RX スマート・コンフィグレータ
ユーザーガイド: e² studio 編

R20AN0451JS0160
Rev.1.60
2024.04.16

要旨

本アプリケーションノートでは、e² studio のプラグインツールである RX スマート・コンフィグレータ（以下、スマート・コンフィグレータと略す）の基本的な使用方法について説明します。

統合開発環境 e² studio の対象バージョンは以下の通りです。

・e² studio 2024-01 以降

対象デバイス/対応コンパイラ

サポートしているデバイス及びコンパイラは、以下の URL をご参照ください。

https://www.renesas.com/rx-smart-configurator

目次

1. 概要

1.1 目的

本アプリケーションノートは、統合開発環境 e² studio でスマート・コンフィグレータを使用したプロジェクトの作成、基本的な使用方法について説明しています。

e² studio の使い方は、e² studio のユーザーズマニュアルを参照してください。

1.2 特長

スマート・コンフィグレータは、「ソフトウェアを自由に組み合わせられる」をコンセプトとしたユーティリティです。FIT(Firmware Integration Technology)モジュールのミドルウェアをインポート、ドライバコード生成、端子設定の 3 つの機能でお客様のシステムへのルネサス製ドライバの組み込みを容易にします。タイミング波形などのスマート・コンフィグレータのグラフィカルな表示により、ミドルウェアとドライバの設定が簡単になります。

1.3 ソフトウェアコンポーネント

スマート・コンフィグレータは、2 種類のソフトウェアコンポーネント（コード生成（CG）と Firmware Integration Technology（FIT））に対応します。それぞれのソフトウェアが対応するドライバとミドルウェアは、以下の通りです。

- ベーシックドライバ
  - CG ドライバ（CMT、A/D コンバータ、SCI など）
  - FIT モジュール（CMT、DTC、DMAC、RSPI、SCIFA など）
- ミドルウェア
  - FIT モジュール（USB、Ethernet、フラッシュメモリ （内蔵フラッシュメモリ書き換え）など）

ベーシックドライバは、CMT、AD コンバータ、CSI などのマイコン周辺機能の制御プログラムです。

コード生成機能を使用したソフトウェアコンポーネント（CG ドライバ/ FIT モジュール）の組み込みが便利です。

また、USB、Ethernet、フラッシュメモリ（内蔵フラッシュメモリ書き換え）などのミドルウェアを含んだ FIT モジュールをソフトウェアコンポーネントとして組み込むことができます。

2. プロジェクトの作成

スマート・コンフィグレータを使用して C/C++プロジェクトを生成する手順を、以下に説明します。

e² studio のプロジェクト作成ウィザードの詳細は、e² studio の関連ドキュメントを参照してください。

2.1 RTOS を使用しないプロジェクトの作成

RTOS を使用しないプロジェクトの作成手順を、以下に説明します。

(1) e² studio を起動し、ワークスペースを指定します。起動後、［ファイル］→［新規］→［Renesas C/C++プロジェクト］→［Renesas RX］の順に選択してプロジェクト作成ウィザードを開きます。

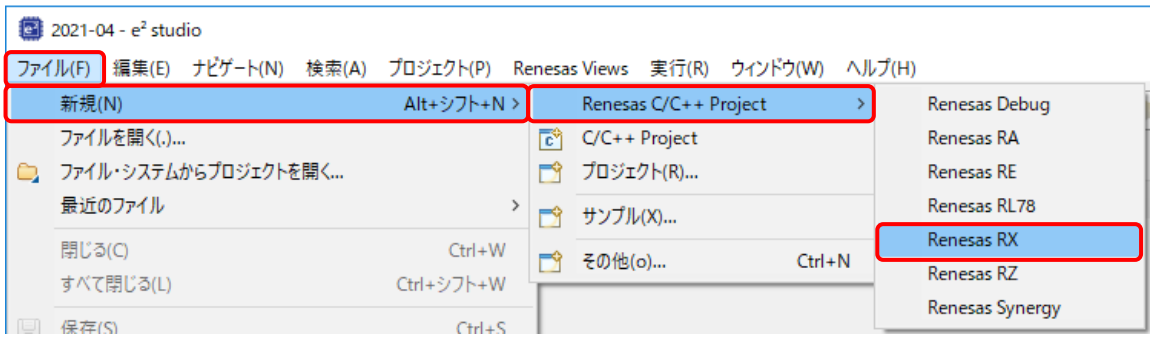

図 2-1 新規プロジェクトの作成

(2) プロジェクト作成ウィザードで、［Renesas RX］→［Renesas CC-RX C/C++ Executable Project］または［GCC for Renesas RX C/C++ Executable Project］を選択し、［次へ］ボタンをクリックします。

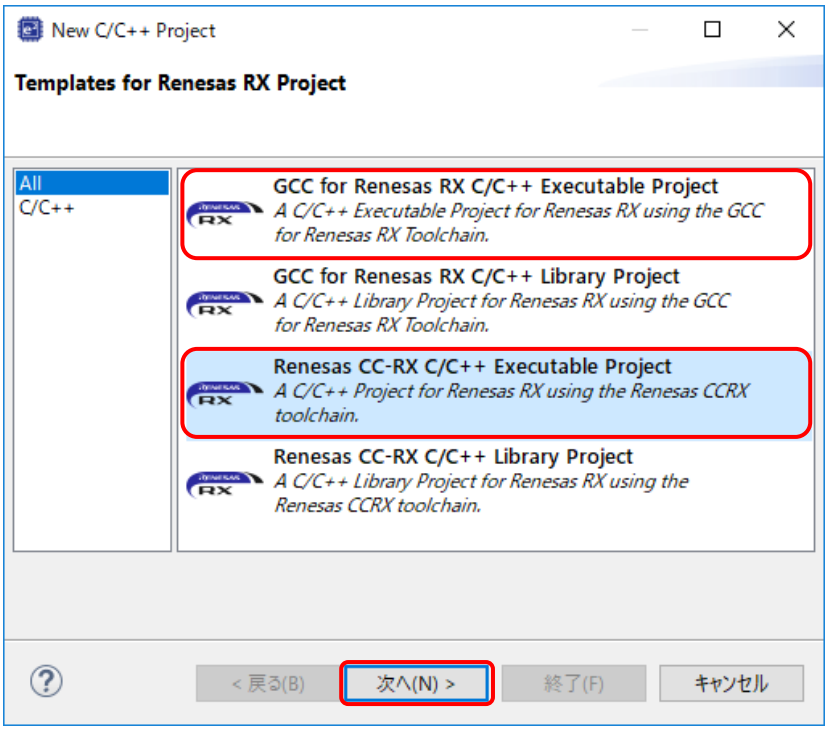

図 2-2 新規 C/C++プロジェクトのテンプレート

(3) プロジェクト名を入力し、 ［次へ］ボタンをクリックして次に進みます。

（例：CC-RX executable project, プロジェクト名：“Smart_Configurator_Example”）

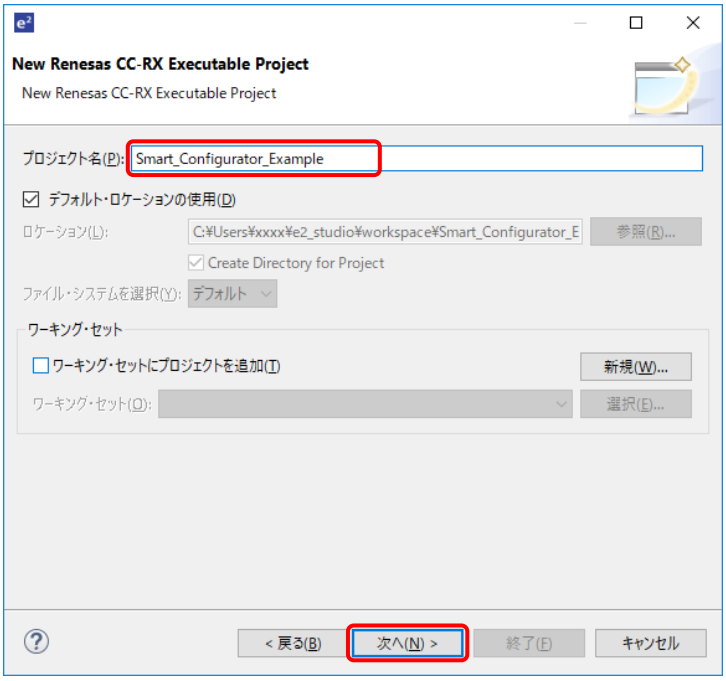

図 2-3 New Renesas CC-RX executable project の作成

(4) 言語、ツールチェーン、ボード、デバイスおよびデバッグ設定を選択します。RTOS は「なし」のまま [次へ] をクリックします。（例：Target Board: RSKRX64M）

【注】 ボードを選択すると、初期端子設定、クロック周波数、ターゲット・デバイスの設定が自動で選択されます。

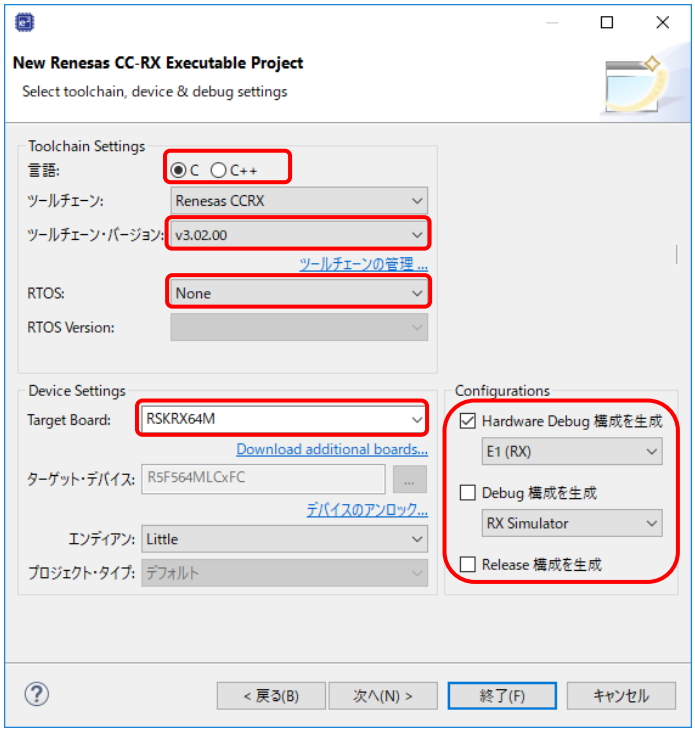

図 2-4 ツールチェーン、デバイス、デバッグ設定の選択

(5) ［コーディング・アシスタントツールの選択］ダイアログボックスで、［スマート・コンフィグレータを使用する］のチェックボックスを選び、［終了］をクリックします。

【注】 (4)で、スマート・コンフィグレータが対応しているデバイスを選択時のみ、［スマート・コンフィグレータを使用する］のチェックボックスが選択可能になります。

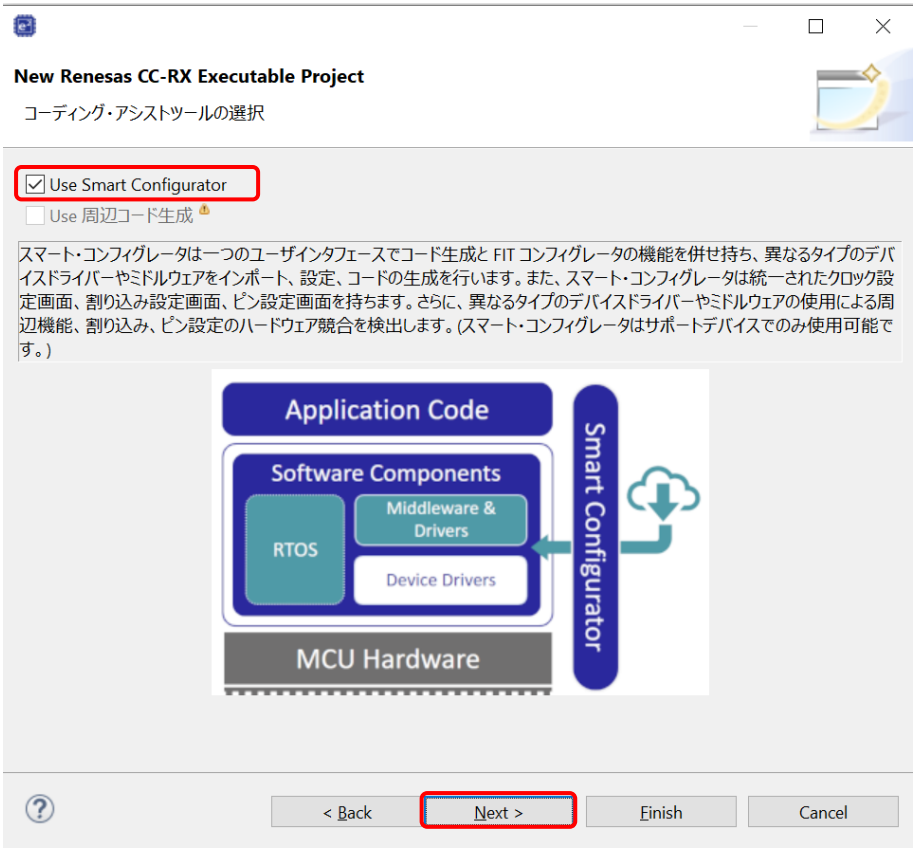

図 2-5 コーディング・アシストツールの選択

(6) プロジェクト作成の完了を待ちます。

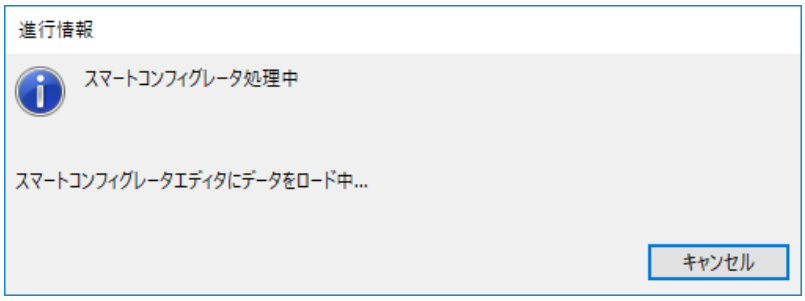

図 2-6 プロジェクト作成の処理

(7) 新規 C/C++プロジェクトの作成が成功すると、作成したプロジェクトがスマート・コンフィグレータ・パースペクティブ上で開きます。

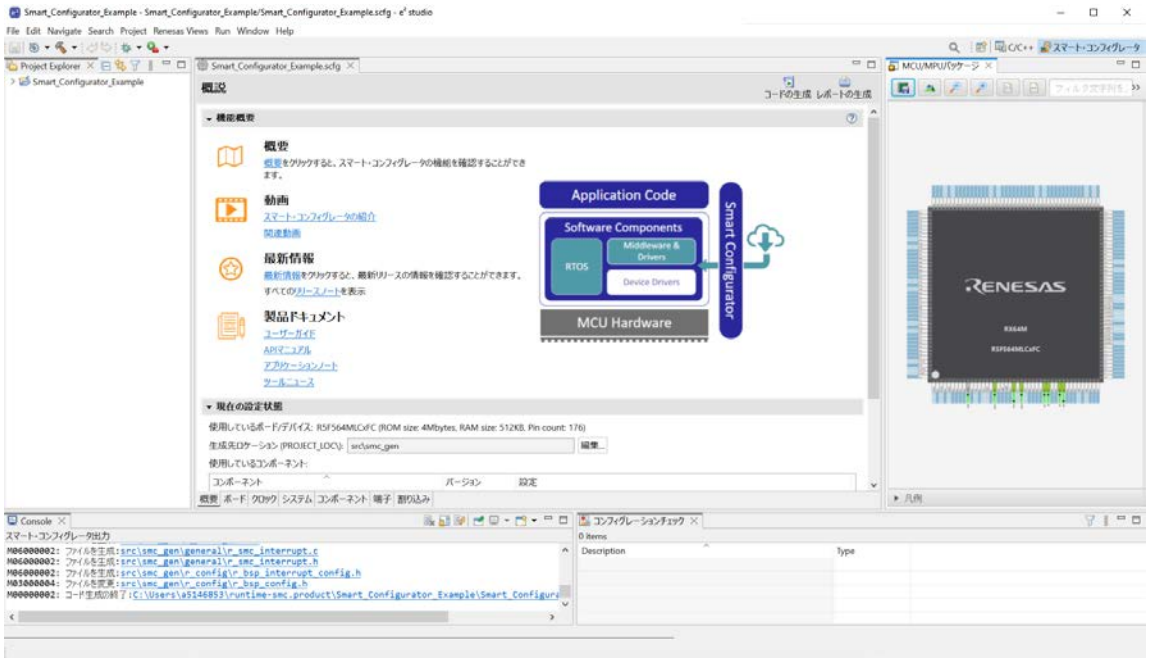

図 2-7 ［スマート・コンフィグレータ］パースペクティブ

2.2 RTOS プロジェクトの作成

RTOS プロジェクトの作成手順を、以下に説明します。

【注】FreeRTOS プロジェクトに対応しているデバイスおよび Renesas FreeRTOS については、Renesas FreeRTOS の関連ドキュメントを参照してください。（「12 参考ドキュメント」参照）

(1) e² studio を起動し、ワークスペースを指定します。起動後、［ファイル］→［新規］→［Renesas C/C++プロジェクト］→［Renesas RX］の順に選択してプロジェクト作成ウィザードを開きます。

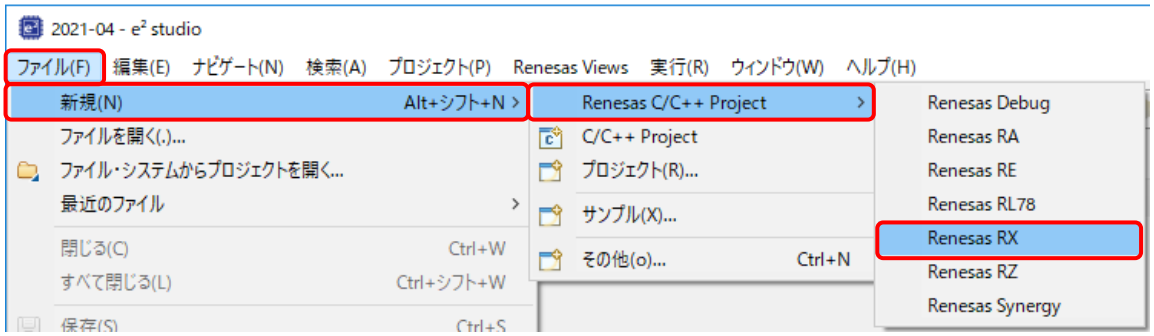

図 2-8 新規プロジェクトの作成

(2) プロジェクト作成ウィザードで、［Renesas RX］→［Renesas CC-RX C/C++ Executable Project］を選択し、［次へ］ボタンをクリックします。

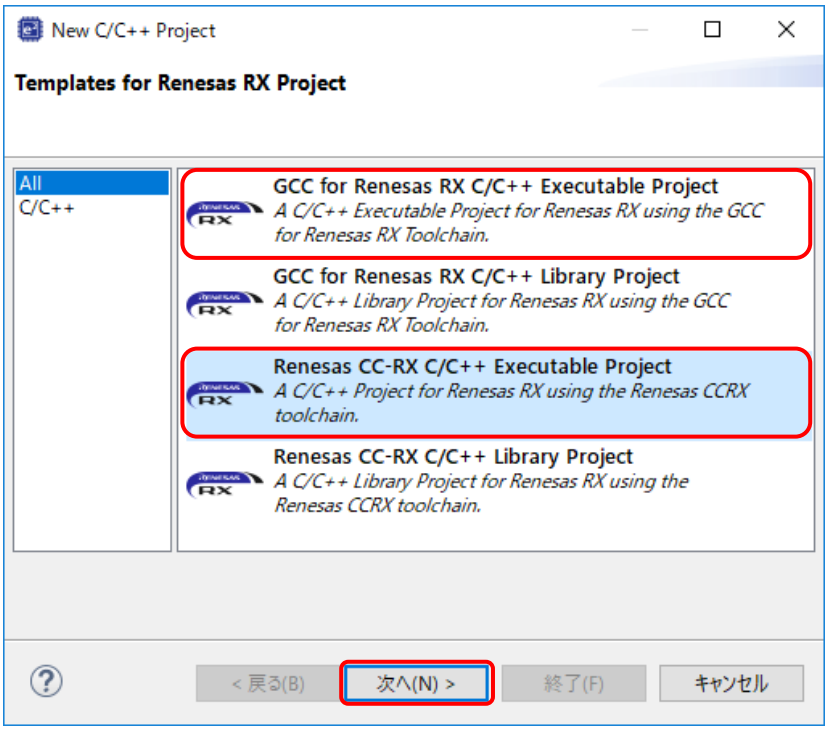

図 2-9 新規 C/C++プロジェクトのテンプレート

(3) プロジェクト名を入力し、［次へ］ボタンをクリックして次に進みます。

（例：CC-RX executable project，プロジェクト名：“Smart_Configurator_Example”）

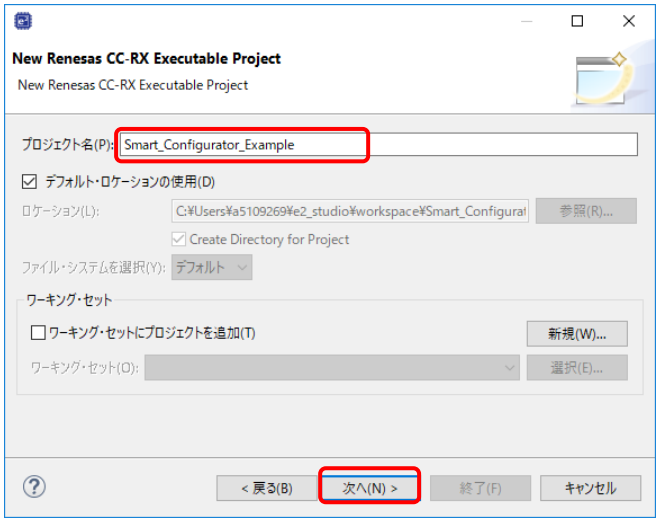

図 2-10 New Renesas CC-RX executable project の作成

(4) ツールチェーンと RTOS を選択します。（例：RTOS：FreeRTOS）

- FreeRTOS (kernel only) ：FreeRTOS (kernel only) でプロジェクトを作成します。
- FreeRTOS (with IoT libraries) ：FreeRTOS (with IoT libraries) でプロジェクトを作成します。
- Azure RTOS：Azure RTOS でプロジェクトを作成します。
- FreeRTOS (with IoT libraries)(deprecated structure): FreeRTOS (with IoT libraries)(deprecated structure)でプロジェクトを作成します。
- RI600V4：RI600V4 Real-time OS でプロジェクトを作成します。

RI600V4 を選択するには、プロジェクト作成前に RI600V4 Real-time OS をインストールする必要があります（https://www.renesas.com/software-tool/ri600v4-real-time-os-rx-family）。

FreeRTOS または Azure RTOS をダウンロードしていない場合は手順 (5) 、ダウンロード済みの場合は手順 (7) に進みます。

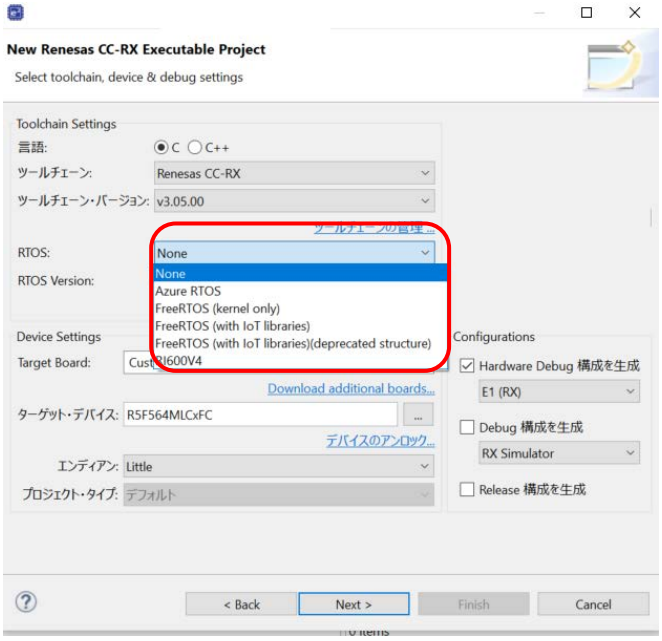

図 2-11 RTOS 設定の選択

(5) ［RTOS モジュールダウロード］ダイアログが表示されますので、RTOS パッケージを選択し、［ダウンロード］をクリックします。

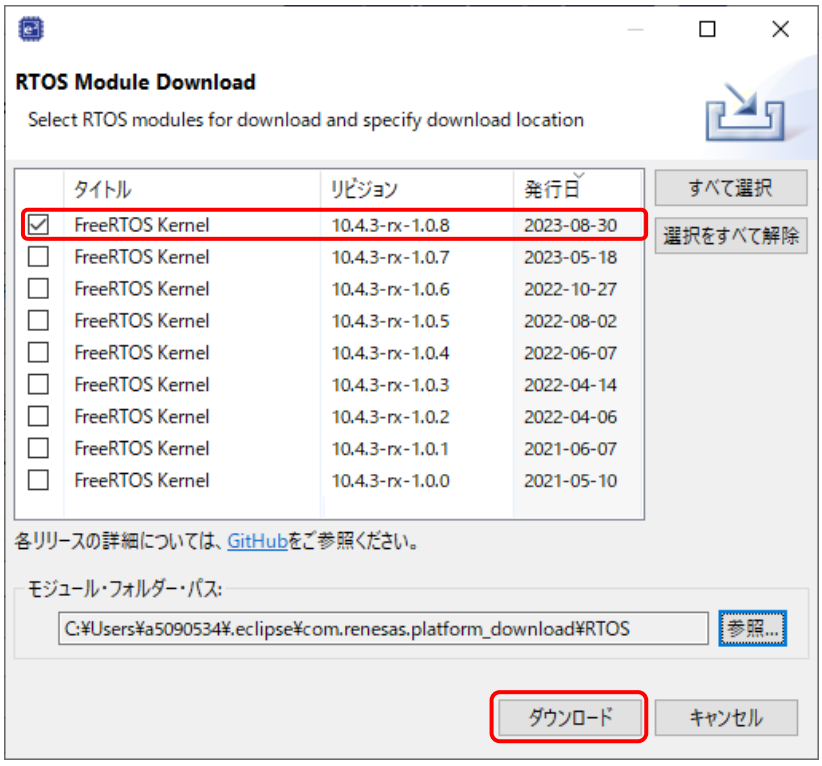

図 2-12 ［RTOS モジュールダウンロード］ダイアログ

(6) 免責事項が表示されますので、［同意する］をクリックします。

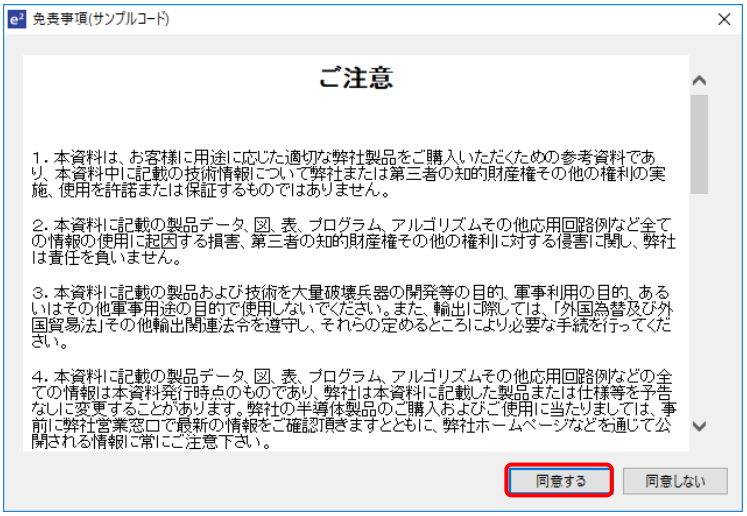

図 2-13 免責事項

(7) ボード、デバイスおよびデバッグ設定を選択します。RTOS パッケージでサポートされているデバイスのみ選択できます。［次へ］をクリックします。 (例：ターゲット・ボード：RSKRX64M)

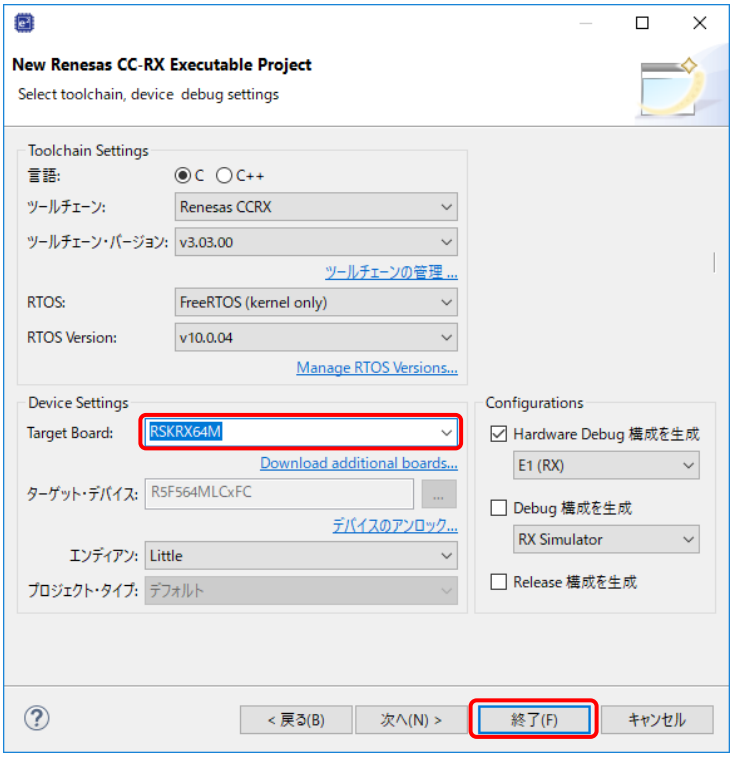

図 2-14 デバイス、デバック設定の選択

(8) ［コーディング・アシスタントツールの選択］ダイアログボックスで、［スマート・コンフィグレータを使用する］のチェックボックスを選び、［終了］をクリックします。

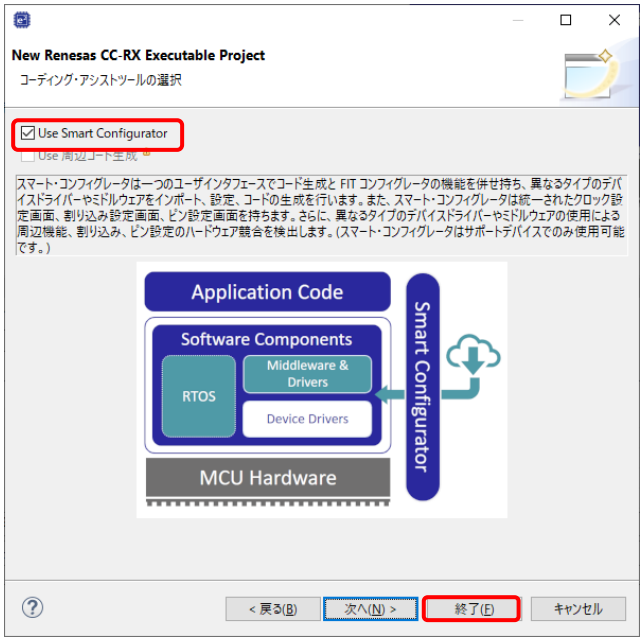

図 2-15 コーディング・アシストツールの選択

(8) プロジェクト作成の完了を待ちます。

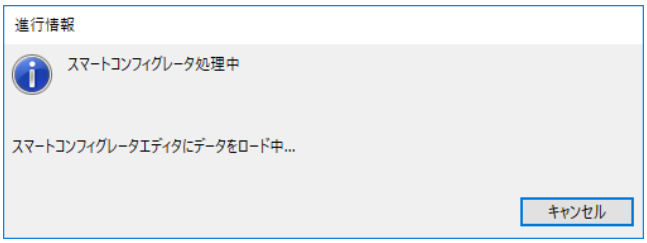

図 2-16 プロジェクト作成の処理

(9) 新規 C プロジェクトの作成が成功すると、作成したプロジェクトがスマート・コンフィグレータ・パースペクティブ上で開きます。

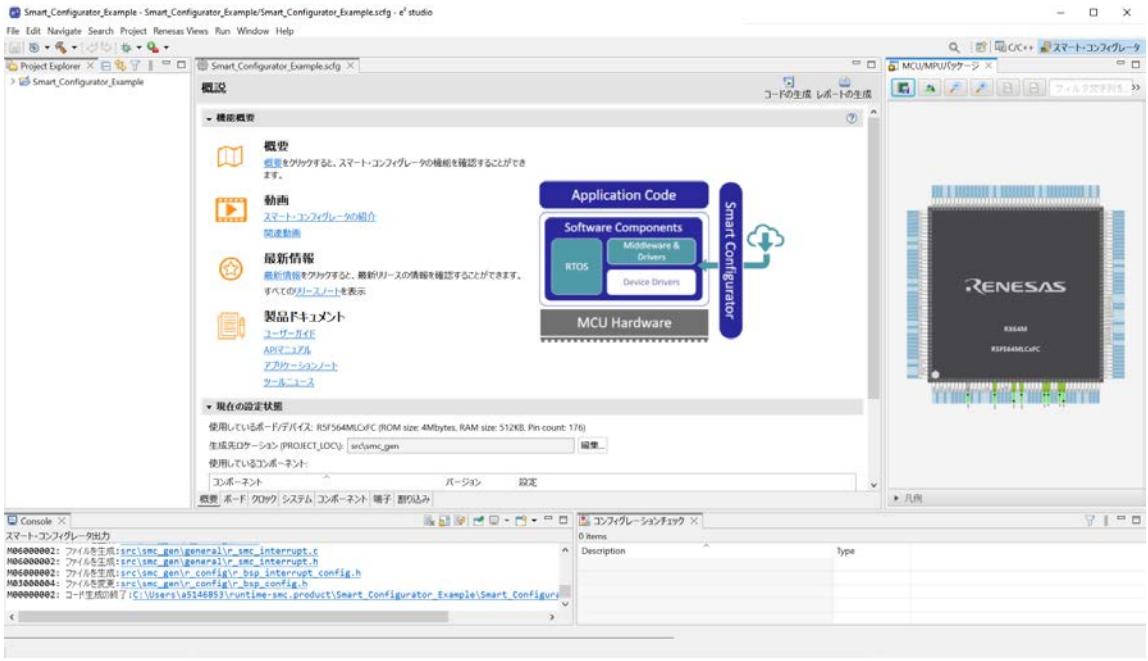

図 2-17 ［スマート・コンフィグレータ］パースペクティブ

2.3 Blinky プロジェクトの作成

サンプルプロジェクトを使用してプロジェクトを生成する手順を、以下に説明します。

(1) e² studio を起動し、ワークスペースを指定します。起動後、［ファイル］→［新規］→［Renesas C/C++プロジェクト］→［Renesas RX］の順に選択してプロジェクト作成ウィザードを開きます。

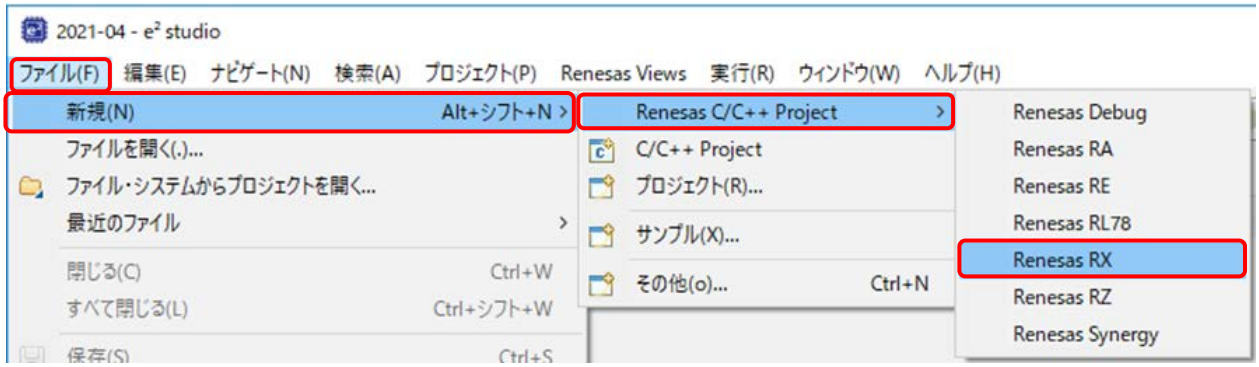

図 2-18 新規プロジェクトの作成

(2) プロジェクト作成ウィザードで、［Renesas RX］→［Renesas CC-RX C/C++ Executable Project］を選択し、［次へ］ボタンをクリックします。

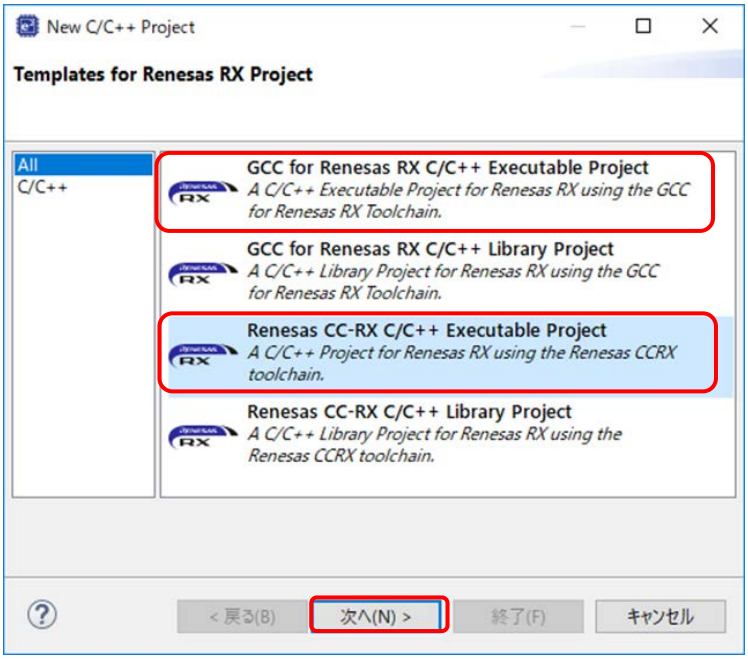

図 2-19 新規 C/C++プロジェクトのテンプレート

(3) プロジェクト名を入力し、 ［次へ］ボタンをクリックして次に進みます。

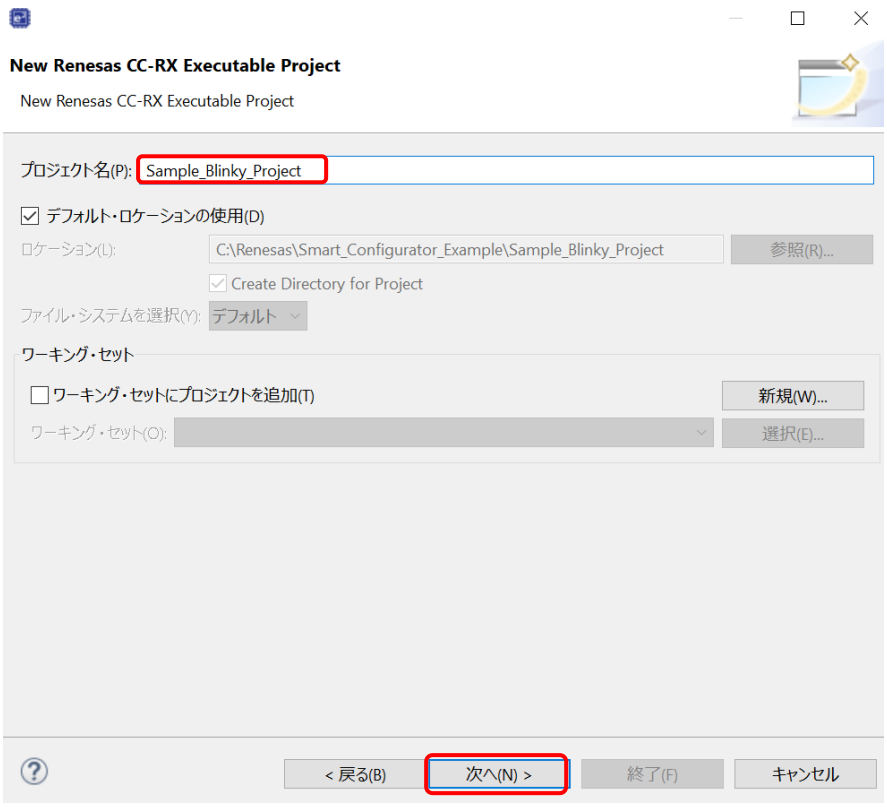

図 2-20 Blinky Project のプロジェクト名

(4) 言語、ツールチェーン、ボード、デバイスおよびデバッグ設定を選択します。RTOS は「なし」のまま [次へ] をクリックします。

【注】 Blinky Project は、ルネサスボードでのみ使用可能です。（例：Target Board: CK-RX64M）

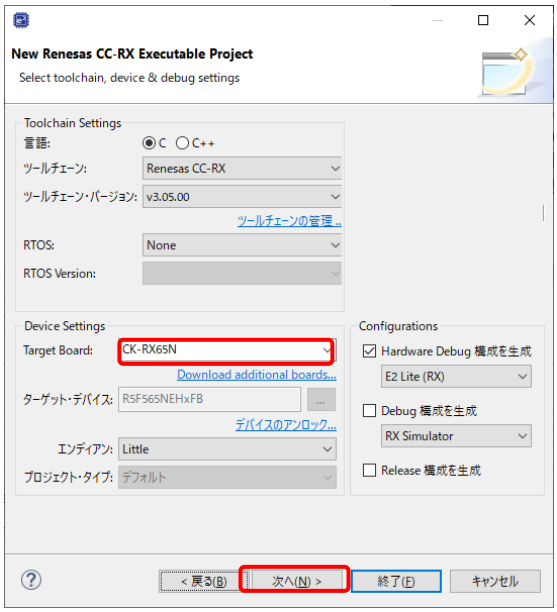

図 2-21 デバイス、デバック設定の選択

(5) ［コーディング・アシスタントツールの選択］ダイアログボックスで、［スマート・コンフィグレータを使用する］のチェックボックスを選び、［次へ］をクリックします。

図 2-22 コーディング・アシストツールの選択

(6) ［プロジェクトテンプレートの選択］ダイアログボックスで、［Bare Metal - Blinky］のチェックボックスを選び、［終了］をクリックします。

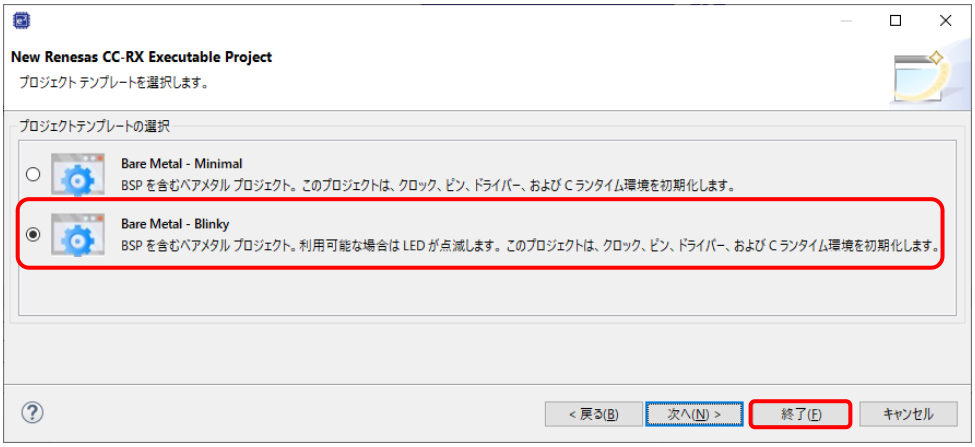

図 2-23 プロジェクトテンプレートの選択

(7) プロジェクトの作成が成功すると、作成したプロジェクトがスマート・コンフィグレータ・パースペクティブ上で開きます。

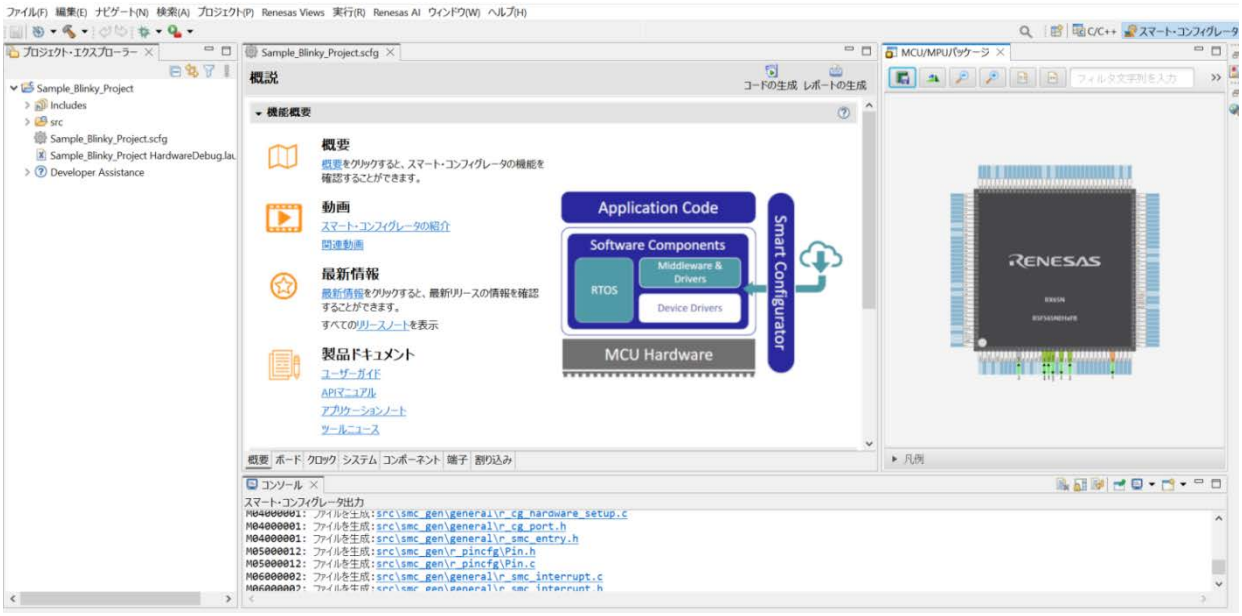

図 2-24 ［スマート・コンフィグレータ］パースペクティブ

(8) コンポーネントページに移動すると、既存のコンポーネントと設定を確認できます。

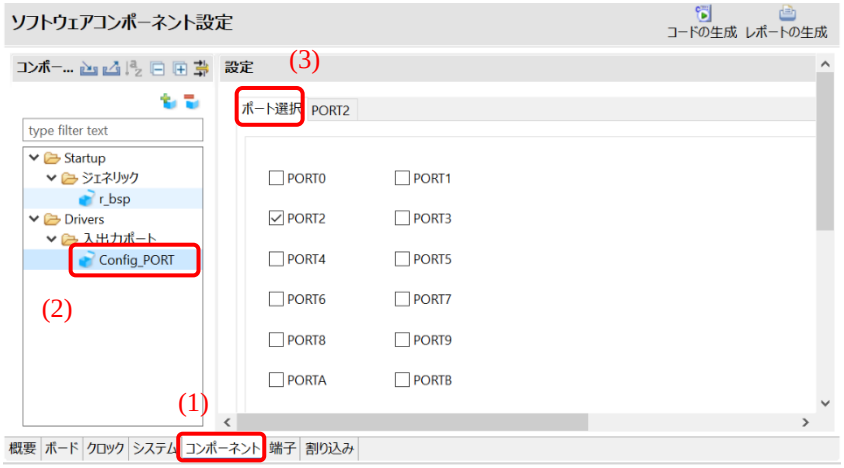

図 2-25 コンポーネント設定(1)

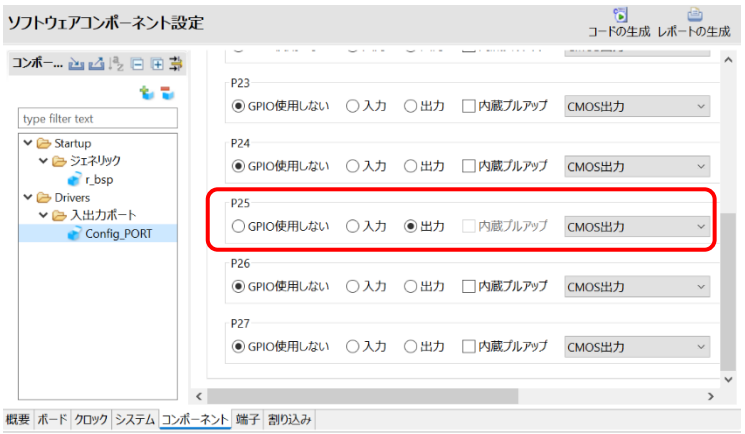

図 2-26 コンポーネント設定(2)

(9) プロジェクト・エクスプローラーで、[src]フォルダを展開し、[Sample_Blinky_Project.c]をダブルクリックして、Main 関数を開きます。サンプルコードが表示されます。

図 2-27 メイン関数

(10) コンピュータをターゲットボードに接続することで、このプロジェクトをビルドし、ターゲットボード上でデバッグすることができ、LED の点滅を確認できます。

3. スマート・コンフィグレータの操作方法

3.1 スマート・コンフィグレータの表示

スマート・コンフィグレータの機能を十分に活用するためには、スマート・コンフィグレータ・パースペクティブを確実に開いていることが必要です。開いていない場合は、e² studio ウィンドウ右上角のパースペクティブを選択してください。

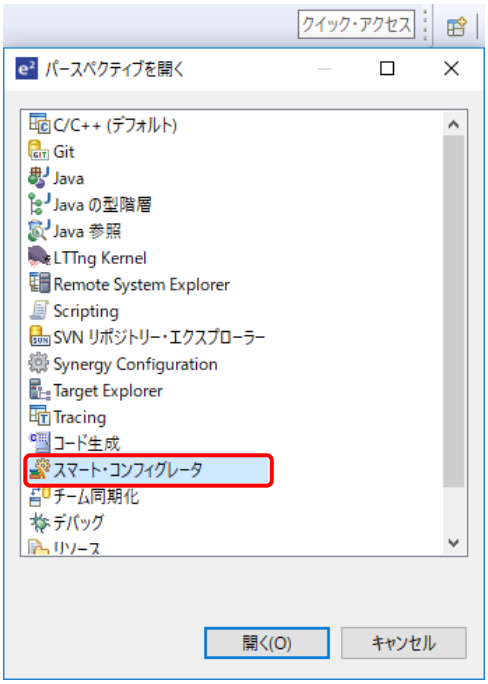

図 3-1 ［スマート・コンフィグレータ］パースペクティブを開く

3.2 操作手順

e² studio 上のスマート・コンフィグレータで周辺機能の設定し、ビルドするまでの手順を**図 3-2 操作手順**に示します。e² studio の操作については、e² studio の関連ドキュメントを参照してください。

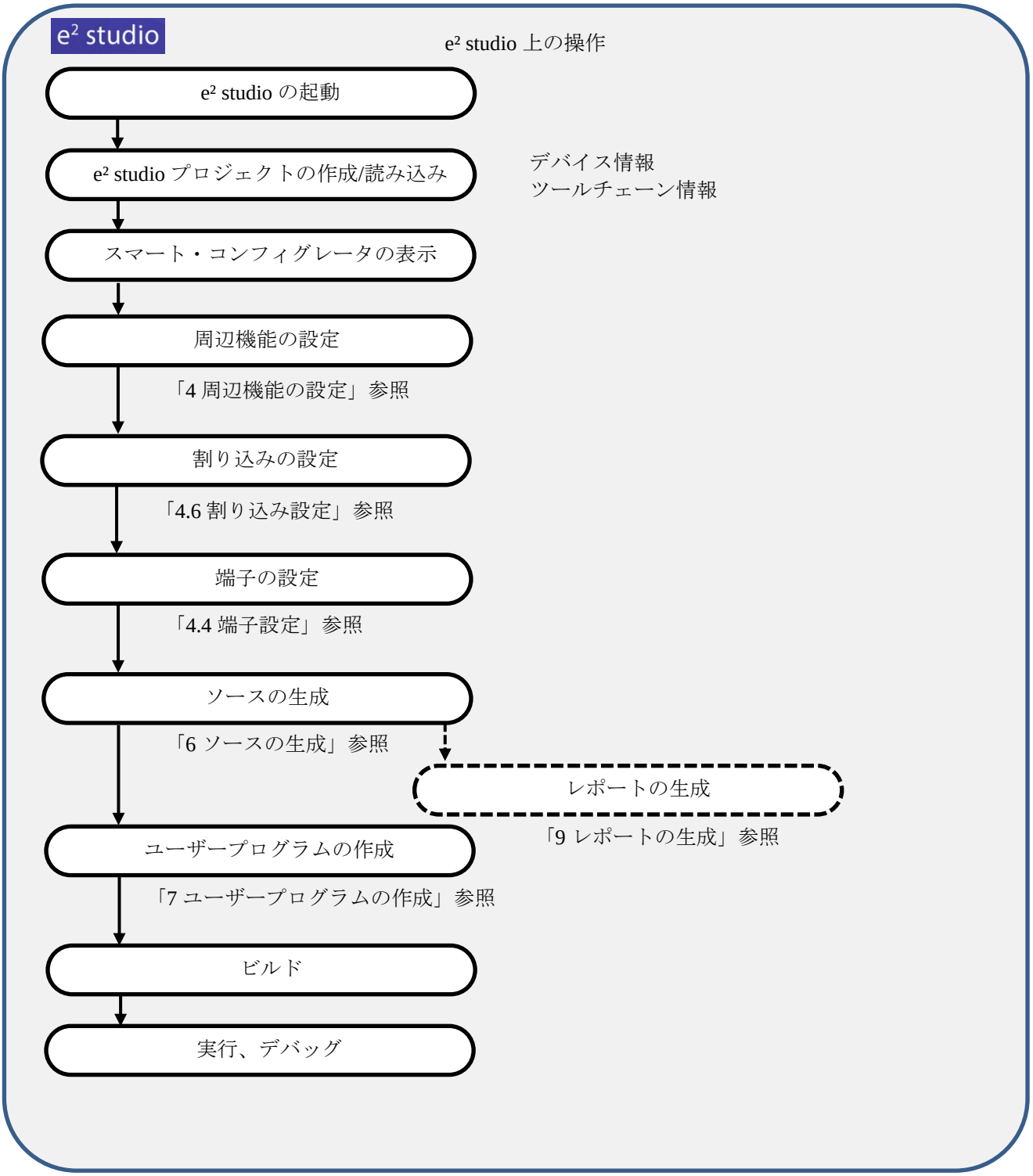

図 3-2 操作手順

3.3 プロジェクト情報の保存先

スマート・コンフィグレータは、プロジェクトで使用するマイクロコントローラ、ビルド・ツール、周辺機能、端子機能などの設定情報をプロジェクト・ファイル(*.scfg)に保存し、参照します。

スマート・コンフィグレータのプロジェクト・ファイルは、e² studio のプロジェクト・ファイル(.project)と同階層にある“プロジェクト名.scfg”に保存します。

3.4　ウィンドウ

［スマート・コンフィグレータ］パースペクティブの構成を図 3-3 ［スマート・コンフィグレータ］パースペクティブに示します。

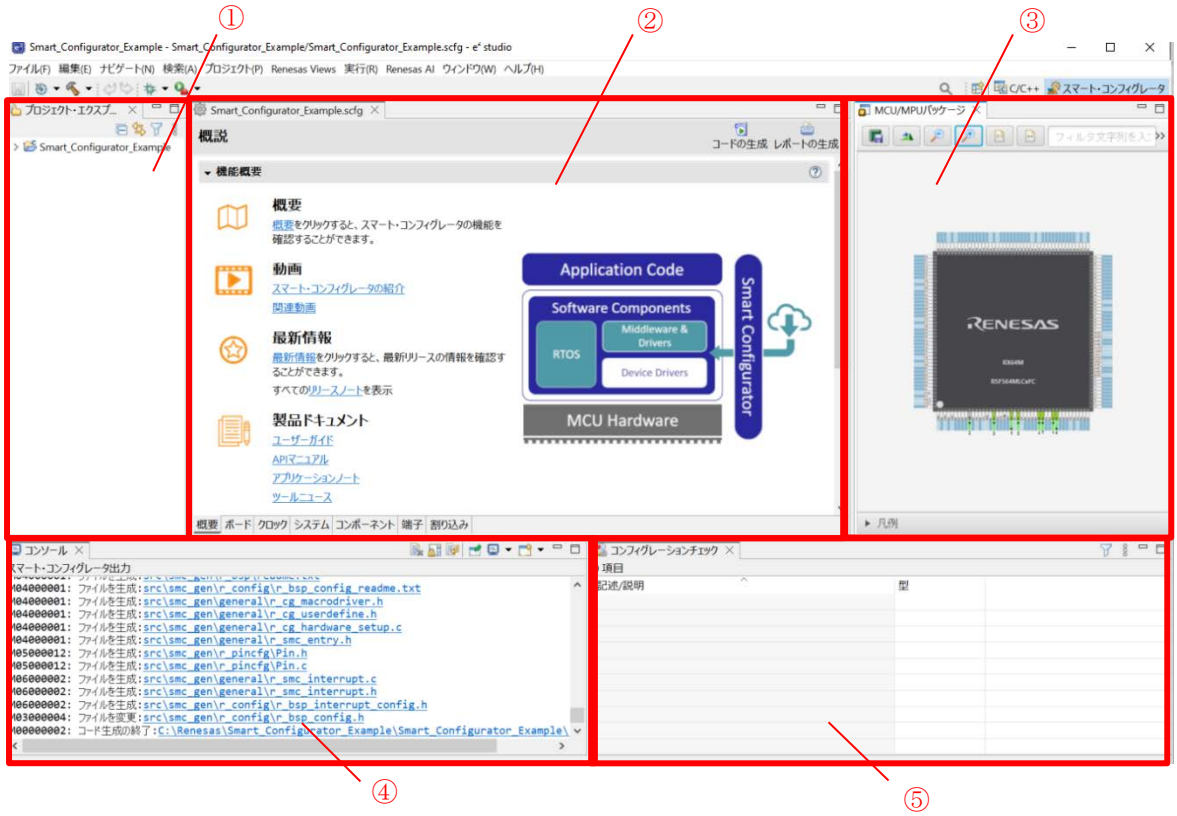

図 3-3 ［スマート・コンフィグレータ］パースペクティブ

① プロジェクト・エクスプローラー

② スマート・コンフィグレータビュー

③ MCU/MPU パッケージビュー

④ コンソールビュー

⑤ コンフィグレーションチェックビュー

3.4.1　プロジェクト・エクスプローラー

プロジェクトのフォルダ構成をツリーで表示します。

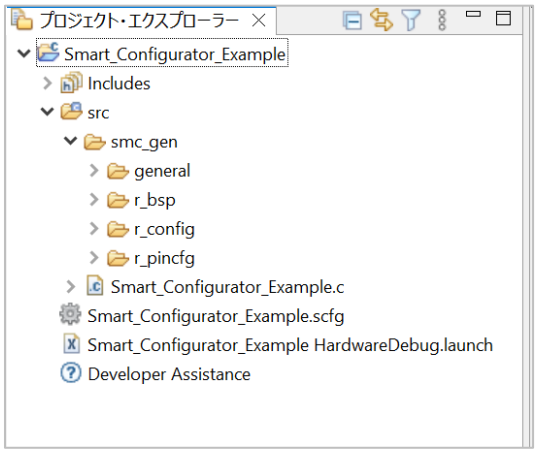

図 3-4 プロジェクト・エクスプローラー

ビューが開いていない場合は、e² studio メニュー上の［ウィンドウ］→［ビューの表示］→［その他］を選択し、開いた［ビューの表示］ダイアログボックスから［一般］→［プロジェクト・エクスプローラー］を選択してください。

3.4.2　スマート・コンフィグレータビュー

［概要］，［ボード］，［クロック］，［システム］，［コンポーネント］，［端子］，［割り込み］の 7 つのページから構成されます。タブをクリックして、ページを選択すると選択したタブに応じて内容が切り替わります。

図 3-5 スマート・コンフィグレータビュー

ビューが開いていない場合は、［プロジェクト・エクスプローラー］からプロジェクト・ファイル(*.scfg)を右クリックし、コンテキストメニューから［開く］を選択してください。

3.4.3　MCU/MPU パッケージビュー

MCU/MPU パッケージ図上に端子状態を表示します。端子設定を変更することもできます。

表示は、[割り当てられた機能]、[ボード機能]、[シンボリック名]の 3 種類の切り替えが行えます。[割り当てられた機能]は、端子設定の割り当て状態を、[ボード機能]は、ボードの初期端子設定情報を、[シンボリック名]は、端子のシンボリック名情報を表示します。ボードの初期端子設定情報は、［ボード］ページの[ボード:]で選択したボードの端子情報です（「4.1.1 ボード選択」、「4.5.6 ボード端子設定情報を使用した端子設定」参照）。

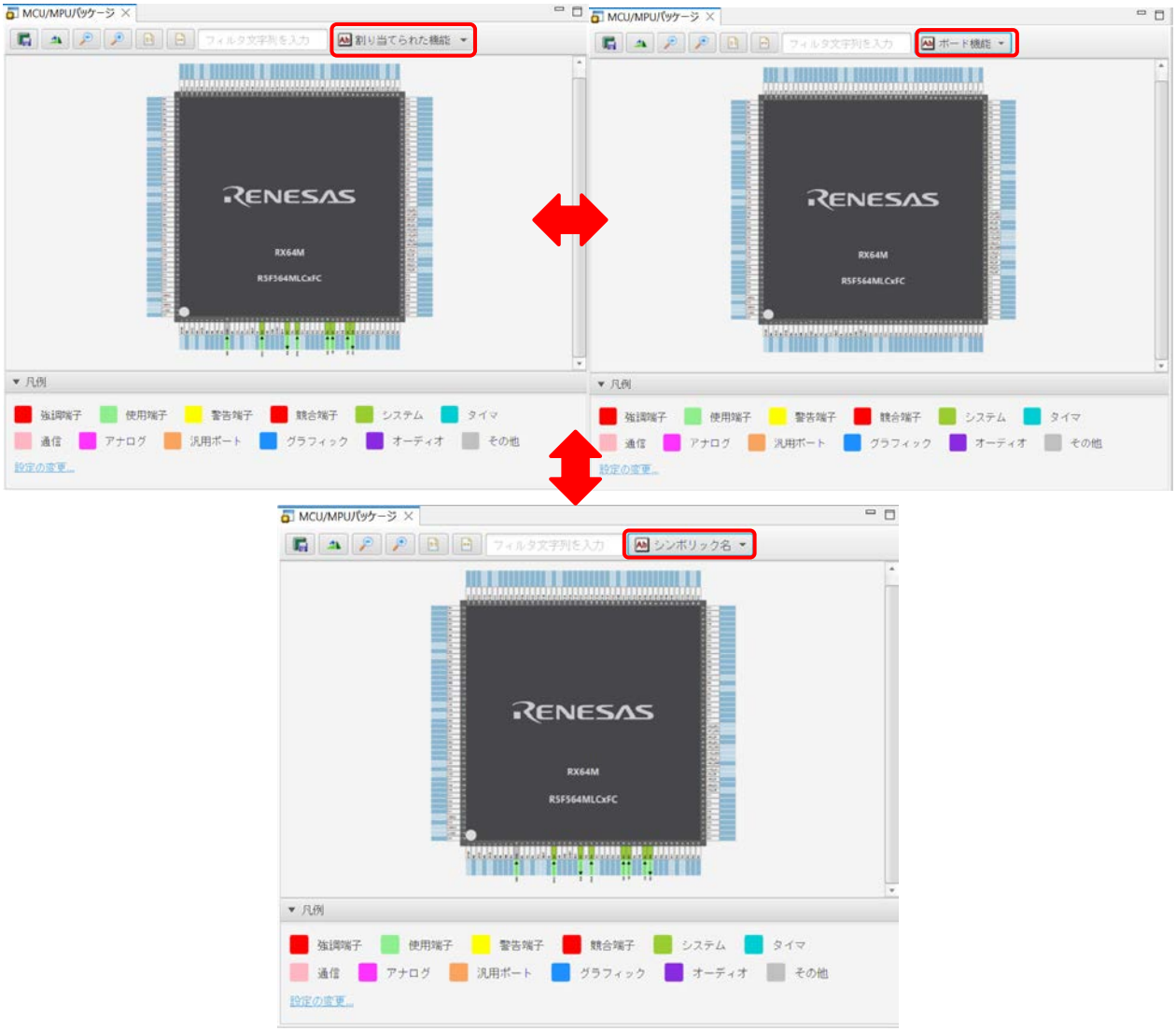

図 3-6 MCU/MPU パッケージビュー

ビューが開いていない場合は、e² studio メニュー上の［Renesas Views］→［スマート・コンフィグレータ］→［MCU/MPU パッケージ］を選択してください。

3.4.4　コンソールビュー

スマート・コンフィグレータビューまたは MCU/MPU パッケージビューでの設定変更内容が表示されます。

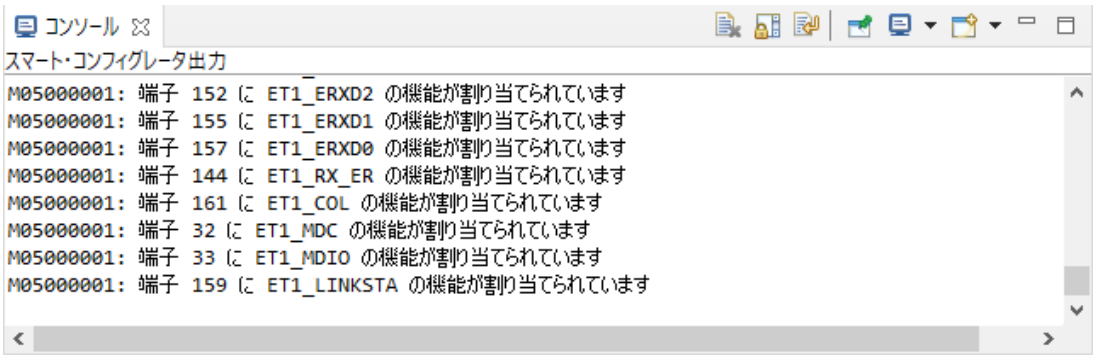

図 3-7 コンソールビュー

ビューが開いていない場合は、e² studio メニュー上の［ウィンドウ］→［ビューの表示］→［その他］を選択し、開いた［ビューの表示］ダイアログボックスから［一般］→［コンソール］を選択してください。

3.4.5　コンフィグレーションチェックビュー

端子競合が発生した場合に、その内容を表示します。

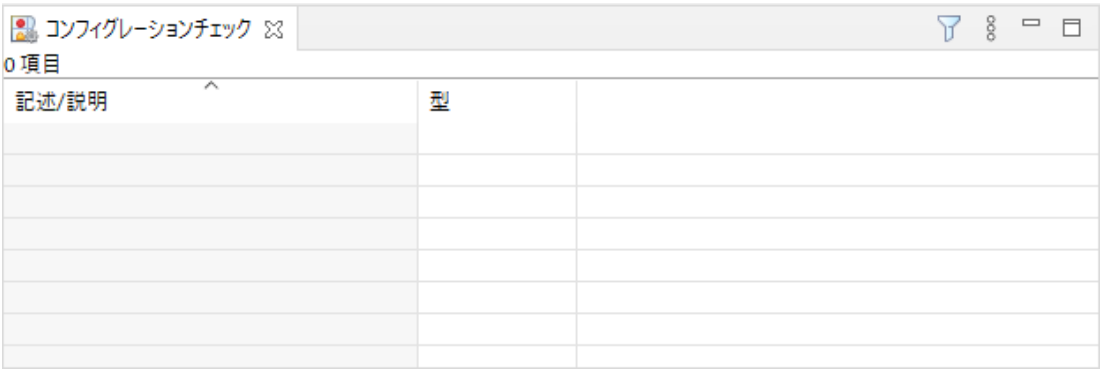

図 3-8 コンフィグレーションチェックビュー

ビューが開いていない場合は、e² studio メニュー上の［Renesas Views］→［スマート・コンフィグレータ］→［コンフィグレーションチェック］を選択してください。

4. 周辺機能の設定

周辺機能は、スマート・コンフィグレータビューから選択します。

4.1 ボード設定

ボードページでは、ボードおよび、デバイスの変更が可能です。

4.1.1 ボード選択

[...] ボタンをクリックすると、ボードが選択できます。

ボード選択により、以下の一括変更が可能です。

- 端子割り当て（初期端子設定）
- メインクロック周波数
- サブクロック周波数
- デバイスの変更

上記ボード設定情報は、Board Description File（.bdf）に定義されています。

Renesas 製ボード(Renesas Starter Kit 等)の.bdf ファイルを WEB からダウンロードし、インポートが可能です。

また、アライアンスパートナーが公開している.bdf ファイルを WEB からダウンロードし、インポートすることで、アライアンスパートナー製ボードの選択が可能となります。

選択したボードに応じて、デバイスが変更され、デバイスの変更は e² studio プロジェクトのターゲット・デバイスに反映されます。「4.7 MCU マイグレーション機能」の手順と同じです。

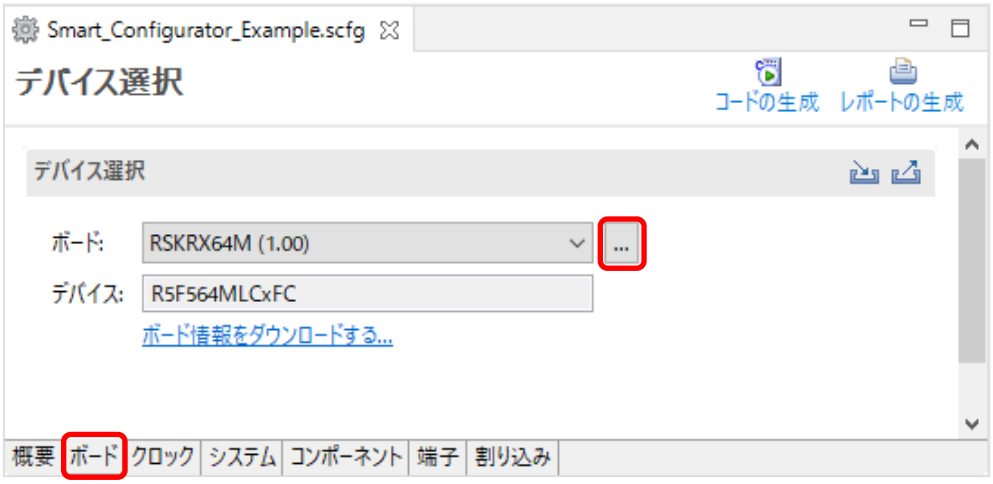

図 4-1 ボード選択

4.1.2　ボード設定のエクスポート

ボード設定のエクスポートは、以下の手順で行います。

(1) ボードページで、［ボードの設定をエクスポート］ ボタンをクリックします。

(2) 出力場所を選択し、エクスポートするファイル名(表示名)を入力します。

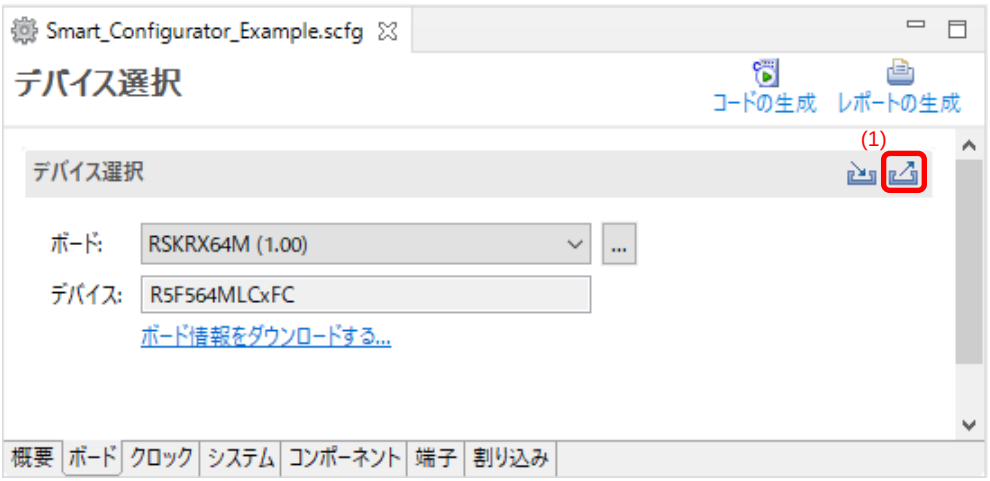

図 4-2 ボード設定のエクスポート(bdf 形式)

4.1.3　ボード設定のインポート

ボード設定のインポートは、以下の手順で行います。

(1) ［ボードの設定をインポート］ ボタンをクリックし、bdf ファイルを選択してください。

(2) インポートしたボード設定がボード選択の選択肢に追加されます。

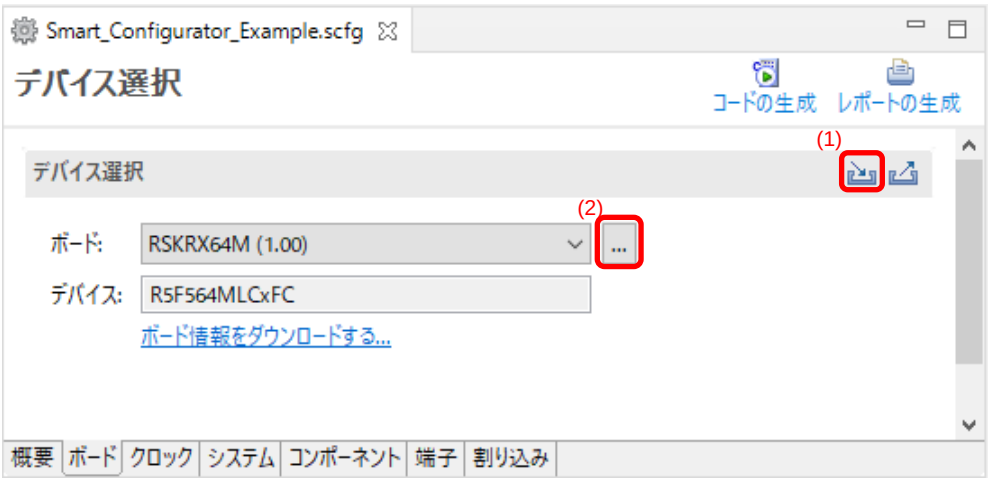

図 4-3 ボード設定のインポート(bdf 形式)

一度インポートしたボード設定は、同じデバイスグループのプロジェクトにおいてボード選択の選択肢に表示されます。

4.2　クロック設定

クロックページでは、システムクロックを設定することができます。クロックページで作成した設定は、全てのドライバおよびミドルウェアで使用されます。

クロック設定を更新するには、以下の手順で行います。

(1) VCC を設定してください。

(2) ボード上で動作するために必要なクロックを選択します（メインクロックがデフォルトで選択されています）。

(3) ボード条件を元に、各クロックに周波数を入力します（内部クロックの周波数は固定されているものもあります）。

(4) PLL 回路を使用されている場合、PLL のクロックソースを選択します。

(5) マルチプレクサの図形では、出力クロックのためにクロックソースを選択します。

(6) 期待した出力クロック周波数を得るために、ドロップダウンリストで分周比を設定します。

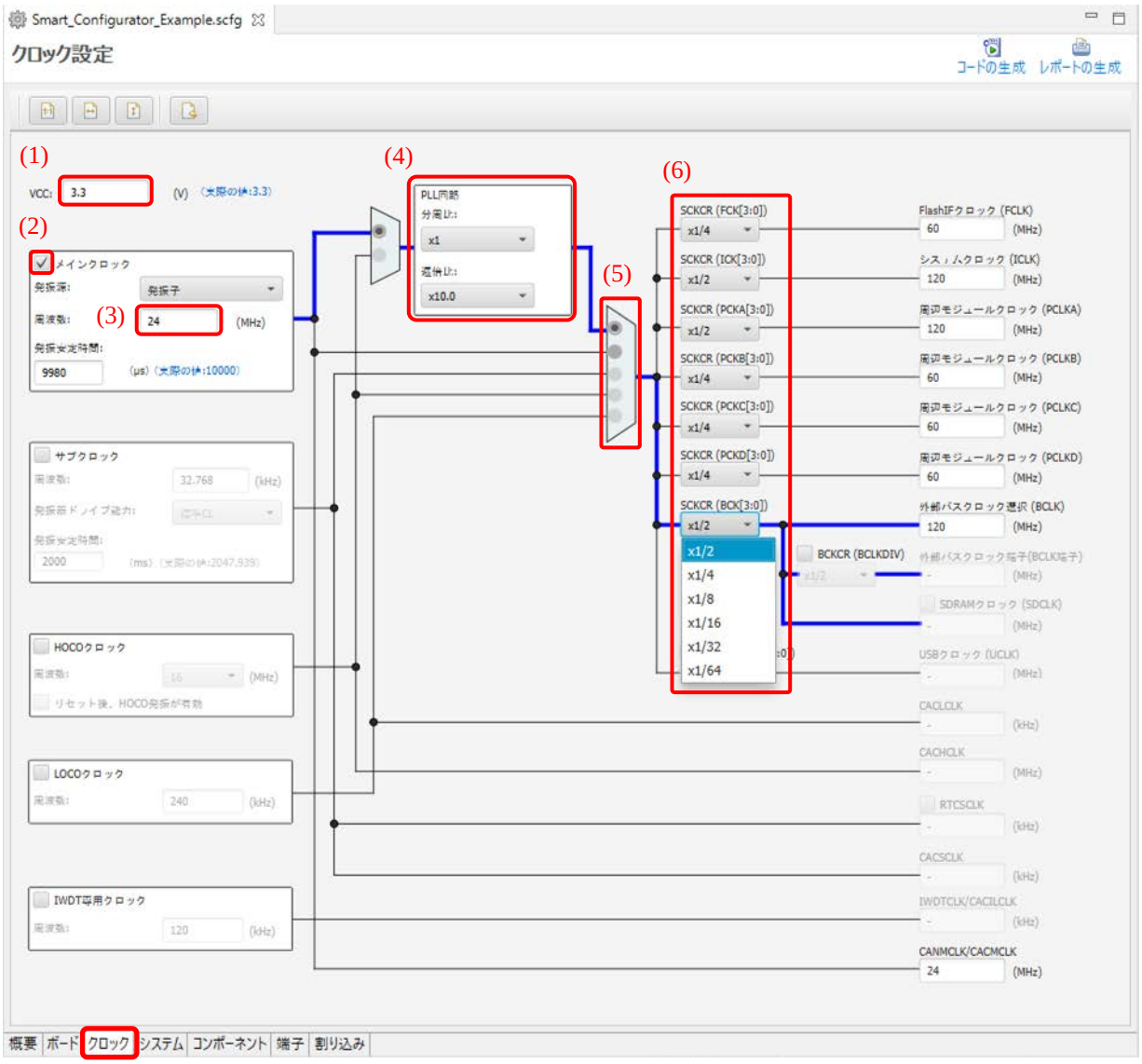

図 4-4 クロック設定

4.3　システム設定

［システム］ページでは、デバッグインターフェイスの端子を設定できます。

使用可能なデバッグインターフェイスには、FINE, JTAG, JTAG (トレース) の 3 種類があります。

コンソールメッセージまたは MCU/MPU パッケージビューから設定された端子を確認できます。

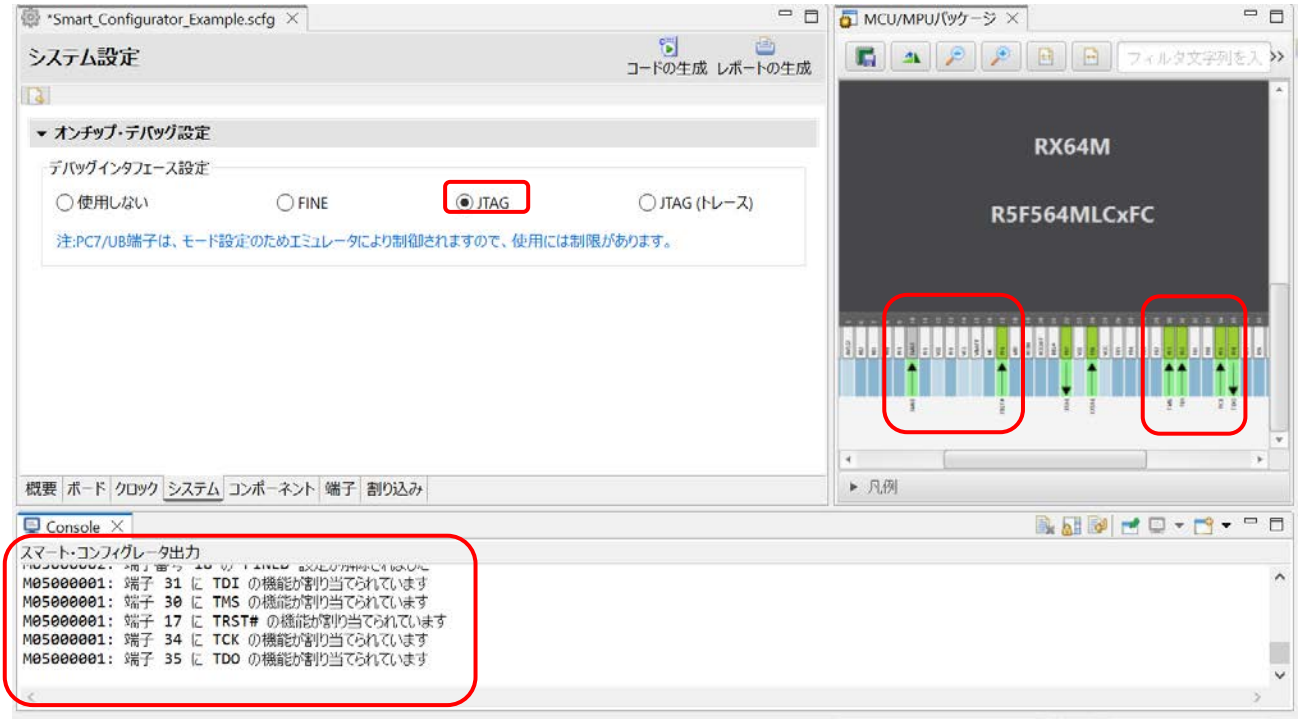

図 4-5 システム設定

4.4　コンポーネント設定

コンポーネントページは、ドライバやミドルウェアをソフトウェアコンポーネントとして組み合わせます。追加したコンポーネントは、左側のコンポーネントツリーに表示されます。

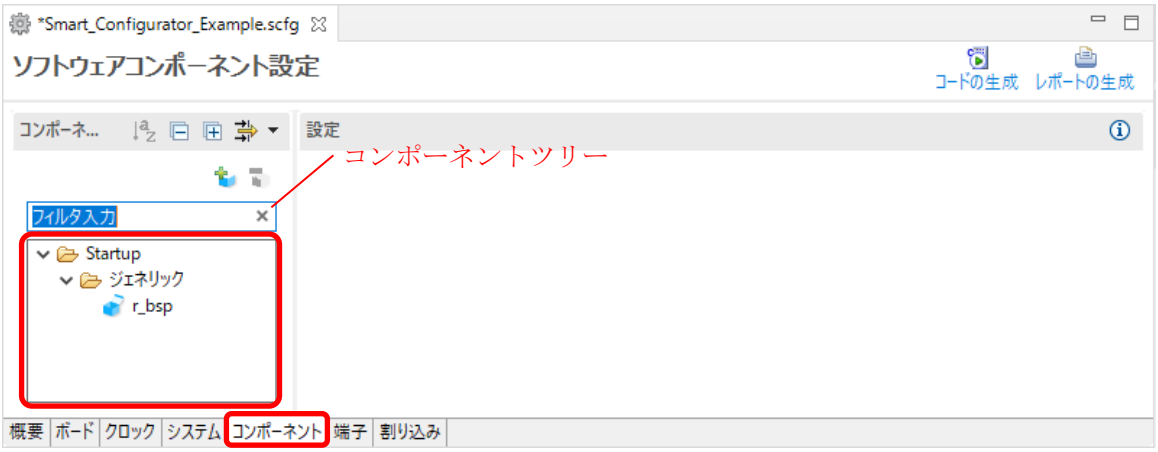

図 4-6 コンポーネントページ

スマート・コンフィグレータは、コード生成（CG）と Firmware Integration Technology (FIT)の２種類のソフトウェアコンポーネントを対象とします。

4.4.1　コード生成コンポーネントの追加方法

コポーネントの追加は、以下の手順で行います。

(1) ［コンポーネントの追加］ アイコンをクリックします。

図 4-7 コンポーネントの追加

(2) ［コンポーネントの追加］ダイアログボックスの［ソフトウェアコンポーネントの選択］のリストからコンポーネントを選択します。（例：シングルスキャンモード S12AD）

(3) ［タイプ］は［コード生成］であることを確認してください。

(4) ［次へ］をクリックします。

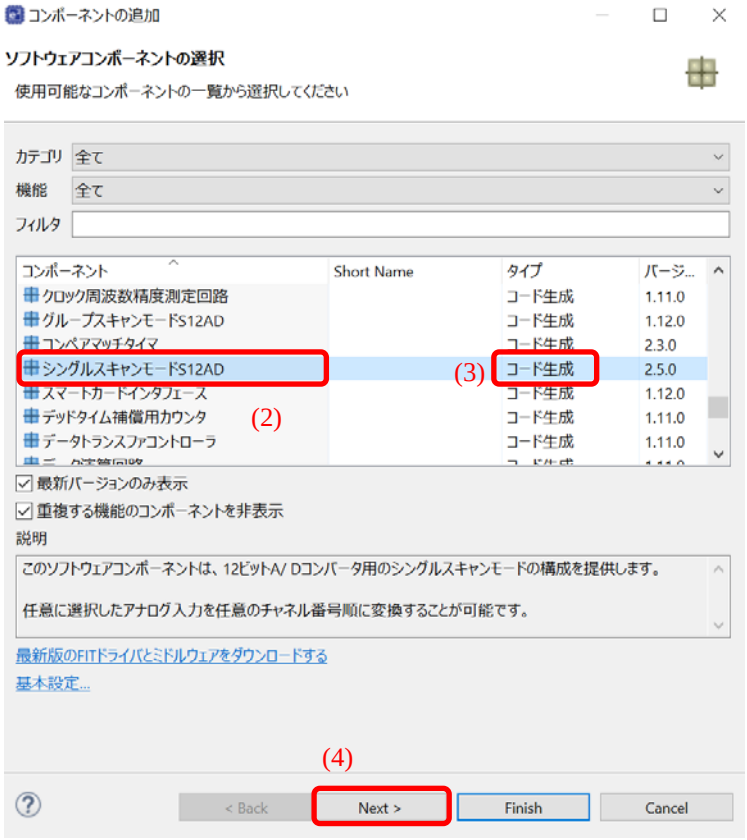

図 4-8 コード生成コンポーネントの追加

(5) ［コンポーネントの追加］ダイアログボックスの［選択したコンポーネントのコンフィグレーションを追加します］ページで、適切なコンフィグレーション名を入力、またはデフォルト名を使用します。（例：Config_S12AD0）

(6) リソースを選択、またはデフォルトのリソースを使用します。（例：S12AD0）

(7) ［終了］をクリックします。

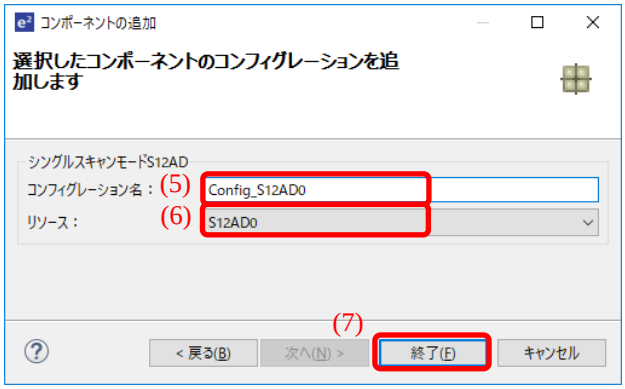

図 4-9 コンポーネントのコンフィグレーション追加

4.4.2　ソフトウェアコンポーネントの削除

プロジェクトからソフトウェアコンポーネントを削除するには、以下の手順で行います。

(1)　コンポーネントツリーからソフトウェアコンポーネントを選択します。

(2)　［コンポーネントの削除］ アイコンをクリックします。

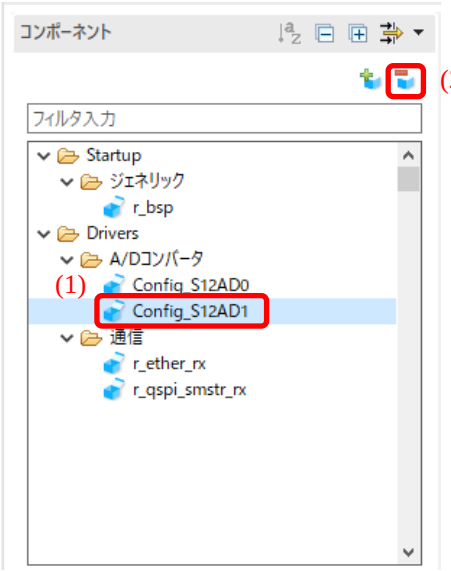

図 4-10 ソフトウェアコンポーネントの削除

コンポーネントツリーから、選択したソフトウェアコンポーネントが削除されます。

[Ctrl]を押しながら、ソフトウェアコンポーネントをクリックすることで複数のソフトウェアコンポーネントが選択できます。

［コンポーネントの削除］ アイコンをクリックすることで、複数のソフトウェアコンポーネントを同時に削除することができます

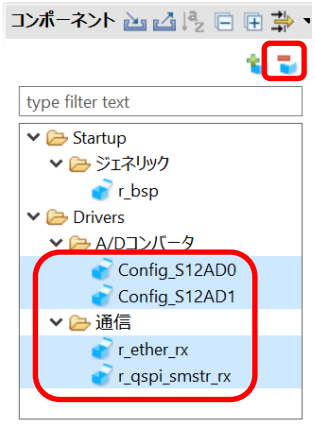

図 4-11 複数のソフトウェアコンポーネントの削除

この操作では、プロジェクトに登録されているソースファイルは削除されません。[コード生成] ボタンでソースファイル出力を行うと、削除したソフトウェアコンポーネントのソースファイルが削除されます。

4.4.3　コンポーネント一覧とハードウェア一覧との切り替え

コンポーネントツリーのノードを直接クリックすることで、新しいコンポーネントの追加をスマート・コンフィグレータがサポートできるようになります。この機能を使用するには、ユーザーはコンポーネント一覧から、ハードウェア一覧表示へコンポーネントツリーを変更する必要があります。

(1) ［表示メニュー］ アイコンをクリックし、［ハードウェア一覧表示］を選択します。コンポーネントツリーは、ハードウェアリソース階層の中にコンポーネントを表示します。

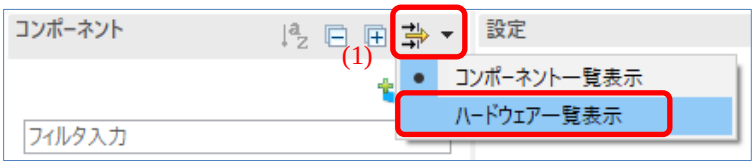

図 4-12 ハードウェア一覧への切り替え

(2) ハードウェアリソースノード(例：12 ビット A/D コンバータ →S12AD1)をダブルクリックし、［コンポーネントの追加］ダイアログボックスを開きます。

(3) リストからコンポーネントを選択し(例：シングルスキャンモード S12AD)、「4.4.1 コード生成コンポーネントの追加方法」の章に記載されている手順で新しい構成を開きます。

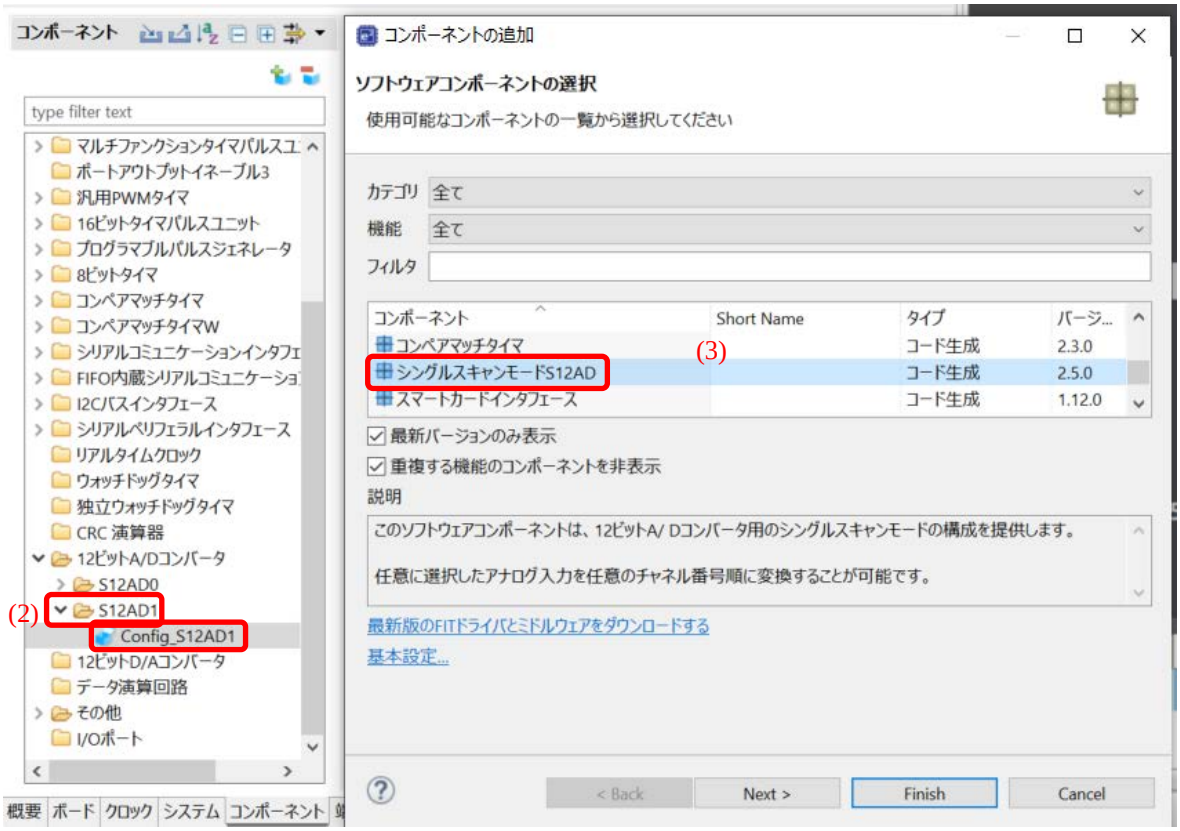

図 4-13 ハードウェア一覧表示の CG コンポーネント追加

4.4.4 CG ドライバの設定

CG コンフィグレーションを設定するには、以下の手順で行います。

(1) コンポーネントツリーにある CG コンフィグレーションをクリックし、選択します。（例：Config_S12AD0）

(2) 右側の設定パネルでドライバを設定します。図は例です。

a. AN000 を選択します。

b. ［変換開始トリガ設定］の項目で、［トリガ入力端子］を選択します。

c. ［詳細設定］をクリックして、表示を広げます。

d. ［チャージ設定］の項目の［ディスチャージ］を選択します。

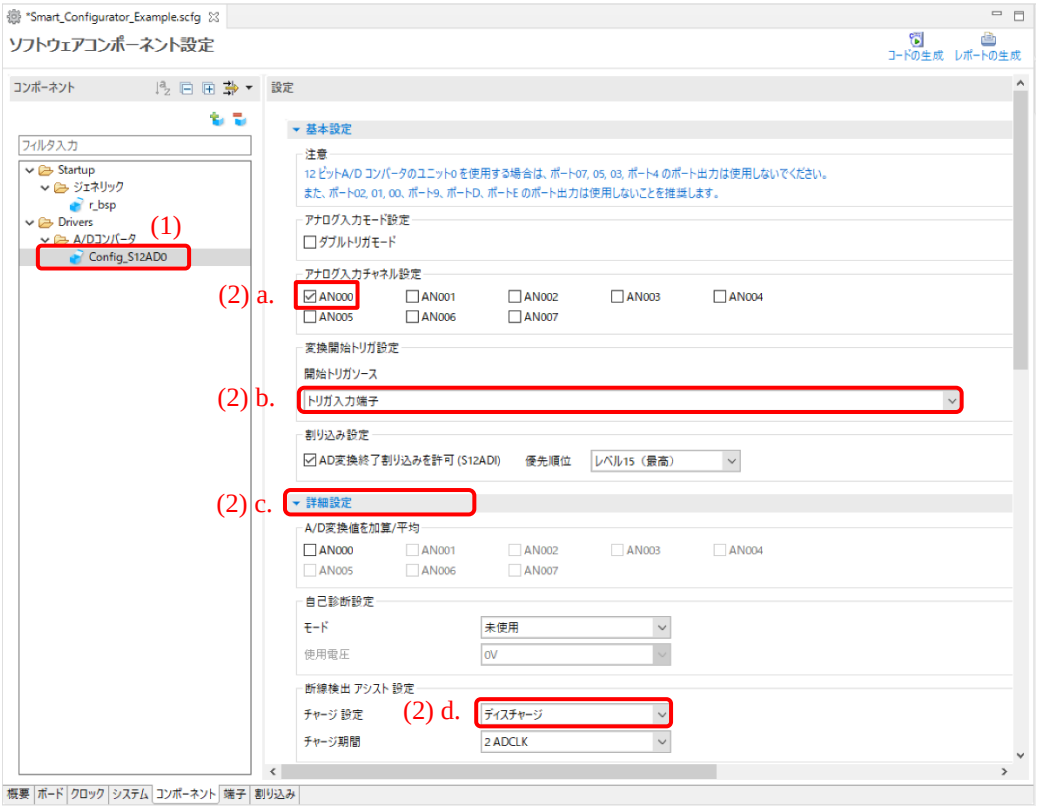

図 4-14 CG ドライバの設定

CG コンフィグレーションのコード生成は、デフォルトで生成する設定になっています。

CG コンフィグレーションを右クリックし、［ ✓ コード生成 ］をクリックすると、［ コード生成 ］に変わりコードを生成しません。

［ コード生成 ］をクリックすると、［ ✓ コード生成 ］に変わりコードを生成します。

4.4.5　CG コンフィグレーションのリソース変更

スマート・コンフィグレータでは、ユーザーは CG コンフィグレーションのリソースを変更することができます（例：S12AD0 から S12AD1 に変更）。互換性のある設定は、現在のリソースから新しく選択したリソースへ移行することができます。

現在のソフトウェアコンポーネント用にリソースを変更するには、以下の手順で行います。

(1) CG コンフィグレーションを右クリックします（例：Config_S12AD0）。

(2) コンテキストメニューから［リソースの変更］を選択します。

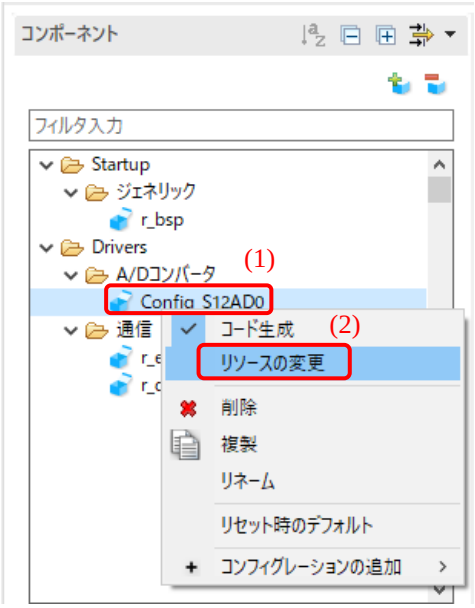

図 4-15 リソースの変更

(3) ［リソースの選択］ダイアログボックスにある新しいリソースを選択します（例：S12AD1）。

(4) ［次へ］ボタンが有効になるので、クリックします。

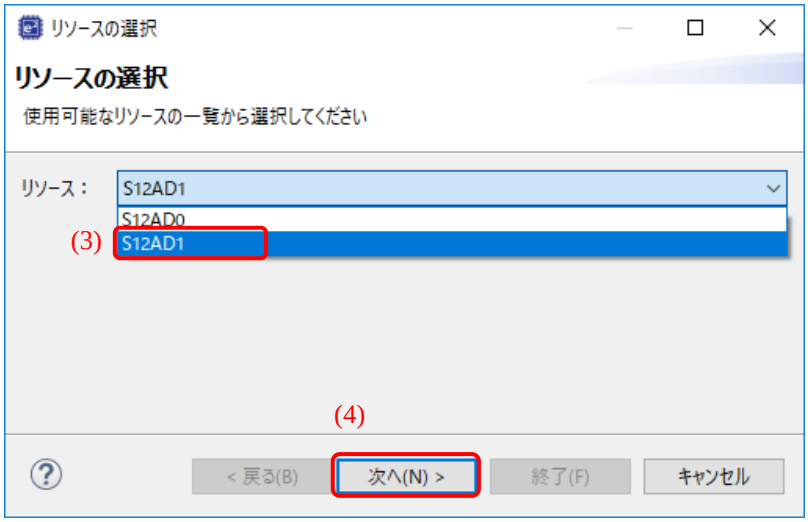

図 4-16 コンポーネントページ-新しいリソースの選択

(5) コンフィグレーション設定は、［コンフィグレーション設定の選択］ダイアログボックスに表示されます。

(6) 設定が変更可能であるかを確認します。

(7) テーブル内の設定を使用するか、デフォルト設定を使用するか選択します。

(8) ［終了］をクリックします。

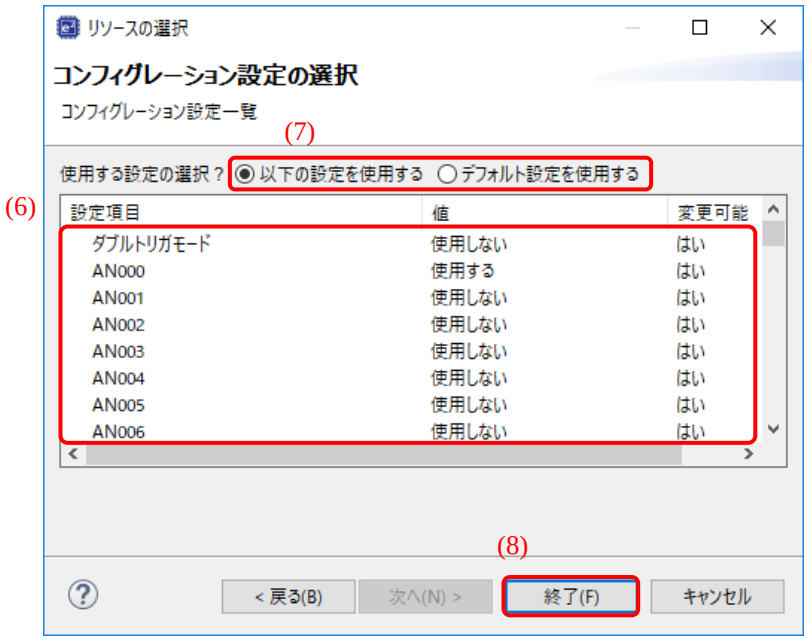

図 4-17 新しいリソース設定の確認

リソースは、自動的に更新されます。（例：S12ADI0 から S12ADI1）

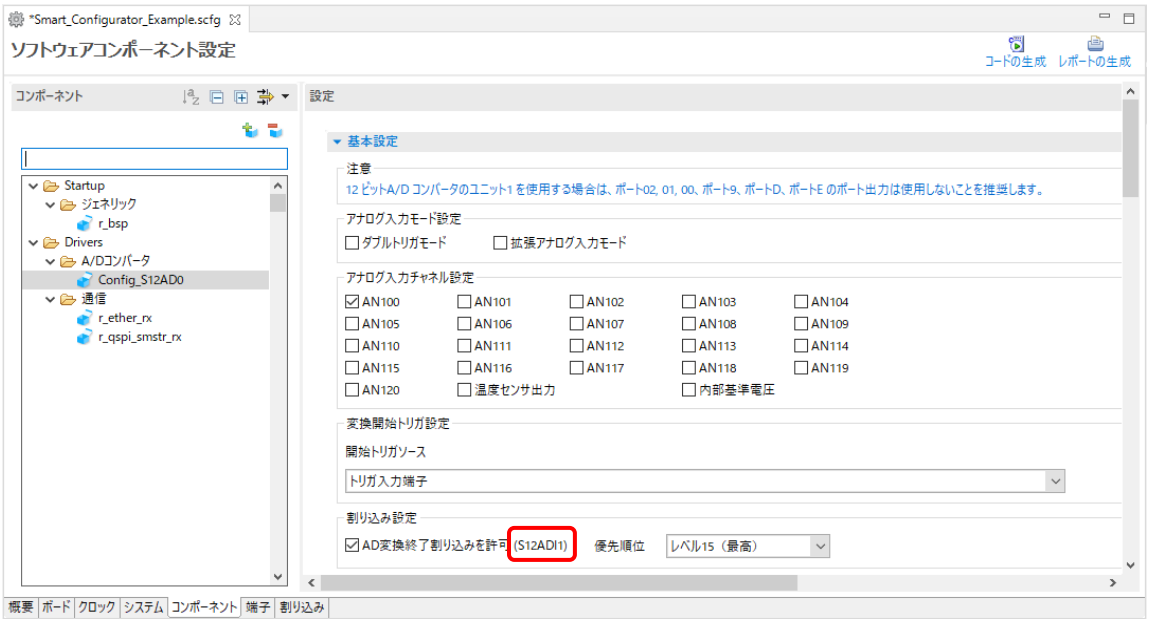

図 4-18　自動的に更新されるリソース

コンフィグレーション名を変更する場合は、以下の手順で行います。

(9) CG コンフィグレーションを右クリックします。

(10) ［リネーム］を選択して、コンフィグレーションに再度名前をつけます（例：Config_S12AD0 を Config_S12AD1）。

図 4-19 コンフィグレーションに再度名前をつける

4.4.6　FIT モジュールのダウンロード

FIT ドライバまたはミドルウェアは、ルネサスエレクトロニクスホームページからダウンロードする必要があります。

(1) ［コンポーネントの追加］ アイコンをクリックします。

(2) ［コンポーネントの追加］ダイアログの［ソフトウェアコンポーネントの選択］ページの［他のソフトウェアコンポーネントをダウンロードする］リンクをクリックし、FIT モジュールをダウンロードします。

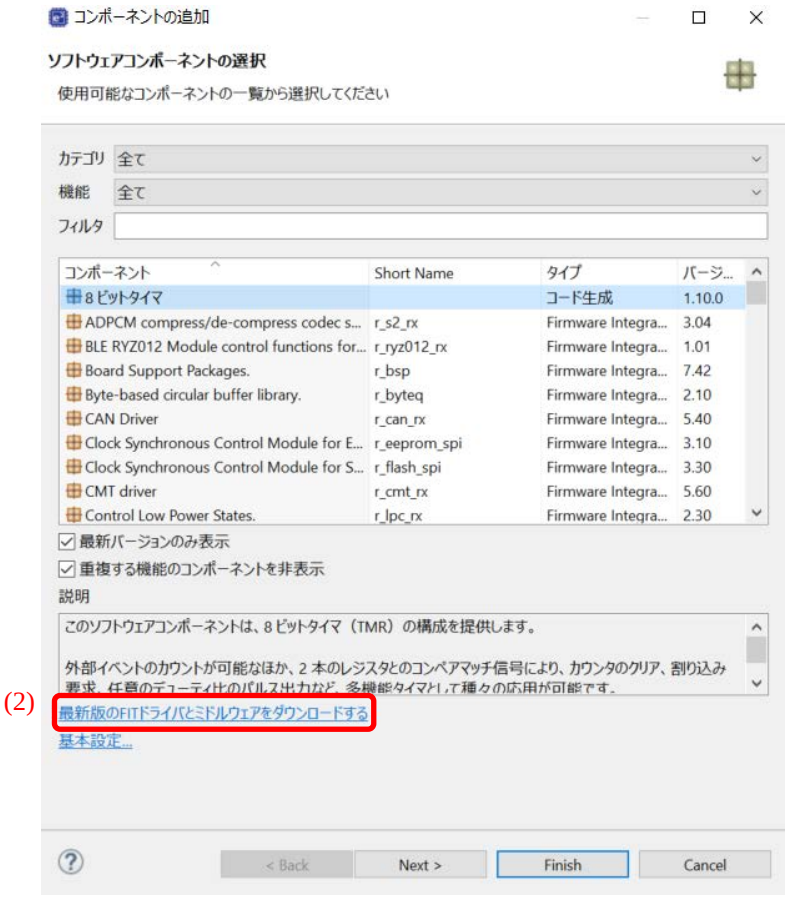

図 4-20 他のソフトウェアコンポーネントのダウンロード

【注】ダウンロードには、“My Renesas”へのログインが必要です。ログインされていない場合は、以下のダイアログが表示されますので、ログインを行ってください。［My Renesas について］ボタンをクリックすると、新しいユーザー登録が可能になります。

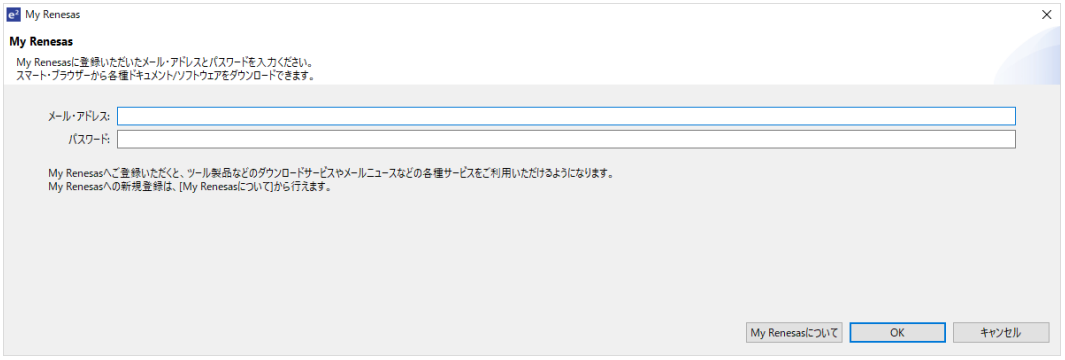

図 4-21 My Renesas へのログイン

(3) ［FIT モジュールのダウンロード］ダイアログボックスで、必要なモジュールのチェックボックスにチェックを入れてください。［RX Driver Package のみ表示する］のチェックを外すとモジュールの絞り込みが解除されます。

(4) ［参照...］をクリックしてダウンロードする場所を指定します。

(5) ［ダウンロード］をクリックして、選択したモジュールのダウンロードを開始します。

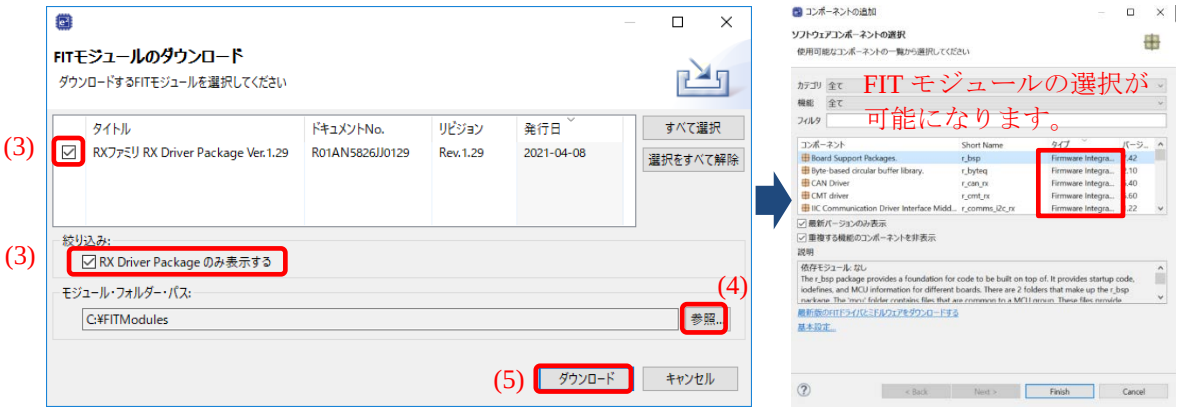

図 4-22 FIT モジュールのダウンロード

4.4.7　FIT ドライバまたはミドルウェアの追加方法

FIT ドライバまたはミドルウェアは、以下の手順で追加してください。

(1) ［コンポーネントの追加］ アイコンをクリックします。

(2) ［コンポーネントの追加］ダイアログボックスの［ソフトウェアコンポーネントの選択］ページのリストからコンポーネントを選択します（例：r_ether_rx と r_qspi_smstr_rx）。Ctrl を押しながらの選択で、複数コンポーネントの選択が可能です。

(3) ［タイプ］は［Firmware Integration Technology］であることを確認してください。

(4) ［終了］をクリックします。

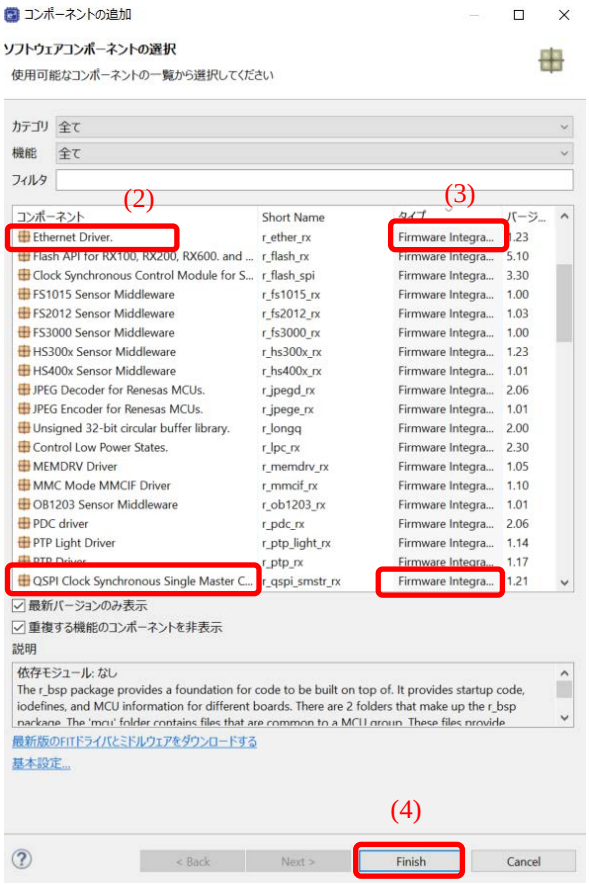

図 4-23 FIT モジュールの追加

4.4.8　FIT サンプルプロジェクトのダウンロードとインポート

FIT ドライバまたはミドルウェアのアイコンが［ ］表示の場合に、サンプルプロジェクトをダウンロードできます。

(1) ［ ］表示の FIT ドライバまたはミドルウェアを選択して、右メニューから［サンプルプロジェクトのダウンロード］を選択します。(例：CMT)

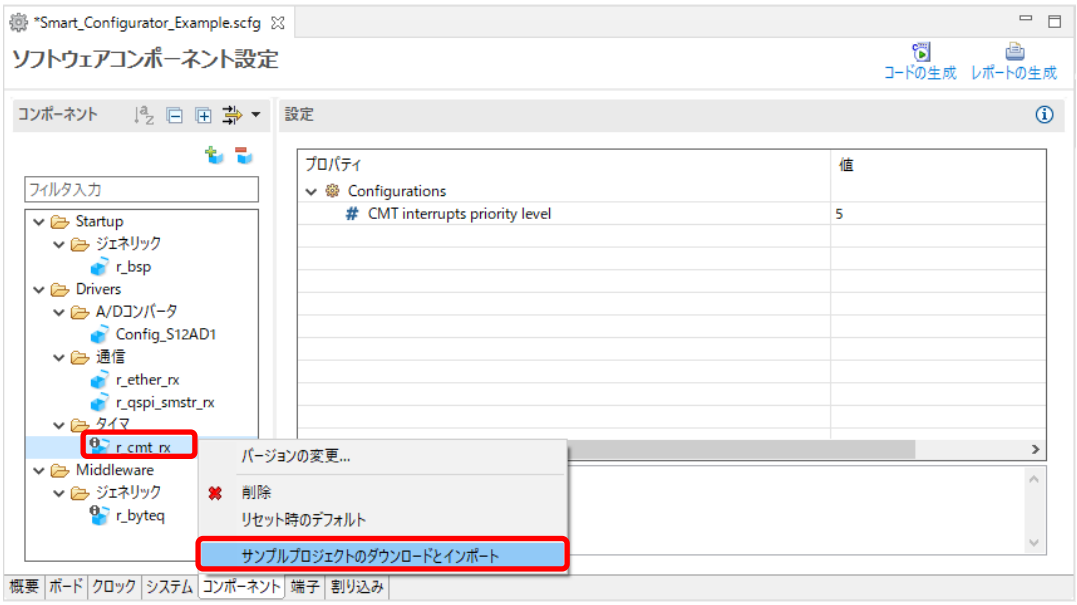

図 4-24 サンプルプロジェクトのダウンロードとインポート

(2) スマート・ブラウザーの［アプリケーション・ノート］タブにサンプル・コードが表示されますので、右メニューの［サンプル・コード（プロジェクトのインポート）］を選択します。

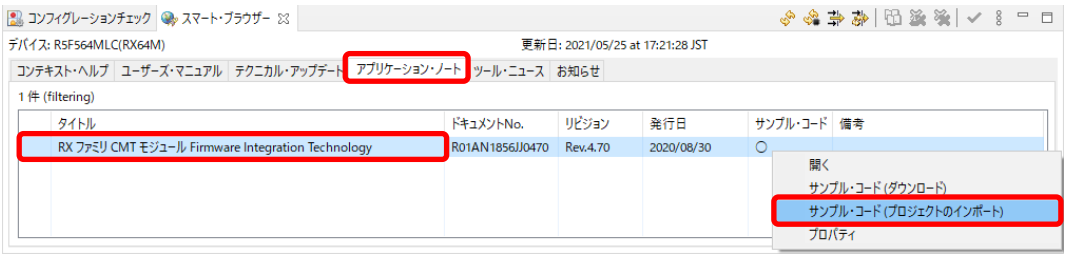

図 4-25 サンプル・コード（プロジェクトのインポート）

(3) サンプル・コードの保存先を指定して、［保存］をクリックします。

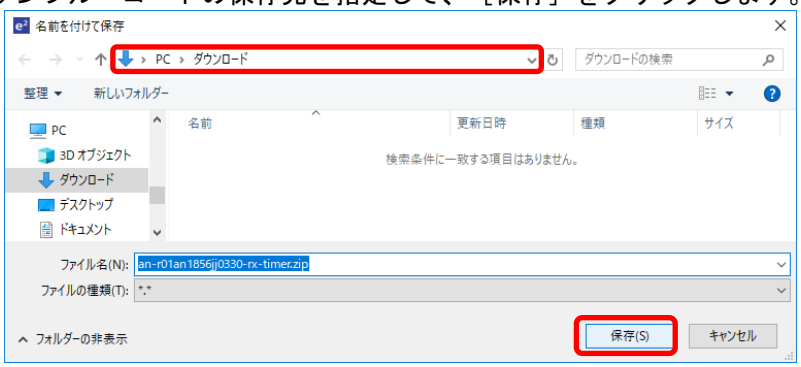

図 4-26 サンプル・コードの保存先

(4) サンプル・コードの免責事項が表示されますので［同意する］をクリックします。

図 4-27 サンプル・コードの免責事項

(5) ［インポートするパッケージの選択］ダイアログが表示されない場合は、手順（6）に進んでください。［インポートするパッケージの選択］ダイアログが表示された場合は、インポートするパッケージを選択して、［OK］をクリックします。

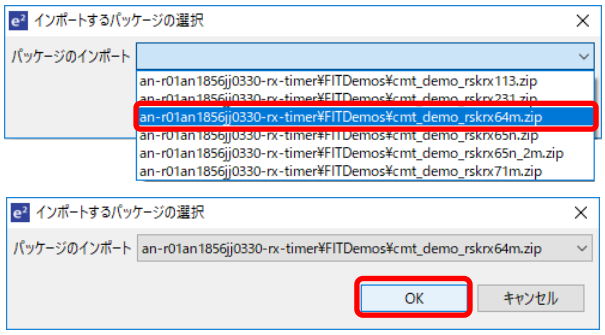

図 4-28 インポートするパッケージの選択

(6) プロジェクトの［インポート］ダイアログが表示されますので、プロジェクトを選択して［終了］ボタンをクリックします。

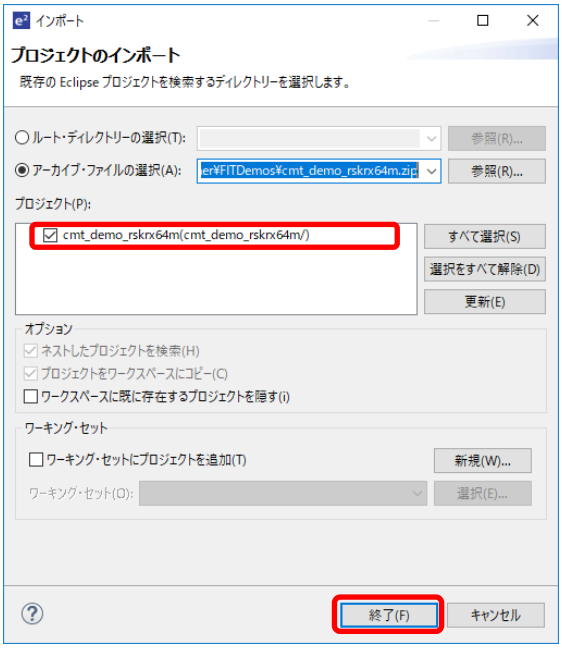

図 4-29 プロジェクトのインポート

(7) ［プロジェクト・エクスプローラー］に追加され、サンプルプロジェクトのインポートは完了です。

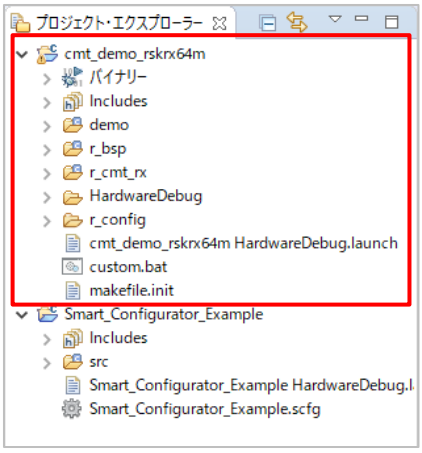

図 4-30 プロジェクト・エクスプローラーへの追加

4.4.9　FIT ソフトウェアコンポーネントの設定

FIT ドライバまたはミドルウェアは、コンフィグレーションオプションを設定して使用します。設定方法は、コンポーネントにより異なります。

- 設定パネルの設定をコード生成の実行により、FIT モジュール用コンフィグレーションファイルに設定を自動的に生成する。
- FIT モジュール用コンフィグレーションファイルを手動で設定する。

FIT モジュール用コンフィグレーションファイルは、ソースコード出力後の r_config フォルダに生成されます。コンフィグレーションオプションの設定については、「7.1 Firmware Integration Technology(FIT)の場合のカスタムコード追加方法」を参照してください。

また、一部のコンポーネントは、設定パネルで端子設定を提供します。 以下は、設定パネルでの端子設定例です。

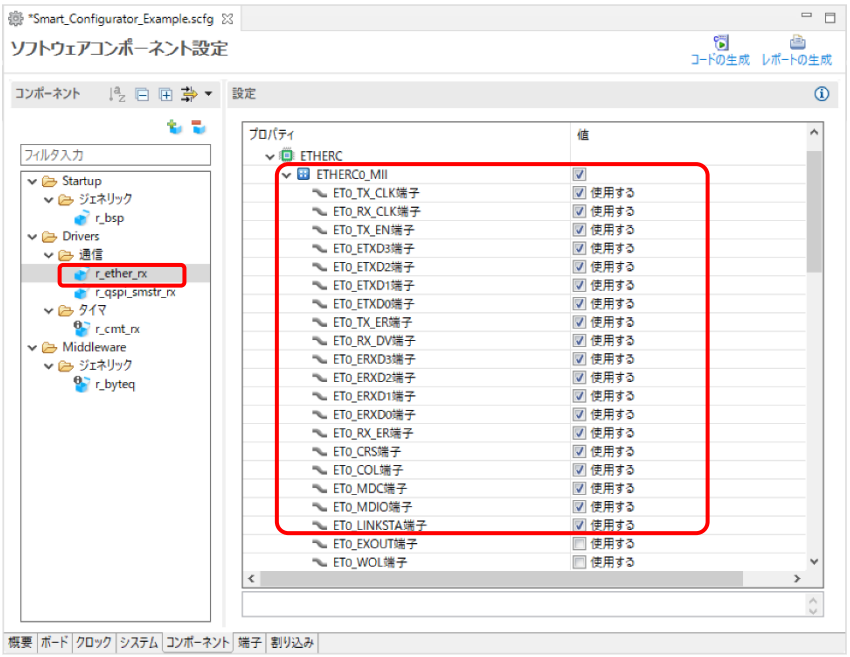

図　4-31 r_ether_rx の端子設定

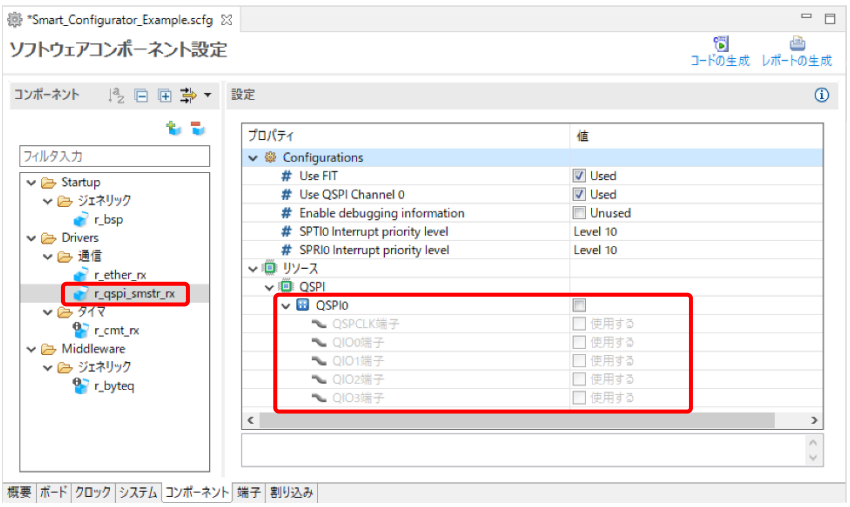

図　4-32 r_qspi_smstr_rx の端子設定

4.4.10　FIT ソフトウェアコンポーネントのバージョン変更

プロジェクト内の FIT ソフトウェアコンポーネントを異なるバージョンに変更するには、以下の手順で行います。

(1) コンポーネントツリーから、バージョンを変更したい FIT ソフトウェアコンポーネントを右クリックします。

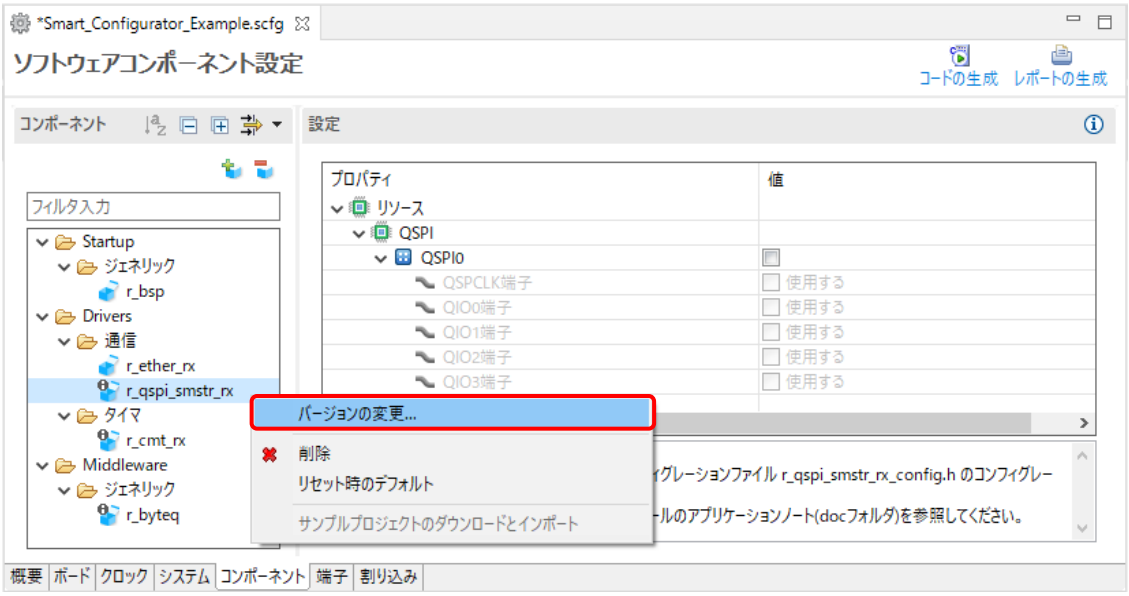

図 4-33 FIT ソフトウェアコンポーネントのバージョン変更

(2) コンテキストメニューから［バージョンの変更...］を選択します。

(3) ［バージョンの変更］ダイアログボックスで変更したいバージョンを選択します。デバイスが対応していないバージョンを選択した場合、［選択されたバージョンはターゲット・デバイスまたはツール・チェーンをサポートしていません。］と表示されますので、対応しているバージョンを選択してください。

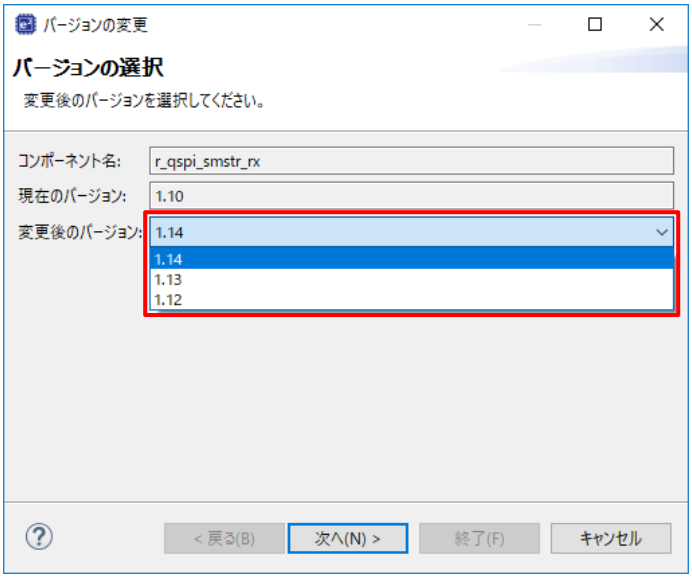

図 4-34 FIT ソフトウェアコンポーネントのバージョン選択

(4) ［次へ］が有効になるので、クリックします。

(5) バージョン変更により、変更する設定項目の一覧が表示されますので、問題ないことを確認し、［終了］をクリックします。

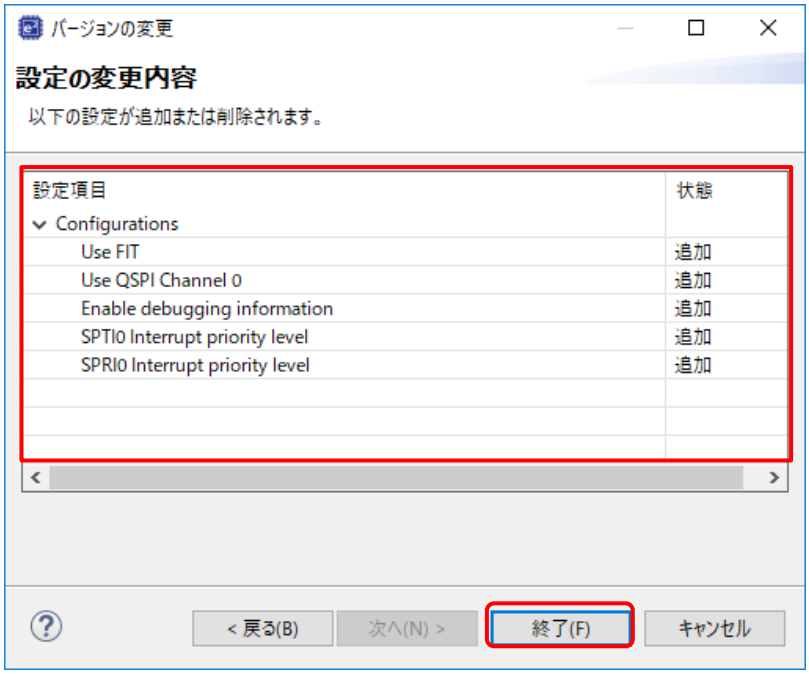

図 4-35 設定変更項目の確認

(6) ［バージョンを変更し、コードを生成しますか。］と表示されますので、問題なければ［はい］をクリックします。

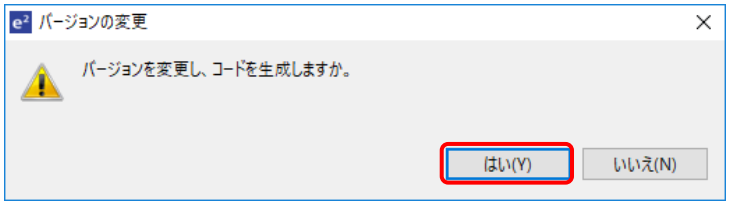

図 4-36 バージョンの変更確認

(7) FIT ソフトウェアコンポーネントのバージョンが変更され、自動的にコード生成が実行されます。

4.4.11　グレーアウト・コンポーネントの更新

コンポーネントのバージョンが利用できない場合、グレーアウトで表示されます。 グレーアウト表示のコンポーネントを利用できるように更新するには、以下の手順で行います。

(1) コンポーネントツリーからグレー表示されたコンポーネントを右クリックして、［バージョンの変更…］を選択します。利用可能なバージョンに変更するには、「4.3.10 FIT ソフトウェアコンポーネントのバージョン変更」の章を参照してください。

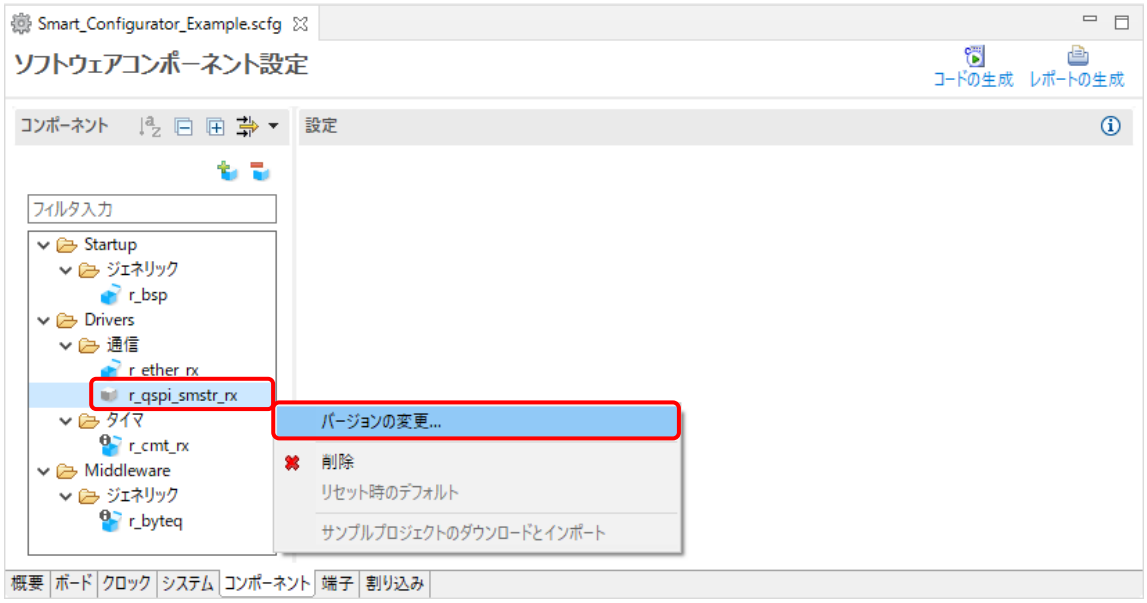

図 4-37 グレーアウト・コンポーネントのバージョン変更

(2) このコンポーネントに使用できるバージョンがない場合は、「4.4.6 FIT モジュールのダウンロード」の章を参照して、このコンポーネントをルネサスエレクトロニクスホームページからダウンロードしてください。

4.4.12　RTOS カーネルの設定

FreeRTOS カーネルを設定するには、以下の手順で行います。

Renesas FreeRTOS については、Renesas FreeRTOS の関連ドキュメントを参照してください。

(1)　コンポーネントツリーの［FreeRTOS_Kernel］を選択します。

(2)　［設定］パネルに RTOS カーネルに対応したパラメータが表示され、コンフィグレーション設定を変更することができます。

(3)　［設定］パネルで選択したパラメータの説明と対応するマクロ定義が表示されます。

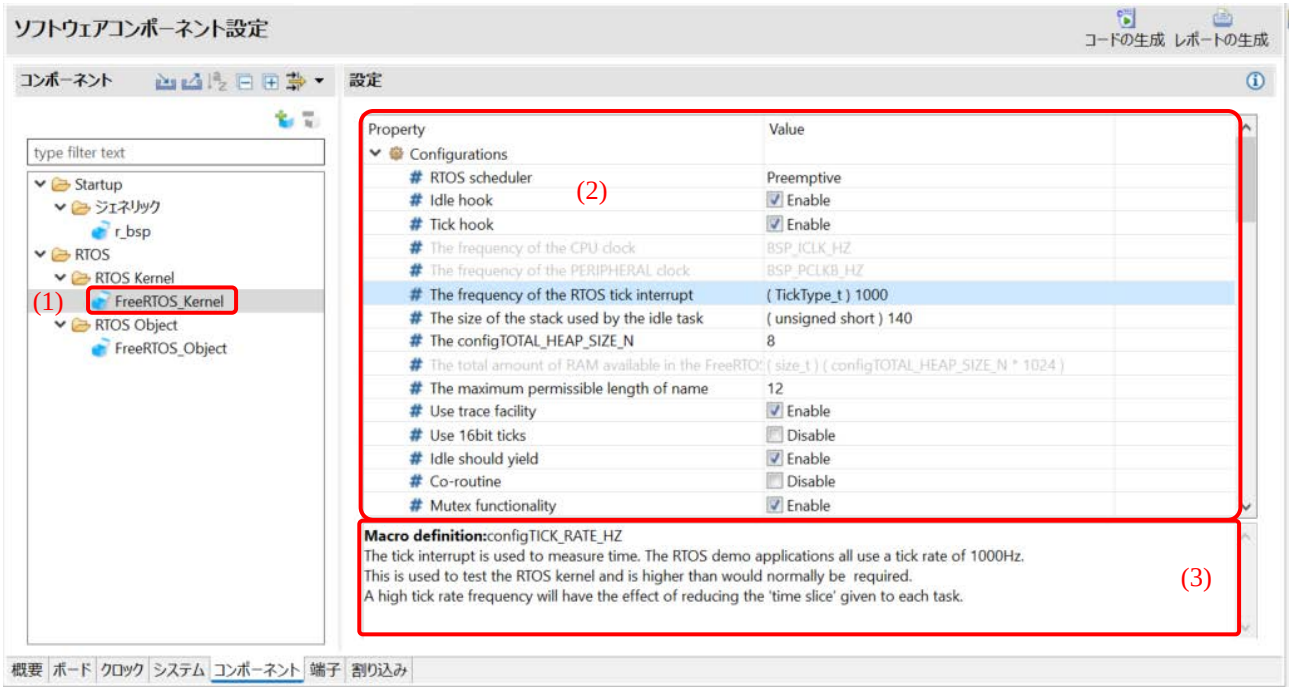

図 4-38 FreeRTOS_Kernel の設定

4.4.13　RTOS オブジェクトの設定

FreeRTOS オブジェクトを設定するには、以下の手順で行います。

(1) コンポーネントツリーの［RTOS_Libary］フォルダ下のライブラリを選択します。

【注】RTOS ライブラリは、FreeRTOS (with IoT libraries) プロジェクトでのみ使用できます。

(2) ［設定］パネルに RTOS ライブラリに対応した構成タブが表示されます。作成するオブジェクトに対応するタブを選択します。

(3) 新しいオブジェクトを作成する場合は［＋］ボタンをクリックして、オブジェクトを削除する場合は、［－］ボタンをクリックします。

(4) テキストボックスとコンボボックスを編集して、オブジェクトを設定します。

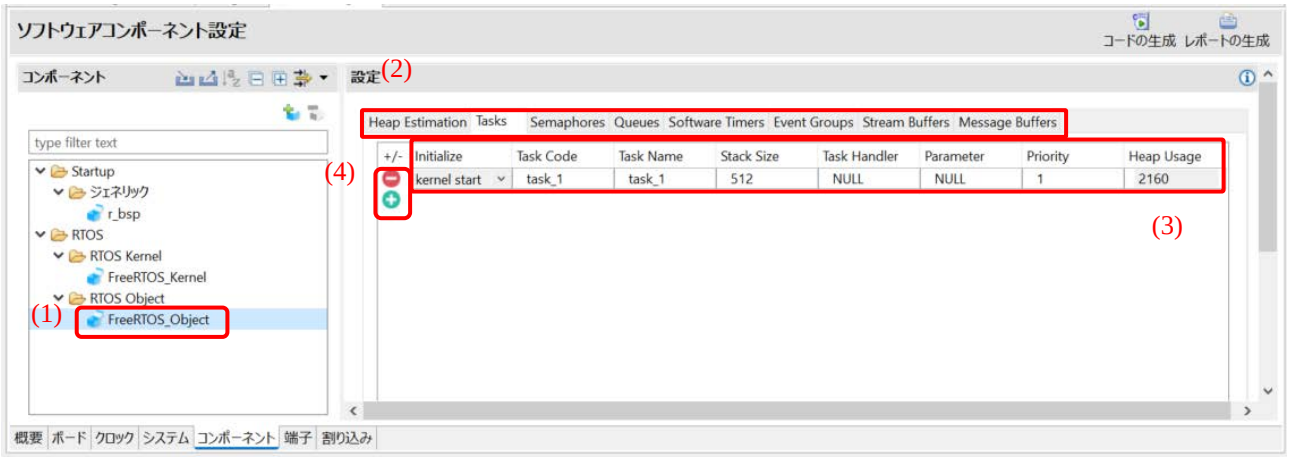

図 4-39 FreeRTOS_Object の設定

4.4.14　RTOS ライブラリの設定

FreeRTOS ライブラリを設定するには、以下の手順で行います。

(1) コンポーネントツリーの［FreeRTOS_Object］を選択します。

(2) ［設定］パネルに RTOS ライブラリに対応したパラメータが表示され、コンフィグレーション設定を変更することができます。

(3) ［設定］パネルで選択したパラメータの説明と対応するマクロ定義が表示されます。

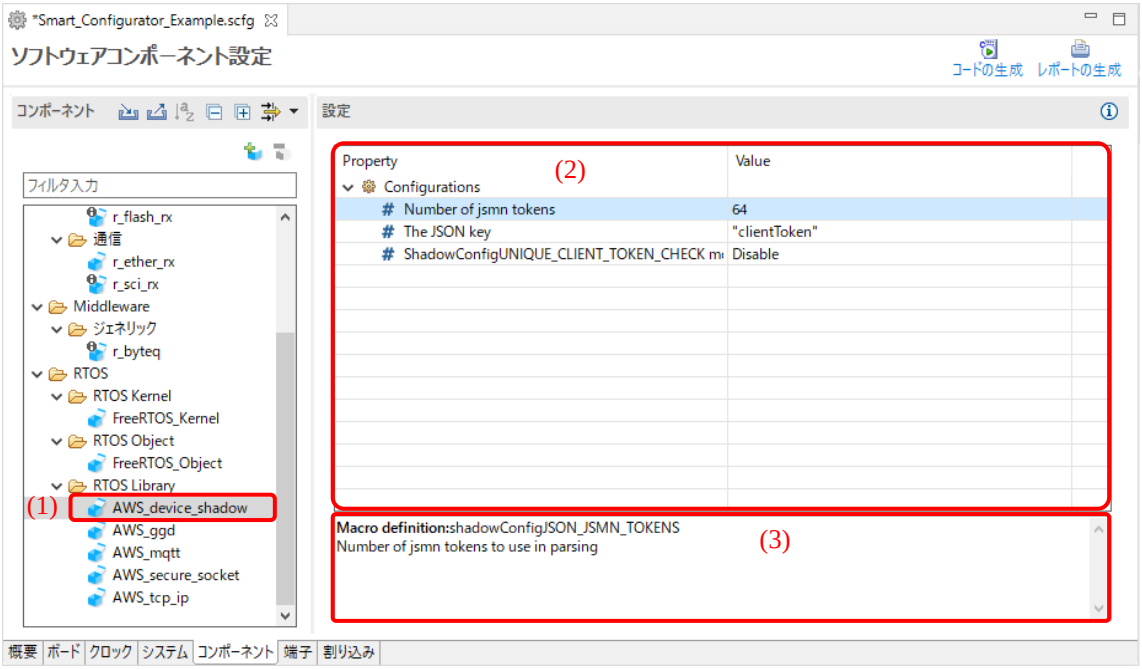

図 4-40 FreeRTOS_Libary の設定

4.4.15　アナログフロントエンドコンポーネントの設定

RX23E-A グループのマイクロコントローラには、温度、圧力、流量および重量をキャリブレーションなしで 0.1%未満の精度で測定できるアナログフロントエンド（AFE）が装備されているため、高精度のセンシング、テストおよび測定機器に最適です。

RX23E-A のプロジェクトを作成すると、AFE コンフィグレーションツールを以下の目的で使用できます。

- GUI で AFE を簡単に設定
- 端子競合を簡単にチェック
- アナログマルチプレクサ接続を簡単にチェック

この章では、アナログマルチプレクサ接続の使用方法について説明します。

(1) スマート・コフィグレータを開き、［コンポーネント］ページで［アナログフロントエンド］と［シングルスキャンモード DSAD］を追加します。

(2) コンポーネントツリーの［Config_DSAD0］を選択し、以下のように設定します。

a. ［アナログ入力チャネル設定］のチャネル 0 を有効にします

b. ［チャネル設定］のチャネル 0 の＋側入力信号を AIN1、－側入力信号を AIN3 にします。

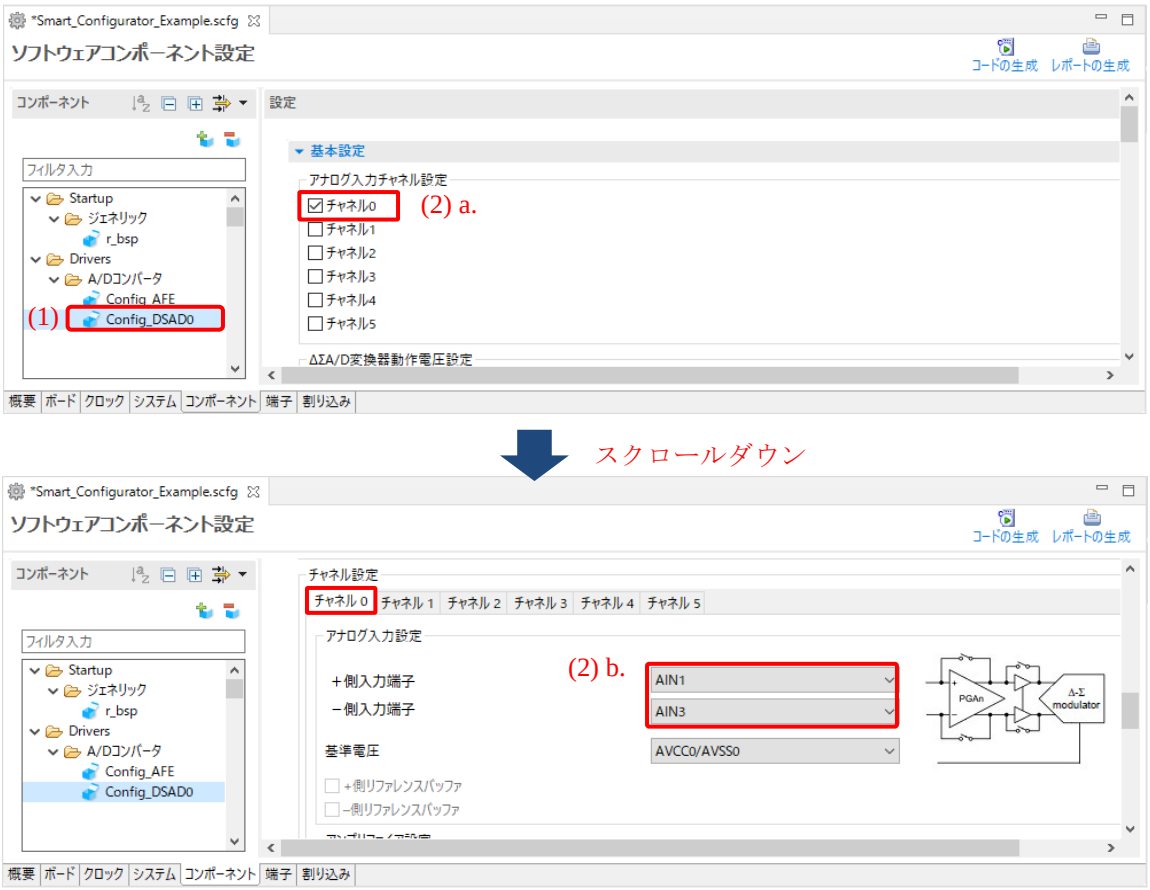

図 4-41 Config_DSAD0 の設定

(3) コンポーネントツリーの［Config_AFE］を選択し、以下のように設定します。

a. ［バイアス出力設定］を有効にします。

b. AIN1 端子出力、AIN3 端子出力を有効にします。

図 4-42 Config_AFE の設定

(4) ［アナログ端子接続］を選択し、AFE マルチプレクサ端子接続のブロック図を表示します。
アナログマルチプレクサのアクティブな接続が強調表示され、接続と構成を簡単に確認できます。

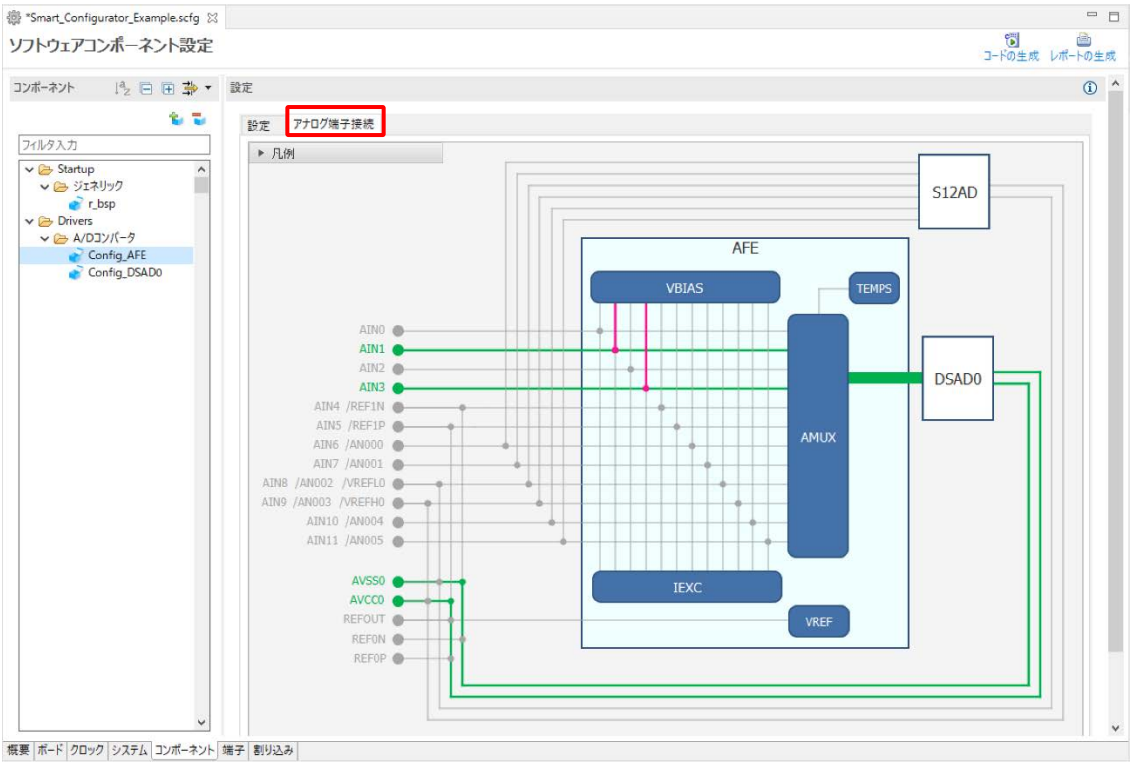

図 4-43 AFE マルチプレクサ端子接続のブロック図

4.4.16　モータコンポーネントの設定

モータドライバジェネレータは、1 つの GUI 設定からモータ制御に使用されるすべての周辺機能のドライバを生成するユーティリティツールです。

【注】サポートしているデバイスは、RX13T, RX23T, RX24T, RX24U, RX26T, RX66T, RX72T, RX72M です。

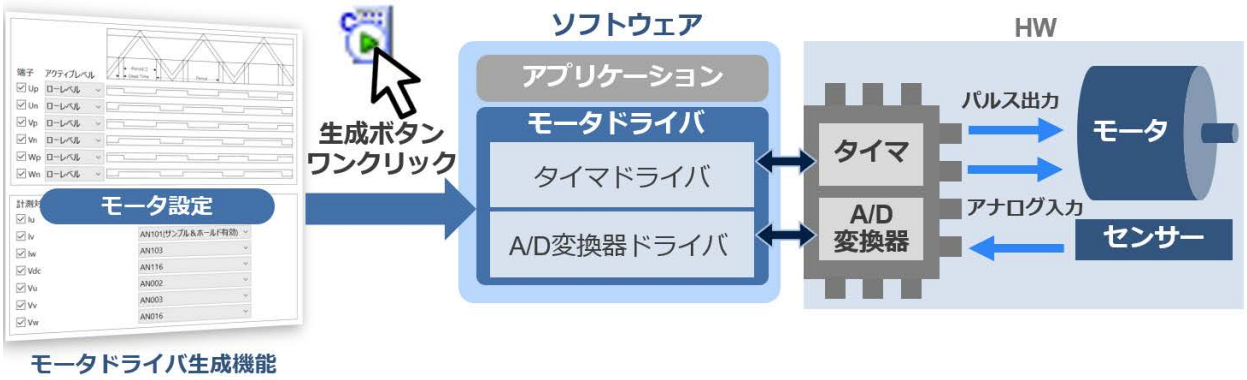

図 4-44 モータドライバジェネレータ

この章では、モータドライバジェネレータの使用方法について説明します。

(1) サポートされているデバイス（例：RX24T）のプロジェクトで、スマート・コンフィグレータの［コンポーネント］ページを開き、新しいコンポーネント［モータ］を追加します。
［新規コンポーネント］ダイアログで、必要に応じてモータ種別を選択し、［終了］ボタンをクリックします。

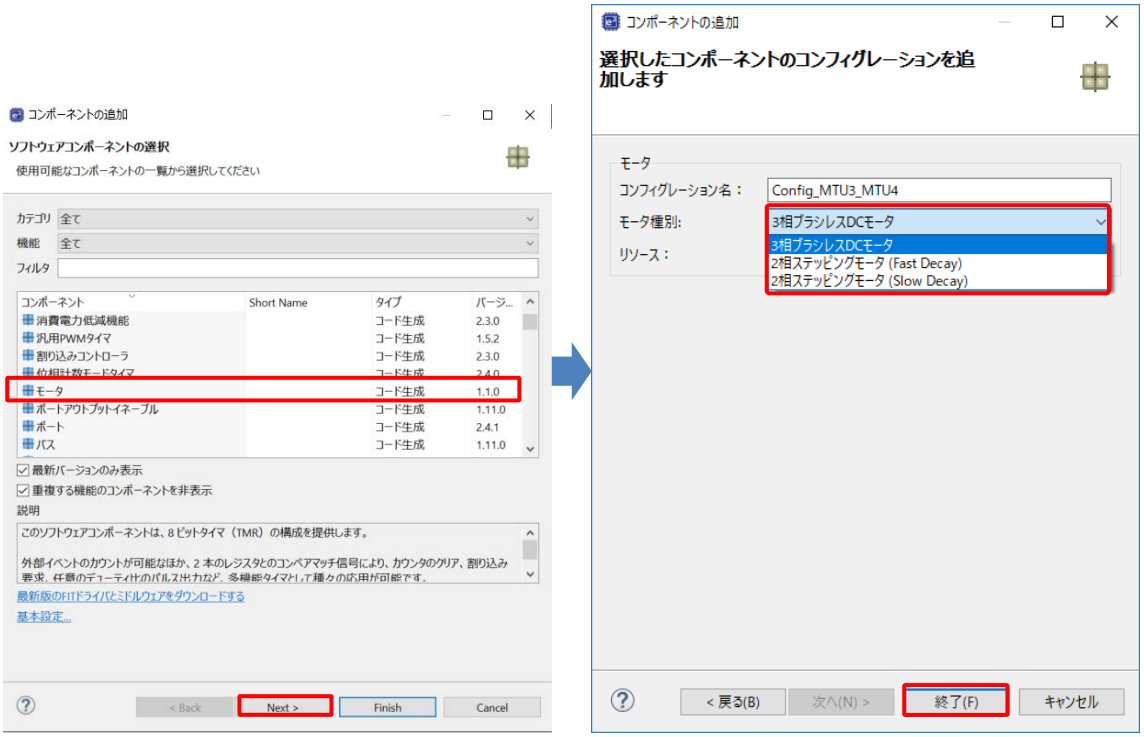

図 4-45 AFE マルチプレクサ端子接続のブロック図

(2) コンポーネントツリーの［Config_MTU3_MTU4］を選択し、［設定］パネルの［タイマ設定］タブを開きます。このタブでは、周期設定、出力レベル設定、出力端子設定、タイマ割り込み設定などのタイマドライバ用の設定が行えます。

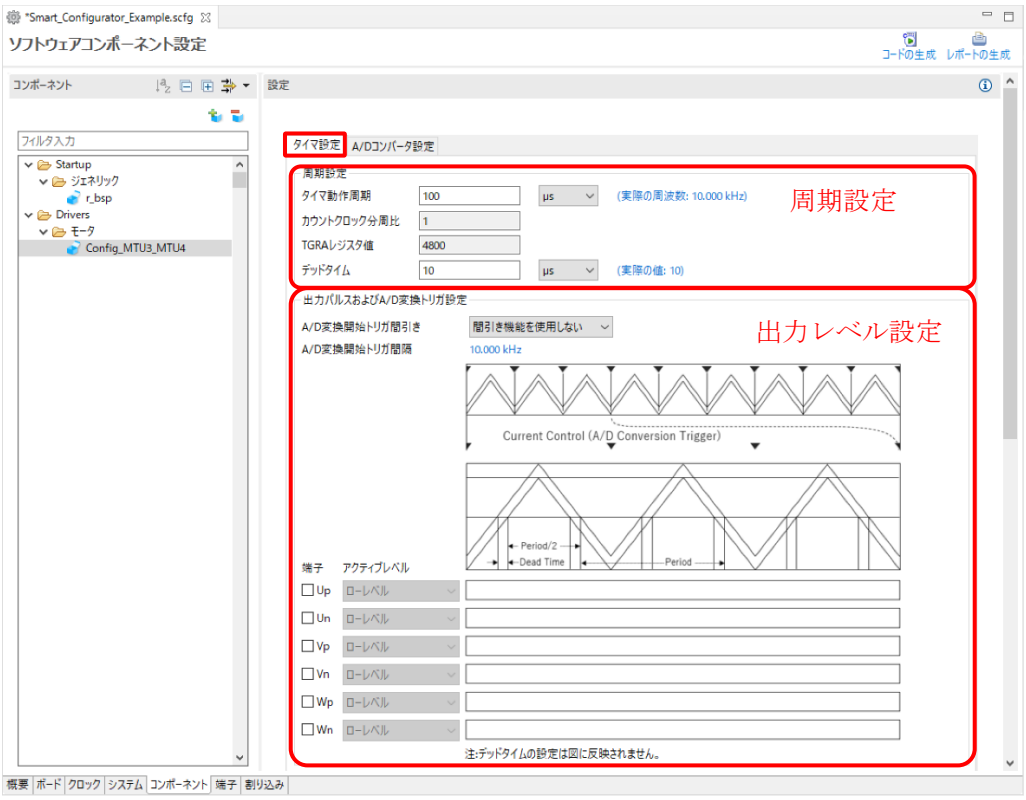

図 4-46 タイマ設定(1)

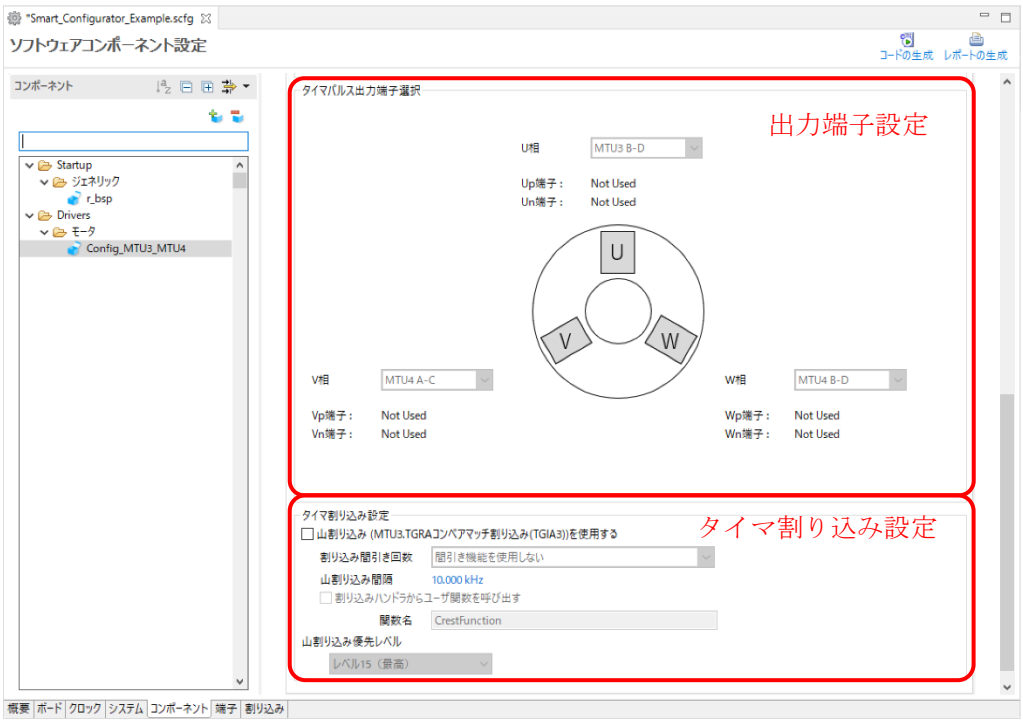

図 4-47 タイマ設定(2)

(3) ［A/D コンバータ設定］では、アナログ端子設定、A/D 割り込み設定などの A/D コンバータドライバ用の設定が行えます。

図 4-48 A/D コンバータ設定

(4) GPT は、一部のデバイスでサポートされています。(例：RX26T). 新しいコンポーネントを追加するとき、コンポーネントの追加ダイアログのリソースに、[Triangle_GPT], [GPT0_GPT1_GPT2], [GPT4_GPT5_GPT6]のいずれかを選択し、[完了]ボタンをクリックします。
【注】[GPT0_GPT1_GPT2]および[GPT4_GPT5_GPT6]は、相補 PWM モードを対象としており、RX26T のみで選択可能です。
[Triangle_GPT]は、三角波 PWM モードを対象としており、マスタとスレーブのチャネル構成として、GPT のチャネルを自由に変更することができます。専用に設計されており、RX23T のみで選択可能です。[Triangle_GPT]は、RX24T, RX24U, RX26T, RX66T, RX72M, RX72T で利用可能です。

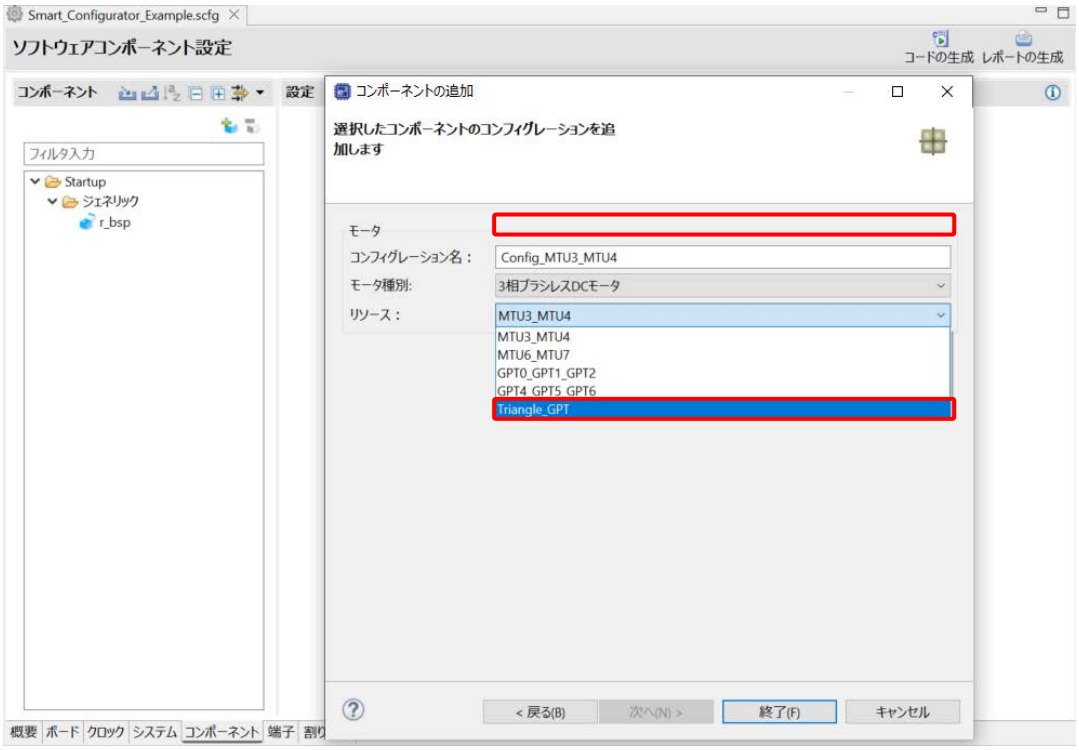

図 4-49 モータリソースの選択

(5) GPT リソースの設定は、MTU リソースの設定といくつかの違いがあります。

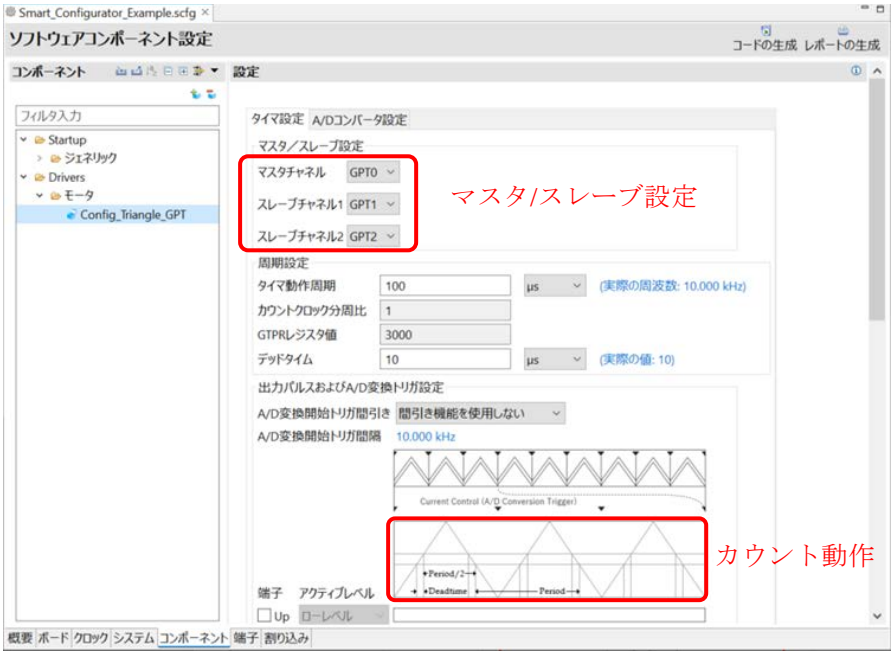

図 4-50 タイマ設定(1)

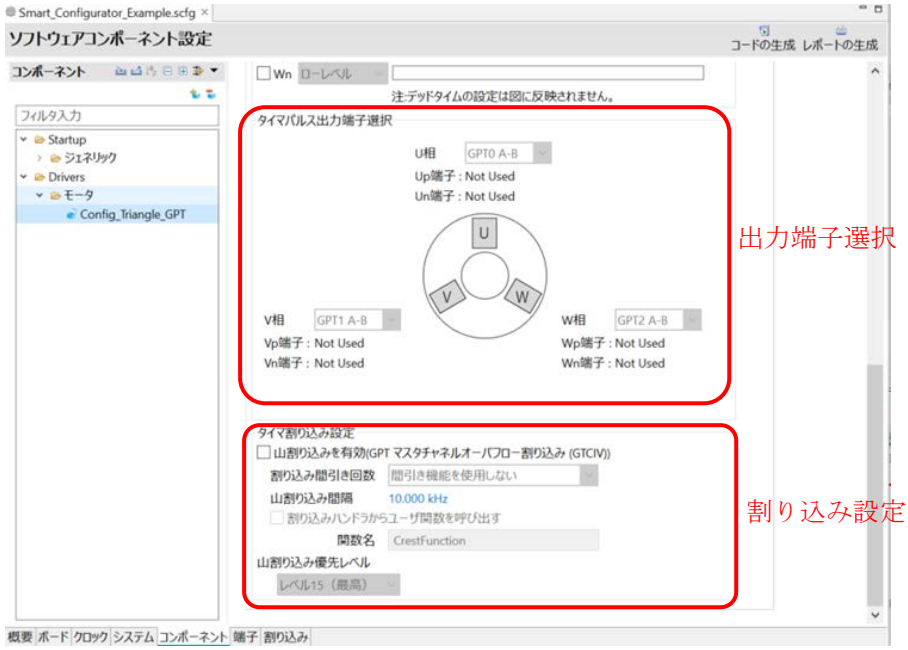

図 4-51 タイマ設定(2)

4.4.17　コンポーネントの基本設定

モジュールの保存先、依存関係などのコンポーネントの基本設定を変更できます。変更するには、［コンポーネントの追加］ダイアログ（**図 4-8 コード生成コンポーネントの追加**）に表示される［ソフトウェアコンポーネントの選択］ページの［基本設定］リンクをクリックし、［設定］ダイアログを表示させます。

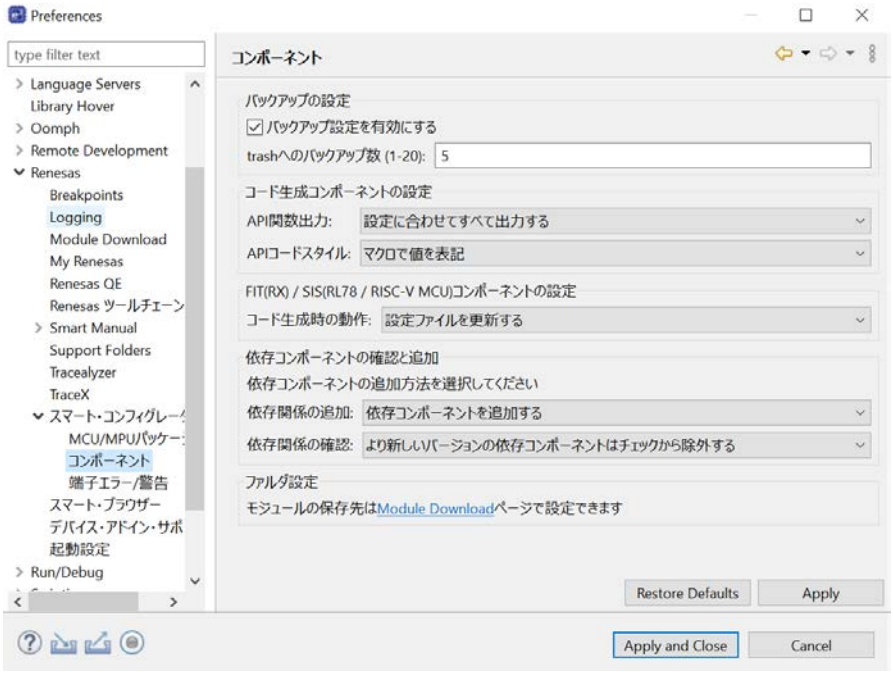

図 4-52 コンポーネント基本設定

【注】 1. モジュールのバージョンとその依存関係が不一致の場合に、警告メッセージ W04020011 を表示します。モジュールとその依存関係の改訂履歴を確認し、使用しているモジュールに変更が不要な場合は、この警告を無視してかまいません。この警告を消すには、コンポーネント基本設定の［依存関係の確認］リストボックスで「依存コポーネントのバージョンをチェックしない」を選択し、［OK］をクリックします。

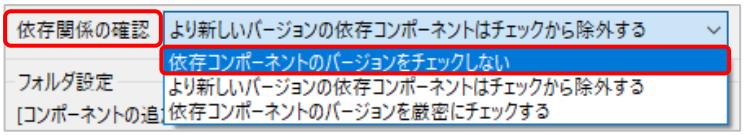

図 4-53 ［依存関係の確認］の変更

2. FIT モジュールを直接 Web サイトからダウロードした場合は、ダウンロードした zip ファイルを展開し、FIT Modules フォルダにある xml ファイルと zip ファイルを［Module Download］ー［保存先（RX）］のフォルダにコピーしてください。保存先を変更するには、［参照］をクリックし、他のフォルダを選択してください。

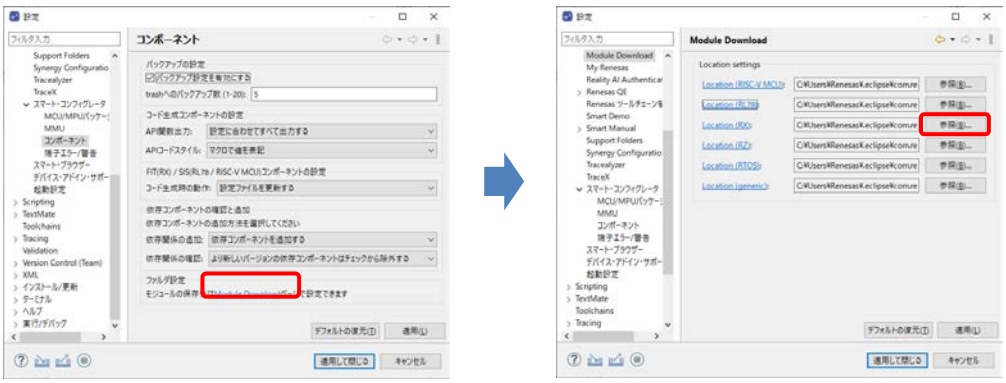

図 4-54 ［保存先 (RX)］の変更

4.5　端子設定

端子ページは、端子機能の割り当てに使用します。周辺機能別に端子機能を表示する［端子機能］リストと、端子番号順に全ての端子を表示する［端子番号］リストの 2 つの表示があり、タブを切り替えることで切り替えることができます。

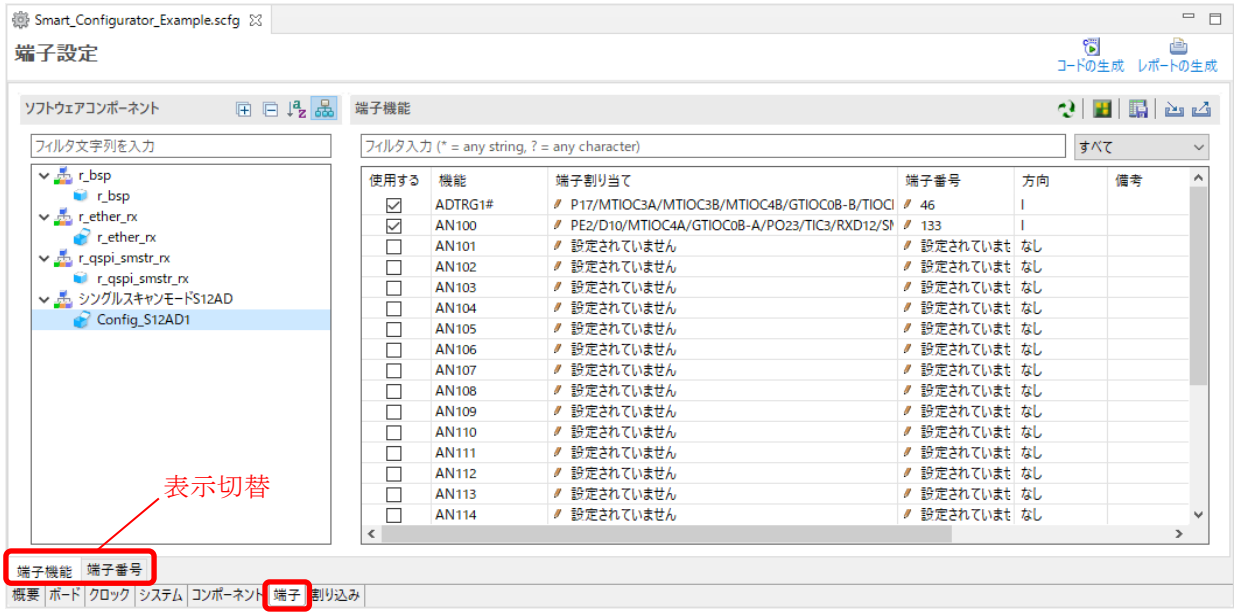

図 4-55 端子ページ（端子機能）

［ボード］ページでボードを選択すると、［デフォルト機能］には、ボードの初期端子設定情報を表示します。また、［機能］の選択リストに表示される［■］アイコンはボードの初期端子機能を示します。

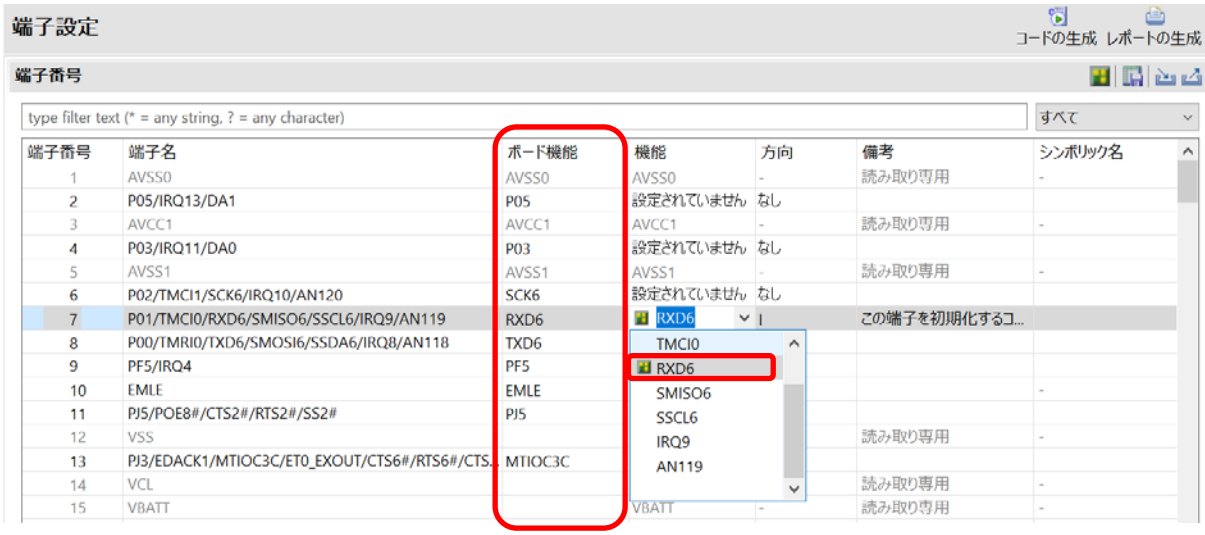

図 4-56 端子ページ（端子番号）

4.5.1　ソフトウェアコンポーネントの端子割り当て変更

スマート・コンフィグレータは、プロジェクトに追加されるソフトウェアコンポーネントに端子を割り当てます。端子の割り当ては端子ページで変更可能です。

このページでは、端子機能と端子番号のリストを表示します。

端子機能リストにあるソフトウェアコンポーネントの端子割り当てを変更するには、以下の手順で行います。

(1) ［ハードウェアリソース表示とソフトウェアコンポーネント表示の切り替え］ をクリックして、ソフトウェアコンポーネントによって表示するように変更します。

(2) ソフトウェアコンポーネントを選択します。（例：Config_S12AD1）

(3) ［使用する］タブをクリックし、使用した端子でソートします。

(4) 端子機能リストの端子割り当てまたは端子番号欄で、端子割り当てを変更します。（例：P17 から P13）

(5) または、［選択されたリソースの次の端子割り当て先］ ボタンをクリックし、端子割り当てを変更します。クリックするごとに、機能を持つ端子が表示されます。

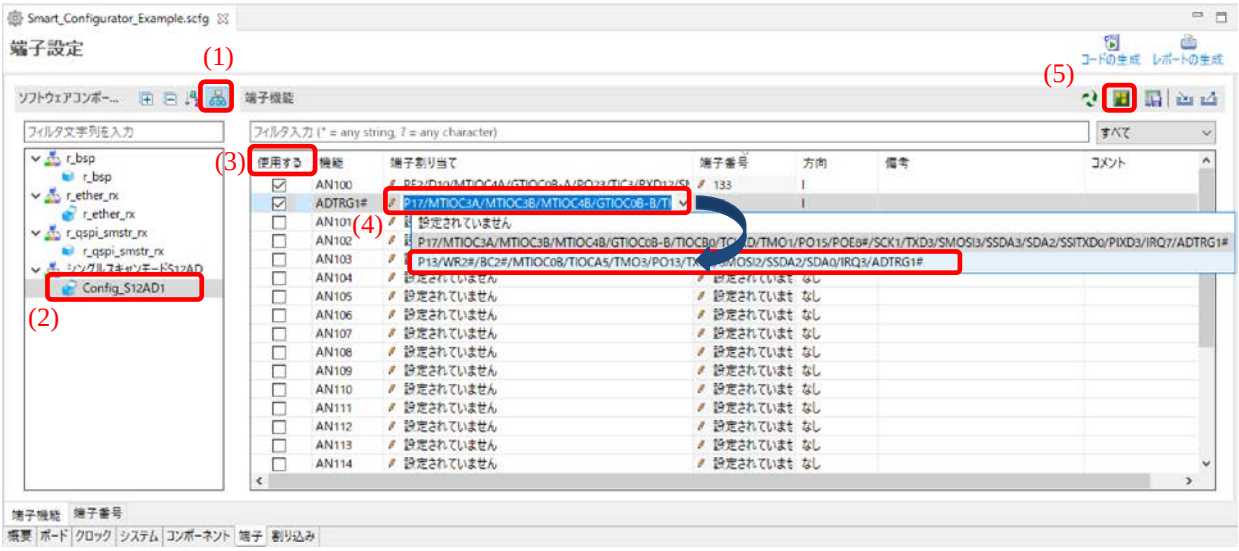

図 4-57 端子設定-［端子機能］リストの端子配置設定

スマート・コンフィグレータでは、ユーザーは他のソフトウェアコンポーネントにリンクすることなく、端子ページで端子機能を有効にすることができます。それらの端子をソフトウェアコンポーネントが使用する他の端子と区別するため、表の中に“この端子を使用するコンポーネントはない”という注意書きがつけられます。

【注】 いくつかの FIT モジュールの端子設定は、サポートされません。
サポートされていない FIT モジュールの端子設定については、対応する FIT モジュール <ProjectDir>\src\smc_gen\r_xxx\doc フォルダにあるアプリケーションノートを参照してください。

4.5.2 MCU/MPU パッケージの端子割り当て

スマート・コンフィグレータでは、 MCU/MPU パッケージビューで端子割り当てを視覚化します。

MCU/MPU パッケージビューをイメージファイルにキャプチャーし、回転や拡大、縮小ができます。

MCU/MPU パッケージビューで端子を設定するには、以下の手順で行います。

(1) ［拡大］ボタンをクリックるすか、マウスホイールをスクロールして、ビュー内を拡大します。

(2) 端子の上で右クリックします。

(3) 割り当てを選択します。

(4) ［設定の変更...］で、端子の色をカスタマイズすることができます。

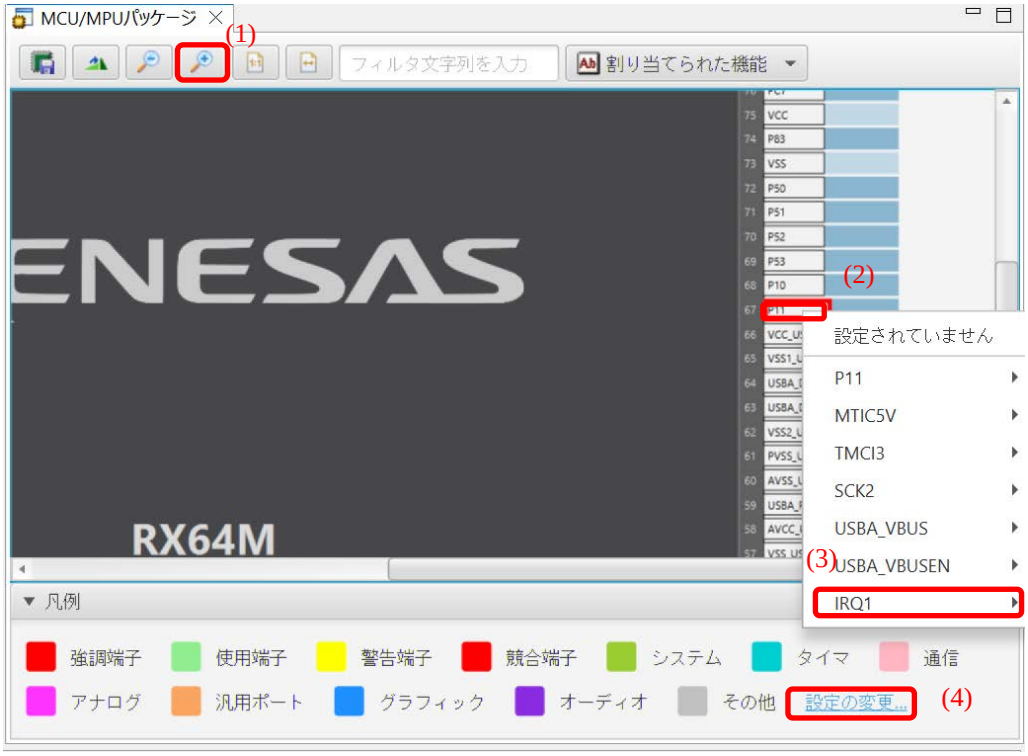

図 4-58 MCU/MPU パッケージを使用した端子設定

4.5.3　端子機能から端子番号の表示

端子機能に関連付けられている端子番号に移動できます。

以下の手順で端子機能から端子番号に移動します。

(1) ［端子機能］タブで、端子機能を右クリックしてポップアップメニューを開きます。

(2) ［端子番号タブにジャンプ］を選択します。

(3) ［端子番号］タブが開き、該当する端子番号が選択されます。 端子機能の端子番号です。

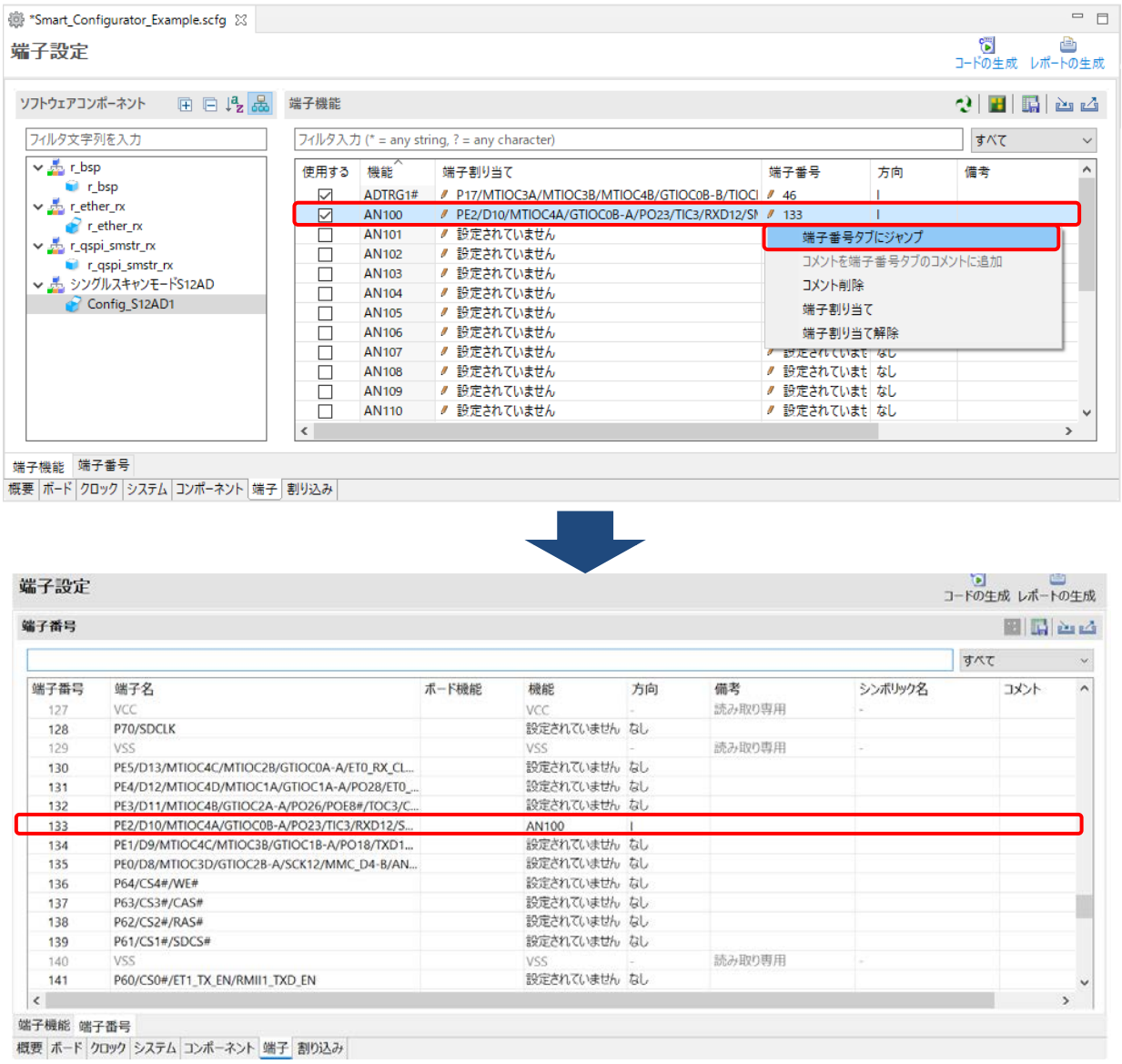

図 4-59 端子機能から端子番号への移動

4.5.4　端子設定のエクスポート

端子設定をエクスポートして、参照することができます。端子設定のエクスポートは、以下の手順で行います。

(1)　端子ページで、［ボードの設定をエクスポート］ボタンをクリックします。

(2)　出力場所を選択し、エクスポートするファイル名を入力します。

XML フォーマットでエクスポートしたファイルは、同じデバイスの型名がある他のプロジェクトにインポートすることができます。

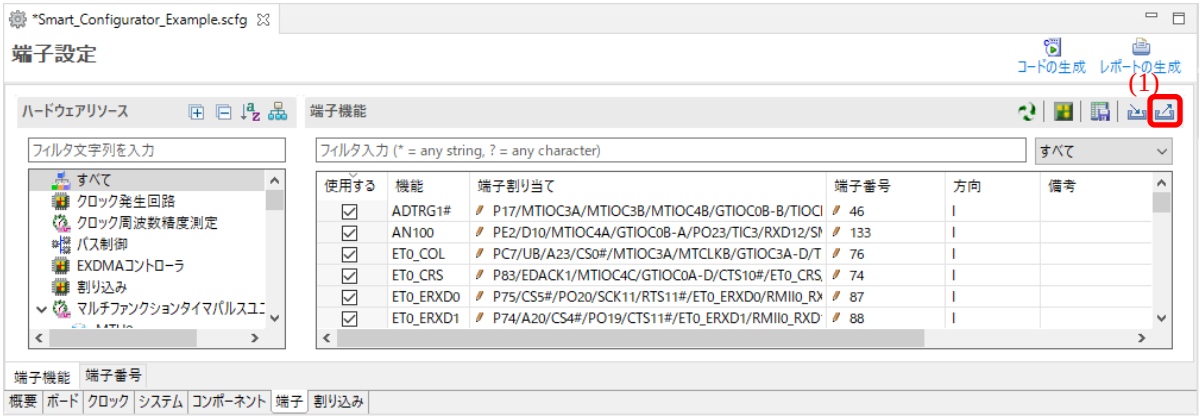

図 4-60 端子設定を XML ファイルへエクスポートする

端子ページの［.csv ファイルにリストを保存］ボタンをクリックすることで、端子設定を CSV 形式で保存します。

4.5.5　端子設定のインポート

現在のプロジェクトに端子設定をインポートするには、［ボードの設定をインポート］ボタンをクリックし、端子設定を含む XML ファイルを選択してください。設定がプロジェクトにインポートされると、このファイルに指定された設定は、端子設定ページに反映されます。

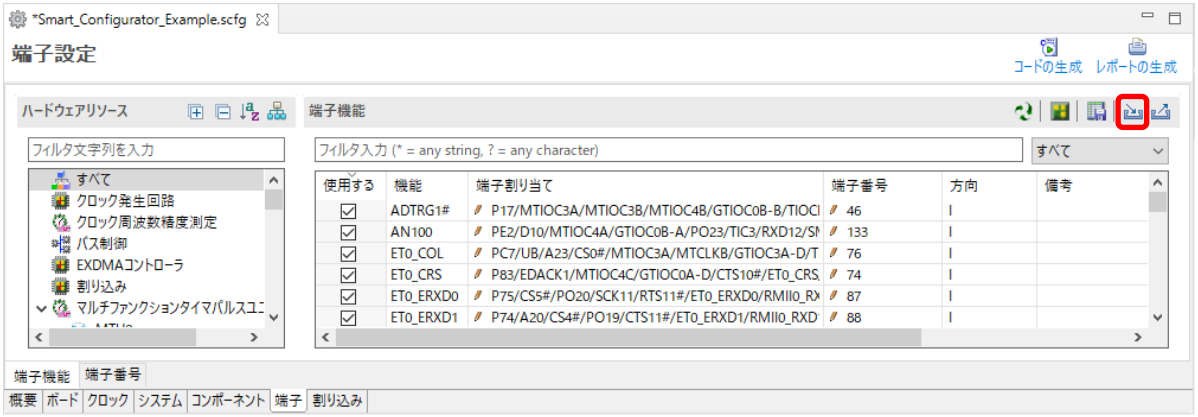

図 4-61 端子設定を XML ファイルからインポートする

【注】 端子設定は反映されますが、コンポーネント設定には反映されません。

4.5.6　ボード端子設定情報を使用した端子設定

ボードの初期端子設定を一括で行えます。選択したボードは［ボード］ページで確認できます。

端子を一括で設定するには、以下の手順で行います。

(1)　MCU/MPU パッケージで［Default Board］を選択します。(ボードの初期端子設定が確認できます)

(2)　［端子設定］ページを開き、［Assign default board pins］アイコンをクリックします。

(3)　［Assign default board pins］ダイアログが開いたら［Select all］をクリックしてください。

(4)　[OK]をクリックします。

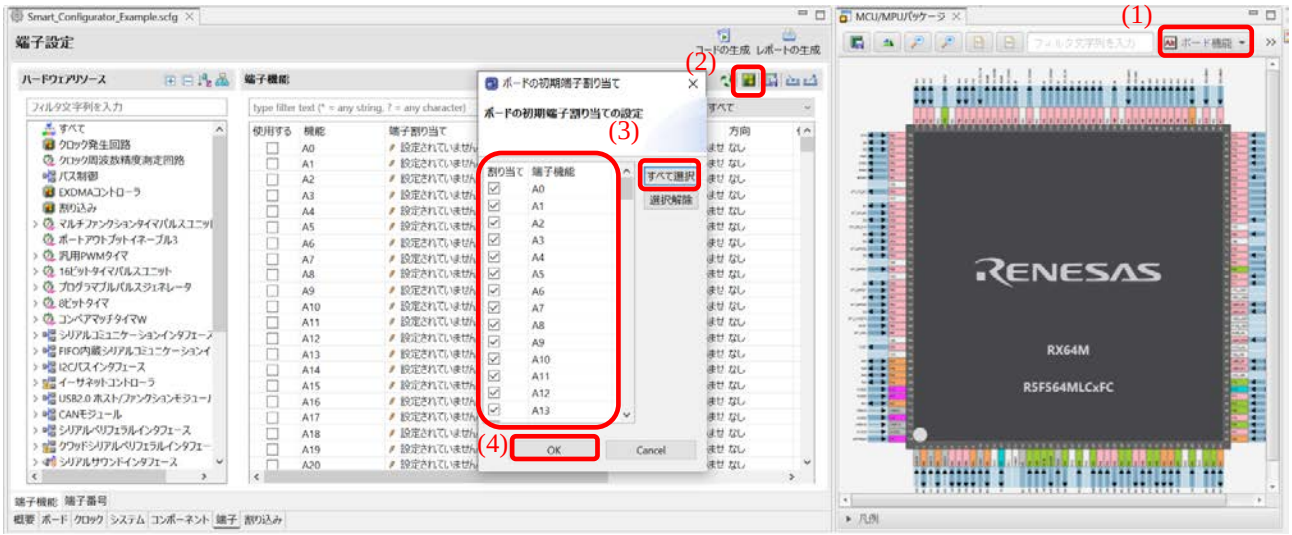

図 4-62 ボードの初期端子設定

端子設定を一括で行わない場合は、手順（3）で個別に指定してください。

4.5.7　端子のフィルタ機能

「端子」ページの[端子機能]タブ、[端子番号]タブでフィルタ範囲を指定し、より簡単に参照することができます。

図 4-63 ［端子機能］タブのフィルタ

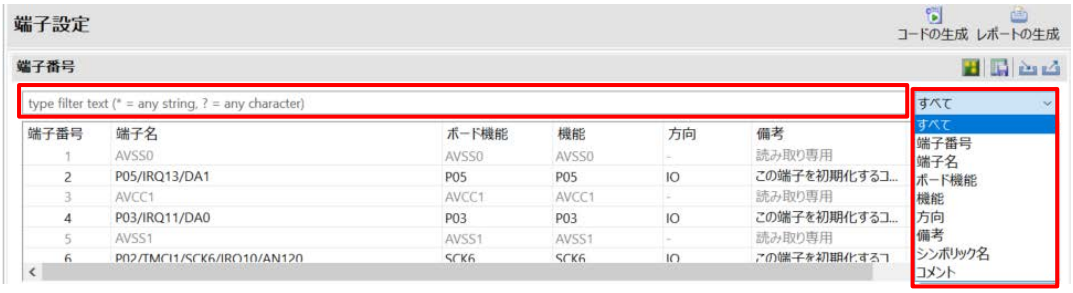

図 4-64 ［端子番号］タブのフィルタ

4.5.8　端子エラー／警告設定

［端子エラー／警告］設定を使用して、［コフィグレーションチェック］ビューの表示方法を制御できます。制御する場合は、［コンポーネントの追加］ダイアログの［基本設定...］リンクから［設定］ダイアログを表示させます。［設定］ダイアログのメニュー［Renesas］＞［スマート・コンフィグレータ］＞［端子エラー／警告］を選択し、設定を変更します。

図 4-65 端子エラー/警告の設定

以下は、［ソフトウェアコンポーネント］の設定を［Info］から［Error］に変更した例です。

図 4-66 ［ソフトウェアコンポーネント］を［**Info**］から［**Error**］に変更

4.5.9　シンボリック名設定

[シンボリック名]は端子の属性で、端子番号ページと MCU/MPU パッケージページで確認できます。
[シンボリック名]を設定することで、ユーザ独自のシンボル名を使用でき、端子の割り当てが変更された場合でも、ソースコードを変更せず端子を制御することができます。
端子番号ページでシンボリック名が設定されると、マクロ定義は、Pin.h ファイルに生成されます。

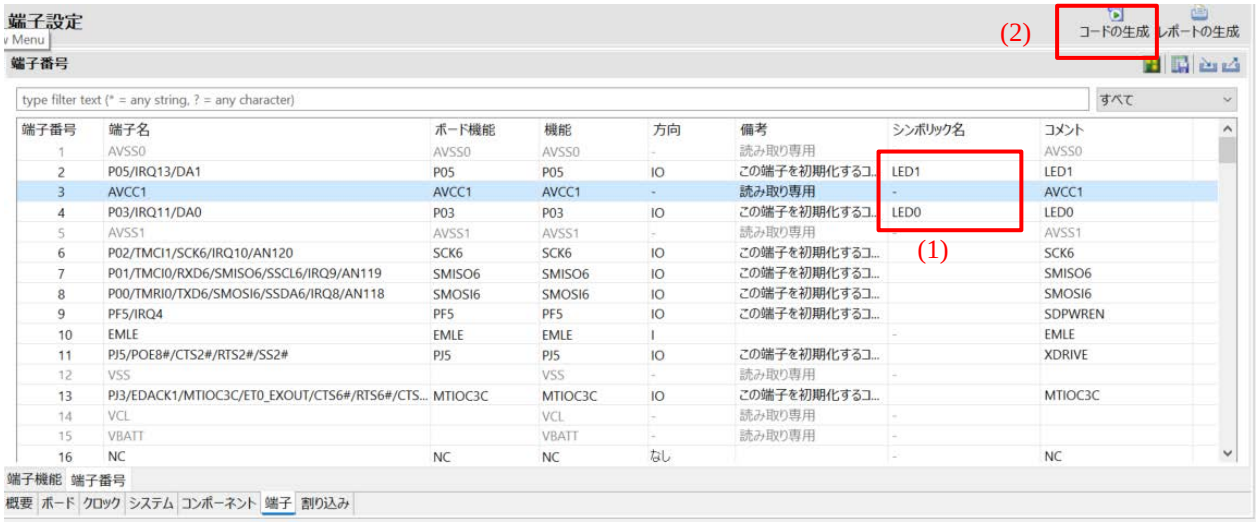

図 4-67 シンボリック名の設定

コード生成後、Pin.h ファイルで定義をチェックできます。

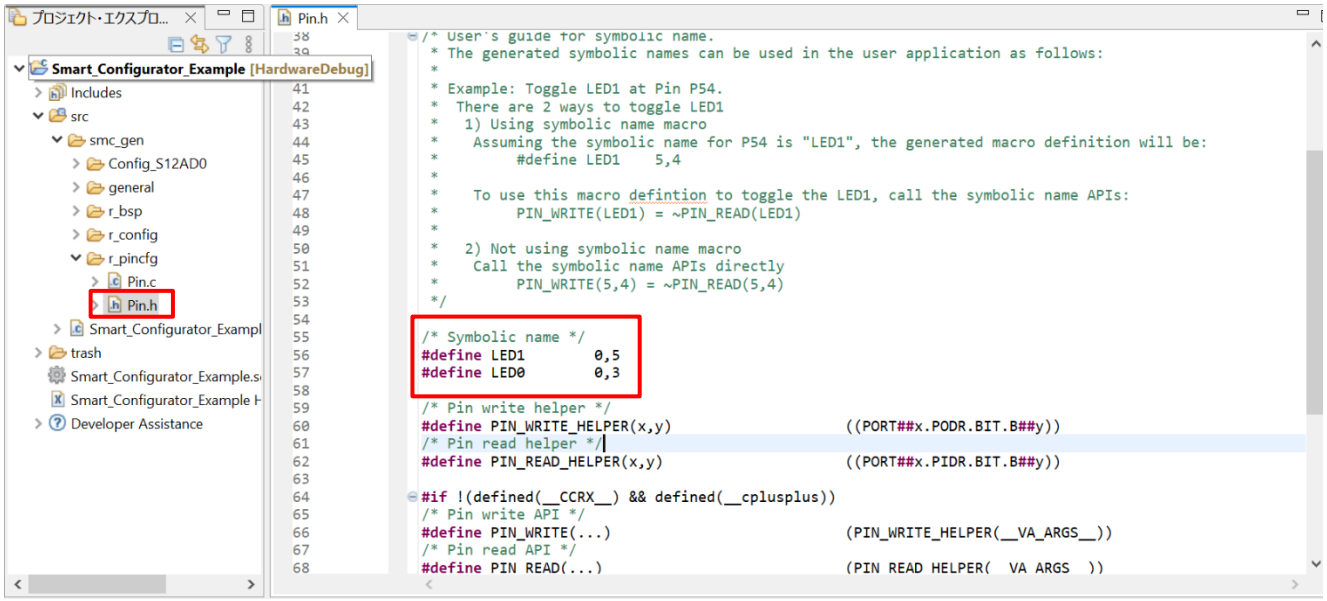

図 4-68 生成コードのシンボリック名

```
void main(void)
{
    // Set port direction
    PORTB.PDR.BYTE = 0x08U;
    PORTE.PDR.BYTE = 0x20U;

    //Init LED status
    PIN_WRITE(LED1) = 1U;
    PIN_WRITE(LED0) = 0U;

    //Toggle LEDs
    while(1){
        PIN_WRITE(LED0) = ~PIN_READ(LED1);
        PIN_WRITE(LED1) = ~PIN_READ(LED0);

        //Toggle LEDs
        for (int i = 0;i < 100000;i++){
            nop();
        }
    }
}
```

図 4-69 Main 関数でシンボリック名を使用

4.6　割り込み設定

割り込みページは、［コンポーネント］ページで設定した周辺機能の割り込みの確認および設定を行います。ベクタ番号別に割り込みが表示されますので、割り込み優先レベル、高速割り込み要因、動的割り込みベクタ番号を設定します。

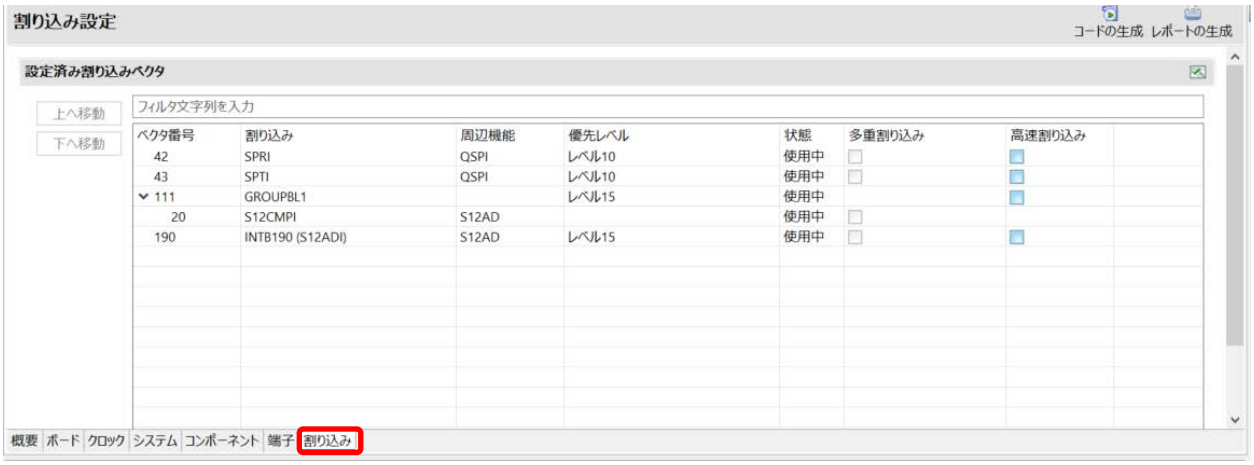

図 4-70 割り込みページ

4.6.1　割り込み優先レベルと高速割り込みの設定

コンポーネントページでの CG コンフィグレーションに割り込みを使用する場合、割り込みの状態は“使用中”に変わります。使用中の割り込みだけを表示する場合は、［設定した割り込みのみ表示］ ボタンをクリックしてください。

(1) 割り込みページで、割り込み優先レベルを変えることができます。

(2) 割り込みを高速割り込みに設定するには、高速割り込み欄のチェックボックスを選択します。一度に一つの割り込みしか、高速割り込みに設定できません。

(3) グループ割り込みは、割り込みテーブルでは折りたたまれます。グループ割り込みリストの割り込みを見るには、［展開］ ボタンをクリックしてください。

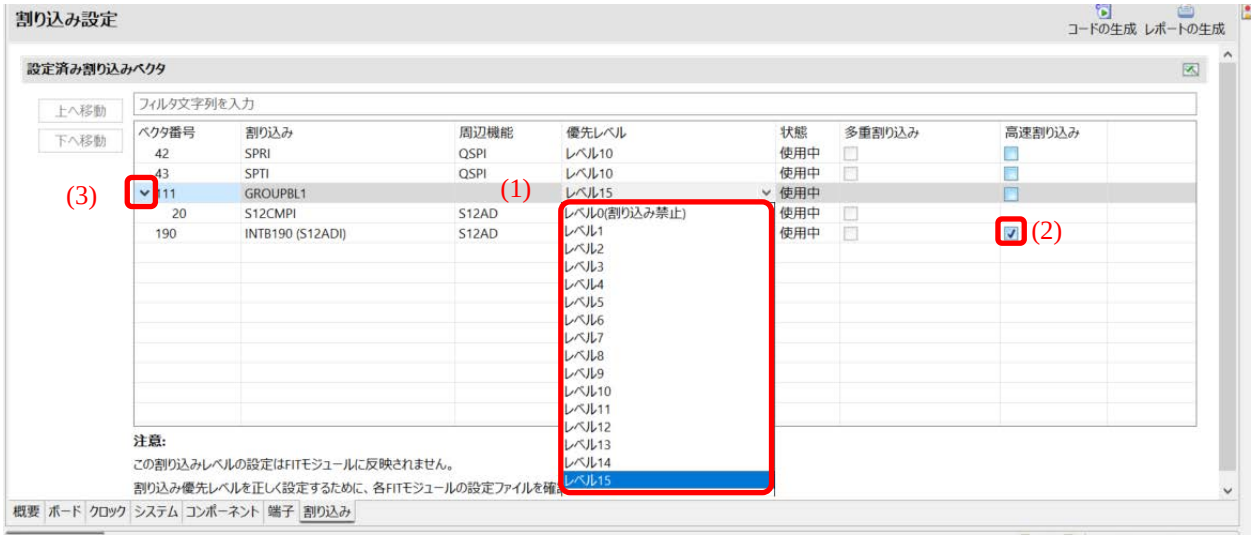

図 4-71 割り込み設定

【注】 FIT モジュールの割り込み設定は、サポートされていません。
それぞれの FIT モジュールの割り込み設定については、対応する FIT モジュール <ProjectDir>¥src¥smc_gen¥r_xxx¥doc フォルダにあるアプリケーションノートを参照してください。

4.6.2　割り込みベクタ番号の変更

割り込み設定ページで、ユーザーは選択型割り込み A、B のベクタ番号を変更することができます。

(1) 選択型割り込み A、B を選択します。

(2) ［上へ移動］と［下へ移動］ボタンが使用可能となります。このボタンをクリックして、ベクタ番号を変更します。

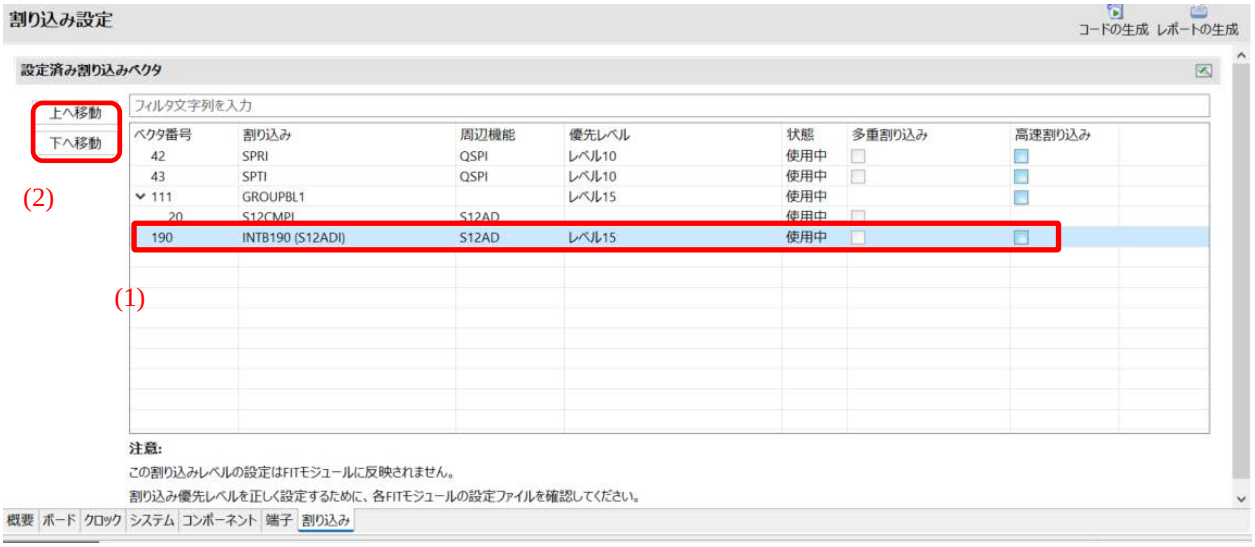

図 4-72 選択型割り込み A、B

4.6.3　多重割り込み設定

RX MCU の多重割り込み機能により、割り込み実行中に別の割り込みを処理することができます。
多重割り込みの設定は、割り込みページと、コンポーネントの割り込み設定で設定することができます。

(1) 多重割り込みをサポートしているコンポーネントを選択し、複数の割り込みを許可をチェックします。

(2) 多重割り込みの設定は、割り込みページとコンポーネントページで同期しています。

(3) 生成されたファイルをプロジェクトエクスプローラーで開くと、生成されたコードを確認できます。

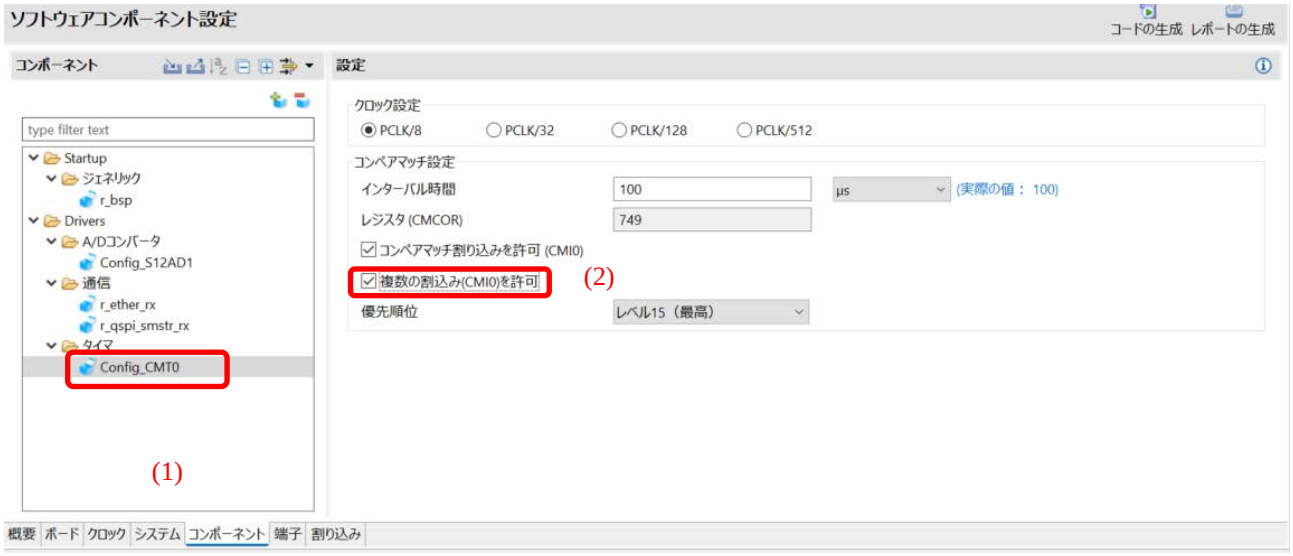

図 4-73 コンポーネントページの多重割り込み設定

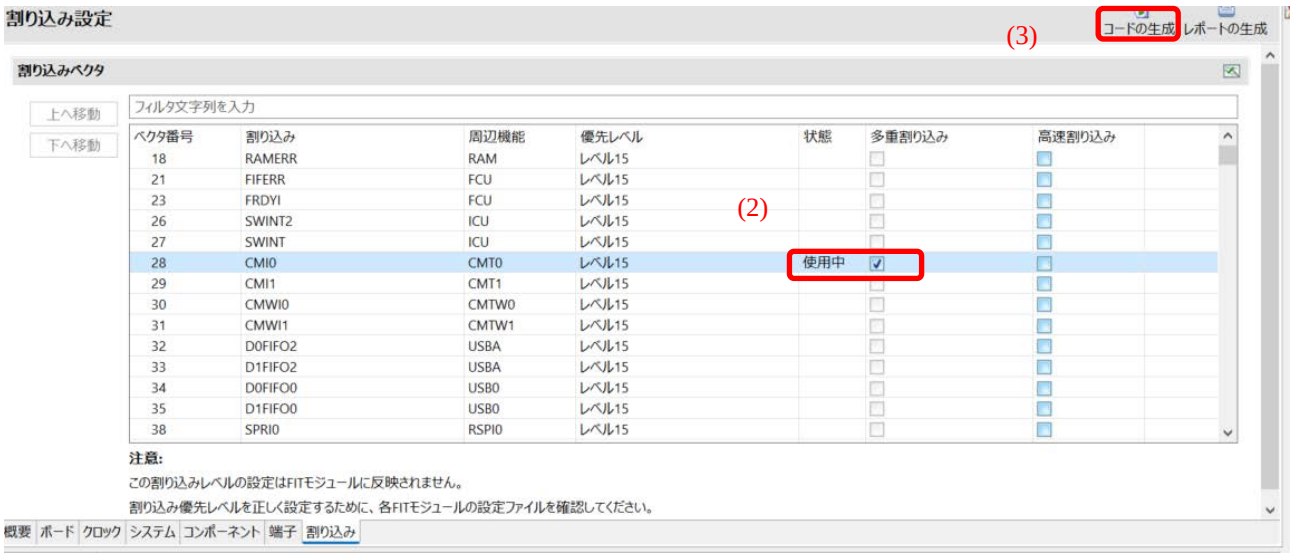

図 4-74 割り込みページの多重割り込み設定

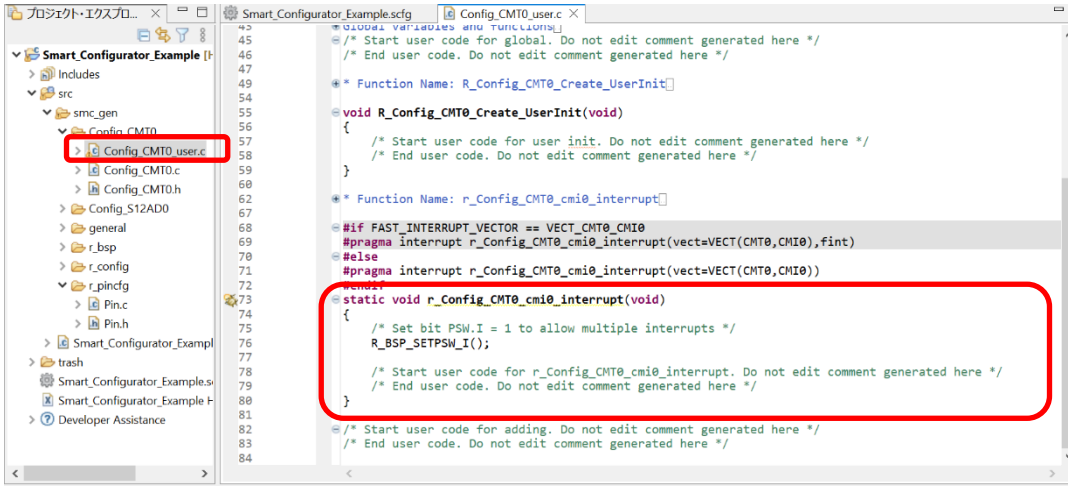

図 4-75 多重割り込みの生成コード

4.7　MCU マイグレーション機能

MCU マイグレーション機能は、異なるデバイス間でプロジェクト設定の移行を行います。プロジェクト設定の変換は、同一ファミリ内で可能で、e² studio の［プロジェクト］メニューから以下の手順で行います。

【注】 デバイスの変更により、プロジェクトの設定が変わる場合があります。
デバイス変更を実行する前にプロジェクトのバックアップを行ってください。

(1) プロジェクトを選択し、［プロジェクト］メニューから［Change Device］を選択します。

図 4-76 ［Change Device］選択

(2) ボードリストからターゲット・ボードを選択すると、デバイスが自動的に選択されます。

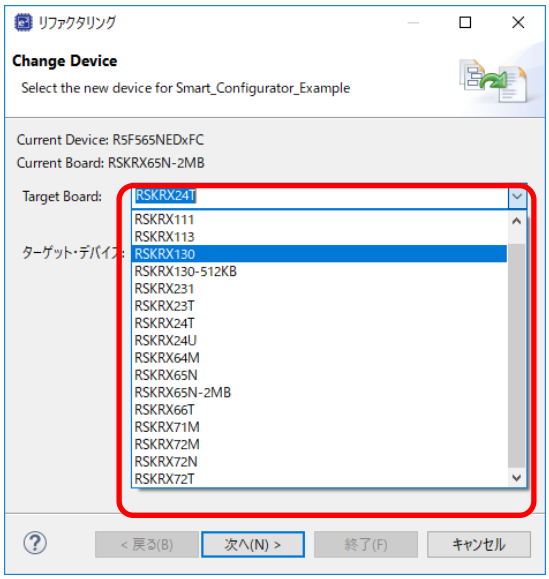

図 4-77 ターゲット・ボード選択

ターゲット・ボードを「カスタム」として選択した場合は、デバイスリストからターゲット・デバイスを手動で選択し、「次へ」をクリックします (例：RSKRX65N ボードからカスタムボードの RX651 デバイスに変更)。

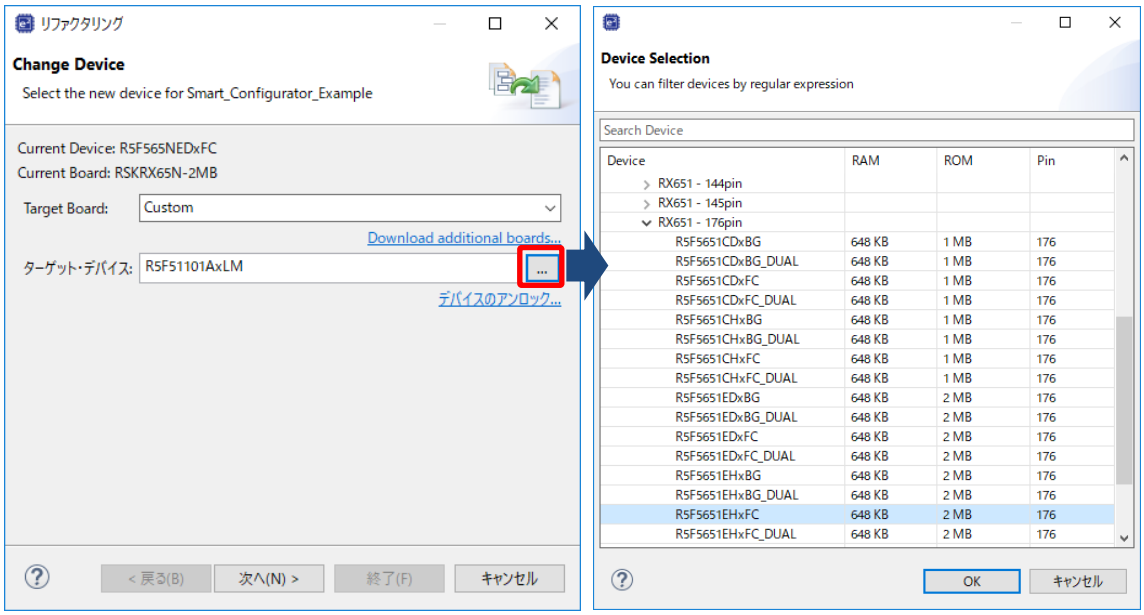

図 4-78 ターゲット・デバイス選択

(3) ［検出された問題］に表示されたメッセージを確認して［次へ］をクリックします。

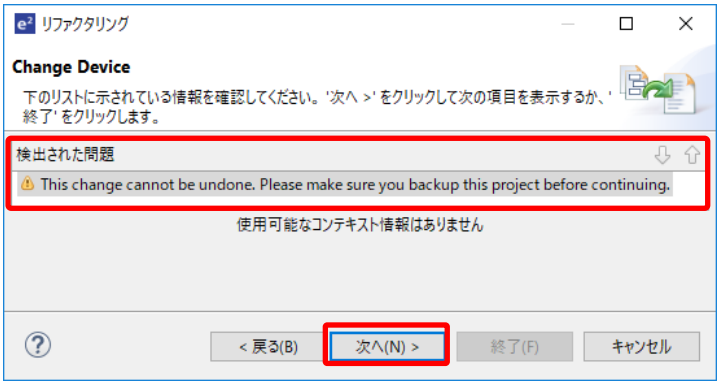

図 4-79 検出された問題

メッセージ	説明
ターゲット・デバイスはスマート・コンフィグレータでサポートされていません	スマート・コンフィグレータがサポートしていないデバイスへの変更時に表示されます。スマート・コンフィグレータの変換は実行できませんが、プロジェクト、ビルダー、リンカー、デバッカーは変換できます。
This change cannot be undone. Please make sure you backup this project before continuing.	デバイスを変更すると変更前に戻すことができませんので、プロジェクトのバックアップ後に実行してください。

(4) ［実行される変更］で、変更する項目を選択してマイグレーションを実行します。

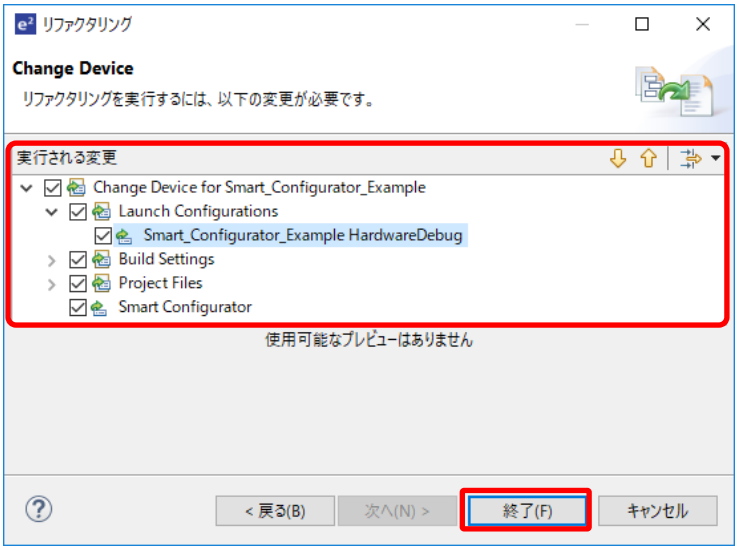

図 4-80 実行される変更

(5) デバイスの変更が完了すると、「概要」ページのデバイス名が更新されます。

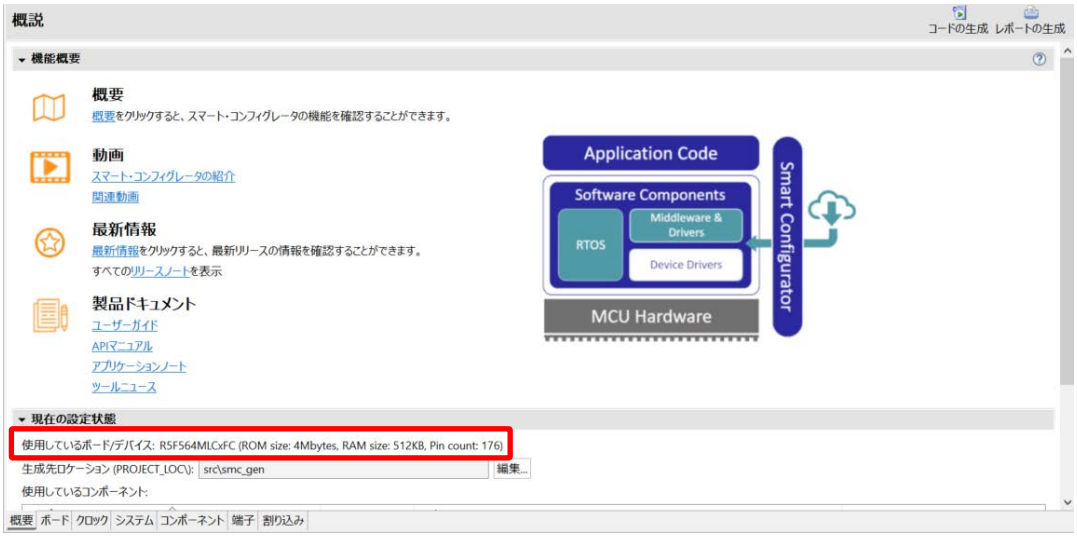

図 4-81 デバイス更新確認

(6) コンソールにデバイス変更結果レポートが出力されます。

図 4-82 設定変換ステータスレポート

【注】 デバイス変更結果レポートが正しく表示されない場合は、右メニューより「エンコード」の設定を“Unicode(UTF-8)”に変更してください。

図 4-83 エンコード変更

5. 競合の管理

コンポーネントの追加、端子や割り込みの設定をすると、リソースの不一致や失われた依存関係にあるモジュールに関連する問題が起こる可能性があります。この情報はコンフィグレーションチェックビューに表示されます。表示された情報を参照して、競合問題を解決してください。

5.1 リソースの競合

同じリソース（例：S12AD0）を使うために、二つのソフトウェアのコンフィグレーションを設定した場合、コンポーネントツリーにエラーマーク が表示されます。

コンフィグレーションチェックビューに周辺機能の競合に関するメッセージが表示され、周辺機能に競合が見つかったソフトウェア設定を知らせます。

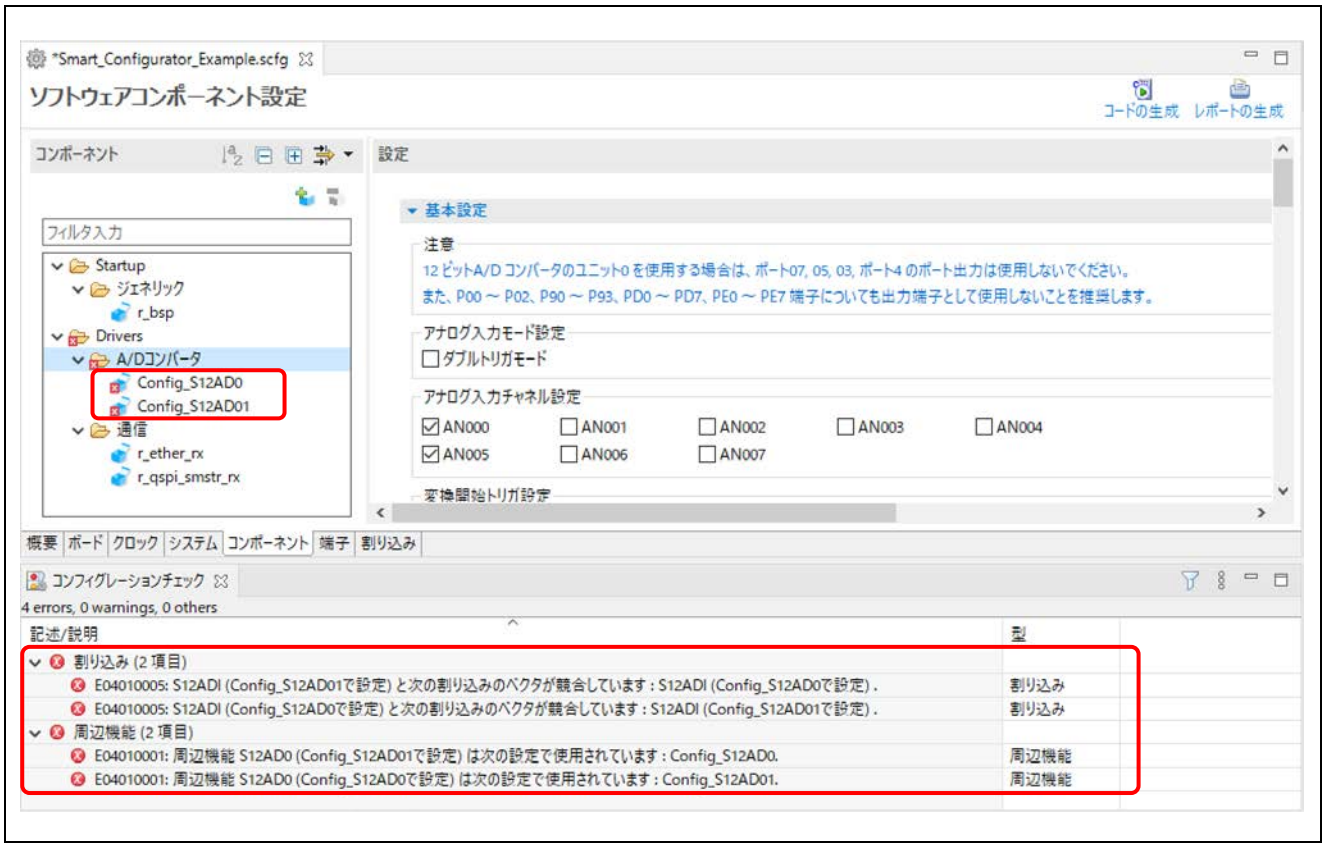

図 5-1 リソースの競合

5.2　端子の競合

端子の競合がある場合、エラーマーク　がツリーと端子機能リストに表示されます。

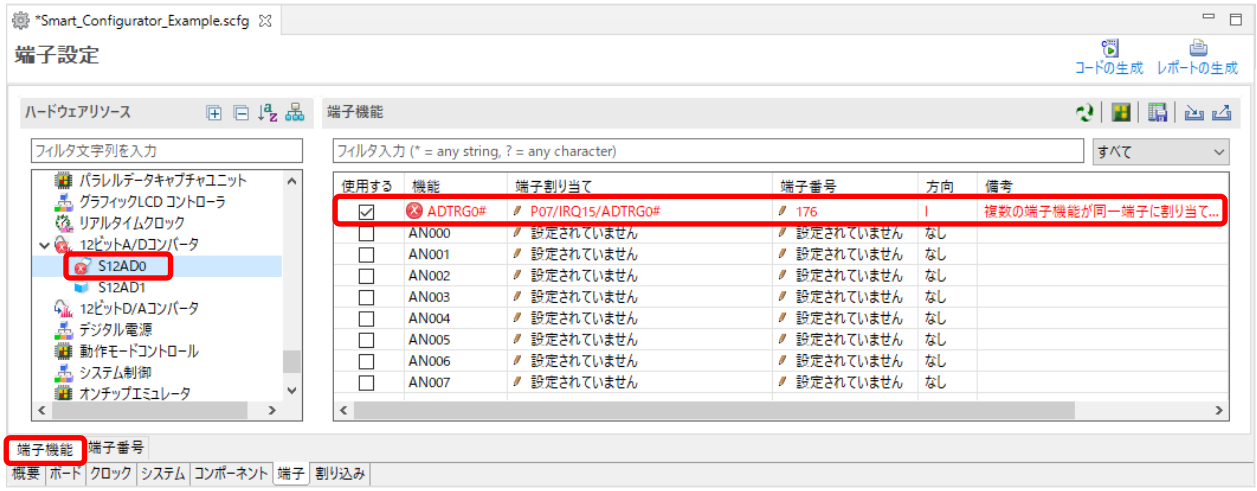

図 5-2 端子の競合

競合情報の詳細は、コンフィグレーションチェックビューに表示されます。

図 5-3 端子競合のメッセージ

エラーマークのあるツリーノードを右クリックし、［競合の解決］を選択して競合を解決してください。

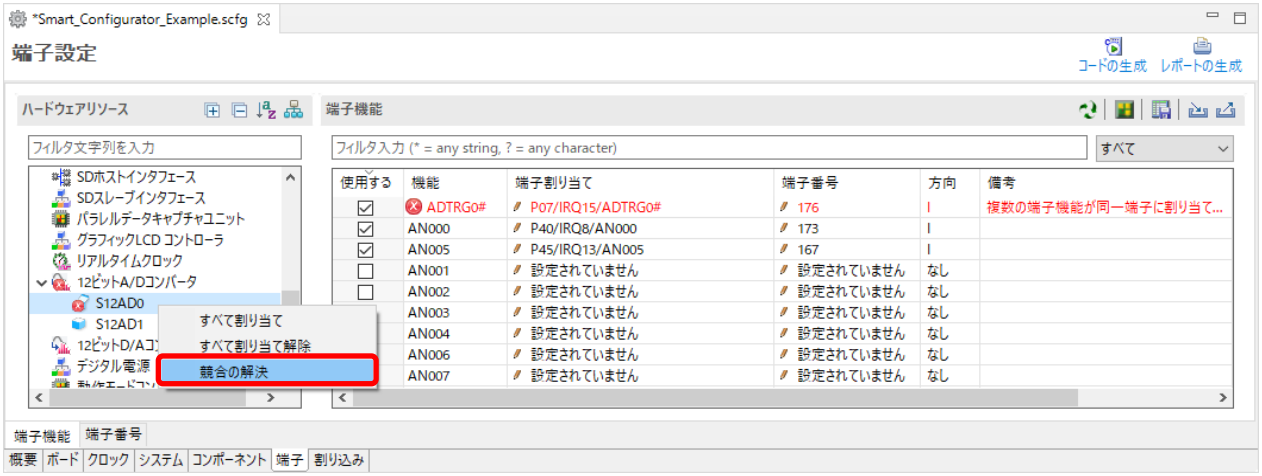

図 5-4 端子競合の解決

選択されたノードの端子は、他の端子に再度割り当てられます。

6. ソースの生成

6.1 生成ソースの出力

スマート・コンフィグレータビューの［コードの生成］ボタンをクリックすると、設定した内容に応じたソースファイルを出力します。

図 6-1 ソースファイルの生成

スマート・コンフィグレータは、<PeojectDir>¥src¥smc_gen にファイルを生成し、プロジェクト・エクスプローラー内のソースファイルを更新します。すでにスマート・コンフィグレータでファイルを生成している場合、バックアップも生成します。（「8 生成ソースのバックアップ」を参照ください。）

【注】ユーザー作成のソースファイルを sms_gen フォルダに置くと、ソースコード生成時に消去されます。

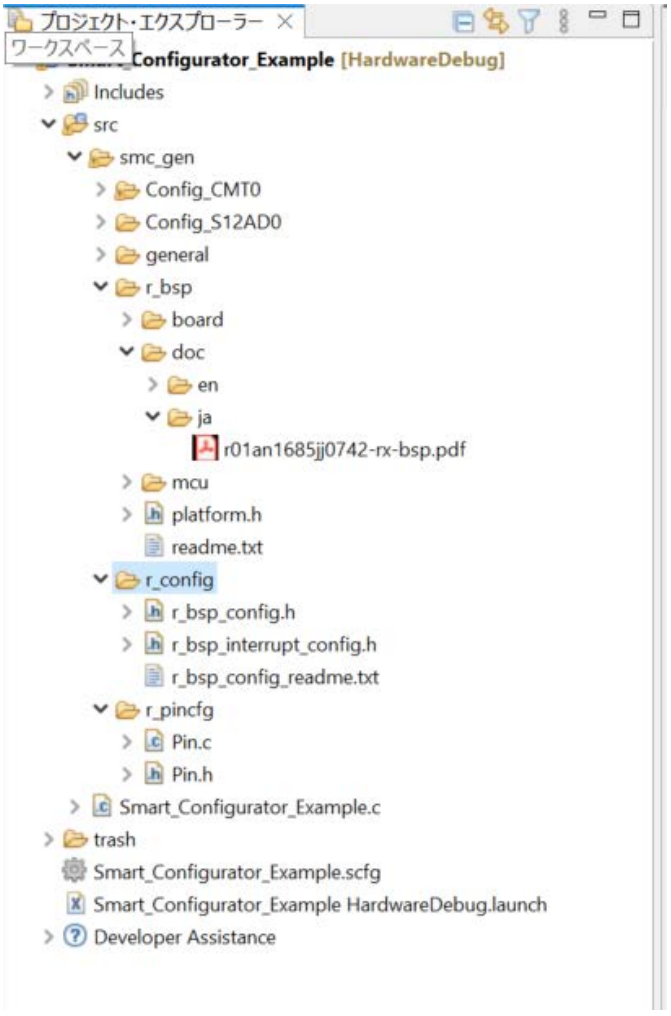

図 6-2 プロジェクト・エクスプローラー内のソースファイル

6.2　ソース生成先の設定

ソースの生成先は、スマート・コンフィグレータの［概要］ページから設定できます。

(1) ［概要］ページの［現在の設定状態］の下にある［編集］ボタンをクリックします。

図 6-3 ソース生成先の変更

(2) ［フォルダの選択］ダイアログで、生成先のフォルダを選択するか、新しいフォルダを作成し、［OK］ボタンをクリックします。

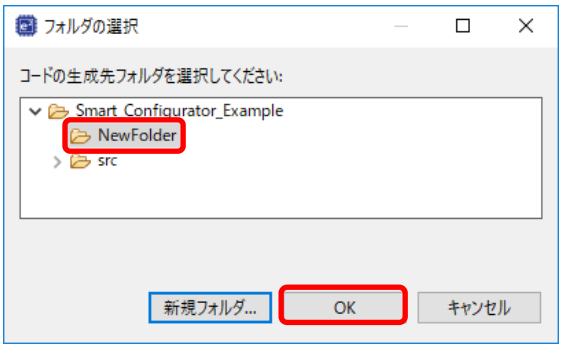

図 6-4 フォルダ選択

(3) ［コード生成  ］ボタンをクリックすると、(2)で選択したフォルダにソースファイルが生成されます。［概要］ページで現在の生成コードの場所を確認することもできます。

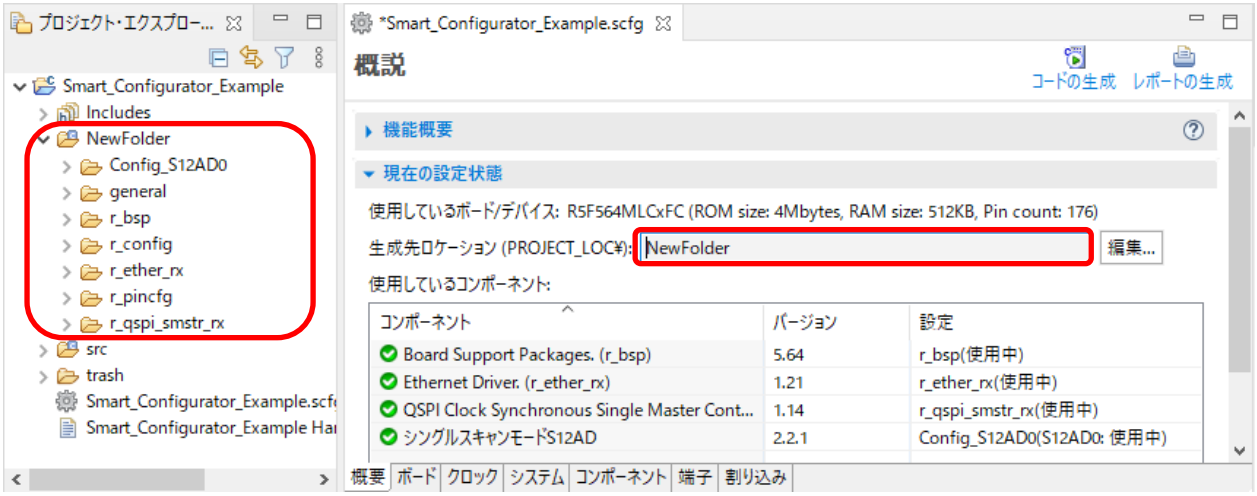

図 6-5 プロジェクト・エクスプローラー内のソースファイル

6.3 生成ファイルの構成とファイル名

スマート・コンフィグレータが出力するフォルダとファイルを下図に示します。なお、*main()*関数は e² studio でプロジェクト作成時に生成する*{Project name}.c* に含まれます。

r_xxx は FIT モジュール名、“ConfigName”はコンポーネント設定で設定したコンフィグレーション名を示します。

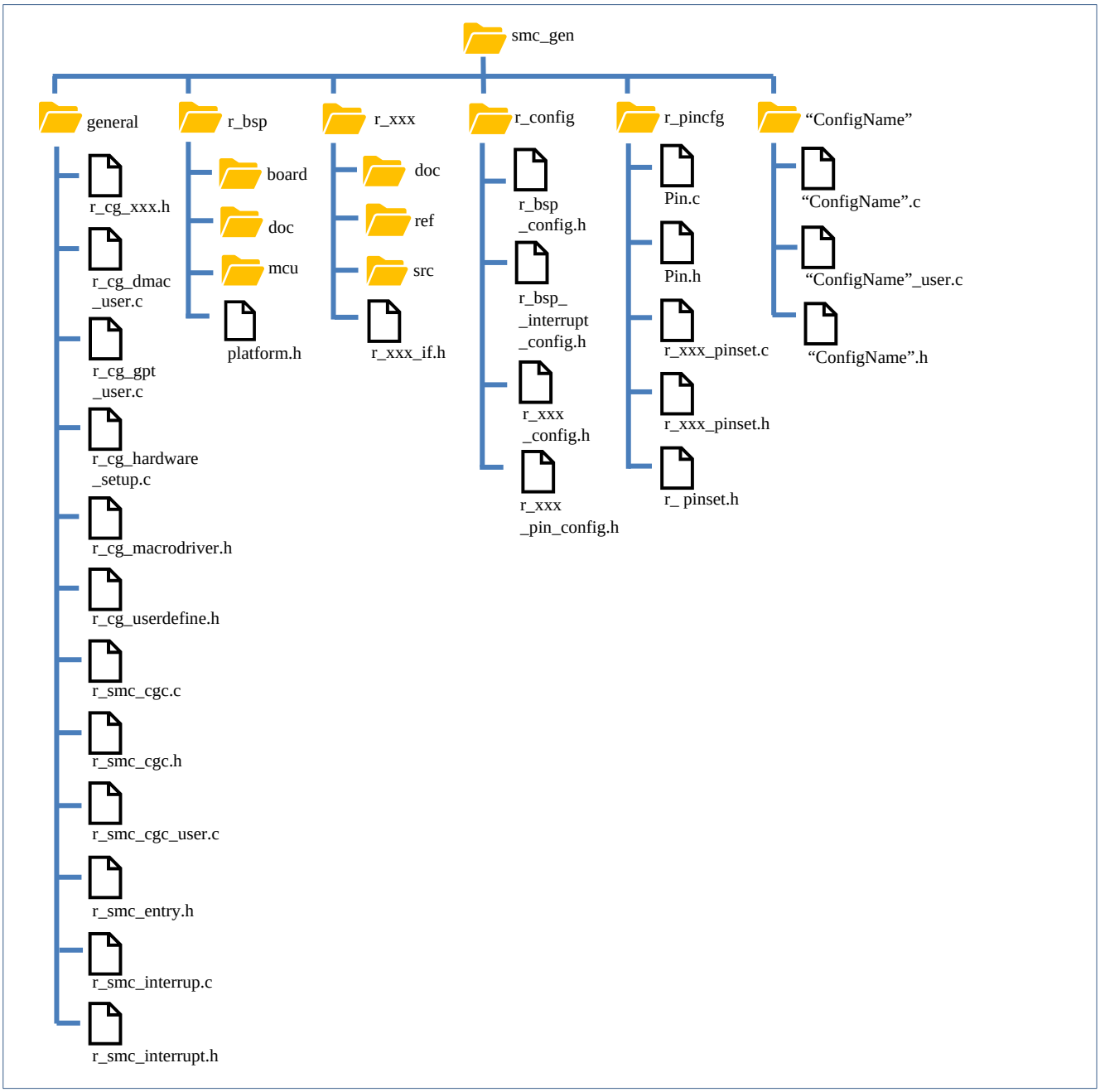

図 6-6 生成ファイルの構成とファイル名

フォルダ	ファイル	説明
general		このフォルダは常時生成されます。同じ周辺機能の CG ドライバで共通に使用される、ヘッダファイルとソースファイルを含みます。
	r_cg_xxx.h(*1)	これらのファイルは常時生成されます。SFR レジスタを設定するためのマクロ定義を含みます。
	r_cg_dmac_user.c	このファイルは、DMAC 機能を持つデバイス用に生成されます。いくつかの DMAC チャネルで共有される割り込みサービスルーチンやコールバック関数を含みます（ハードウェア仕様に依存します）。
	r_cg_gpt_user.c	このファイルは、GPT 機能を持つデバイス用に生成されます。いくつかの GPT チャネルで共有される割り込みサービスルーチンやコールバック関数を含みます（ハードウェア仕様に依存します）。
	r_cg_hardware_setup.c	このファイルは常時生成されます。 *R_ConfigName_Create* の名前を持つ全てのドライバの初期化関数を呼び出す *R_Systeminit* を含みます。*R_ Systeminit* は、クロックソース、高速割り込み、グループ割り込み以外のクロックの初期化も呼び出します。
	r_cg_macrodriver.h	このファイルは常時生成されます。 このヘッダファイルは、ドライバで使用される共通のマクロ定義を含みます。
	r_cg_userdefine.h	このファイルは常時生成されます。 ユーザーは、専用のユーザーコード領域にマクロ定義を追加することができます。
	r_smc_cgc.c	このファイルは常時生成されます。 クロックページの設定を基にしたクロックソース以外のクロックソースを設定する初期化を含みます。
	r_smc_cgc.h	このファイルは常時生成されます。 このヘッダファイルは、クロックソース以外のクロックを初期化するマクロ定義を含みます。
	r_smc_cgc_user.c	このファイルは、CGC 初期化後にユーザーがコードを *R_CGC_Create* に追加する関数を含みます。 ユーザーは、コードと関数を専用のユーザーコード領域に追加することができます。
	r_smc_entry.h	このファイルは常時生成されます。 このファイルには、プロジェクトに追加される CG ドライバのヘッダファイルが含まれます。 ユーザーが追加するソースファイルで、CG ドライバの関数を使用する場合、このファイルのインクルードが必要です。
	r_smc_interrupt.c	このファイルは常時生成されます。高速割り込み、グループ割り込みの初期化を含みます（ハードウェア仕様に依存します）。
	r_smc_interrupt.h	このファイルは常時生成されます。 高速割り込み、グループ割り込みを初期化するマクロ定義を含みます。また、［割り込み］タブで設定されるすべての割り込みの優先レベルを含みます。ユーザーは、アプリケーションコードにこれらのマクロ定義を使用することができます。
r_bsp		このフォルダは常時生成されます。 以下を含む複数のサブフォルダ（*board, doc, mcu*）から構成されます。 - *main()*実行前に MCU を起動する初期化コード（例：スタックのセットアップ、メモリの初期化） - *iodefine.h* (*mcu* フォルダ)にあるすべての SFR レジスタの定義 - *r_bsp* のアプリケーションノート プロジェクトで使用されるデバイスの *r_bsp.h* を含む、*platform.h* もこのフォルダに生成されます。

フォルダ	ファイル	説明
r_xxx *(*1)*		このフォルダは、プロジェクトに追加される FIT モジュール用に生成され、以下によって構成されます。 *-doc* フォルダ：この FIT モジュールのアプリケーションノート - *ref* フォルダ：FIT モジュール構成ファイルと端子構成ファイルのリファレンス - *src* フォルダ：FIT モジュールソースファイルとヘッダファイル - *r_xxx_if.h* *(*1)*：この FIT モジュールのすべての API 呼び出しとインタフェース定義のリスト 注： *r_xxx* にあるフォルダは、各 FIT モジュールの要求に依存します。
r_config		このフォルダは常時生成されます。 MCU パッケージ、クロック、割り込み、*R_xxx_Open* *(*1)*の名前を持つドライバ初期化関数のコンフィグレーションヘッダファイルを含みます。
	r_bsp_config.h	このファイルは常時生成されます。 クロック初期化と他の MCU に関連する *r_bsp* の設定を含みます。いくつかの MCU 関連の設定はスマート・コンフィグレータが生成し（例：パッケージタイプ）、他の設定（例：スタックサイズ）はユーザーが手動で設定します。
	r_bsp_interrupt_config.h	このファイルは常時生成されます。 選択型割り込み A、B のマッピングを含みます（ハードウェア仕様に依存します）。
	r_xxx_config.h *(*1)*	プロジェクトに追加される全 FIT ドライバ用のコンフィグレーションヘッダファイルです。ユーザーは手動でこのファイルを設定することができます。
	r_xxx_pin_config.h *(*1)*	端子設定シーケンスで特定要求のある FIT ドライバ専用の、端子コンフィグレーションヘッダファイルです。
r_pincfg	*Pin.c*	このファイルは常時生成されます。 ［端子］タブで設定される全周辺機能に使用する端子機能初期化のリファレンスです（I/O ポート以外）。
	Pin.h	このファイルは常時生成されます。 *Pin.c* での端子設定の関数プロトタイプを含みます。
	r_xxx_pinset.c *(*1)*	プロジェクトに追加される FIT ドライバの端子機能初期化を含みます。このファイルの API 関数は、アプリケーションコード内で呼び出すことができます。
	r_xxx_pinset.h *(*1)*	このファイルは *r_xxx_pinset.c* での端子設定機能のプロトタイプを含みます。
	r_pinset.h	このファイルは、*r_pincfg* フォルダに *r_xxx_pinset.h* *(*1)* の名前がついたすべてのピン設定ヘッダファイルを含みます。
{ConfigName}		このフォルダは、プロジェクトに追加される CG ドライバ用に生成されます。 このフォルダの API 関数は、*ConfigName*（設定名）の後に命名します。
	{ConfigName}.c	このファイルは、ドライバ(*R_ConfigName_Create*)を初期化する関数、ドライバに特有な操作、例えばスタート(*R_ConfigName_Start*)やストップ(*R_ConfigName_Stop*)を実行する関数を含みます。
	{ConfigName}_user.c	ドライバの初期化(*R_ConfigName_Create*)の後に追加することができる割り込みサービスルーチンと関数を含みます。 ユーザーは、専用のユーザーコード領域にコードと関数を追加することができます。
	{ConfigName}.h	*{ConfigName}.c* と *{ConfigName}_user.c* のヘッダファイルです。

＊1：*xxx* は周辺機能名を意味します。

6.4　クロック設定

クロックページにあるクロックソースの設定は、¥src¥smc_gen¥r_config フォルダにある *r_bsp_config.h* ファイルのマクロに生成されます。*main()*を実行する前に *r_bsp* によって、クロック初期化コードは処理されます。*r_bsp_config.h* ファイルには、他の MCU 関連の設定（例：パッケージ、スタックサイズ）も含みます。他のクロックの構成は、¥src¥smc_gen¥general フォルダに生成されます。

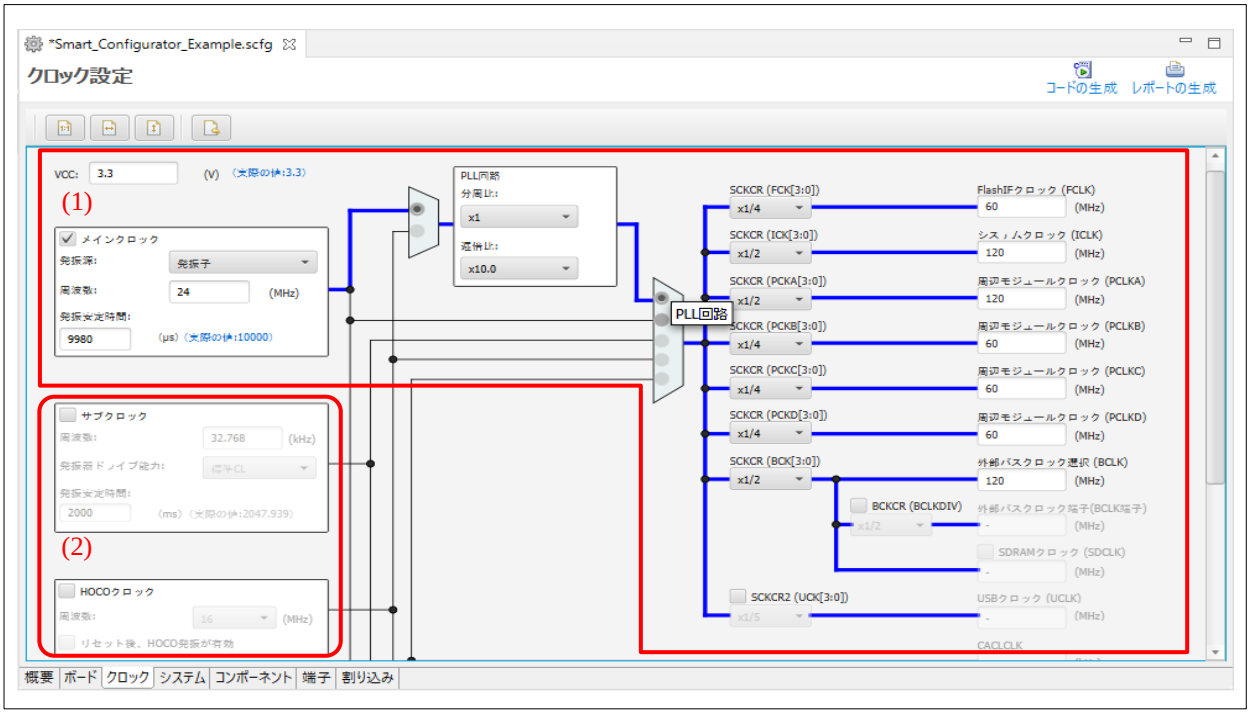

図 6-7 メインクロックをクロックソースとして選択した場合のクロック設定

No	フォルダ	ファイル	マクロ/関数	説明
(1)	**r_config**	*r_bsp_config.h*	クロックに関連するマクロ	これらの設定は、クロックソースの**クロック**ページにあるユーザーの選択を基に、スマート・コンフィグレータによって生成されます。一度に一つのクロックだけをクロックソースとして選択することができます。*main()*を実行する前に、*r_bsp* はクロックの初期化を処理します。
			MCU 設定に関連するマクロ	MCU 関連の設定は、スマート・コンフィグレータが生成し（例：パッケージタイプ）、他の設定（例：スタックサイズ）はユーザーが手動で設定します。これらのマクロを設定する前に、*r_bsp* フォルダのアプリケーションノート（*¥src¥smc_gen¥r_bsp¥doc*）を参照してください。
(2)	**general**	*r_smc_cgc.c*	*R_CGC_Create*	この API 関数は、クロックソース以外のクロックを初期化します。*r_cg_hardware_setup.c* の *R_Systeminit* は、*main()*関数を実行する前に、この関数を呼び出します。
		r_smc_cgc.h	クロックに関連するマクロ	これらのマクロは、*R_CGC_Create* のクロックの初期化に使用されます。
		r_smc_cgc_user.c	*R_CGC_Create_UserInit*	CGC 初期化後に、ユーザーが *R_CGC_Create* にコードを追加する際にこの API 関数を使用します。

コードの生成実行前の *r_bsp_config.h* は trash フォルダにバックアップされます。

6.5　端子設定

端子設定は、コンポーネントにより下記(1)から(4)に示すようなソースファイルに生成されます。

(1) {ConfigName}を使用したドライバの端子初期化

端子機能は¥src¥smc_gen¥{ConfigName}¥{ConfigName}.c の R_ConfigName_Create で初期化されます。

端子初期化コードは、main()を実行する前に処理されます。

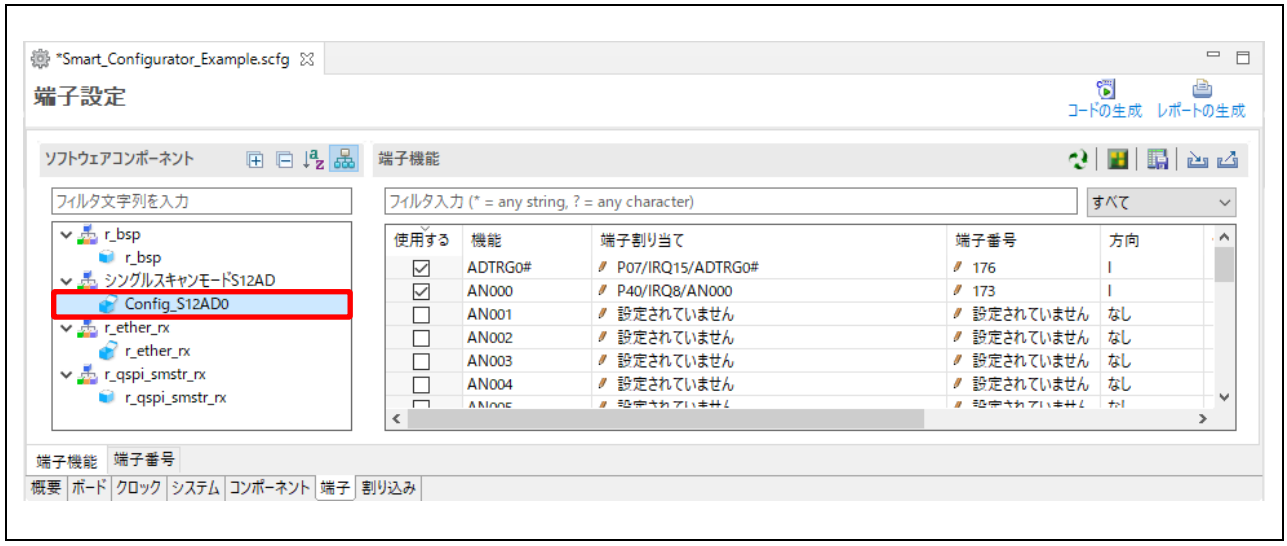

図 6-8 *Config_S12AD0* の端子設定

フォルダ	ファイル	関数	ドライバ	説明
{ConfigName}	*{ConfigName}.c*	*R_ConfigName_Create*	CG	このドライバが使用した端子を API 関数が初期化します。*main()*関数を実行する前に、*r_cg_hardware_setup.c* の *R_Systeminit* はこの関数を呼び出します。

(2) r_xxx (*2)を含むドライバの端子初期化

端子設定ソースファイル r_xxx_pinset.c は、¥src¥smc_gen¥r_pincfg フォルダに生成されます。

このファイルの API 関数は、ユーザーがアプリケーションコードで呼び出します。

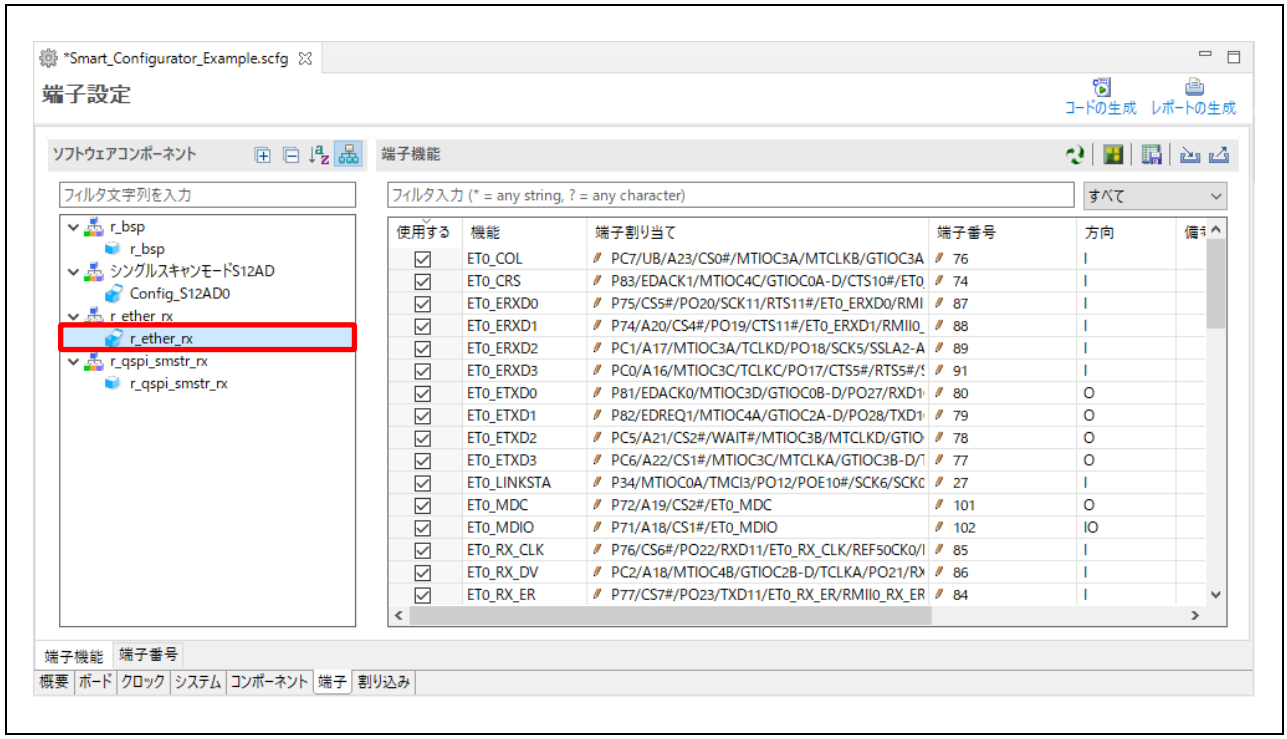

図 6-9 *r_ether_rx* の端子設定

フォルダ	ファイル	関数	ドライバ	説明
r_pincfg	*r_xxx_pinset.c* *(*2)*	*R_xxx_PinSet_xxxn* (*2、3)	FIT	このドライバで使用される端子を、API 関数が初期化します。この API 関数を呼び出す前に、対応する *r_xxx* フォルダにあるアプリケーションノート（*¥src¥smc_gen¥r_xxx¥doc* (*2)）を参照してください。

＊2：*xxx* は周辺機能名を意味します。

＊3：*n* は周辺チャネル番号を意味します。

(3) r_xxx_smstr (*4)を含むドライバの端子初期化

端子設定ヘッダファイルは、r_xxx_smstr_rx_pin_config.h の名前が付いた¥src¥smc_gen¥r_config フォルダに生成されます。

このファイルのマクロ定義は、r_xxx_smstr ソースファイルで処理されます。

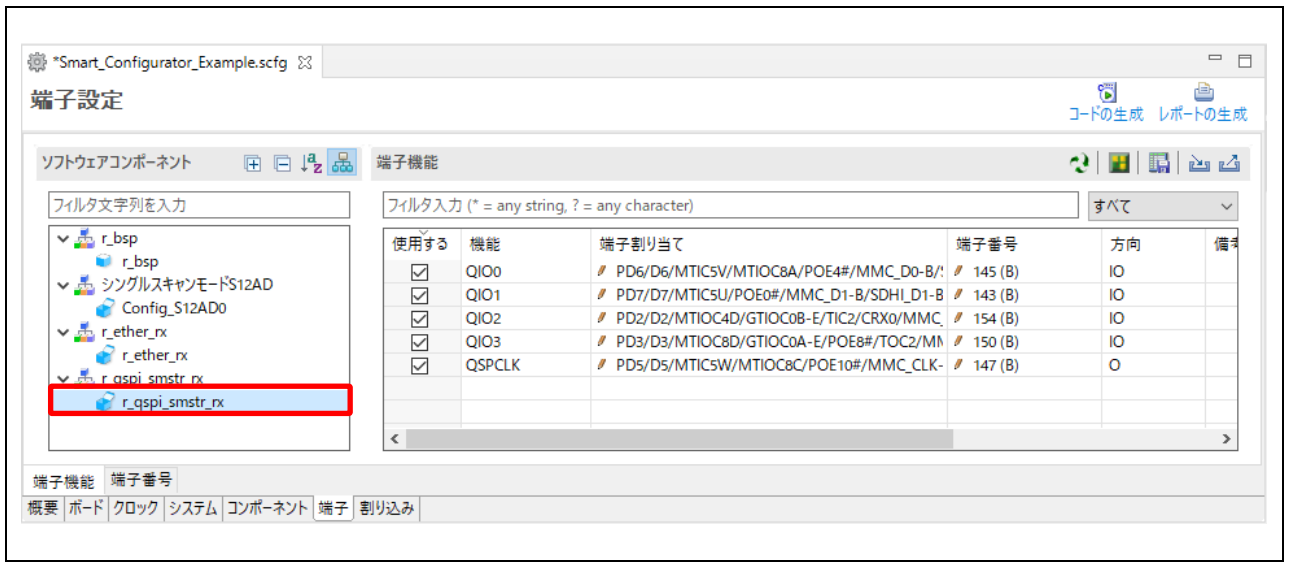

図 6-10 *r_qspi_smstr_rx* の端子設定

フォルダ	ファイル	関数	ドライバ	説明
r_config	*r_xxx_smstr_rx_pin_config.h (*4)*	-	FIT	このヘッダファイルのマクロ定義は、このドライバで使用される端子を初期化します。これらのマクロは *r_xxx_smstr* ソースファイルで呼び出されます。

＊4：*xxx* は周辺機能名を意味します。

(4) 端子初期化コードの参照

プロジェクトで使用されるすべての周辺端子機能については、¥src¥smc_gen¥r_pincfg フォルダにある Pin.c を参照してください（I/O ポート以外）。

フォルダ	ファイル	関数	ドライバ	説明
r_pincfg	*Pin.c*	*R_Pins_Create*	-	［端子］ページで設定された、I/O ポート以外の全端子機能の初期化コードを含みます。

6.6　割り込み設定

割り込みページの設定は、いくつかのソースファイルに生成されます。

r_xxx モジュールで使用される割り込みを初期化する場合、対応する¥src¥smc_gen¥r_xxx¥doc フォルダのアプリケーションノートを参照してください（xxx は周辺機能名を意味します）。

図 6-11 割り込み設定

No	項目	フォルダ	ファイル	ドライバ	説明
(1)	優先レベル	**general**	*r_smc_interrupt.c*	CG	この割り込み優先レベル設定は、グループ割り込み(*5)用です。このファイルの *R_Interrupt_Create* で初期化されます。*main()*関数を実行する前に、*r_cg_hardware_setup.c* の *R_Systeminit* はこの関数を呼び出します。
(2)	優先レベル	**{ConfigName}**	*{ConfigName}.c*	CG	この割り込み優先レベル設定は、通常割り込みと、選択型割り込み A、B(*5)用です。 このファイルの *R_ConfigName_Create* で初期化されます。*main()*関数を実行する前に、*r_cg_hardware_setup.c* の *R_Systeminit* はこの関数を呼び出します。
(3)	ベクタ番号	**r_config**	*r_bsp_interrupt_config.h*	CG FIT	［割り込み］タブの選択型割り込み A、B(*5)のベクタ番号は、このファイル上でマップされ、*r_bsp* によって処理されます。
(4)	高速割り込み	**general**	*r_smc_interrupt.c*	CG	高速割り込み設定は、このファイルの *R_Interrupt_Create* で初期化されます。*main()*関数を実行する前に、*r_cg_hardware_setup.c* の *R_Systeminit* はこの関数を呼び出します。
			r_smc_interrupt.h	CG	高速割り込みのベクタ番号はこのファイルで定義されます。 *{ConfigName}_user.c* は、高速割り込みサービスルーチンを準備するためこのマクロ定義を使用します。
(1) (2)	優先レベル	**general**	*r_smc_interrupt.h*	-	［割り込み］タブで設定される全割り込みの優先レベルは、このファイルで定義されます。 ユーザーは、アプリケーションコードにこれらのマクロ定義を使用することができます。

＊5：割り込みの種類は、ハードウェア仕様に依存します。

6.7　コンポーネント設定

6.7.1　FIT モジュール設定

1) *r_bsp* の設定

r_bsp の設定ファイルは、*r_bsp_config.h* として¥src¥smc_gen¥r_config フォルダの下に生成されます。そのファイルは、クロック初期化と他の MCU 関連の設定（例：パッケージ）を含みます。

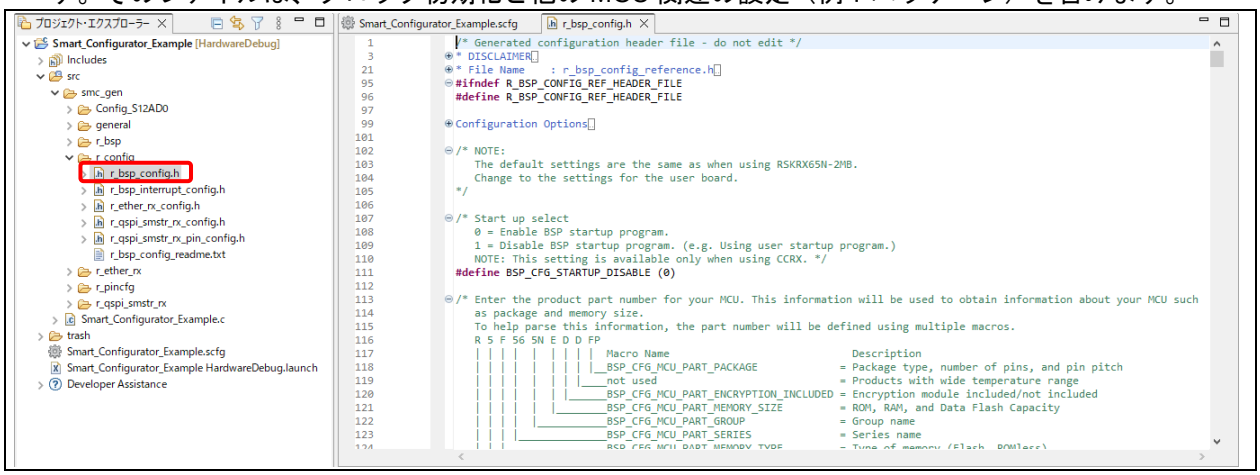

図 6-12 *r_bsp_config.h*

2) FIT モジュールの設定

プロジェクトに追加される FIT モジュールの設定ファイルは、*r_xxx_config.h* として¥src¥smc_gen¥r_config フォルダの下に生成されます（*r_xxx* は FIT モジュール名を意味します）。

［コンポーネント］ページに設定 GUI がある FIT モジュールの場合、設定はスマート・コンフィグレータによって生成されるため、*.h ファイルを手動で変更する必要はありません。

［コンポーネント］ページに構成 GUI がない FIT モジュールの場合、*.h ファイルの設定を手動で変更する必要があります。下図に示すように、*(2) コンフィグレーション*のマクロ定義値を設定する前に、*(1) 説明*を参照してください。

r_xxx_config.h の変更については、¥src¥smc_gen¥r_xxx¥doc フォルダのアプリケーションノートを参照してください。

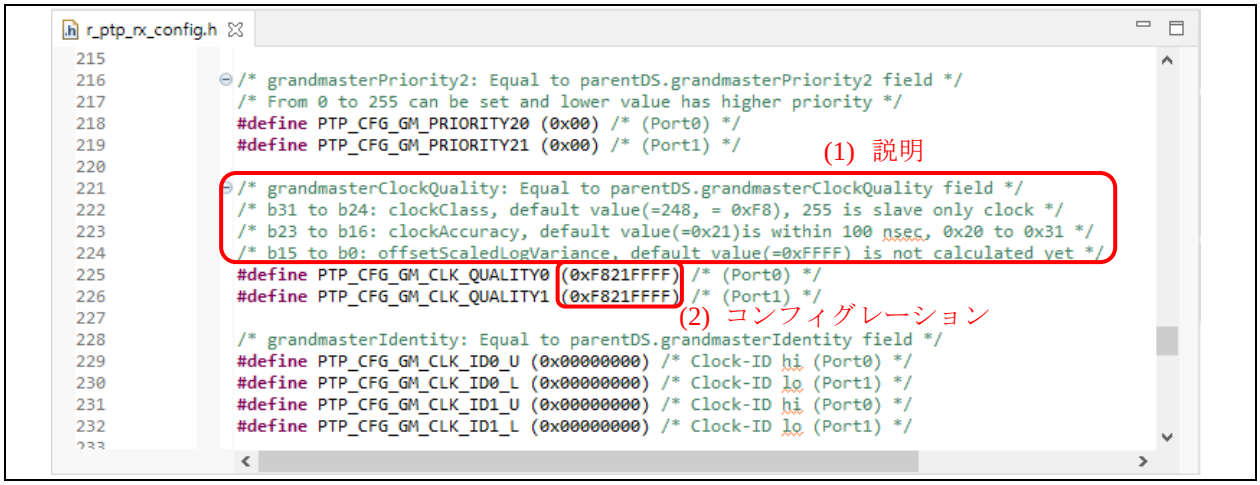

図 6-13 *r_xxx_config.h (r_ether_rx_config.h)*の例

6.7.2 FreeRTOS カーネル設定

Renesas FreeRTOS_Kernel の設定ファイルは、*FreeRTOSConfig.h* として¥src¥frtos_config フォルダに生成されます。

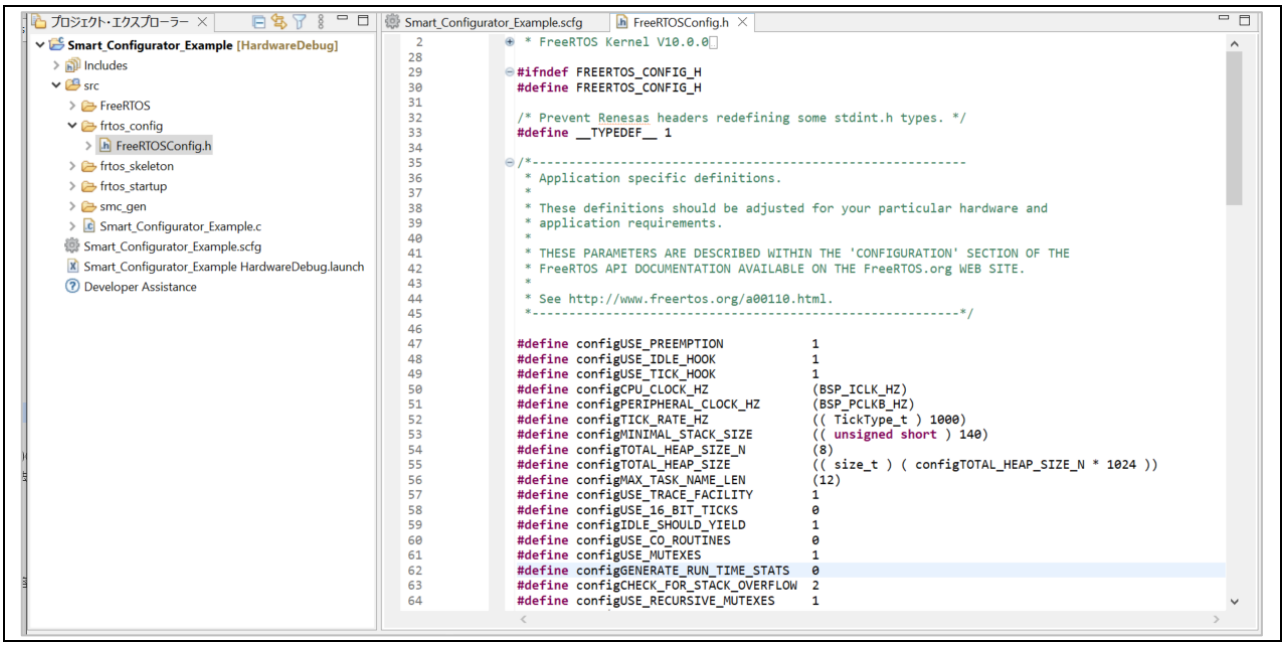

図 6-14 *FreeRTOSConfig.h*

7. ユーザープログラムの作成

スマート・コンフィグレータのコンポーネントタイプには、［Firmware Integration Technology］と［コード生成］の 2 つがあり、出力したソースファイルにカスタムコードを追加する方法が異なります。ここでは、［Firmware Integration Technology］と［コード生成］それぞれのカスタムコード追加方法について説明します。

7.1 Firmware Integration Technology(FIT)の場合のカスタムコード追加方法

コンポーネントのタイプで［Firmware Integration Technology］を選択した場合、r_config フォルダの中にある r_xxx_config.h のコンフィグレーションオプションを設定します。コンフィグレーションオプションの設定については、プロジェクト・ツリーに追加される FIT モジュール（r_xxx）のアプリケーションノート（doc フォルダ）を参照ください。

ソースコード出力の際、対象ファイルが存在すれば、ファイルの内容を保護します。

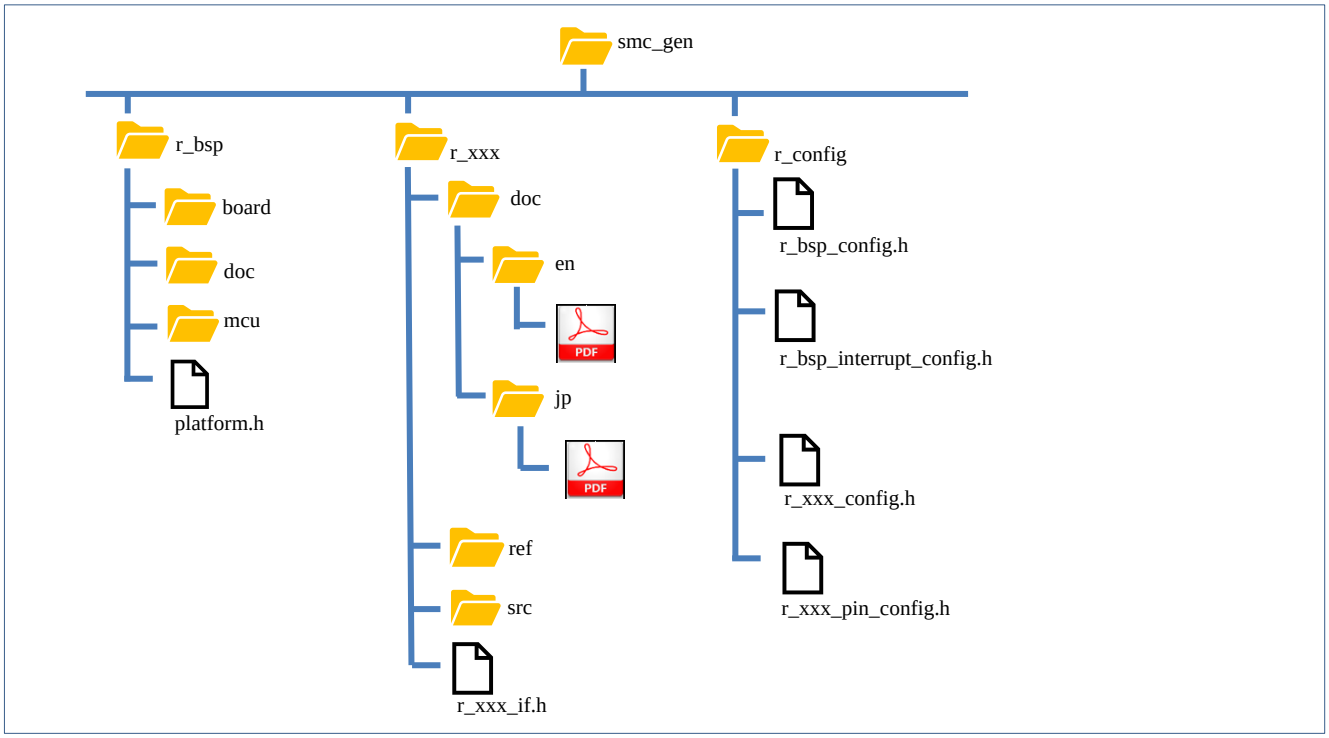

図 7-1 FIT モジュールのファイル構成

7.2　コード生成の場合のカスタムコード追加方法

コンポーネントのタイプで［コード生成］を選択した場合、ソースコード出力の際、同一ファイルが存在する場合には、以下のコメントで囲まれた部分に限り、該当ファイルをマージします。

```
/* Start user code for xxxx. Do not edit comment generated here */

/* End user code. Do not edit comment generated here */
```

［コード生成］の場合、特定の周辺機能ごとに 3 つのファイルを生成します。デフォルトのファイル名は、「Config_xxx.h」、「Config_xxx.c」、「Config_xxx_user.c」となり、xxx は周辺機能を表します。（例えば、コンペアマッチタイマ(リソース CMT3)の場合、xxx は“CMT3”と名付けられます。）カスタムコードを追加するためのコメントは、3 つのファイルそれぞれの先頭と最後に設けられる他、「Config_xxx_user.c」にある周辺機能の割り込み関数内にも追加されます。以下に CMT3 の例（Config_CMT3_user.c）を示します。

```
/***********************************************************************
Pragma directive
***********************************************************************/
/* Start user code for pragma. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */

/***********************************************************************
Includes
***********************************************************************/
#include "r_cg_macrodriver.h"
#include "r_cg_userdefine.h"
#include "Config_CMT3.h"
/* Start user code for include. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */

/***********************************************************************
Global variables and functions
***********************************************************************/
/* Start user code for global. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */

/***********************************************************************
* Function Name: R_Config_CMT3_Create_UserInit
* Description  : This function adds user code after initializing the CMT3 channel
* Arguments    : None
* Return Value : None
***********************************************************************/

void R_Config_CMT3_Create_UserInit(void)
{
    /* Start user code for user init. Do not edit comment generated here */
    /* End user code. Do not edit comment generated here */
}
```

```
/***********************************************************************
* Function Name: r_Config_CMT3_cmi3_interrupt
* Description  : This function is CMI3 interrupt service routine
* Arguments    : None
* Return Value : None
***********************************************************************/

#if FAST_INTERRUPT_VECTOR == VECT_PERIB_INTB129
#pragma interrupt r_Config_CMT3_cmi3_interrupt(vect=VECT(PERIB,INTB129),fint)
#else
#pragma interrupt r_Config_CMT3_cmi3_interrupt(vect=VECT(PERIB,INTB129))
#endif
static void r_Config_CMT3_cmi3_interrupt(void)
{
    /* Start user code for r_Config_CMT3_cmi3_interrupt. Do not edit comment
generated here */
    /* End user code. Do not edit comment generated here */
}

/* Start user code for adding. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
```

7.3　ユーザーアプリケーションコードの使用

生成された FIT およびコード生成のコードを使用するには、以下の手順で行います。

詳細については、「11 参考ドキュメント」の「アプリケーションノート」を参照してください。

(1) *{Project name}*.c ファイルを開き、使用するモジュールのヘッダファイルをインクルードコードに追加します。

FIT の場合は r_xxx_if.h です。コード生成の場合、自動的に r_smc_entry.h に追加されます。

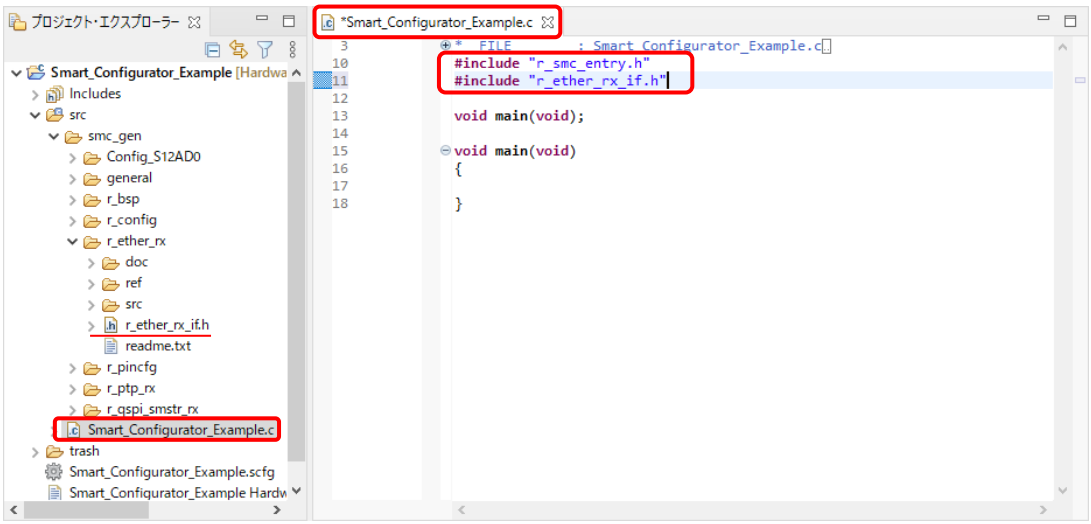

図 7-2 ヘッダファイルの追加

(2) main 関数で生成された関数を呼び出し、アプリケーションコードを追加します。

コード生成の場合、端子初期化を含むドライバ初期化関数（R_ConfigName_Create）は、デフォルトで r_cg_hardware_setup.c の R_Systeminit 関数で呼び出されます。ドライバ固有の処理を実行するには、アプリケーションコードを追加する必要があります。 例えば、開始（R_*ConfigName*_Start）および停止（R_*ConfigName*_Stop）。

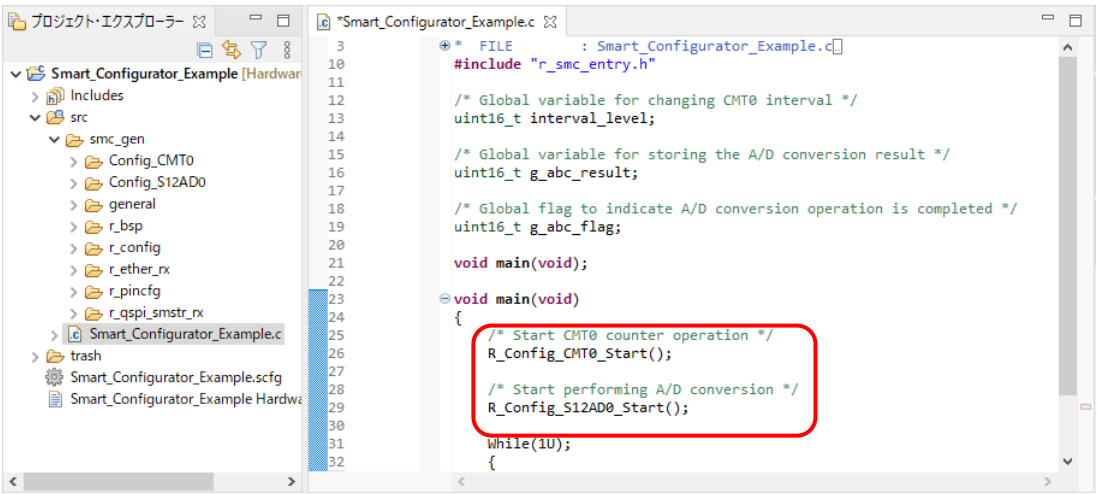

図 7-3 コード生成関数コール

FIT モジュールの場合は、対応するアプリケーションノートの「API 関数」の章に記載されている例を参照してください。 アプリケーションノートは、各 FIT モジュールの下の［doc］フォルダにあります。

8. 生成ソースのバックアップ

スマート・コンフィグレータには、ソースコードのバックアップ機能があります。

［コードの生成］ ボタンをクリックしてコードの生成を行うとき、スマート・コンフィグレータは生成ソースのバックアップを生成します。<Date-and-Time>は、コード生成を実行しバックアップフォルダを作成した日時です。

<ProjectDir>¥trash¥<Date-and-Time>

9. レポートの生成

スマート・コンフィグレータは、設定情報をレポートで提供します。レポートを生成するには、以下の手順で行います。

9.1 全設定内容レポート

スマート・コンフィグレータビューで［レポートの生成］ボタンをクリックし、レポートを出力します。出力形式は 、PDF とテキストの 2 種類が選択可能です。

図 9-1 全設定内容レポート出力(txt 形式)

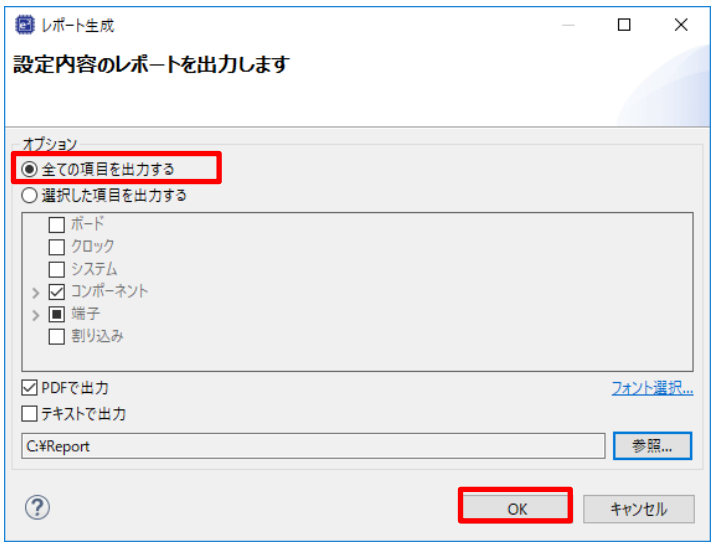

図 9-2 レポート出力ダイアログ

9.2　端子機能リスト、端子番号リスト設定内容(csv 形式)

スマート・コンフィグレータビューの［端子］ページで［.csv ファイルにリストを保存］ボタンをクリックし、端子機能リスト、端子番号リスト設定内容を出力します。

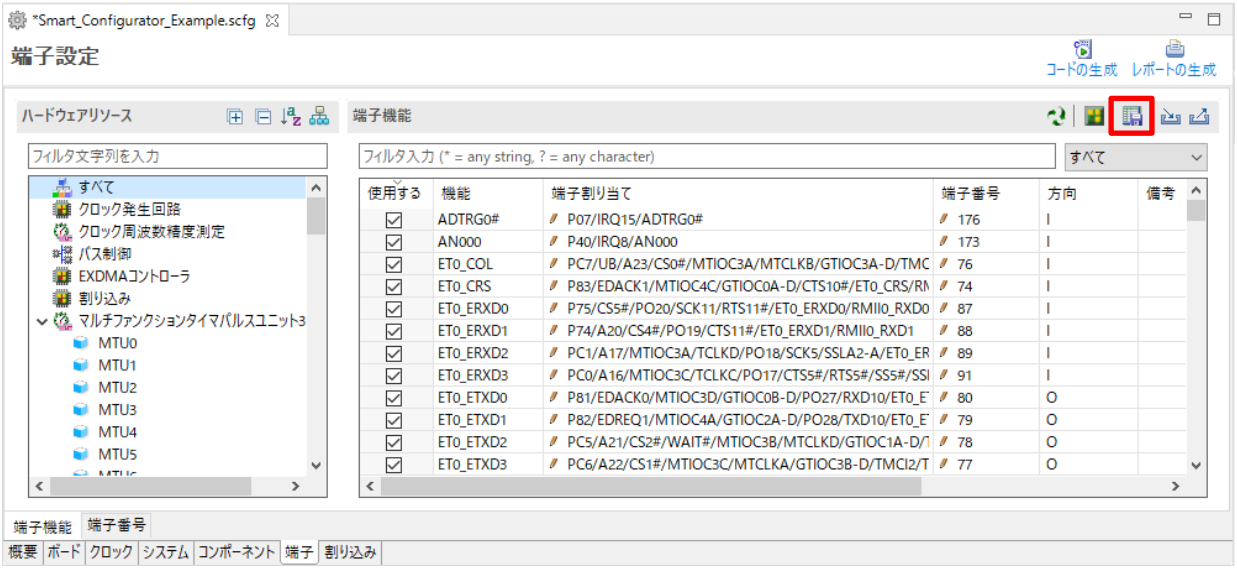

図 9-3 端子機能リスト、端子番号リスト出力(csv 形式)

9.3　MCU/MPU パッケージ図(png 形式)

MCU/MPU パッケージビューの［端子配置図を保存］ボタンをクリックし、MCU/MPU パッケージ図を出力します。

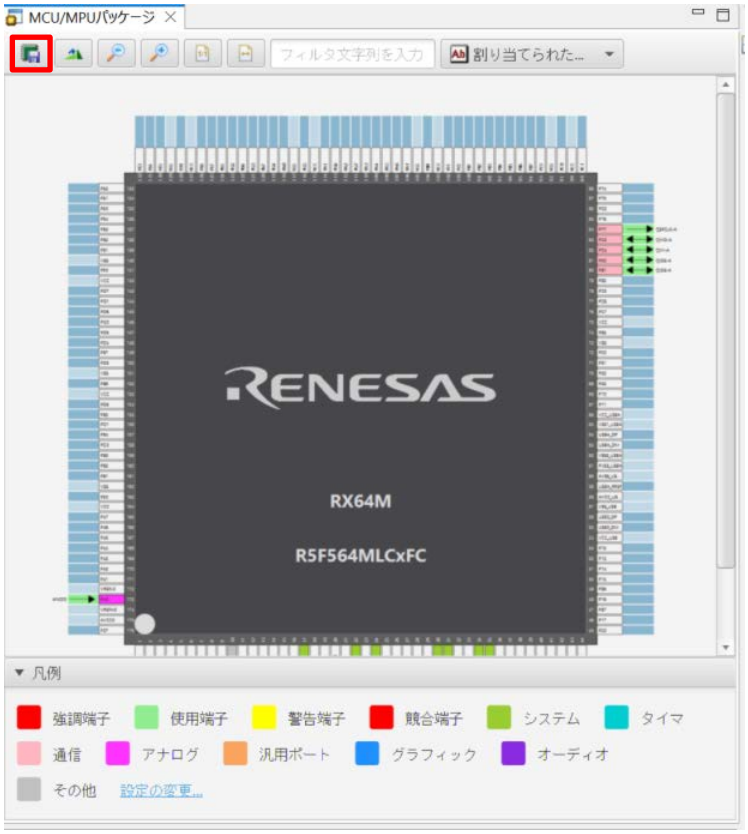

図 9-4 MCU/MPU パッケージ図出力(png 形式)

10. ユーザーコード保護機能

e2 studio 2023-01 Smart Configurator for RX プラグイン以降のバージョンより、新たなユーザーコード保護機能をサポートしました。図 10-1 の指定タグを追加することで、任意の位置にユーザーコードを追加できるようになりました。追加されたユーザーコードはコード生成時に保護されます。

ユーザーコード保護機能は「コード生成コンポーネント」が生成したファイルのみサポート対象となります。

10.1 ユーザーコード保護機能の指定タグ

ユーザーコード保護機能を使用する場合、図 10-1 のように、/* Start user code */ と /* End user code */ を挿入し、このタグの間にユーザーコードを追加してください。指定タグが完全に一致しない場合は、保護されません。

```
/* Start user code */
コメントの間にユーザーコードを追加
/* End user code */
```

図 10-1 ユーザーコード保護機能の指定タグ

10.2 ユーザーコード保護機能の使用例

図 10-2 に示すように、図 10-1 の指定タグを使用し、CMT モジュールの Create 関数の中に新しいユーザーコードを挿入します。その後、CMT の GUI での設定を更新し、再びコード生成すると、挿入されたユーザーコードが新たに生成されたファイルに自動的にマージされます。

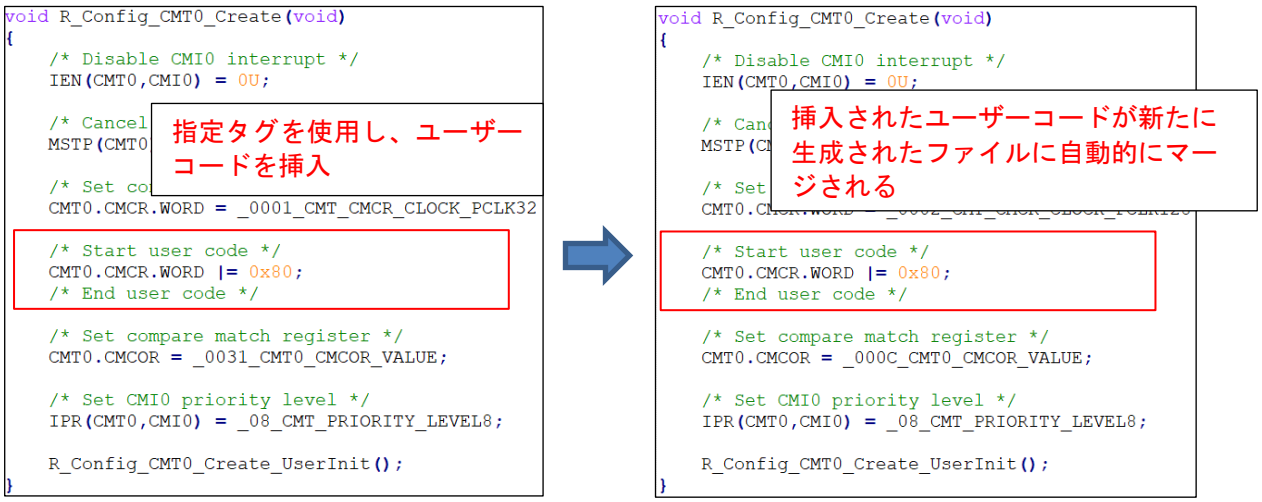

図 10-2 ユーザーコードの保護機能

10.3　競合発生時の対応方法

10.3.1　競合の発生条件

挿入したユーザーコードの前後にある生成コードに変更がある場合(GUI の設定変更、スマート・コンフィグレータのバージョンアップなど)、図 10-3 のように生成コードに競合が発生します。

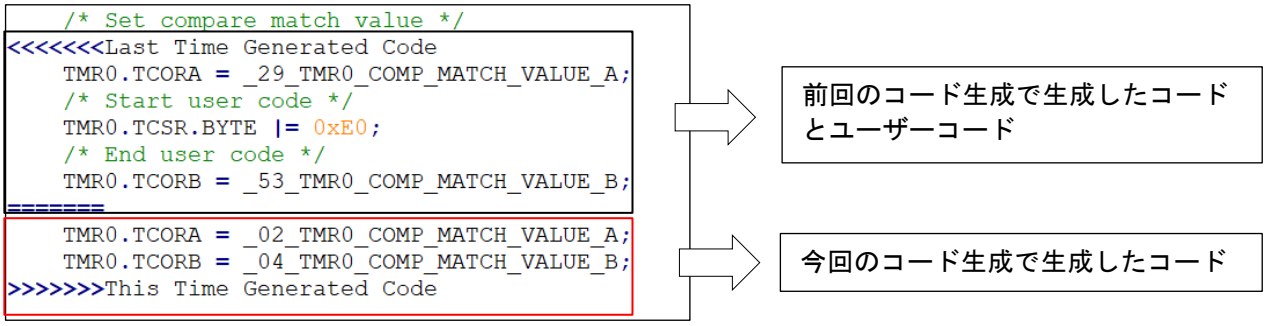

図 10-3 生成された競合コード

競合が発生した場合、コンソールに図 10-4 のようなメッセージが表示されます。

図 10-4 競合のメッセージ

10.3.2　競合の解決方法

競合を解決するには、競合が発生したファイルを開いて、下記の手順に従って手動でコードを修正してください。

1) 図 10-5 に示すように、「Last Time Generated Code」のユーザーコードをコピーして、「This Time Generated Code」の新しい位置に貼り付けてください。

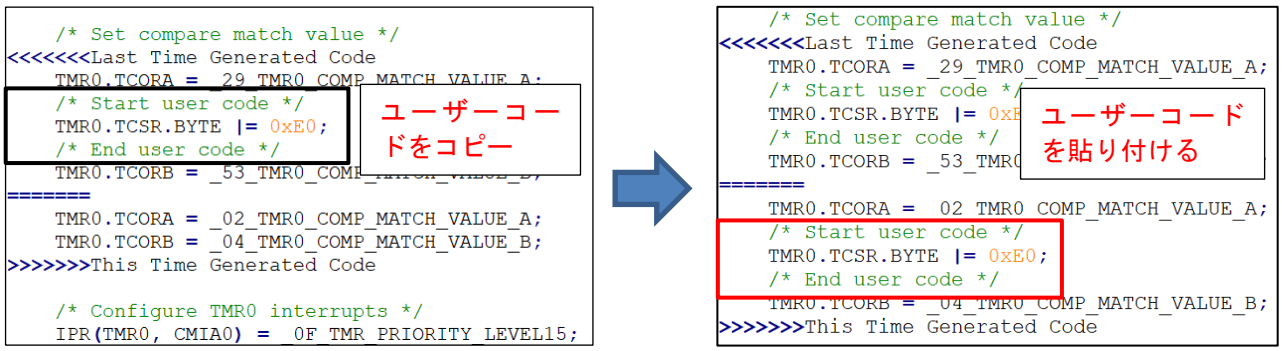

図 10-5 生成された競合コード

2) 図 10-6 のように古い生成コードと競合に関するコメント（<<<<<<<Last Time Generated Code、=======と>>>>>>>This Time Generated Code）を削除してください。

```
/* Set compare match value */
TMR0.TCORA = _02_TMR0_COMP_MATCH_VALUE_A;
/* Start user code */
TMR0.TCSR.BYTE |= 0xE0;
/* End user code */
TMR0.TCORB = _04_TMR0_COMP_MATCH_VALUE_B;
```

図 10-6 競合を解決した後のコード

別の競合の解決方法：

1) コンソールメッセージのファイル名をクリックし、比較ビューアーを開きます。

図 10-7 コンソールのエラーメッセージ

2) 比較ビューアーが開いた後、左側のコードを右側のコードに反映するか、手動で右側のコードを編集できます。

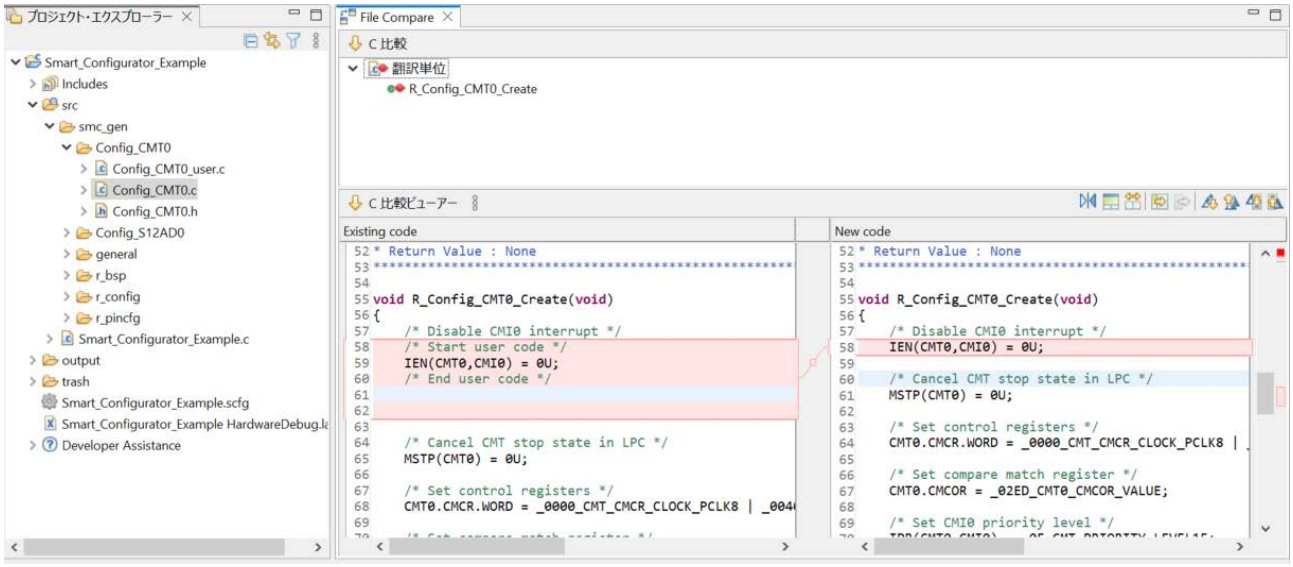

図 10-8 比較ビューアーの競合コード

11. ヘルプ

11.1 ヘルプ

スマート・コンフィグレータの詳細情報は、e² studio メニュー上のヘルプを参照ください。

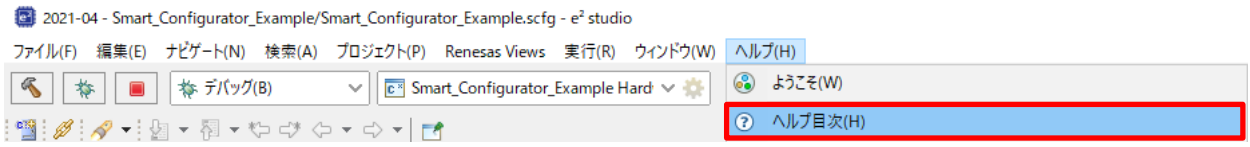

図 11-1 ヘルプ表示

「概要」ページの アイコンクリックにより、ヘルプを参照できます。

図 11-2 クイックスタート

11.2 Developer assistance

Developer assistance は、API 情報と API の呼び出し例を表示ツールです。ユーザは、自動生成されたコードを簡単に使用することができます。

図 11-3 Developer assistance

(1) コンポーネントを追加し、コードを生成します。

(2) 生成されたファイルが存在することを確認します。

(3) プロジェクトエクスプローラーで、[Developer Assistance]を展開し、コンポーネントをダブルクリックします。

(4) API 情報と呼び出し例を読むことができます。

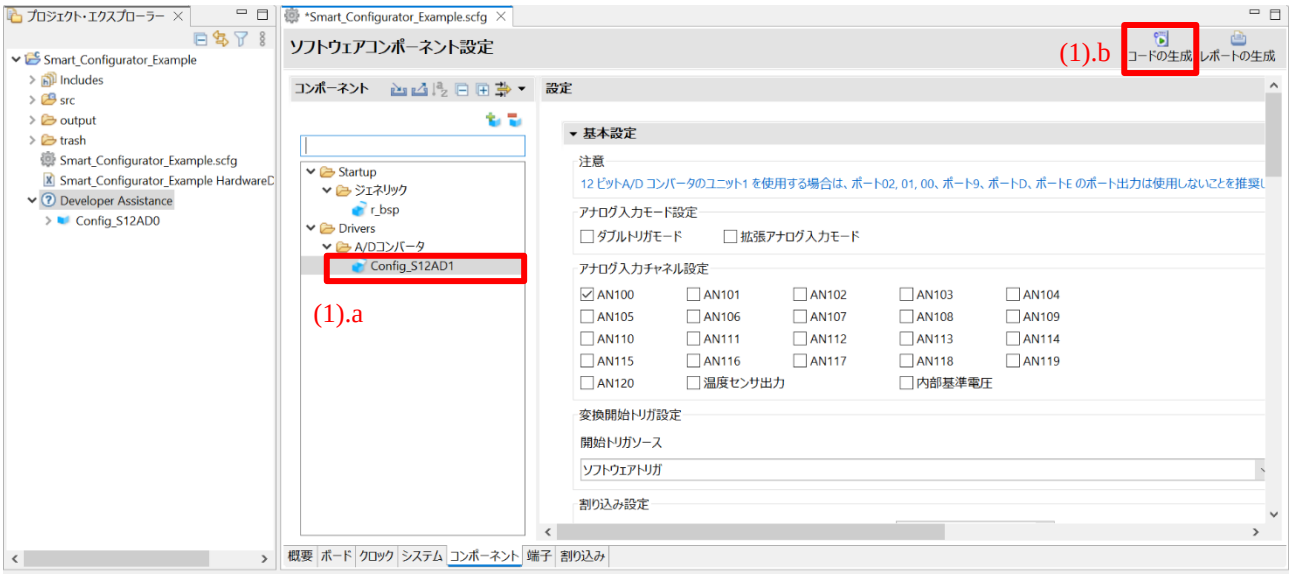

図 11-4　コンポーネントの追加とコード生成

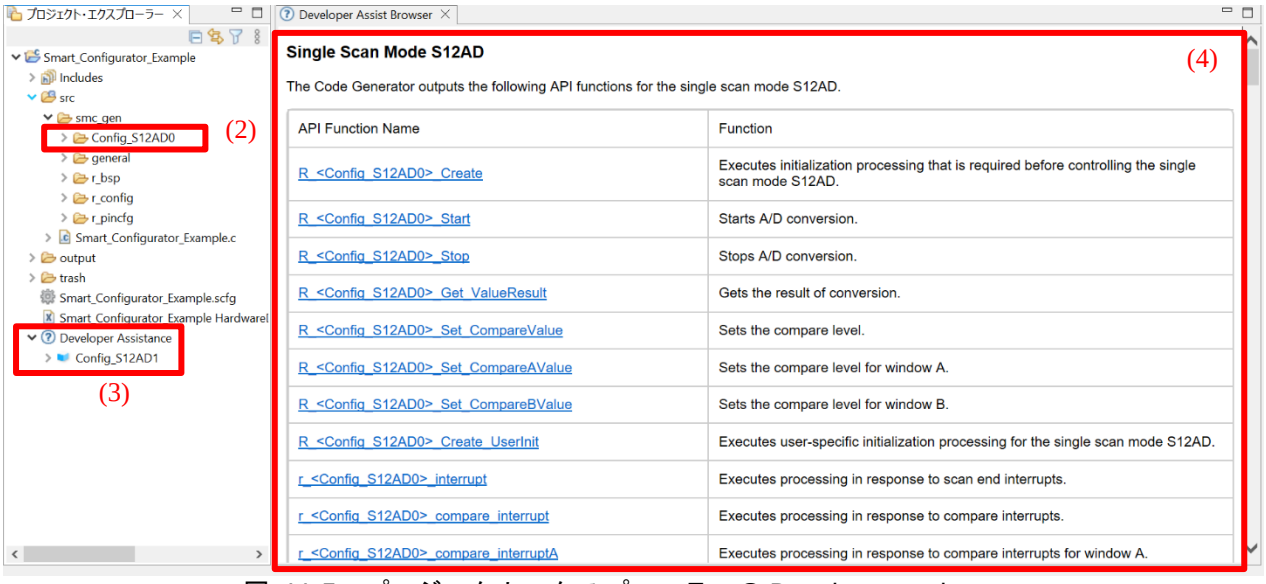

図 11-5　プロジェクトエクスプローラーの Developer assistance

12. 参考ドキュメント

ユーザーズマニュアル：ハードウェア

最新版をルネサスエレクトロニクスホームページから入手してください。

テクニカルアップデート／テクニカルニュース

最新の情報をルネサスエレクトロニクスホームページから入手してください。

ユーザーズマニュアル：開発環境

統合開発環境 e2 studio ユーザーズマニュアル 入門ガイド (R20UT4374)

CC-RX コンパイラ ユーザーズマニュアル (R20UT3248)

RX ファミリ Renesas FreeRTOS アプリケーションノート (R01AN4307)

e2 studio Partner RTOS Aware Debugging for RX (R20AN0586)

（最新版をルネサスエレクトロニクスホームページから入手してください。）

アプリケーションノート

スマート・コンフィグレータ Application Examples: Ethernet (R20AN0495)

スマート・コンフィグレータ Application Examples: CMT, A/D, SCI, DMA, USB (R20AN0496)

（最新版をルネサスエレクトロニクスホームページから入手してください。）

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ
http://www.renesas.com

お問合せ先
http://www.renesas.com/contact/

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2017.07.31	ー	新規作成
1.10	2018.03.20	ー	e² studio v6.2 へのアップデート
		4	動作環境の変更
		8	図 2.4 ツールチェーン、デバイス、デバッグ設定の選択 更新
		15	図 3.6 コンポーネントの追加 更新
		16	図 3.7 コード生成コンポーネントの追加 更新
		19	図 3.12 FIT モジュールの追加 更新
		26	3.22 r_ether_rx の端子設定 更新
		30	注釈の追加
		36	4.1 生成されたファイル構造 の表にある r_bsp フォルダの説明修正
		50	8.1 トラブルシューティング 追加
		53	9.1 応用例 更新
1.20	2018.11.01	ー	e² studio v7.0.0 へのアップデート
		4	1. 概要 更新
		10	図 3-1 操作手順 追加
		11	3.3 プロジェクトの保存先 追加
		12	3.4 ウィンドウ 追加
		20	4.1.3 ボード設定のエクスポート、4.1.4 ボード設定のインポート 追加
		22	図 4-7 コンポーネントページ 追加
		34	図 4-26 端子ページ（端子機能）、図-27 端子ページ（端子番号） 追加
		40	図 4-35 割り込みページ 追加
		43	図 5-1 ソースファイルの生成、図 5-2 プロジェクト・エクスプローラー内のソースファイル 追加
		56	7. ユーザープログラムの作成 追加
		61	9.2 端子機能リスト、端子番号リスト設定内容(csv 形式)、9.3 MCU パッケージ図(png 形式) 追加
		62	10.1 ヘルプ 追加
1.30	2019.01.07	ー	e² studio v7.3.0 へのアップデート
		9	2.2 FreeRTOS プロジェクトの作成 追加
		21	3.4.3 MCU パッケージビュー 更新
		23	4.1.1 デバイス選択 更新
		31	4.3.4 FIT サンプルプロジェクトのダウンロードとインポート 追加
		40	4.3.9 FIT ソフトウェアコンポーネントの設定 更新
		41	4.3.10 FIT ソフトウェアコンポーネントのバージョン変更 追加
		43	4.3.11 FreeRTOS カーネルの設定 追加
		44	4.3.12 コンポーネントの基本設定 追加
		45	図 4-41 端子ページ（端子番号）更新
		49	4.4.5 ボード端子設定情報を使用した端子設定 追加 4.4.6 端子のフィルタ機能 追加
		53	4.6 MCU マイグレーション機能 追加
		71	6.6.2 FreeRTOS カーネル設定 追加

Rev.	発行日	改訂内容	
		ページ	ポイント
1.40	2021.06.21	-	e² studio 2021-04 にアップデート
		5	2. プロジェクトの作成 更新
		21	4.1 ボード選択 更新
		30	4.3.4 FIT モジュールのダウンロード 更新
		31	4.3.5 FIT ドライバまたはミドルウェアの追加方法 更新
		37	4.3.9 FIT ソフトウェアコンポーネントの設定 更新
		38	4.3.10 FIT ソフトウェアコンポーネントのバージョン変更 更新
		40	4.3.11 グレーアウト・コンポーネントの更新 追加
		41	4.3.12 RTOS カーネルの設定 追加
		42	4.3.13 RTOS オブジェクトの設定 追加
		43	4.3.14 RTOS ライブラリの設定 追加
		44	4.3.15 アナログフロントエンドコンポーネントの設定 追加
		46	4.3.16 モータコンポーネントの設定 追加
		56	4.4.8 端子エラー／警告設定 追加
		64	5. 競合の管理 更新
		67	6.2 ソース生成先の設定 追加
		81	7.3 ユーザーアプリケーションコードの使用 追加
1.50	2023.04.16	1	RX スマート・コンフィグレータの URL 更新
		87-89	10 ユーザーコード保護機能 追加
1.60	2024.04.16	-	図を e² studio 2024-01 に更新
		15-19	2.3 Blinky プロジェクトの作成 追加
		25	3.4.3 MCU/MPU パッケージビュー 更新
		33	4.4.2 ソフトウェアコンポーネントの削除 更新
		56-57	4.4.16 モータコンポーネントの設定 更新
		58	4.4.17 コンポーネントの基本設定 更新
		67-68	4.5.9 シンボリック名設定 追加
		72-73	4.6.3 多重割り込み設定 追加
		103-104	10.3.2 競合の解決方法 更新
		106	11.2 Developer assistance 追加

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 未使用端子の処理
   【注意】未使用端子は、本文の「未使用端子の処理」に従って処理してください。
   CMOS製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI周辺のノイズが印加され、LSI内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。
2. 電源投入時の処置
   【注意】電源投入時は，製品の状態は不定です。
   電源投入時には、LSIの内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。
   外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。
   同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. リザーブアドレス（予約領域）のアクセス禁止
   【注意】リザーブアドレス（予約領域）のアクセスを禁止します。
   アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
4. クロックについて
   【注意】リセット時は、クロックが安定した後、リセットを解除してください。
   プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。
   リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子
   （または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
5. 製品間の相違について
   【注意】型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。
   同じグループのマイコンでも型名が違うと、内部ROM、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ輻射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

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。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、お客様の責任において、お客様の機器・システムを設計ください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品または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またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 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. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング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。
6. 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）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等

   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、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（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（「脆弱性問題」といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因しまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めていますが、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、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他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。
10. 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。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 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。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
12. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
13. 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。
14. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注 1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配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。

(Rev.5.0-1　2020.10)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。