag;                          /* 受信完了フラグ */
__interrupt void MD_INTSR6 ( void ) {        /* 割り込み INTSR6 発生時の割り込み処理 */
    .....
    if ( gUart6RxLen > gUart6RxCnt ) {
        .....
        if ( gUart6RxLen == gUart6RxCnt ) {
            UART6_ReceiveEndCallback ();    /* コールバック・ルーチンの呼び出し */
        }
    }
}

void UART6_ReceiveEndCallback ( void ) {     /* 割り込み INTSR6 発生時のコールバック・ルーチン */
    gFlag = 0;                               /* 受信完了フラグの設定 */
}

```

UART6_SendEndCallback

UART 送信完了割り込み INTST6 の発生に伴う処理を行います。

備考 本 API 関数は、UART 送信完了割り込み INTST6 に対応した割り込み処理 MD_INTST6 のコールバック・ルーチン（[UART6_SendData](#) の引数 *txnum* で指定された数のデータ送信が完了した際の処理）として呼び出されます。

[所属]

CG_serial_user.c

[指定形式]

```
void UART6_SendEndCallback ( void );
```

[引数]

なし

[戻り値]

なし

UART6_ReceiveEndCallback

UART 受信完了割り込み INTSR6 の発生に伴う処理を行います。

備考 本 API 関数は、UART 受信完了割り込み INTSR6 に対応した割り込み処理 MD_INTSR6 のコールバック・ルーチン（UART6_ReceiveData の引数 *rxnum* で指定された数のデータ受信が完了した際の処理）として呼び出されます。

[所属]

CG_serial_user.c

[指定形式]

```
void UART6_ReceiveEndCallback ( void );
```

[引数]

なし

[戻り値]

なし

UART6_SoftOverRunCallback

UART 受信完了割り込み INTSR6 の発生に伴う処理を行います。

備考 本 API 関数は、UART 受信完了割り込み INTSR6 に対応した割り込み処理 MD_INTSR6 のコールバック・ルーチン（[UART6_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]

- [Ix2]

```
void UART6_SoftOverRunCallback ( USHORT rx_data );
```

- [Kx2-L]

```
void UART6_SoftOverRunCallback ( UCHAR rx_data );
```

[引数]

I/O	引数	説明
O	USHORT <i>rx_data</i> ;	受信したデータ（ UART6_ReceiveData の引数 <i>rxnum</i> で指定された数以上に受信したデータ）
O	UCHAR <i>rx_data</i> ;	受信したデータ（ UART6_ReceiveData の引数 <i>rxnum</i> で指定された数以上に受信したデータ）

[戻り値]

なし

UART6_ErrorCallback

UART 通信におけるエラー割り込み INTSRE6 の発生に伴う処理を行います。

備考 本 API 関数は、エラー割り込み INTSRE6 に対応した割り込み処理 MD_INTSRE6 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 UART6_ErrorCallback ( UCHAR err_type );
```

[引数]

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

[戻り値]

なし

[使用例]

以下に、エラー割り込みの発生要因別にコールバック処理を行う際の例を示します。

【CG_serial_user.c】

```
#include "CG_macrodriver.h"
__interrupt void MD_INTSRE6 ( void ) { /* 割り込み INTSRE6 発生時の割り込み処理 */
    UCHAR err_type;
    .....
    UART6_ErrorCallback ( err_type ); /* コールバック・ルーチンの呼び出し */
}

void UART6_ErrorCallback ( UCHAR err_type ) { /* 割り込み INTSRE6 発生時のコールバック・ルーチン */
    if ( err_type & 0x1 ) { /* 発生要因の判別 */
```

```
..... /* オーバーラン・エラーが発生した際のコールバック処理 */  
} else if ( err_type & 0x2 ) {          /* 発生要因の判別 */  
..... /* フレーミング・エラーが発生した際のコールバック処理 */  
} else if ( err_type & 0x4 ) {          /* 発生要因の判別 */  
..... /* パリティ・エラーが発生した際のコールバック処理 */  
}  
}
```

CSI1*n*_Init

シリアル・インタフェース (CSI1*n*) の初期化処理を行います。

[所属]

CG_serial.c

[指定形式]

```
void CSI1n_Init ( void );
```

備考 *n*は、チャンネル番号を意味します。

[引数]

なし

[戻り値]

なし

CSI1*n*_UserInit

シリアル・インタフェース（CSI1*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、[CSI1*n*_Init](#)のコールバック・ルーチンとして呼び出されます。

[所属]

CG_serial_user.c

[指定形式]

```
void CSI1n_UserInit ( void );
```

備考 *n*は、チャンネル番号を意味します。

[引数]

なし

[戻り値]

なし

CSI1*n*_Start

CSI1*n* 通信を待機状態にします。

[所属]

CG_serial.c

[指定形式]

```
void CSI1n_Start ( void );
```

備考 *n*は、チャンネル番号を意味します。

[引数]

なし

[戻り値]

なし

CSI1*n*_Stop

CSI1*n*通信を終了します。

[所属]

CG_serial.c

[指定形式]

```
void CSI1n_Stop ( void );
```

備考 *n*は、チャンネル番号を意味します。

[引数]

なし

[戻り値]

なし

CSI1n_ReceiveData

データのCSI1n受信を開始します。

- 備考 1.** 本API関数では、1バイト単位のCSI1n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.** CSI1n 受信を行う際には、本API関数の呼び出し以前に [CSI1n_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 CSI1n_ReceiveData ( UCHAR *rxbuf, USHORT rxnum );
```

備考 *n* は、チャンネル番号を意味します。

[引数]

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

[戻り値]

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

[使用例]

以下に、チャンネル 10 から 4 バイトの固定長データを 1 回だけ CSI10 受信する際の例を示します。

【CG_main.c】

```
#include "CG_macrodriver.h"
BOOL gFlag; /* 受信完了フラグ */
void main ( void ) {
    UCHAR rxbuf[10];
```



```

USHORT rxnum = 4;

gFlag = 1; /* 受信完了フラグの初期化 */
.....

CSI10_Start (); /* CSI10 通信の開始 */
CSI10_ReceiveData ( &rxbuf, rxnum ); /* CSI10 受信の開始 */
while ( gFlag ); /* rxnum 個の受信待ち */
.....
}

```

【CG_serial_user.c】

```

#include "CG_macrodriver.h"

extern BOOL gFlag; /* 受信完了フラグ */

__interrupt void MD_INTCSI10 ( void ) { /* 割り込み INTCSI10 発生時の割り込み処理 */
    if ( gCsi10RxCnt < gCsi10RxLen ) {
        .....
        if ( gCsi10RxCnt == gCsi10RxLen ) {
            CSI10_ReceiveEndCallback (); /* コールバック・ルーチンの呼び出し */

        } else {
            .....
        }
    }
}

void CSI10_ReceiveEndCallback ( void ) { /* 割り込み INTCSI10 発生時のコールバック・ルーチン */
    gFlag = 0; /* 受信完了フラグの設定 */
}

```

CSI1n_SendReceiveData

データのCSI1n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CSI1n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CSI1n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.** CSI1n送受信を行う際には、本API関数の呼び出し以前に [CSI1n_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 CSI1n_SendReceiveData ( UCHAR *txbuf, USHORT txnum, UCHAR *rxbuf );
```

備考 *n*は、チャンネル番号を意味します。

[引数]

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

[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、チャンネル10から4バイトの固定長データを1回だけCSI10送受信する際の例を示します。

【CG_main.c】

```

#include    "CG_macrodriver.h"

BOOL      gSflag;                                /* 送信完了フラグ */
void main ( void ) {
    UCHAR   txbuf[] = "0123";
    USHORT  txnum = 4;
    UCHAR   rxbuf[10];

    gSflag = 1;                                    /* 送信完了フラグの初期化 */
    .....

    CSI10_Start ();                               /* CSI10 通信の開始 */
    CSI10_SendReceiveData ( &txbuf, txnum, &rxbuf ); /* CSI10 送受信の開始 */
    while ( gSflag );                             /* txnum 個の送受信待ち */
    .....
}

```

【CG_serial_user.c】

```

#include    "CG_macrodriver.h"

extern BOOL  gSflag;                              /* 送信完了フラグ */
__interrupt void MD_INTCSI10 ( void ) {          /* 割り込み INTCSI10 発生時の割り込み処理 */
    if ( gCsi10TxCnt > 0 ) {
        .....
    } else {
        .....
        CSI10_SendEndCallback ();                /* コールバック・ルーチンの呼び出し */
    }
}

void CSI10_SendEndCallback ( void ) {           /* 割り込み INTCSI10 発生時のコールバック・ルーチン */
    gSflag = 0;                                  /* 送信完了フラグの設定 */
}

```

CSI1n_SendEndCallback

CSI1n 通信完了割り込み INTCSI1n の発生に伴う処理を行います。

備考 本 API 関数は、CSI1n 通信完了割り込み INTCSI1n に対応した割り込み処理 MD_INTCSI1n のコールバック・ルーチン（CSI1n_SendReceiveData の引数 *txnum* で指定された数のデータ送信が完了した際の処理）として呼び出されます。

[所属]

CG_serial_user.c

[指定形式]

```
void CSI1n_SendEndCallback ( void );
```

備考 *n* は、チャンネル番号を意味します。

[引数]

なし

[戻り値]

なし

CSI1n_ReceiveEndCallback

CSI1n 通信完了割り込み INTCSI1n の発生に伴う処理を行います。

備考 本 API 関数は、CSI1n 通信完了割り込み INTCSI1n に対応した割り込み処理 MD_INTCSI1n のコールバック・ルーチン（CSI1n_ReceiveData の引数 rxnum で指定された数のデータ受信が完了した際の処理）として呼び出されます。

[所属]

CG_serial_user.c

[指定形式]

```
void CSI1n_ReceiveEndCallback ( void );
```

備考 n は、チャンネル番号を意味します。

[引数]

なし

[戻り値]

なし

IICA_Init

シリアル・インタフェース（IICA）の初期化処理を行います。

[所属]

CG_serial.c

[指定形式]

```
void IICA_Init ( void );
```

[引数]

なし

[戻り値]

なし

IICA_UserInit

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

備考 本API関数は、[IICA_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 IICA_UserInit ( void );
```

[引数]

なし

[戻り値]

なし

IICA_Stop

IICA 通信を終了します。

[所属]

CG_serial.c

[指定形式]

```
void IICA_Stop ( void );
```

[引数]

なし

[戻り値]

なし

IICA_MasterSendStart

IICA マスタ送信を開始します。

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

[所属]

CG_serial.c

[指定形式]

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

[引数]

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

[戻り値]

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

IICA_MasterReceiveStart

IICA マスタ受信を開始します。

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

[所属]

CG_serial.c

[指定形式]

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

[引数]

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

[戻り値]

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

IICA_StopCondition

ストップ・コンディションを発生します。

[所属]

CG_serial.c

[指定形式]

```
void IICA_StopCondition ( void );
```

[引数]

なし

[戻り値]

なし

IICA_MasterSendEndCallback

IICA 通信完了割り込み INTIICA0 の発生に伴う処理を行います。

備考 本 API 関数は、IICA 通信完了割り込み INTIICA0 に対応した割り込み処理 MD_INTIICA0 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 IICA_MasterSendEndCallback ( void );
```

[引数]

なし

[戻り値]

なし

IICA_MasterReceiveEndCallback

IICA 通信完了割り込み INTIICA0 の発生に伴う処理を行います。

備考 本 API 関数は、IICA 通信完了割り込み INTIICA0 に対応した割り込み処理 MD_INTIICA0 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 IICA_MasterReceiveEndCallback ( void );
```

[引数]

なし

[戻り値]

なし

IICA_MasterErrorCallback

IICA マスタ通信におけるエラーの検出に伴う処理を行います。

[所属]

CG_serial_user.c

[指定形式]

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

[引数]

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

[戻り値]

なし

IICA_SlaveSendStart

IICA スレーブ送信を開始します。

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

[所属]

CG_serial.c

[指定形式]

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

[引数]

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

[戻り値]

なし

IICA_SlaveReceiveStart

IICA スレーブ受信を開始します。

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

[所属]

CG_serial.c

[指定形式]

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

[引数]

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

[戻り値]

なし

IICA_SlaveSendEndCallback

IICA 通信完了割り込み INTIICA0 の発生に伴う処理を行います。

備考 本 API 関数は、IICA 通信完了割り込み INTIICA0 に対応した割り込み処理 MD_INTIICA0 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 IICA_SlaveSendEndCallback ( void );
```

[引数]

なし

[戻り値]

なし

IICA_SlaveReceiveEndCallback

IICA 通信完了割り込み INTIICA0 の発生に伴う処理を行います。

備考 本 API 関数は、IICA 通信完了割り込み INTIICA0 に対応した割り込み処理 MD_INTIICA0 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 IICA_SlaveReceiveEndCallback ( void );
```

[引数]

なし

[戻り値]

なし

IICA_SlaveErrorCallback

IICA スレーブ通信におけるエラーの検出に伴う処理を行います。

[所属]

CG_serial_user.c

[指定形式]

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

[引数]

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

[戻り値]

なし

IICA_GetStopConditionCallback

IICA スレーブ通信におけるストップ・コンディションの検出に伴う処理を行います。

[所属]

CG_serial_user.c

[指定形式]

```
void IICA_GetStopConditionCallback ( void );
```

[引数]

なし

[戻り値]

なし

C.3.5 オペアンプ

以下に、コード生成がオペアンプ用として出力するAPI関数の一覧を示します。

表 C—6 オペアンプ用 API 関数

API 関数名	機能概要
OPAMP_Init	オペアンプの機能を制御するうえで必要となる初期化処理を行います。
OPAMP_UserInit	オペアンプに関するユーザ独自の初期化処理を行います。
PGA_Start	オペアンプ (PGA モード) の動作を開始します。
PGA_Stop	オペアンプ (PGA モード) の動作を停止します。
PGA_ChangePGAFactor	オペアンプ (PGA モード) における入力電圧の増幅率を設定します。
AMP_Start	オペアンプ (シングル・アンプ・モード) の動作を開始します。
AMP_Stop	オペアンプ (シングル・アンプ・モード) の動作を停止します。
AMPn_Start	オペアンプ n (シングル・アンプ・モード) の動作を開始します。
AMPn_Stop	オペアンプ n (シングル・アンプ・モード) の動作を停止します。

OPAMP_Init

オペアンプの機能を制御するうえで必要となる初期化処理を行います。

[所属]

CG_opamp.c

[指定形式]

```
void OPAMP_Init ( void );
```

[引数]

なし

[戻り値]

なし

OPAMP_UserInit

オペアンプに関するユーザ独自の初期化処理を行います。

備考 本 API 関数は、OPAMP_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_opamp_user.c

[指定形式]

```
void OPAMP_UserInit ( void );
```

[引数]

なし

[戻り値]

なし

PGA_Start

オペアンプ (PGA モード) の動作を開始します。

[所属]

CG_opamp.c

[指定形式]

```
void PGA_Start ( void );
```

[引数]

なし

[戻り値]

なし

PGA_Stop

オペアンプ（PGA モード）の動作を停止します。

[所属]

CG_opamp.c

[指定形式]

```
void PGA_Stop ( void );
```

[引数]

なし

[戻り値]

なし

PGA_ChangePGAFactor

オペアンプ（PGA モード）における入力電圧の増幅率を設定します。

備考 引数 *factor* に指定された値は、オペアンプ制御レジスタ（AMP0M）に設定されます。

[所属]

CG_opamp.c

[指定形式]

```
#include "CG_macrodriver.h"
#include "CG_opamp.h"
MD_STATUS PGA_ChangePGAFactor ( enum PGAFactor factor );
```

[引数]

I/O	引数	説明
I	enum PGAFactor <i>factor</i> ;	入力電圧の増幅率 PGAFACTOR0 : 4 倍 PGAFACTOR1 : 8 倍 PGAFACTOR2 : 16 倍 PGAFACTOR3 : 32 倍

[戻り値]

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

AMP_Start

オペアンプ（シングル・アンプ・モード）の動作を開始します。

[所属]

CG_opamp.c

[指定形式]

```
void AMP_Start ( void );
```

[引数]

なし

[戻り値]

なし

AMP_Stop

オペアンプ（シングル・アンプ・モード）の動作を停止します。

[所属]

CG_opamp.c

[指定形式]

```
void AMP_Stop ( void );
```

[引数]

なし

[戻り値]

なし

AMP n _Start

オペアンプ n (シングル・アンプ・モード) の動作を開始します。

[所属]

CG_opamp.c

[指定形式]

```
void AMPn_Start ( void );
```

備考 n は、チャンネル番号を意味します。

[引数]

なし

[戻り値]

なし

AMP n _Stop

オペアンプ n (シングル・アンプ・モード) の動作を停止します。

[所属]

CG_opamp.c

[指定形式]

```
void AMPn_Stop ( void );
```

備考 n は、チャンネル番号を意味します。

[引数]

なし

[戻り値]

なし

C.3.6 コンパレータ

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

表 C-7 コンパレータ用 API 関数

API 関数名	機能概要
Comparator_Init	コンパレータの機能を制御するうえで必要となる初期化処理を行います。
Comparator_UserInit	コンパレータに関するユーザ独自の初期化処理を行います。
Comparatorn_Start	コンパレータ <i>n</i> の動作を開始します。
Comparatorn_Stop	コンパレータ <i>n</i> の動作を停止します。

Comparator_Init

コンパレータの機能を制御するうえで必要となる初期化処理を行います。

[所属]

CG_comparator.c

[指定形式]

```
void Comparator_Init ( void );
```

[引数]

なし

[戻り値]

なし

Comparator_UserInit

コンパレータに関するユーザ独自の初期化処理を行います。

備考 本API関数は、[Comparator_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_comparator_user.c

[指定形式]

```
void Comparator_UserInit ( void );
```

[引数]

なし

[戻り値]

なし

Comparator n _Start

コンパレータ n の動作を開始します。

[所属]

CG_comparator.c

[指定形式]

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

備考 n は、チャンネル番号を意味します。

[引数]

なし

[戻り値]

なし

Comparator n _Stop

コンパレータ n の動作を停止します。

[所属]

CG_comparator.c

[指定形式]

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

備考 n は、チャンネル番号を意味します。

[引数]

なし

[戻り値]

なし

C. 3.7 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—8 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_ComparatorOn	電圧コンパレータを動作許可状態に設定します。
AD_ComparatorOff	電圧コンパレータを動作停止状態に設定します。
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_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_ComparatorOn

電圧コンパレータを動作許可状態に設定します。

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

[所属]

CG_ad.c

[指定形式]

```
void AD_ComparatorOn ( void );
```

[引数]

なし

[戻り値]

なし

AD_ComparatorOff

電圧コンパレータを動作停止状態に設定します。

[所属]

CG_ad.c

[指定形式]

```
void AD_ComparatorOff ( void );
```

[引数]

なし

[戻り値]

なし

AD_Start

A/D 変換を開始します。

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

[所属]

CG_ad.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

AD_Stop

A/D 変換を終了します。

備考 電圧コンパレータは、本 API 関数の処理完了後も動作を継続しています。

したがって、電圧コンパレータの動作を停止する場合は、本 API 関数の処理完了後、[AD_ComparatorOff](#) を呼び出す必要があります。

[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。

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

[所属]

CG_ad.c

[指定形式]

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

[引数]

I/O	引数	説明
I	enum ADChannel <i>channel</i> ;	アナログ電圧の入力端子 ADCHANNEL n : 入力端子 ADCHANNELPGAIN : オペアンプ出力端子【1x2】 ADCHANNEL12V : 内部電圧 (1.2 V)【1x2】

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_Read

A/D 変換結果（10 ビット）を読み出します。

備考 10 ビット A/D 変換結果レジスタ（ADCR）の内容が引数 *buffer* で指定された領域に格納されます。

[所属]

CG_ad.c

[指定形式]

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

[引数]

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

[戻り値]

なし

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。

備考 8 ビット A/D 変換結果レジスタ H (ADCRH) の内容が引数 *buffer* で指定された領域に格納されます。

[所属]

CG_ad.c

[指定形式]

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

[引数]

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

[戻り値]

なし

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
TMX_Init	16 ビット・タイマ Xn の機能を制御するうえで必要となる初期化処理を行います。
TMXn_Start	16 ビット・タイマ Xn のカウントを開始します。
TMXn_Stop	16 ビット・タイマ Xn のカウントを終了します。
TMXn_ChangeDuty	TOX0n 端子にシングル出力する PWM 信号のデューティ比を変更します。
TMXn_ChangeDualDuty	TOX0n 端子にデュアル出力する PWM 信号のデューティ比を変更します。
TMX_EnableHighImpedanceState	16 ビット・タイマ Xn のハイ・インピーダンス出力を開始します。
TMX_DisableHighImpedanceState	16 ビット・タイマ Xn のハイ・インピーダンス出力を終了します。
TM00_Init	16 ビット・タイマ/イベント・カウンタ 00 の機能を制御するうえで必要となる初期化処理を行います。
TM00_UserInit	16 ビット・タイマ/イベント・カウンタ 00 に関するユーザ独自の初期化処理を行います。
TM00_Start	16 ビット・タイマ/イベント・カウンタ 00 のカウントを開始します。
TM00_Stop	16 ビット・タイマ/イベント・カウンタ 00 のカウントを終了します。
TM00_ChangeTimerCondition	キャプチャ/コンペア・コントロール・レジスタ 00 (CRC00) の内容を変更します。
TM00_GetFreeRunningValue	キャプチャ・レジスタ (CR0n0) の内容を獲得します。
TM00_SoftwareTriggerOn	ワンショット・パルス出力のためのトリガ (ソフトウェア・トリガ) を発生させます。
TM00_ChangeDuty	TO00 端子に出力する信号のデューティ比を変更します。
TM00_GetPulseWidth	TI0n0 端子に対する入力信号 (入力パルス) のハイ/ロウ・レベルの測定幅を獲得します。
TM5n_Init	8 ビット・タイマ/イベント・カウンタ 5n の機能を制御するうえで必要となる初期化処理を行います。
TM5n_UserInit	8 ビット・タイマ/イベント・カウンタ 5n に関するユーザ独自の初期化処理を行います。
TM5n_Start	8 ビット・タイマ/イベント・カウンタ 5n のカウントを開始します。
TM5n_Stop	8 ビット・タイマ/イベント・カウンタ 5n のカウントを終了します。
TM5n_ChangeTimerCondition	8 ビット・タイマ・コンペア・レジスタ 5n (CR5n) の内容を変更します。
TM5n_ChangeDuty	TO5n 端子に出力する PWM 信号のデューティ比を変更します。
TMHn_Init	8 ビット・タイマ Hn の機能を制御するうえで必要となる初期化処理を行います。
TMHn_UserInit	8 ビット・タイマ Hn に関するユーザ独自の初期化処理を行います。
TMHn_Start	8 ビット・タイマ Hn のカウントを開始します。

API 関数名	機能概要
TMHn_Stop	8 ビット・タイマ Hn のカウントを終了します。
TMHn_ChangeTimerCondition	8 ビット・タイマ H コンペア・レジスタ 0n / 1n (CMP0n / CMP1n) の内容を変更します。
TMHn_ChangeDuty	TOHn 端子に出力する PWM 信号のデューティ比を変更します。
TMH1_CarrierOutputEnable	8 ビット・タイマ H1 (キャリア・ジェネレータ・モード) のキャリア・パルス出力を開始します。
TMH1_CarrierOutputDisable	8 ビット・タイマ H1 (キャリア・ジェネレータ・モード) のキャリア・パルス出力を終了します。

TMX_Init

16ビット・タイマXnの機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    TMX_Init ( void );
```

[引数]

なし

[戻り値]

なし

TMX n _Start

16 ビット・タイマ X n のカウントを開始します。

備考 本 API 関数を呼び出してからカウント処理を開始するまでの時間は、該当機能の種類（PWM 出力、A/D 変換スタート・タイミング信号出力など）により異なります。

[所属]

CG_timer.c

[指定形式]

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

備考 n は、チャンネル番号を意味します。

[引数]

なし

[戻り値]

なし

TMX n _Stop

16 ビット・タイマ X n のカウントを終了します。

[所属]

CG_timer.c

[指定形式]

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

備考 n は、チャンネル番号を意味します。

[引数]

なし

[戻り値]

なし

TMX n _ChangeDuty

TOX n 端子にシングル出力する 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 X n をシングル出力用に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 TMXn_ChangeDuty ( UCHAR ratio );
```

備考 n は、チャネル番号を意味します。

[引数]

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

備考 デューティ比 *ratio* 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 ) {
    UCHAR ratio = 25;
    .....
    TMX0_Start (); /* カウントの開始 */
    .....
    TMX0_ChangeDuty ( ratio ); /* デューティ比の変更 */
    .....
}
```

```
}  
}
```

TMXn_ChangeDualDuty

TOXn端子にデュアル出力する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Xnをデュアル出力用に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 TMXn_ChangeDualDuty ( UCHAR ratio, UCHAR ratio1, UCHAR delay );
```

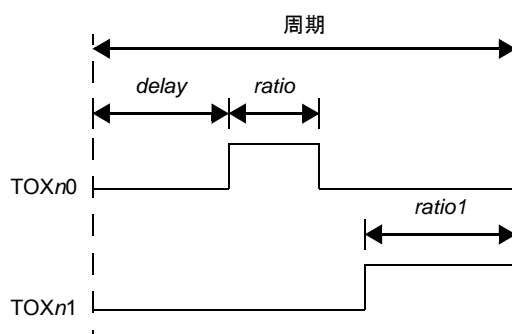
備考 nは、チャンネル番号を意味します。

[引数]

I/O	引数	説明
I	UCHAR ratio;	TOXn0のデューティ比 (0 ~ 100, 単位: %)
I	UCHAR ratio1;	TOXn1のデューティ比 (0 ~ 100, 単位: %)
I	UCHAR delay;	TOXn0の遅延間隔 (0 ~ 100, 単位: %)

備考 1. デューティ比 ratio, ratio1, delayに設定する値は、10進数に限られます。

2. 各引数は、以下の意味を持ちます。



[戻り値]

なし

TMX_EnableHighImpedanceState

16 ビット・タイマ Xn のハイ・インピーダンス出力を開始します。

[所属]

CG_timer.c

[指定形式]

```
void    TMX_EnableHighImpedanceState ( void );
```

[引数]

なし

[戻り値]

なし

TMX_DisableHighImpedanceState

16ビット・タイマXnのハイ・インピーダンス出力を終了します。

[所属]

CG_timer.c

[指定形式]

```
void    TMX_DisableHighImpedanceState ( void );
```

[引数]

なし

[戻り値]

なし

TM00_Init

16ビット・タイマ/イベント・カウンタ 00 の機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    TM00_Init ( void );
```

[引数]

なし

[戻り値]

なし

TM00_UserInit

16 ビット・タイマ/イベント・カウンタ 00 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、[TM00_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    TM00_UserInit ( void );
```

[引数]

なし

[戻り値]

なし

TM00_Start

16ビット・タイマ/イベント・カウンタ 00 のカウントを開始します。

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

[所属]

CG_timer.c

[指定形式]

```
void    TM00_Start ( void );
```

[引数]

なし

[戻り値]

なし

TM00_Stop

16ビット・タイマ/イベント・カウンタ 00 のカウントを終了します。

[所属]

CG_timer.c

[指定形式]

```
void    TM00_Stop ( void );
```

[引数]

なし

[戻り値]

なし

TM00_ChangeTimerCondition

16ビット・タイマ・キャプチャ／コンペア・コントロール・レジスタ 0n0 (CR0n0) の内容を変更します。

備考 CR0n0 の内容を変更する際には、本 API 関数の呼び出し以前に **TM00_Stop** を呼び出す必要があります。

[所属]

CG_timer.c

[指定形式]

```
#include "CG_macrodriver.h"
#include "CG_timer.h"
MD_STATUS TM00_ChangeTimerCondition ( USHORT *array_reg, USHORT array_num );
```

[引数]

I/O	引数	説明
I	USHORT *array_reg;	変更対象レジスタに設定する値を格納した領域へのポインタ
I	USHORT array_num;	変更対象レジスタ 1: CR000 2: CR000, CR010

[戻り値]

マクロ	説明
MD_OK	正常終了
MD_ERROR	引数 array_num の指定が不正

TM00_GetFreeRunningValue

キャプチャ・レジスタ (CR0n0) の内容を獲得します。

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00がフリー・ランニング・タイマ・モードで動作し、16ビット・タイマ・キャプチャ/コンペア・コントロール・レジスタ0n0 (CRC0n0) がキャプチャ・レジスタとして使用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 TM00_GetFreeRunningValue ( ULONG *count, enum TMChannel channel );
```

[引数]

I/O	引数	説明
O	ULONG *count;	獲得した値を格納する領域へのポインタ
I	enum TMChannel channel;	獲得対象端子 TMCHANNEL0 : TI000 端子 TMCHANNEL1 : TI010 端子

[戻り値]

マクロ	説明
MD_OK	正常終了
MD_ERROR	異常終了 - CR0n0 がコンペア・レジスタとして動作しています。
MD_ARGERROR	引数の指定が不正

TM00_SoftwareTriggerOn

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

備考 本API関数の呼び出しは、16ビット・タイマ/イベント・カウンタ00をワンショット・パルス出力用に使用している場合に限られます。

[所属]

CG_timer.c

[指定形式]

```
void    TM00_SoftwareTriggerOn ( void );
```

[引数]

なし

[戻り値]

なし

TM00_ChangeDuty

TO00 端子に出力する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 00 を PPG 出力用に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 TM00_ChangeDuty ( UCHAR ratio );
```

[引数]

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

備考 デューティ比 *ratio* 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 ) {
    UCHAR ratio = 25;
    .....
    TM00_Start (); /* カウントの開始 */
    .....
    TM00_ChangeDuty ( ratio ); /* デューティ比の変更 */
    .....
}
```

TM00_GetPulseWidth

TI0n0 端子に対する入力信号（入力パルス）のハイ／ロウ・レベルの測定幅を獲得します。

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 00 をパルス幅測定用に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"
void TM00_GetPulseWidth ( ULONG *highwidth, ULONG *lowwidth, enum TMChannel channel );
```

[引数]

I/O	引数	説明
O	ULONG *highwidth;	ハイ・レベルの測定幅（0x0 ~ 0xffff）を格納する領域へのポインタ
O	ULONG *lowwidth;	ロウ・レベルの測定幅（0x0 ~ 0xffff）を格納する領域へのポインタ
I	enum TMChannel channel;	測定対象端子 TMCHANNEL0 : TI000 端子 TMCHANNEL1 : TI010 端子

[戻り値]

なし

TM5n_Init

8ビット・タイマ/イベント・カウンタ 5nの機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    TM5n_Init ( void );
```

備考 nは、チャンネル番号を意味します。

[引数]

なし

[戻り値]

なし

TM5n_UserInit

8ビット・タイマ/イベント・カウンタ 5n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、TM5n_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    TM5n_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

TM5n_Start

8ビット・タイマ/イベント・カウンタ 5n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・カウンタなど）により異なります。

[所属]

CG_timer.c

[指定形式]

```
void    TM5n_Start ( void );
```

備考 nは、チャンネル番号を意味します。

[引数]

なし

[戻り値]

なし

TM5n_Stop

8ビット・タイマ/イベント・カウンタ 5nのカウントを終了します。

[所属]

CG_timer.c

[指定形式]

```
void    TM5n_Stop ( void );
```

備考 nは、チャンネル番号を意味します。

[引数]

なし

[戻り値]

なし

TM5n_ChangeTimerCondition

8ビット・タイマ・コンペア・レジスタ 5n (CR5n) の内容を変更します。

備考 CR5nの内容を変更する際には、本API関数の呼び出し以前に **TM5n_Stop** を呼び出す必要があります。

[所属]

CG_timer.c

[指定形式]

```
#include "CG_macrodriver.h"
void TM5n_ChangeTimerCondition ( UCHAR value );
```

備考 nは、チャンネル番号を意味します。

[引数]

I/O	引数	説明
I	UCHAR value;	CR5nに設定する値

[戻り値]

なし

TM5n_ChangeDuty

TO5n 端子に出力する 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、8 ビット・タイマ/イベント・カウンタ 5n を PWM 出力用に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 TM5n_ChangeDuty ( UCHAR ratio );
```

備考 n は、チャンネル番号を意味します。

[引数]

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

備考 デューティ比 *ratio* 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 ) {
    UCHAR ratio = 25;
    .....
    TM50_Start (); /* カウントの開始 */
    .....
    TM50_ChangeDuty ( ratio ); /* デューティ比の変更 */
}
```

```
.....  
}
```

TMHn_Init

8ビット・タイマHnの機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    TMHn_Init ( void );
```

備考 nは、チャンネル番号を意味します。

[引数]

なし

[戻り値]

なし

TMHn_UserInit

8ビット・タイマ Hn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、TMHn_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    TMHn_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

TMHn_Start

8ビット・タイマHnのカウントを開始します。

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

[所属]

CG_timer.c

[指定形式]

```
void    TMHn_Start ( void );
```

備考 nは、チャンネル番号を意味します。

[引数]

なし

[戻り値]

なし

TMHn_Stop

8ビット・タイマHnのカウントを終了します。

[所属]

CG_timer.c

[指定形式]

```
void    TMHn_Stop ( void );
```

備考 nは、チャンネル番号を意味します。

[引数]

なし

[戻り値]

なし

TMHn_ChangeTimerCondition

8ビット・タイマHコンペア・レジスタ 0n / 1n (CMP0n / CMP1n) の内容を変更します。

備考 CMP0n / CMP1nの内容を変更する際には、本API関数の呼び出し以前に **TMHn_Stop** を呼び出す必要があります。

[所属]

CG_timer.c

[指定形式]

```
#include "CG_macrodriver.h"
#include "CG_timer.h"
MD_STATUS TMHn_ChangeTimerCondition ( UCHAR *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	UCHAR *array_reg;	変更対象レジスタに設定する値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: CMP0n 2: CMP0n, CMP1n

[戻り値]

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

TMHn_ChangeDuty

TOHn 端子に出力する 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、8 ビット・タイマ Hn を PWM 出力用に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 TMHn_ChangeDuty ( UCHAR ratio );
```

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 ratio;	デューティ比 (0 ~ 100, 単位: %)

備考 デューティ比 ratio 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 ) {
    UCHAR ratio = 25;
    .....
    TMH0_Start (); /* カウントの開始 */
    .....
    TMH0_ChangeDuty ( ratio ); /* デューティ比の変更 */
    .....
}
```

```
}  
}
```

TMH1_CarrierOutputEnable

8ビット・タイマH1（キャリア・ジェネレータ・モード）のキャリア・パルス出力を開始します。

[所属]

CG_timer.c

[指定形式]

```
void    TMH1_CarrierOutputEnable ( void );
```

[引数]

なし

[戻り値]

なし

TMH1_CarrierOutputDisable

8ビット・タイマH1（キャリア・ジェネレータ・モード）のキャリア・パルス出力を終了します。

[所属]

CG_timer.c

[指定形式]

```
void    TMH1_CarrierOutputDisable ( void );
```

[引数]

なし

[戻り値]

なし

C.3.9 ウォッチドッグ・タイマ

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

表 C—10 ウォッチドッグ・タイマ用 API 関数

API 関数名	機能概要
WDT_Restart	ウォッチドッグ・タイマのカウンタをクリアしたのち、カウント処理を再開します。

WDT_Restart

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

[所属]

CG_wdt.c

[指定形式]

```
void WDT_Restart ( void );
```

[引数]

なし

[戻り値]

なし

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_PowerOff	リアルタイム・カウンタに対するクロック供給を停止します。
RTC_CounterEnable	リアルタイム・カウンタ（年、月、曜日、日、時、分、秒）のカウンタを開始します。
RTC_CounterDisable	リアルタイム・カウンタ（年、月、曜日、日、時、分、秒）のカウンタを終了します。
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 INTRTC の発生周期を設定したのち、定周期割り込み機能を開始します。
RTC_ConstPeriodInterruptDisable	定周期割り込み機能を終了します。
RTC_ConstPeriodInterruptCallback	定周期割り込み INTRTC の発生に伴う処理を行います。
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_AlarmInterruptCallback	アラーム割り込み INTRTC の発生に伴う処理を行います。
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 INTRTCI 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_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。

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

RTC_Init

リアルタイム・カウンタの機能を制御するうえで必要となる初期化処理を行います。

[所属]

CG_rtc.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

RTC_UserInit

リアルタイム・カウンタに関するユーザ独自の初期化処理を行います。

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

[所属]

CG_rtc_user.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

RTC_PowerOff

リアルタイム・カウンタに対するクロック供給を停止します。

備考 本 API 関数の呼び出しにより、リアルタイム・カウンタはリセット状態へと移行します。

このため、本 API 関数の呼び出し後、制御レジスタ（リアルタイム・カウンタ・コントロール・レジスタ 0 : RTCC0 など）への書き込みは無視されます。

[所属]

CG_rtc.c

[指定形式]

```
void RTC_PowerOff ( 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_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・カウンタのカウンタ値として、“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_rtc.h"
void main ( main ) {
    struct RTCCounterValue counterwriteval;
    .....

    RTC_CounterEnable ();          /* カウンタの開始 */
    .....

    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 ); /* カウンタ値の設定 */
    .....
}
```

RTC_CounterGet

リアルタイム・カウンタのカウンタ値を読み出します。

[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;	読み出したカウンタ値を格納する構造体へのポインタ

備考 カウンタ値 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	カウンタ処理を実行中（読み出し前）
MD_BUSY2	カウンタ処理を停止中（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・カウンタのカウンタ値を読み出す際の例を示します。

【CG_main.c】

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

[所属]

CG_rtc.c

[指定形式]

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

[引数]

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

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

【CG_main.c】

```
#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_ConstPeriodInterruptCallback

定周期割り込み INTRTC の発生に伴う処理を行います。

備考 本 API 関数は、定周期割り込み INTRTC に対応した割り込み処理 MD_INTRTC 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_ConstPeriodInterruptCallback ( 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_rtc.h"
void RTC_AlarmSet ( struct RTCArmValue alarmval );
```

[引数]

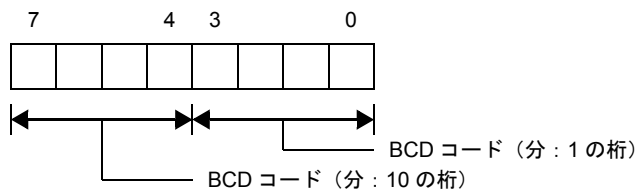
I/O	引数	説明
I	struct RTCArmValue alarmval;	アラームの発生条件（曜日，時，分）

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

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

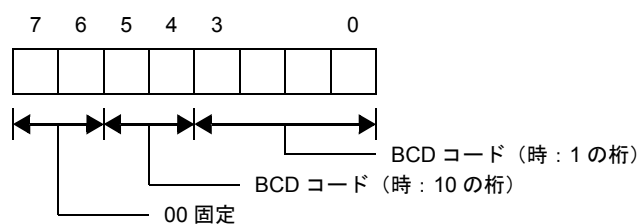


- Alarmwh（時）

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。

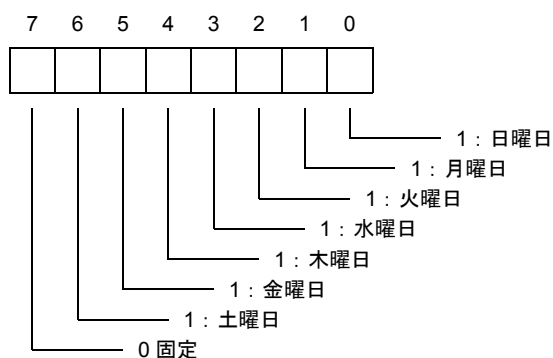
なお，ビット5は，リアルタイム・カウンタが12時間制の場合，以下の意味となります。

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

```
}
```

[使用例 2]

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

【CG_main.c】

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

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

[引数]

I/O	引数	説明
O	struct RTCArmValue *alarmval;	読み出した発生条件を格納する構造体へのポインタ

[戻り値]

なし

[使用例]

以下に、アラームの発生条件を読み出す際の例を示します。

【CG_main.c】

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

RTC_AlarmInterruptCallback

アラーム割り込み INTRTC の発生に伴う処理を行います。

備考 本 API 関数は、アラーム割り込み INTRTC に対応した割り込み処理 MD_INTRTC 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_AlarmInterruptCallback ( void );
```

[引数]

なし

[戻り値]

なし

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 INTRTCI 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 INTRTCI 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 INTRTCI 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_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 INTRTCI の発生周期 INTERVAL0 : 2 ⁶ /fSUB INTERVAL1 : 2 ⁷ /fSUB INTERVAL2 : 2 ⁸ /fSUB INTERVAL3 : 2 ⁹ /fSUB INTERVAL4 : 2 ¹⁰ /fSUB INTERVAL5 : 2 ¹¹ /fSUB INTERVAL6 : 2 ¹² /fSUB

備考 fSUB 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_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_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 RTC_RTC1HZ_OutputEnable ( void );
```

[引数]

なし

[戻り値]

なし

RTC_RTC1HZ_OutputDisable

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

[所属]

CG_rtc.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

RTC_RTCCL_OutputEnable

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

[所属]

CG_rtc.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

RTC_RTCCL_OutputDisable

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

[所属]

CG_rtc.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

RTC_RTCDIV_OutputEnable

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

[所属]

CG_rtc.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

RTC_RTCDIV_OutputDisable

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

[所属]

CG_rtc.c

[指定形式]

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

[引数]

なし

[戻り値]

なし

RTC_ChangeCorrectionValue

時計誤差を補正するタイミング、および補正值を変更します。

[所属]

CG_rtc.c

[指定形式]

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

[引数]

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

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

[戻り値]

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

C. 3. 11 クロック出力

以下に、コード生成がクロック出力用として出力する API 関数の一覧を示します。

表 C—12 クロック出力用 API 関数

API 関数名	機能概要
PCL_Init	クロック出力制御回路の機能を制御するうえで必要となる初期化処理を行います。
PCL_UserInit	クロック出力制御回路に関するユーザ独自の初期化処理を行います。
PCL_Start	クロック出力を開始します。
PCL_Stop	クロック出力を停止します。
PCL_ChangeFreq	PCL 端子への出力クロックを変更します。

PCL_Init

クロック出力制御回路の機能を制御するうえで必要となる初期化処理を行います。

[所属]

CG_pcl.c

[指定形式]

```
void PCL_Init ( void );
```

[引数]

なし

[戻り値]

なし

PCL_UserInit

クロック出力制御回路に関するユーザ独自の初期化処理を行います。

備考 本API関数は、[PCL_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_pcl_user.c

[指定形式]

```
void PCL_UserInit ( void );
```

[引数]

なし

[戻り値]

なし

PCL_Start

クロック出力を開始します。

[所属]

CG_pcl.c

[指定形式]

```
void PCL_Start ( void );
```

[引数]

なし

[戻り値]

なし

PCL_Stop

クロック出力を停止します。

[所属]

CG_pcl.c

[指定形式]

```
void PCL_Stop ( void );
```

[引数]

なし

[戻り値]

なし

PCL_ChangeFreq

PCL 端子への出力クロックを変更します。

備考 引数 *clock* に指定された値は、クロック出力選択レジスタ（CKS）に設定されます。

[所属]

CG_pcl.c

[指定形式]

```
#include "CG_pclbuz.h"
MD_STATUS PCL_ChangeFreq ( enum PCLclock clock );
```

[引数]

I/O	引数	説明
I	enum PCLclock <i>clock</i> ;	出力クロックの種類 FPRS : fPRS FPRS2 : fPRS/2 FPRS4 : fPRS/4 FPRS8 : fPRS/8 FPRS16 : fPRS/16 FPRS32 : fPRS/2048 FPRS64 : fPRS/4096 FPRS128 : fPRS/8192 SUBCLOCK : fSUB

備考 fPRS はメイン・システム・クロックの周波数を、fSUB 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

C. 3.12 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—13 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_Stop	低電圧検出動作を停止します。
LVI_SetLVILevel	低電圧検出レベルを設定します。

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_Ivi.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]

```
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

LVI_SetLVILevel

低電圧検出レベルを設定します。

- 備考 1.** 低電圧検出レベルを変更する際には、本 API 関数の呼び出し以前に **LVI_Stop** を呼び出す必要があります。
- 2.** 引数 *level* に指定された値は、低電圧検出レベル選択レジスタ (LVIS) に設定されます。

[所属]

CG_lvi.c

[指定形式]

```
#include "CG_macrodriver.h"
#include "CG_lvi.h"
MD_STATUS LVI_SetLVILevel ( enum LVILevel level );
```

[引数]

I/O	引数	説明
I	enum LVILevel <i>level</i> ;	<p>低電圧検出対象の電圧レベル</p> <p>LVILEVEL0 : 4.22 V ± 0.1 V</p> <p>LVILEVEL1 : 4.07 V ± 0.1 V</p> <p>LVILEVEL2 : 3.92 V ± 0.1 V</p> <p>LVILEVEL3 : 3.76 V ± 0.1 V</p> <p>LVILEVEL4 : 3.61 V ± 0.1 V</p> <p>LVILEVEL5 : 3.45 V ± 0.1 V</p> <p>LVILEVEL6 : 3.30 V ± 0.1 V</p> <p>LVILEVEL7 : 3.15 V ± 0.1 V</p> <p>LVILEVEL8 : 2.99 V ± 0.1 V</p> <p>LVILEVEL9 : 2.84 V ± 0.1 V</p> <p>LVILEVEL10 : 2.68 V ± 0.1 V</p> <p>LVILEVEL11 : 2.53 V ± 0.1 V</p> <p>LVILEVEL12 : 2.38 V ± 0.1 V</p> <p>LVILEVEL13 : 2.22 V ± 0.1 V</p> <p>LVILEVEL14 : 2.07 V ± 0.1 V</p> <p>LVILEVEL15 : 1.91 V ± 0.1 V</p>

備考 LVILEVEL10 ~ LVILEVEL15 の指定は、対象デバイスが 78K0/Kx2-L の場合に限られます。

[戻り値]

マクロ	説明
MD_OK	正常終了
MD_ERROR	異常終了 - 低電圧検出対象が外部入力端子からの外部入力電圧 (EXLVI)
MD_ARGERROR	引数の指定が不正

備考 戻り値 MD_ERROR は、対象デバイスが 78K0/KB2-L, 78K0/KC2-L の場合に限られます。

付録D 索引

【A】

AD_ComparatorOff ... 192
 AD_ComparatorOn ... 191
 AD_Init ... 189
 AD_Read ... 196
 AD_ReadByte ... 197
 AD_SelectADChannel ... 195
 AD_Start ... 193
 AD_Stop ... 194
 AD_UserInit ... 190
 A/D コンバータ ... 188
 AD_ComparatorOff ... 192
 AD_ComparatorOn ... 191
 AD_Init ... 189
 AD_Read ... 196
 AD_ReadByte ... 197
 AD_SelectADChannel ... 195
 AD_Start ... 193
 AD_Stop ... 194
 AD_UserInit ... 190
 AMPn_Start ... 181
 AMPn_Stop ... 182
 AMP_Start ... 179
 AMP_Stop ... 180
 API 関数 ... 94
 A/D コンバータ ... 188
 ウォッチドッグ・タイマ ... 233
 オペアンプ ... 173
 クロック出力 ... 271
 コンパレータ ... 183
 システム ... 103
 シリアル ... 133
 タイマ ... 198
 低電圧検出回路 ... 277
 ポート ... 114
 リアルタイム・カウンタ ... 235
 割り込み ... 122

【C】

CG_ChangeClockMode ... 107
 CG_ChangeFrequency ... 109
 CG_ChangePIIMode ... 113
 CG_ReadResetSource ... 106
 CG_SelectPowerSaveMode ... 110
 CG_SelectStabTime ... 112
 CLOCK_Init ... 104
 CLOCK_UserInit ... 105
 Comparator_Init ... 184
 Comparatorm_Start ... 186
 Comparatorm_Stop ... 187
 Comparator_UserInit ... 185
 CSI1n_Init ... 148
 CSI1n_ReceiveData ... 152
 CSI1n_ReceiveEndCallback ... 157
 CSI1n_SendEndCallback ... 156
 CSI1n_SendReceiveData ... 154
 CSI1n_Start ... 150
 CSI1n_Stop ... 151
 CSI1n_UserInit ... 149

【I】

IICA_GetStopConditionCallback ... 172
 IICA_Init ... 158
 IICA_MasterErrorCallback ... 166
 IICA_MasterReceiveEndCallback ... 165
 IICA_MasterReceiveStart ... 162
 IICA_MasterSendEndCallback ... 164
 IICA_MasterSendStart ... 161
 IICA_SlaveErrorCallback ... 171
 IICA_SlaveReceiveEndCallback ... 170
 IICA_SlaveReceiveStart ... 168
 IICA_SlaveSendEndCallback ... 169
 IICA_SlaveSendStart ... 167
 IICA_Stop ... 160
 IICA_StopCondition ... 163
 IICA_UserInit ... 159

- INT_MaskableInterruptEnable ... 127
 INTP_Init ... 123
 INTPn_Disable ... 129
 INTPn_Enable ... 130
 INTP_UserInit ... 124
- [K]**
 KEY_Disable ... 131
 KEY_Enable ... 132
 KEY_Init ... 125
 KEY_UserInit ... 126
- [L]**
 LVI_Init ... 278
 LVI_InterruptModeStart ... 280
 LVI_ResetModeStart ... 281
 LVI_SetLVILevel ... 283
 LVI_Stop ... 282
 LVI_UserInit ... 279
- [O]**
 OPAMP_Init ... 174
 OPAMP_UserInit ... 175
- [P]**
 PCL_ChangeFreq ... 276
 PCL_Init ... 272
 PCL_Start ... 274
 PCL_Stop ... 275
 PCL_UserInit ... 273
 PGA_ChangePGAFactor ... 178
 PGA_Start ... 176
 PGA_Stop ... 177
 PORT_ChangePmnInput ... 117
 PORT_ChangePmnOutput ... 120
 PORT_Init ... 115
 PORT_UserInit ... 116
- [R]**
 RTC_AlarmDisable ... 253
 RTC_AlarmEnable ... 252
 RTC_AlarmGet ... 257
 RTC_AlarmInterruptCallback ... 258
 RTC_AlarmSet ... 254
 RTC_ChangeCorrectionValue ... 270
 RTC_ConstPeriodInterruptCallback ... 251
 RTC_ConstPeriodInterruptDisable ... 250
 RTC_ConstPeriodInterruptEnable ... 248
 RTC_CounterDisable ... 241
 RTC_CounterEnable ... 240
 RTC_CounterGet ... 246
 RTC_CounterSet ... 244
 RTC_Init ... 237
 RTC_IntervalInterruptDisable ... 263
 RTC_IntervalInterruptEnable ... 261
 RTC_IntervalStart ... 259
 RTC_IntervalStop ... 260
 RTC_PowerOff ... 239
 RTC_RTC1HZ_OutputDisable ... 265
 RTC_RTC1HZ_OutputEnable ... 264
 RTC_RTCCL_OutputDisable ... 267
 RTC_RTCCL_OutputEnable ... 266
 RTC_RTCDIV_OutputDisable ... 269
 RTC_RTCDIV_OutputEnable ... 268
 RTC_SetHourSystem ... 242
 RTC_UserInit ... 238
- [T]**
 TM00_ChangeDuty ... 215
 TM00_ChangeTimerCondition ... 212
 TM00_GetFreeRunningValue ... 213
 TM00_GetPulseWidth ... 216
 TM00_Init ... 208
 TM00_SoftwareTriggerOn ... 214
 TM00_Start ... 210
 TM00_Stop ... 211
 TM00_UserInit ... 209
 TM5n_ChangeDuty ... 222
 TM5n_ChangeTimerCondition ... 221
 TM5n_Init ... 217
 TM5n_Start ... 219
 TM5n_Stop ... 220
 TM5n_UserInit ... 218

TMH1_CarrierOutputDisable ... 232
 TMH1_CarrierOutputEnable ... 231
 TMHn_ChangeDuty ... 229
 TMHn_ChangeTimerCondition ... 228
 TMHn_Init ... 224
 TMHn_Start ... 226
 TMHn_Stop ... 227
 TMHn_UserInit ... 225
 TMX_DisableHighImpedanceState ... 207
 TMX_EnableHighImpedanceState ... 206
 TMX_Init ... 200
 TMXn_ChangeDualDuty ... 205
 TMXn_ChangeDuty ... 203
 TMXn_Start ... 201
 TMXn_Stop ... 202

【U】

UART6_ErrorCallback ... 146
 UART6_Init ... 135
 UART6_ReceiveData ... 141
 UART6_ReceiveEndCallback ... 144
 UART6_SendData ... 139
 UART6_SendEndCallback ... 143
 UART6_SoftOverRunCallback ... 145
 UART6_Start ... 137
 UART6_Stop ... 138
 UART6_UserInit ... 136

【W】

WDT_Restart ... 234

【あ行】

新しい列 ダイアログ ... 84
 ウィンドウ・リファレンス ... 38
 ウォッチドッグ・タイマ ... 233
 WDT_Restart ... 234
 オペアンプ ... 173
 AMPn_Start ... 181
 AMPn_Stop ... 182
 AMP_Start ... 179
 AMP_Stop ... 180
 OPAMP_Init ... 174

OPAMP_UserInit ... 175
 PGA_ChangePGAFACTOR ... 178
 PGA_Start ... 176
 PGA_Stop ... 177

【か行】

[外部周辺] タブ ... 66
 機能 (コード生成) ... 22
 機能 (端子配置) ... 8
 クロック出力 ... 271
 PCL_ChangeFreq ... 276
 PCL_Init ... 272
 PCL_Start ... 274
 PCL_Stop ... 275
 PCL_UserInit ... 273
 コード生成 パネル ... 71
 コード生成プレビュー パネル ... 74
 コンパレータ ... 183
 Comparator_Init ... 184
 Comparatorm_Start ... 186
 Comparatorm_Stop ... 187
 Comparator_UserInit ... 185

【さ行】

システム ... 103
 CG_ChangeClockMode ... 107
 CG_ChangeFrequency ... 109
 CG_ChangePIIMode ... 113
 CG_ReadResetSource ... 106
 CG_SelectPowerSaveMode ... 110
 CG_SelectStabTime ... 112
 CLOCK_Init ... 104
 CLOCK_UserInit ... 105
 [出力設定] タブ ... 54
 出力パネル ... 77
 シリアル ... 133
 CSI1n_Init ... 148
 CSI1n_ReceiveData ... 152
 CSI1n_ReceiveEndCallback ... 157
 CSI1n_SendEndCallback ... 156
 CSI1n_SendReceiveData ... 154

CSI1n_Start	…	150
CSI1n_Stop	…	151
CSI1n_UserInit	…	149
IICA_GetStopConditionCallback	…	172
IICA_Init	…	158
IICA_MasterErrorCallback	…	166
IICA_MasterReceiveEndCallback	…	165
IICA_MasterReceiveStart	…	162
IICA_MasterSendEndCallback	…	164
IICA_MasterSendStart	…	161
IICA_SlaveErrorCallback	…	171
IICA_SlaveReceiveEndCallback	…	170
IICA_SlaveReceiveStart	…	168
IICA_SlaveSendEndCallback	…	169
IICA_SlaveSendStart	…	167
IICA_Stop	…	160
IICA_StopCondition	…	163
IICA_UserInit	…	159
UART6_ErrorCallback	…	146
UART6_Init	…	135
UART6_ReceiveData	…	141
UART6_ReceiveEndCallback	…	144
UART6_SendData	…	139
UART6_SendEndCallback	…	143
UART6_SoftOverRunCallback	…	145
UART6_Start	…	137
UART6_Stop	…	138
UART6_UserInit	…	136
TM5n_ChangeTimerCondition	…	221
TM5n_Init	…	217
TM5n_Start	…	219
TM5n_Stop	…	220
TM5n_UserInit	…	218
TMH1_CarrierOutputDisable	…	232
TMH1_CarrierOutputEnable	…	231
TMHn_ChangeDuty	…	229
TMHn_ChangeTimerCondition	…	228
TMHn_Init	…	224
TMHn_Start	…	226
TMHn_Stop	…	227
TMHn_UserInit	…	225
TMX_DisableHighImpedanceState	…	207
TMX_EnableHighImpedanceState	…	206
TMX_Init	…	200
TMXn_ChangeDualDuty	…	205
TMXn_ChangeDuty	…	203
TMXn_Start	…	201
TMXn_Stop	…	202
[端子配置図の設定] タブ	…	51
端子配置図 パネル	…	68
[端子配置の設定] タブ	…	48
[端子配置表の情報] タブ	…	50
端子配置表 パネル	…	60
[外部周辺] タブ	…	66
[端子番号] タブ	…	62
[マクロ] タブ	…	64
[端子番号] タブ	…	62
低電圧検出回路	…	277
LVI_Init	…	278
LVI_InterruptModeStart	…	280
LVI_ResetModeStart	…	281
LVI_SetLVILevel	…	283
LVI_Stop	…	282
LVI_UserInit	…	279

【た行】

タイマ	…	198
TM00_ChangeDuty	…	215
TM00_ChangeTimerCondition	…	212
TM00_GetFreeRunningValue	…	213
TM00_GetPulseWidth	…	216
TM00_Init	…	208
TM00_SoftwareTriggerOn	…	214
TM00_Start	…	210
TM00_Stop	…	211
TM00_UserInit	…	209
TM5n_ChangeDuty	…	222

【な行】

名前を付けて保存 ダイアログ	…	87
----------------	---	----

【は行】

[ファイル設定] タブ … 58
 フォルダの参照 ダイアログ … 86
 プロジェクト・ツリー パネル … 42
 プロパティ パネル … 45
 [出力設定] タブ … 54
 [端子配置図の設定] ブ … 51
 [端子配置の設定] タブ … 48
 [端子配置表の情報] タブ … 50
 [ファイル設定] タブ … 58
 [マクロ設定] タブ … 57
 ポート … 114
 PORT_ChangePmnInput … 117
 PORT_ChangePmnOutput … 120
 PORT_Init … 115
 PORT_UserInit … 116

【ま行】

[マクロ設定] タブ … 57
 [マクロ] タブ … 64
 メイン・ウインドウ … 39

【ら行】

リアルタイム・カウンタ … 235
 RTC_AlarmDisable … 253
 RTC_AlarmEnable … 252
 RTC_AlarmGet … 257
 RTC_AlarmInterruptCallback … 258
 RTC_AlarmSet … 254
 RTC_ChangeCorrectionValue … 270
 RTC_ConstPeriodInterruptCallback … 251
 RTC_ConstPeriodInterruptDisable … 250
 RTC_ConstPeriodInterruptEnable … 248
 RTC_CounterEnable … 240
 RTC_CounterGet … 246
 RTC_Disable … 241
 RTC_Init … 237
 RTC_IntervalInterruptDisable … 263
 RTC_IntervalInterruptEnable … 261
 RTC_IntervalStart … 259
 RTC_IntervalStop … 260

RTC_PowerOff … 239
 RTC_RTC1HZ_OutputDisable … 265
 RTC_RTC1HZ_OutputEnable … 264
 RTC_RTCCL_OutputDisable … 267
 RTC_RTCCL_OutputEnable … 266
 RTC_RTCDIV_OutputDisable … 269
 RTC_RTCDIV_OutputEnable … 268
 RTC_SetHourSystem … 242
 RTC_UserInit … 238
 RTC_CounterSet … 244
 列の選択 ダイアログ … 80

【わ行】

割り込み … 122
 INT_MaskableInterruptEnable … 127
 INTP_Init … 123
 INTPn_Disable … 129
 INTPn_Enable … 130
 INTP_UserInit … 124
 KEY_Disable … 131
 KEY_Enable … 132
 KEY_Init … 125
 KEY_UserInit … 126

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2012.09.01	－	初版発行

CubeSuite+ V1.03.00 ユーザーズマニュアル
78K0 設計編

発行年月日 2012年9月1日 Rev.1.00

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



ルネサスエレクトロニクス株式会社

■営業お問合せ窓口

<http://www.renesas.com>

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

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

(03)5201-5307

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

CubeSuite+ V1.03.00