RL78 スマート・コンフィグレータ
ユーザーガイド: e² studio 編

R20AN0579JJ0105
Rev.1.05
2025.10.20

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 RL78 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 2025-10 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/rl78-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。SW 統合機能を使用したミドルウェアのインポート、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。

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、3 種類のソフトウェア・コンポーネント（コード生成（CG）、Graphical Configurator と RL78 Software Integration System）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。

- コード生成（DTC、A/D コンバータ、割り込みコントローラなど)
  CG ドライバは、DTC、AD コンバータ、割り込みコントローラ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・コンポーネントの組み込みが便利です。

- グラフィカル・コンフィグレータ（SMS、ELCL）
  グラフィカル・コンフィグレータ・モジュールは、他のドライバ設定に比べてグラフィカルな GUI を提供することで、複雑な構成の設定を容易にします。SNOOZE モード・シーケンサ（SMS）とロジック＆イベント・リンク・コントローラ（ELCL）のソフトウェア・コンポーネントを提供しています。

- RL78 Software Integration System（静電容量センサユニット（CTSU2L）など）
  RL78 Software Integration System モジュールは、ドライバ、ミドルウェア、アプリケーション SW のソフトウェア・コンポーネントで、コードを生成するための簡単な GUI を提供します。

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た C/C++プロジェクトの生成手順を、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。

(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++ Project] - [Renesas RL78] 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。

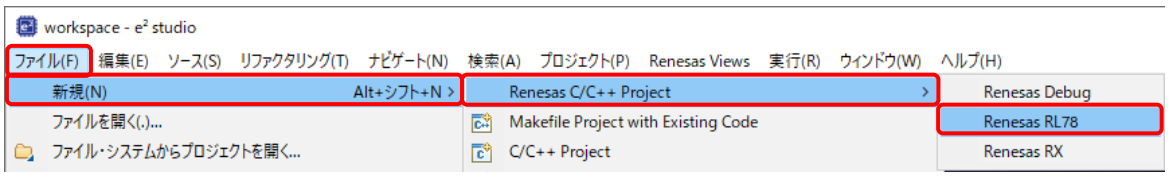

図 2-1 新規プロジェクトの作成

(2) プロジェクト作成ウィザードで、[Renesas CC-RL C Executable Project] または [LLVM for Renesas RL78 C/C++ Executable Project] を選択し、［次へ］ボタンをクリックします。

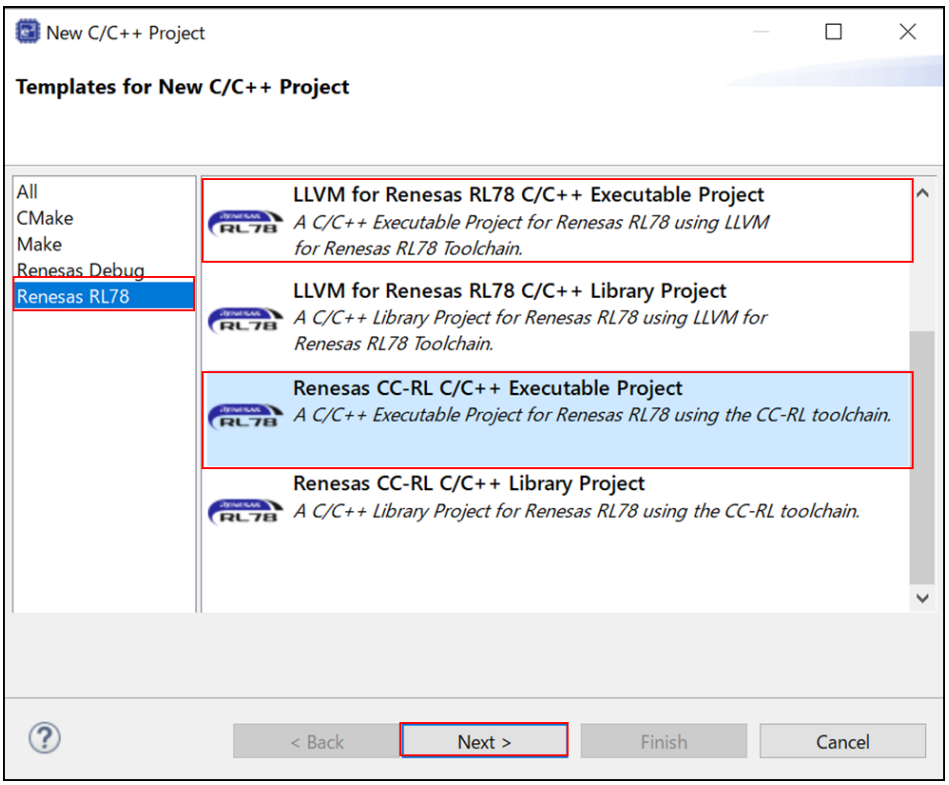

図 2-2 新規 C/C++プロジェクトのテンプレート

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

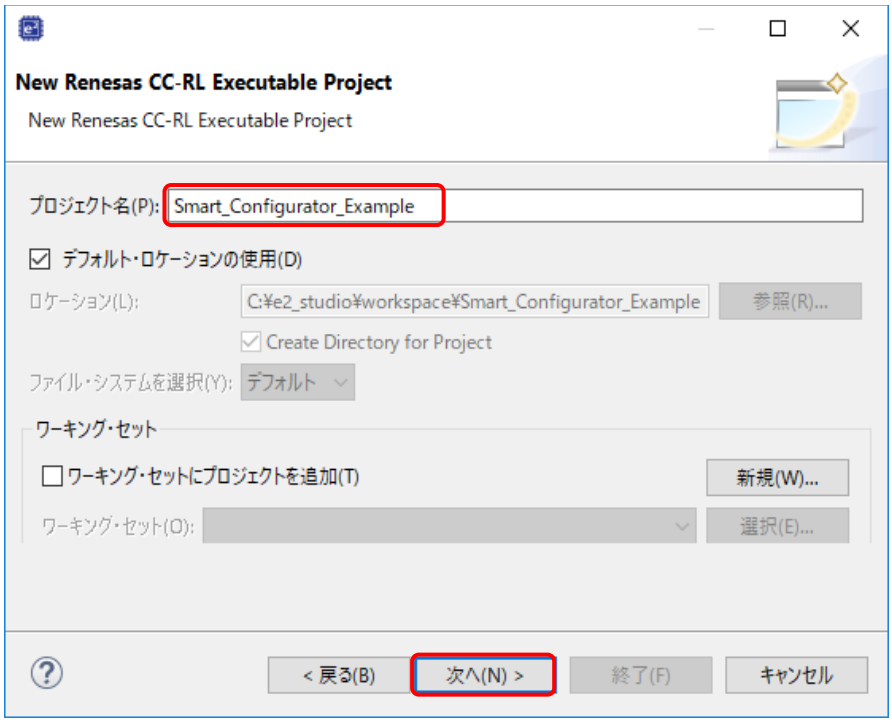

図 2-3 New Renesas CC-RL executable project の作成

(4) ツールチェーン、デバイス、デバッグ設定を選択します。[次へ] をクリックします。

（例： ターゲット・デバイス：RL78/G23 - 128pin（型名: R7F100GSNxFB））

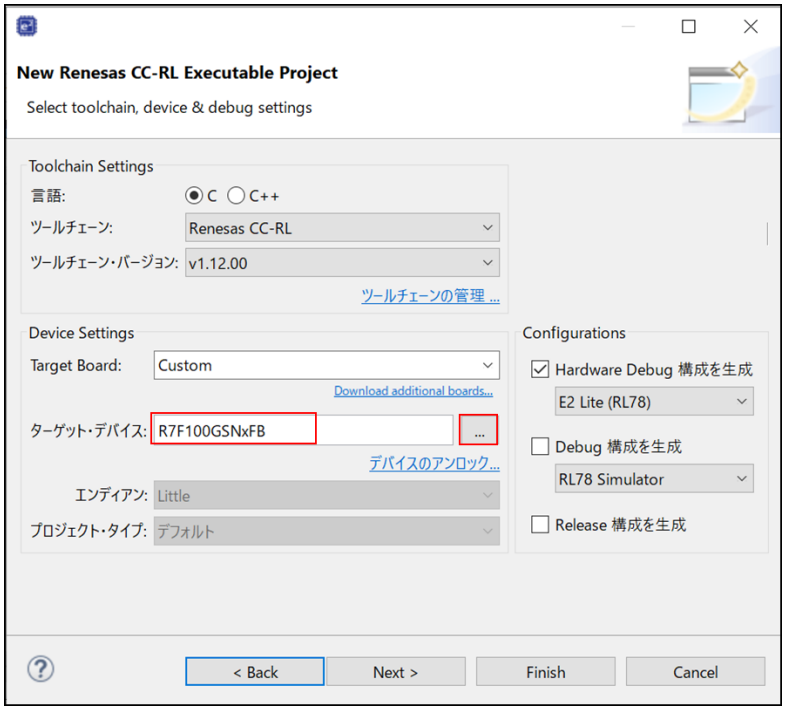

図 2-4 ツールチェーン、デバイス、デバッグ設定の選択

(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。

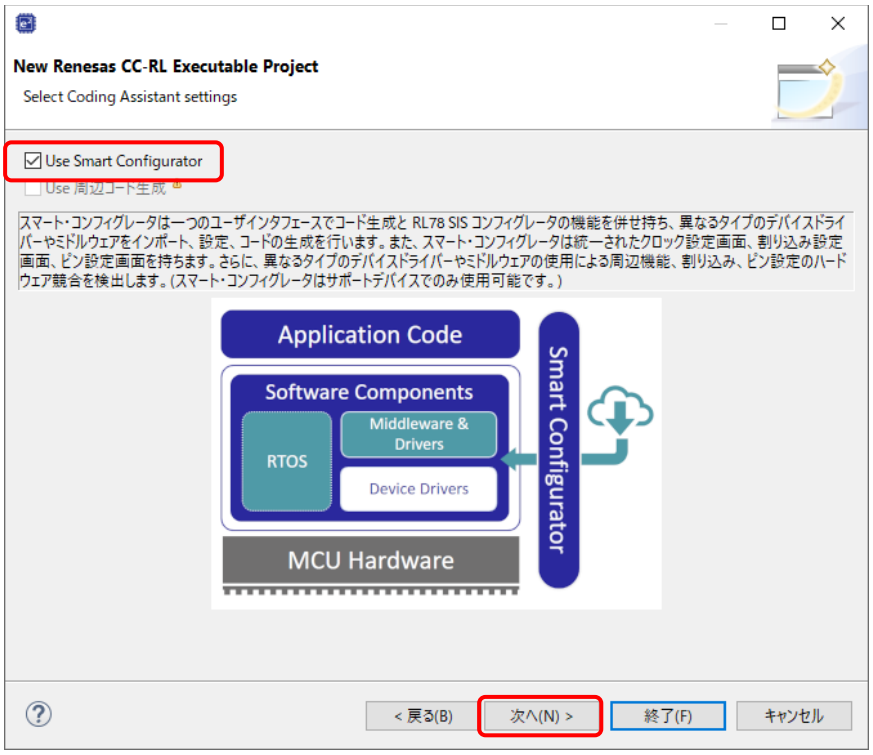

図 2-5 コーディング・アシストツールの選択

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

(6) [プロジェクトテンプレートの選択] ダイアログで、[Bare Metal - Minimal] または [Bare Metal - Blinky] を選択し、[終了] をクリックします。

図 2-6 プロジェクトテンプレートの選択

【注】 [Bare Metal - Blinky] は、(4) の [Target Board] で「Custom」以外の LED リソースを持つボードが選択された場合に表示されます。「Custom」ボード選択時は、[Bare Metal - Minimal] のみが表示されます。

(7)　プロジェクト作成の完了を待ちます。

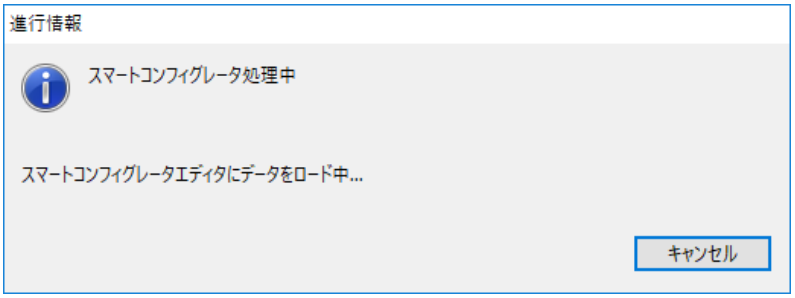

図 2-7 プロジェクト作成の処理

(8)　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。

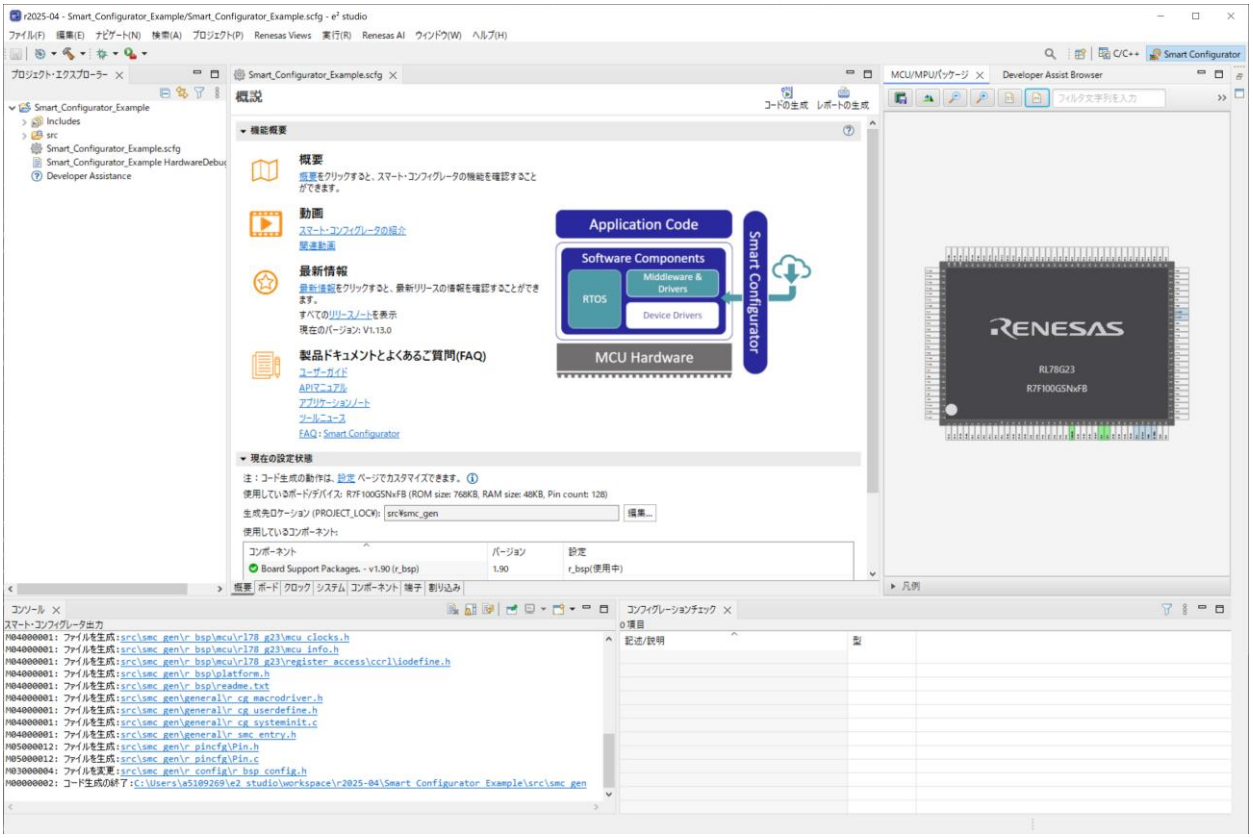

図 2-8［スマート・コンフィグレータ］パースペクティブ

3. スマート・コンフィグレータ 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。

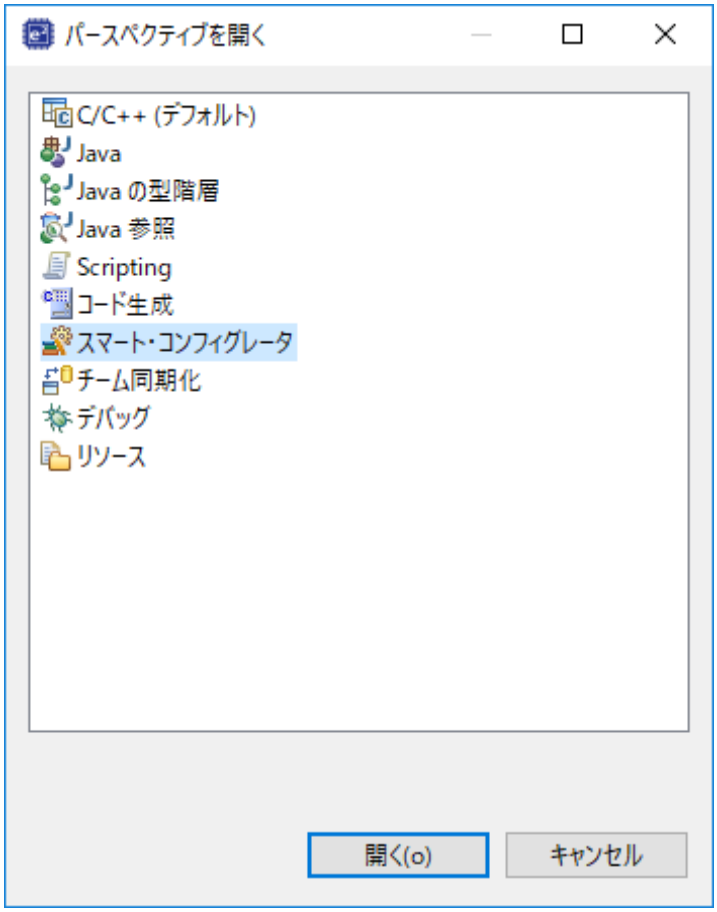

図 3-1 [スマート・コンフィグレータ] パースペクティブを開く

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。

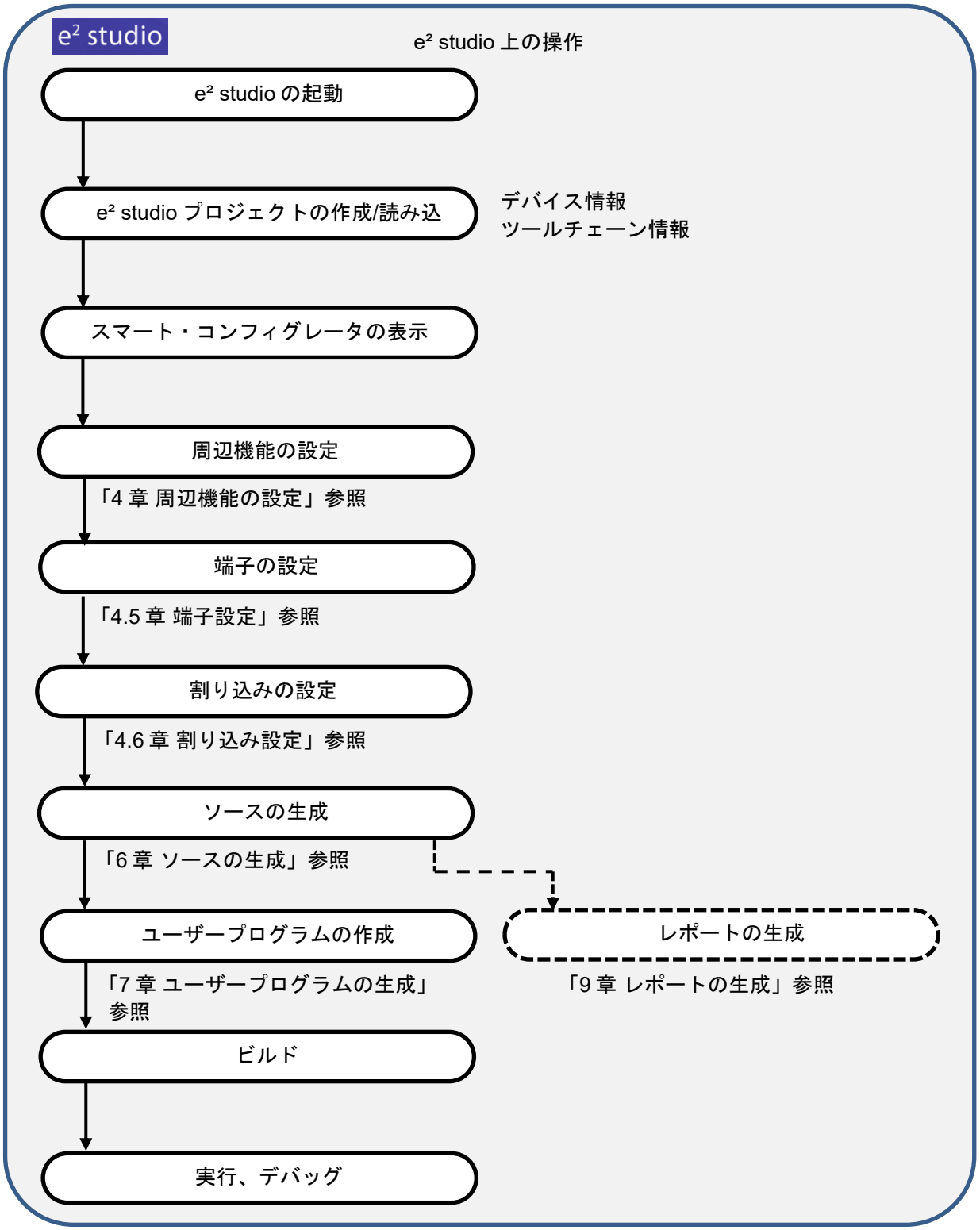

図 3-2 操作手順

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。

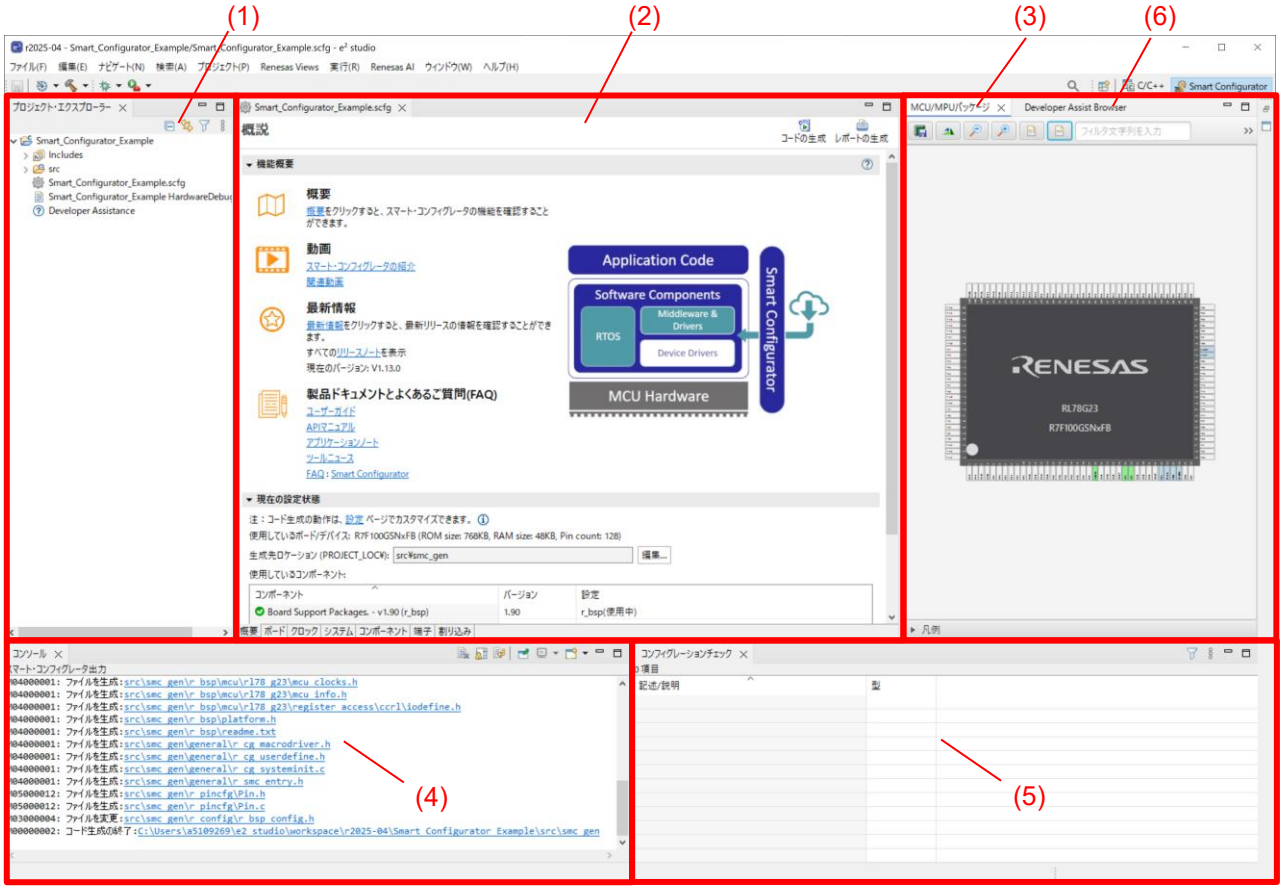

図 3-3［スマート・コンフィグレータ］パースペクティブ

(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

(6) Developer Assist Browser

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。

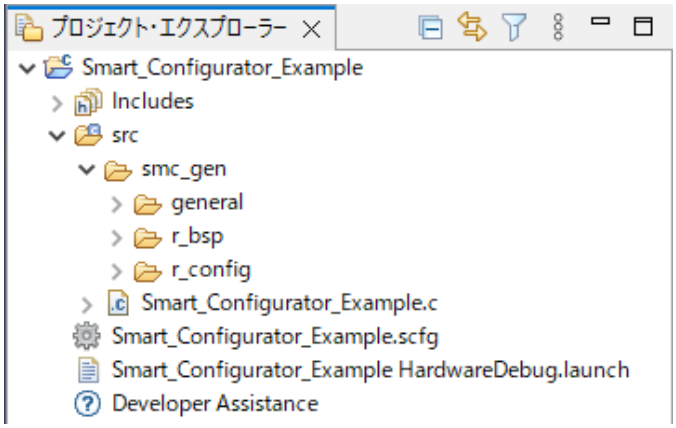

図 3-4　プロジェクト・エクスプローラー

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 6 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。

MCU/MPU パッケージビューは、[割り当てられた機能] 、[ボード機能] 、[シンボリック名] の 3 種類を切り替えることができます。

- [割り当てられた機能] は、端子設定の割り当て状態を表示します。
- [ボード機能] は、ボードの初期設定情報を表示します。
  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 ボード設定、4.5.7 ボード端子設定情報を使用した端子設定を参照ください。）
- [シンボリック名] は、ユーザーが端子に定義したシンボル名を表示します。シンボリック名のマクロ定義は、Pin.h ファイルにポート読み出しまたは書き込みの関数とともに生成します。

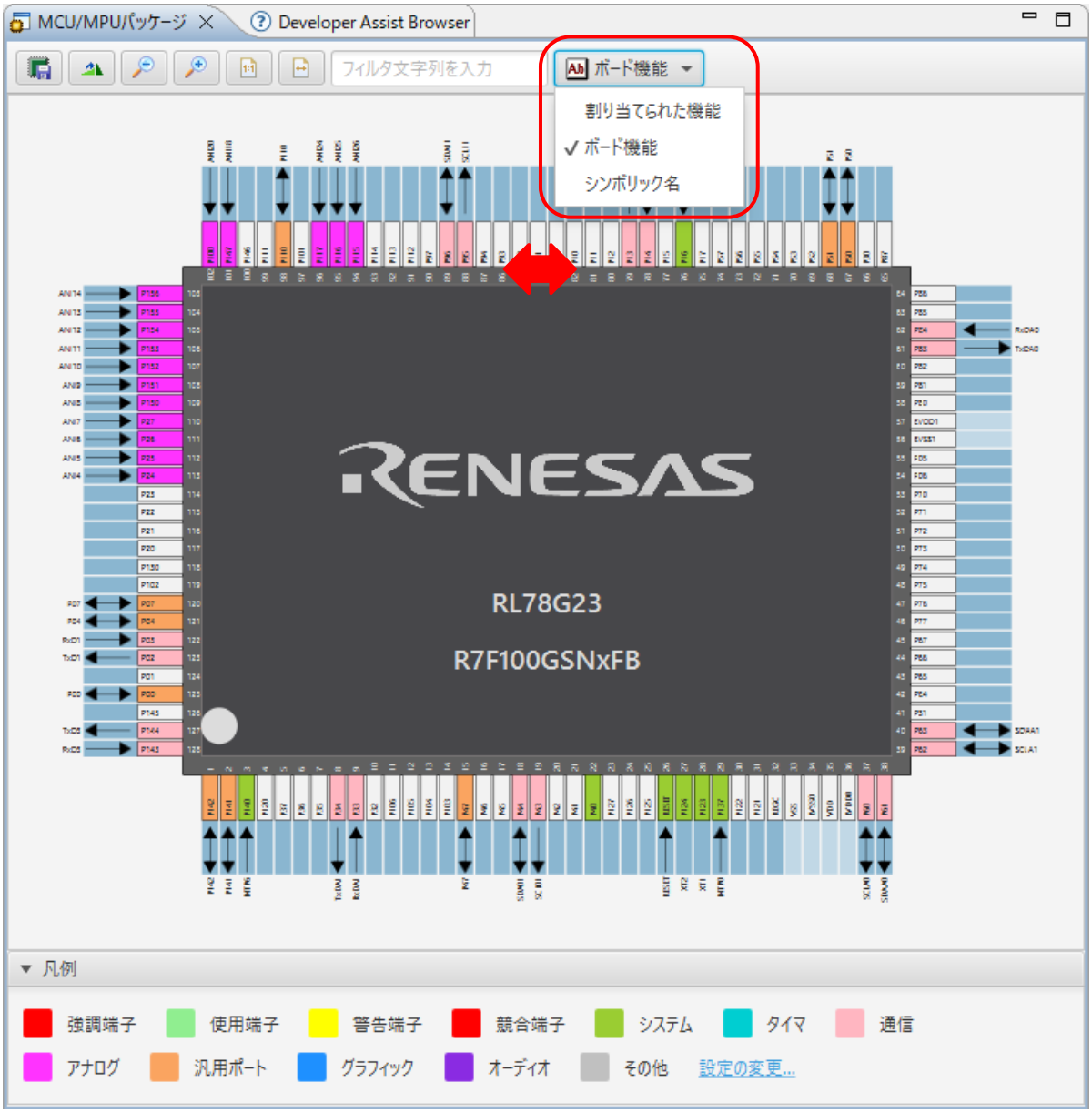

図 3-6 MCU/MPU パッケージビュー

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 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。

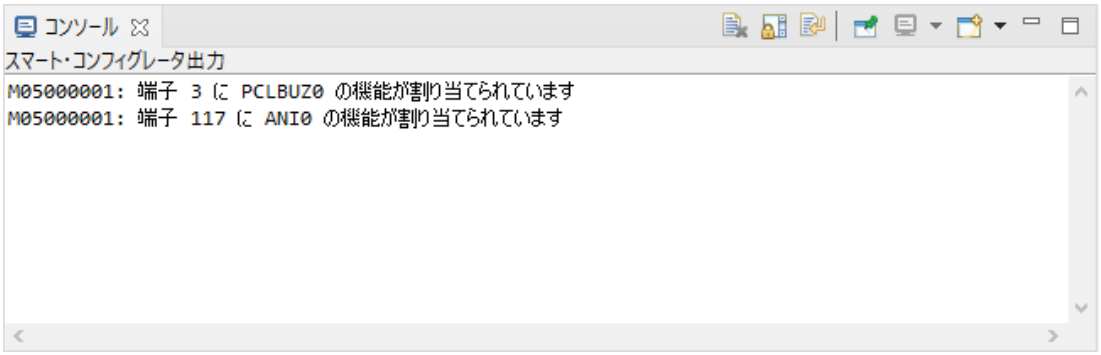

図 3-7 コンソールビュー

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

コンフィグレーションチェックビューには、ドライバが使用する割り込み、周辺機器、端子設定で競合が発生した場合の詳細が表示されます。

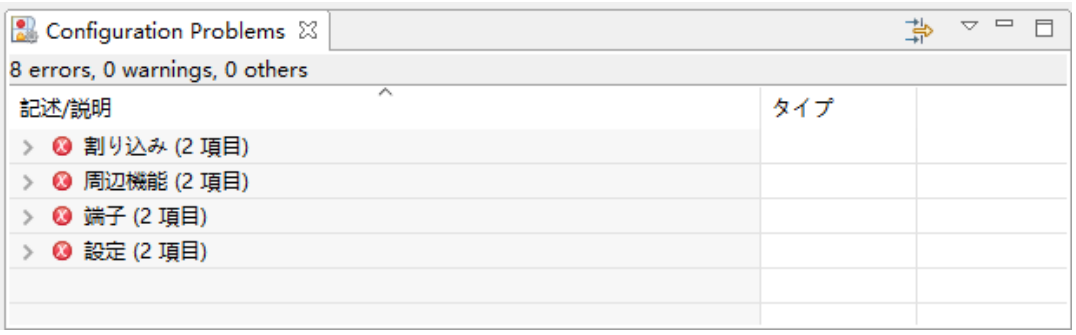

図 3-8 コンフィグレーションチェックビュー

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。

3.4.6　Developer Assist Browser

[Developer Assist Browser] ビューは、スマート・コンフィグレータ開発者支援機能のビューサービスです。API 情報をナビゲートおよび参照し、[Copy] コンテキスト・メニューを使用して、コード生成コンポーネント API 使用例のコードスニペットを C/C++エディタに貼り付けることができます。

このビューが開かれていない場合、e² studio メニュー上の [Renesas Views] - [スマート・コンフィグレータ] - [Developer Assist Browser] を選択してください。

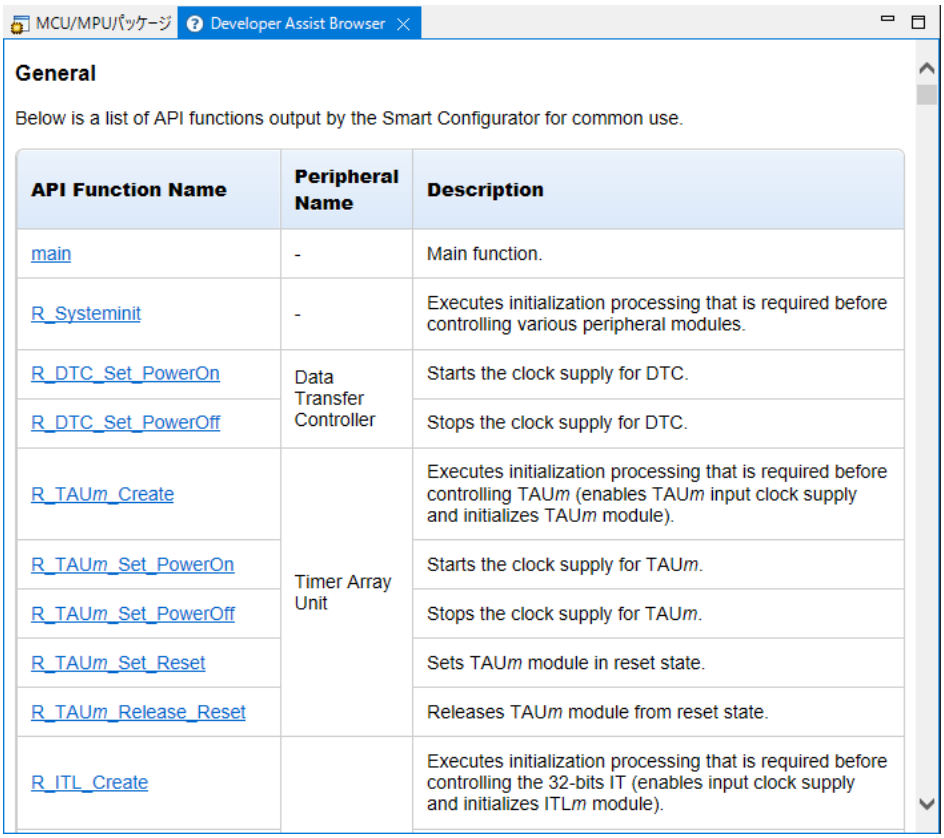

図 3-9 **Developer Assist Browser ビュー**

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　デバイス選択

[...] ボタンをクリックすると、デバイスが選択できます。

「4.7 MCU マイグレーション機能」の手順に従いデバイス変更を行ってください。

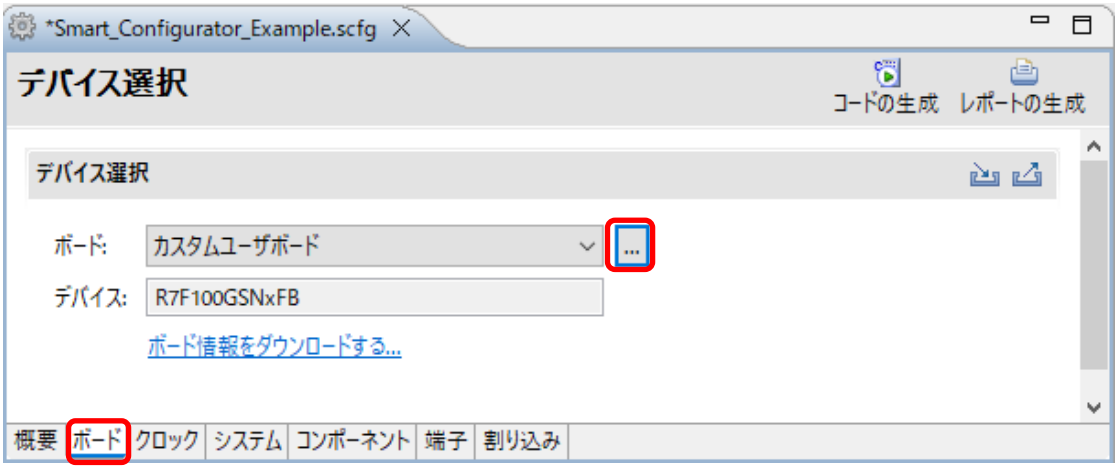

図 4-1 デバイス選択

4.1.2　ボード選択

[...] をクリックすると、リストから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
- オンチップ・デバッグ動作設定とエミュレータ設定

上記ボード設定情報は、Board Description File (.bdf) に定義されています。ルネサス製ボード(ファストプロトタイピングボード等) 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。詳細は 4.7MCU マイグレーション機能を参照してください。

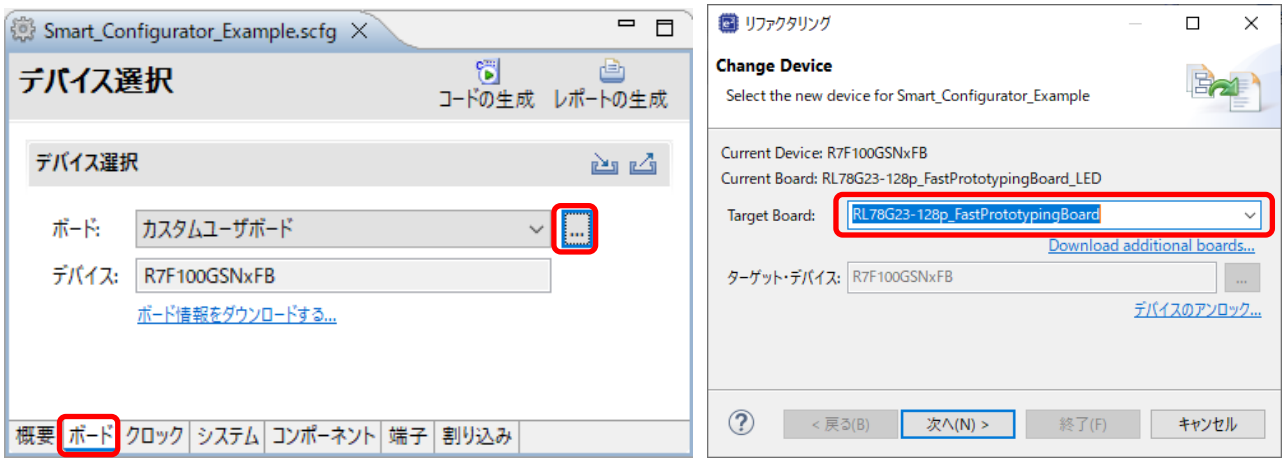

図 4-2 ボード選択

[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。

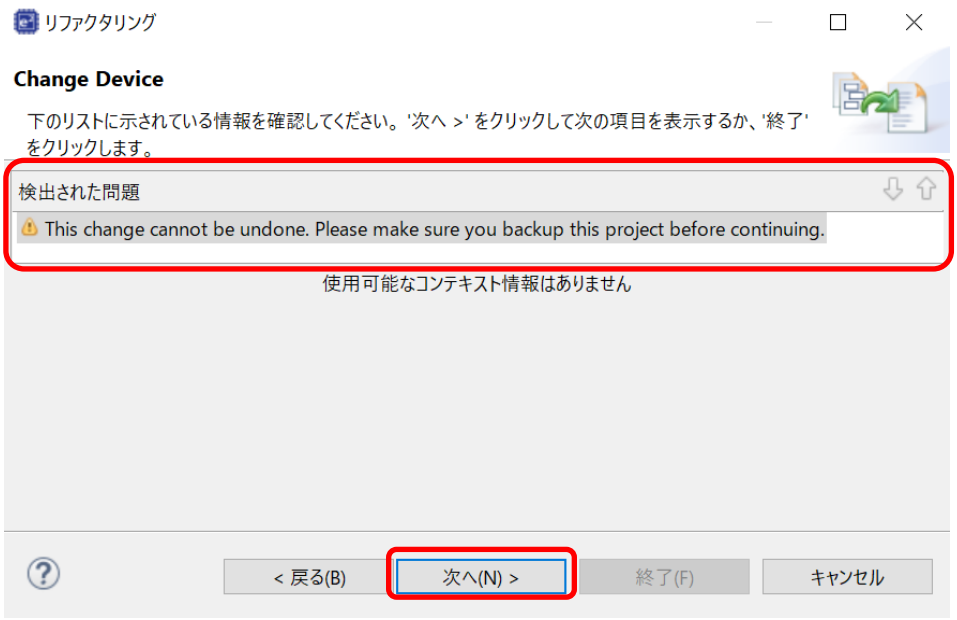

図 4-3 検出された問題

表 4-1　デバイス変更の[検出された問題]の表示一覧

メッセージ	説明
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復元できませんので、プロジェクトのバックアップ後に実行してください。

[実行される変更] で、変更する項目を選択して [終了] をクリックします。

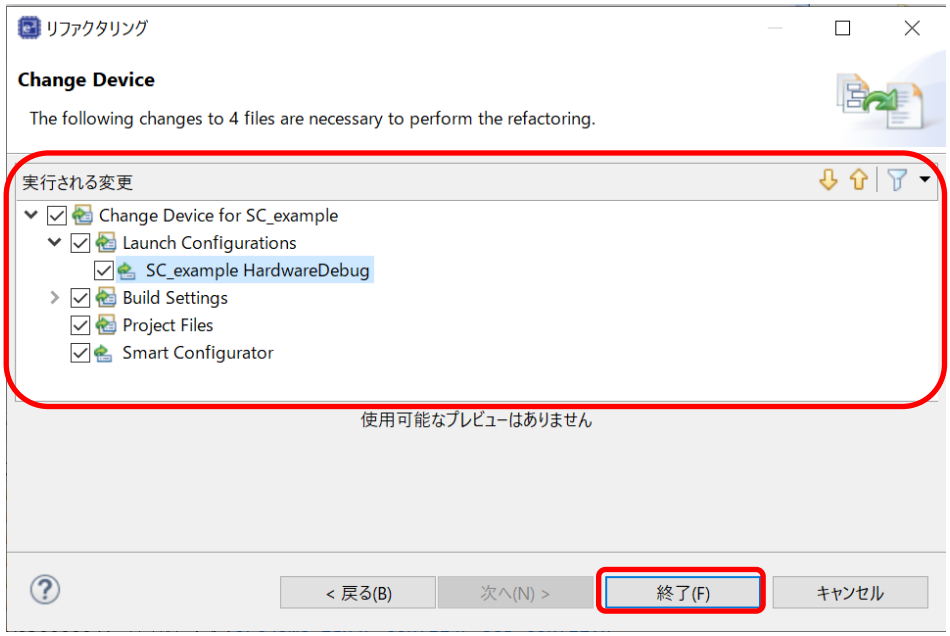

図 4-4 実行される変更項目確認

4.1.3 ボード設定のエクスポート

ボード設定後に今後の参考としてボード設定のエクスポートができます。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。

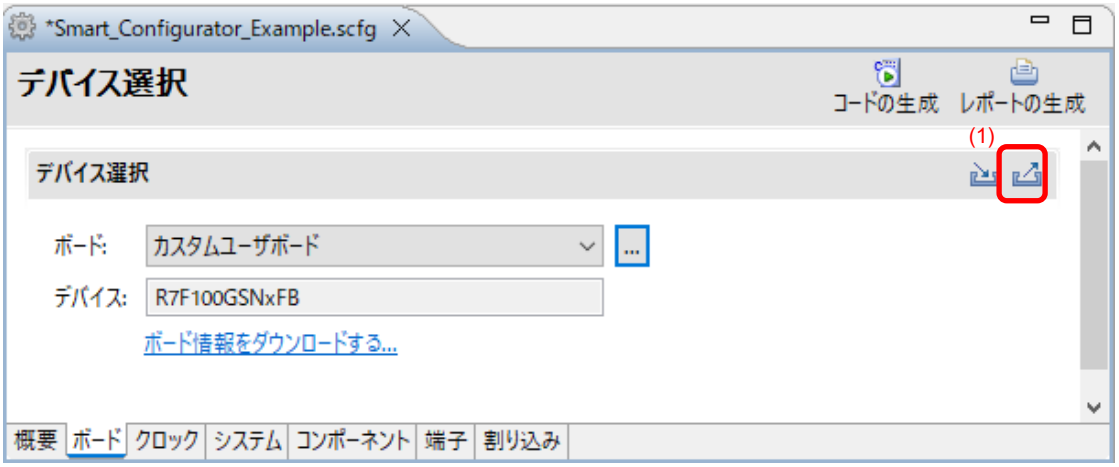

図 4-5 ボード設定のエクスポート（bdf 形式）

4.1.4　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。

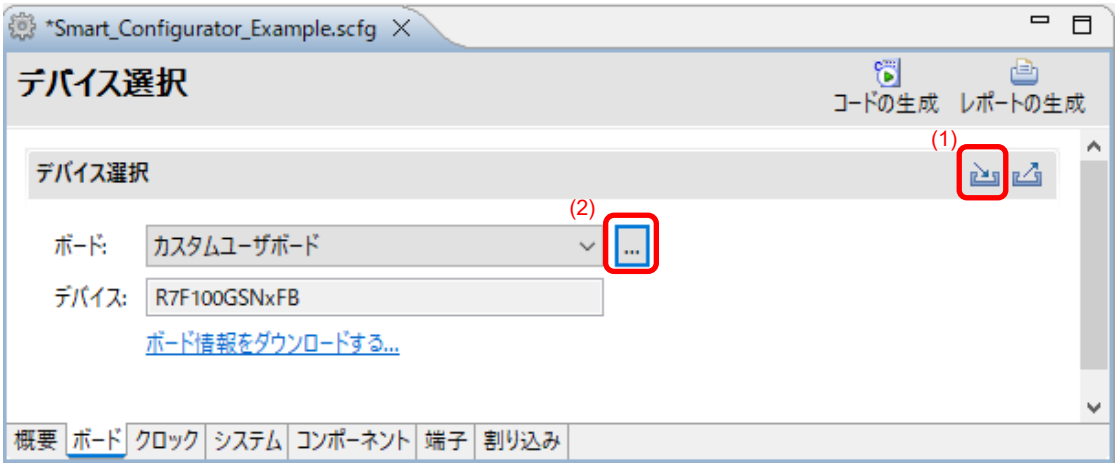

図 4-6 ボード設定のインポート（bdf 形式）

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他のプロジェクトでも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)　動作モードと EVDD 設定を指定します。

(2)　デバイス操作に必要なクロックを選択します（デフォルトは、高速オンチップ・オシレータが選択されています）。

(3)　ボードの仕様に従って各クロックの周波数を指定します（一部の内部クロックでは周波数が固定されていいますので注意してください）

(4)　マルチプレクサ・シンボルで、出力クロックのためのクロック・ソースを選択します。

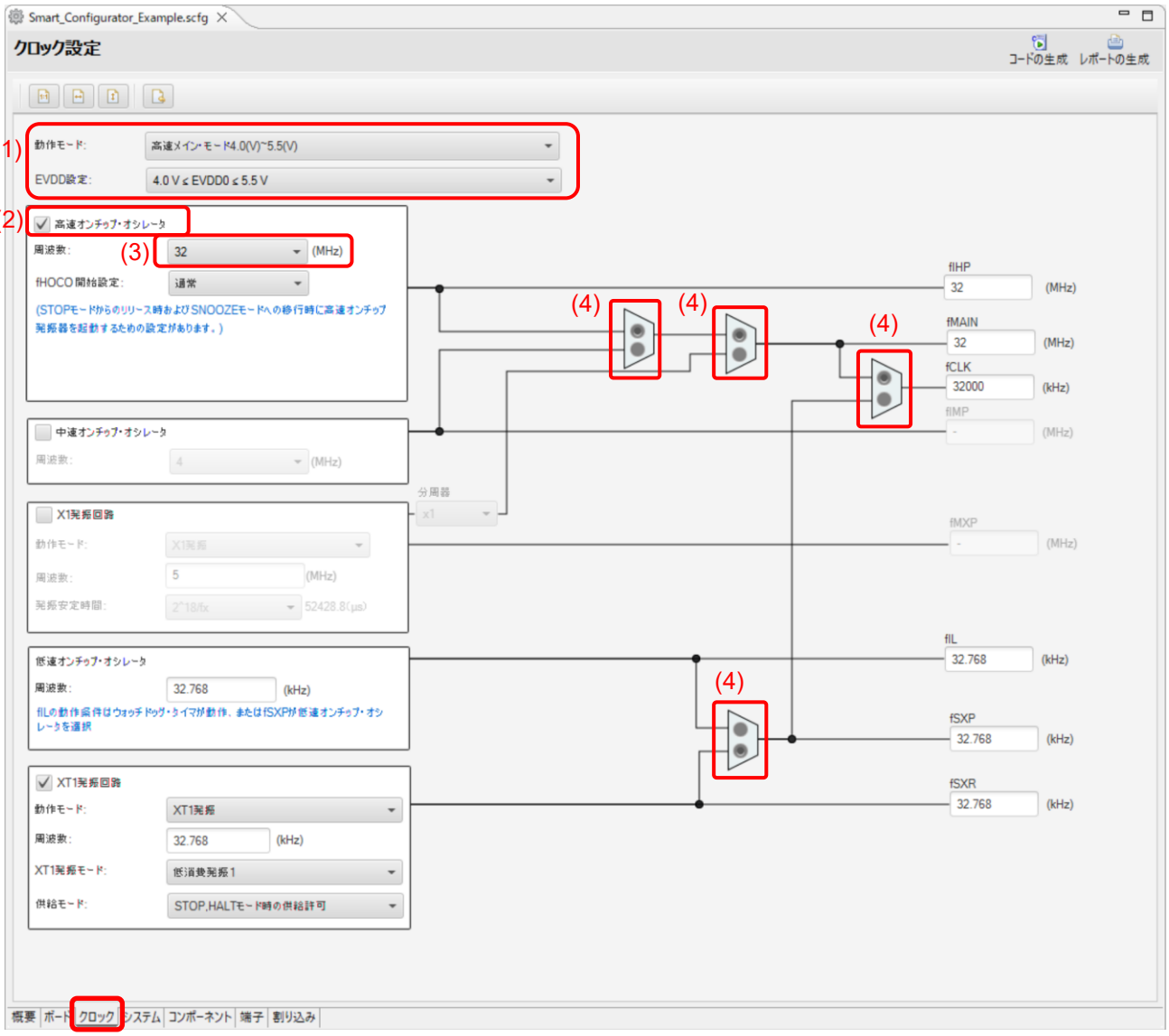

図 4-7［クロック］ページ

4.3　システム設定

[システム] ページでは、オンチップ・デバッグを設定できます。スマート・コンフィグレータは、コード生成時に [システム] ページの設定に応じて、リンカ・オプションを設定します。リンカ・オプションの設定は、プロジェクト・プロパティの [C/C++ ビルド] - [設定] - [Linker] - [デバイス] から確認できます。

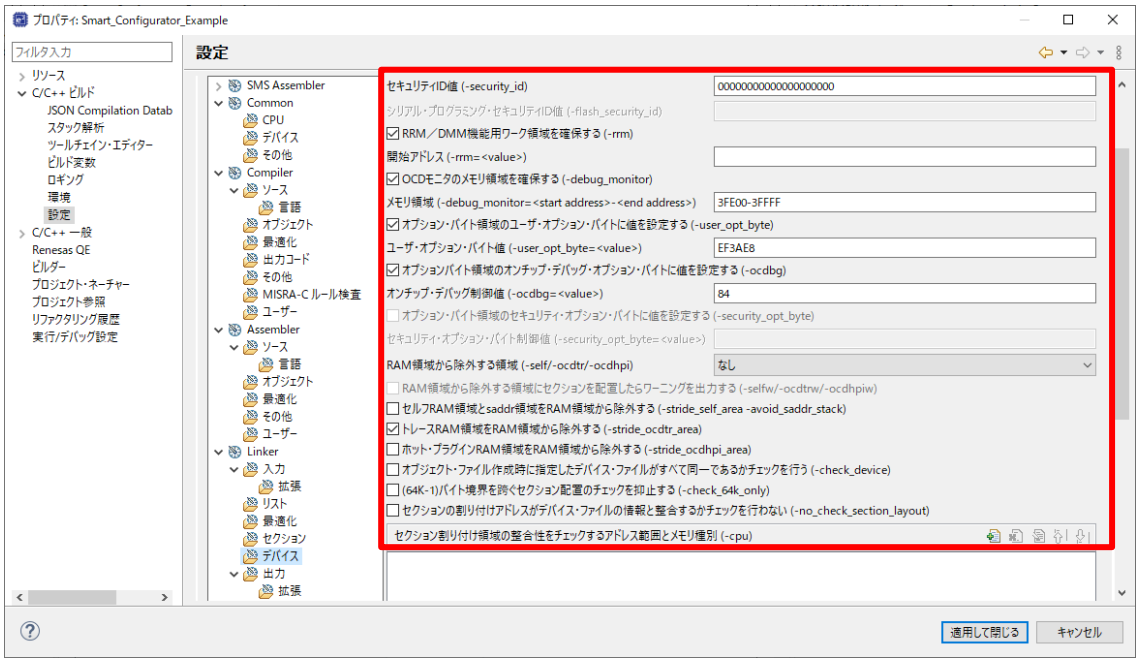

図 4-8 e² studio リンカ・オプション画面（デフォルト）

スマート・コンフィグレータの [システム] ページで、以下のように設定します。

図 4-9 スマート・コンフィグレータ [システム] ページ設定

[システム] タブで、上図の (1) から (3) の設定を行なってから (4) の [コード生成] ボタンをクリックすると、e²studio のリンカ・オプション更新を確認する下図のダイアログが表示されます。

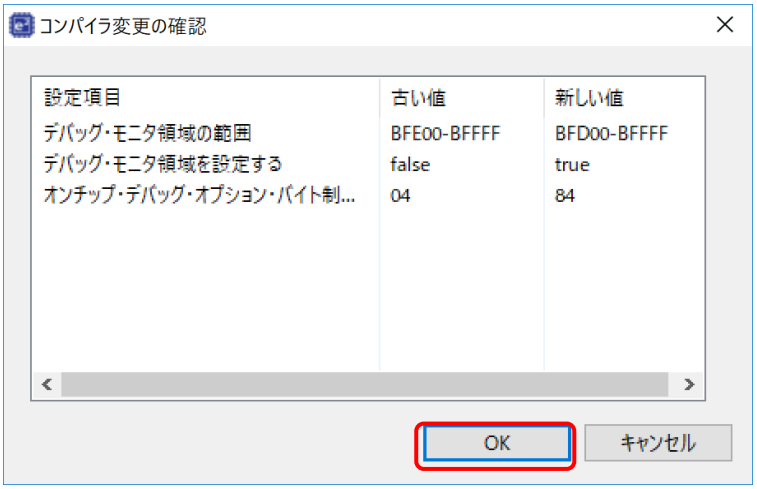

図 4-10 リンカ・オプション確認ダイアログ

[OK] ボタンをクリックし、リンカ・オプション設定を確認すると、以下のように更新されます。

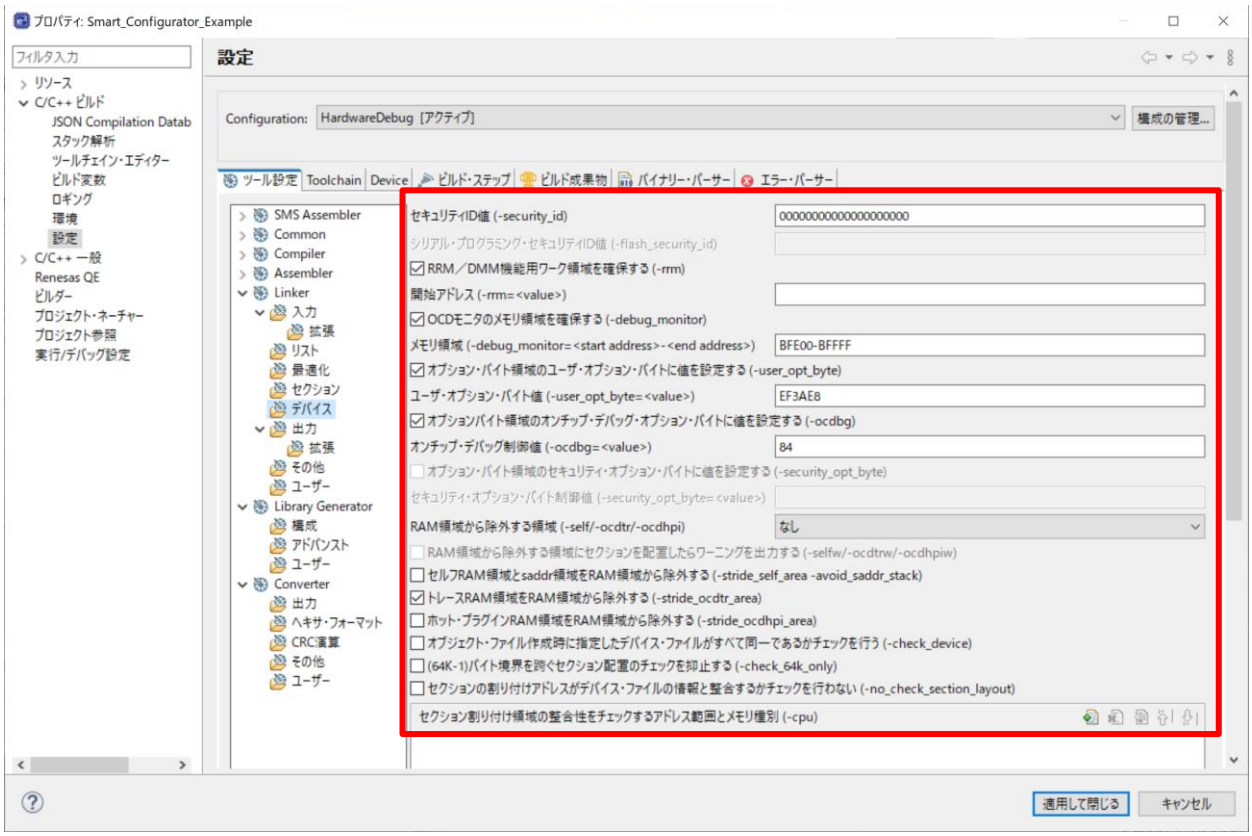

図 4-11 e² studio リンカ・オプション画面（更新後）

【注】 MCU タイプの選択またはチップ部品番号に応じて、これらの設定値は異なります。
詳細設定の構成については、デバイスユーザーズマニュアル：ハードウェア編を参照してください。

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。

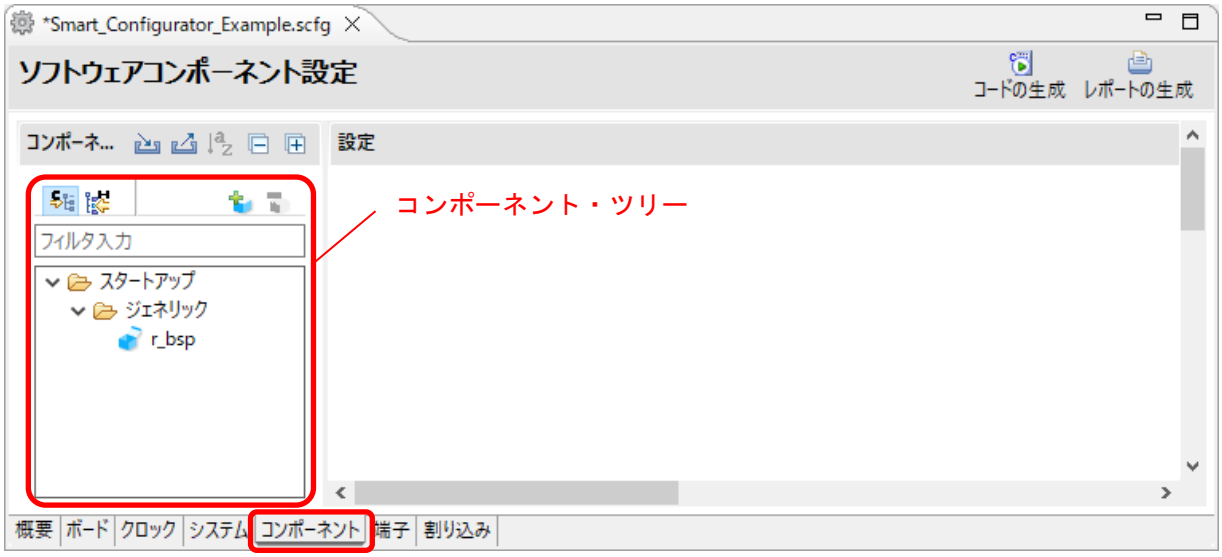

図 4-12 コンポーネント・ページ

4.4.1　コンポーネント・ビューとハードウェア・ビューの切り替え

コンポーネント・ツリーでは、コンポーネント・ビューとハードウェア・ビューの 2 つのツリー表示を提供しています。 以下のアイコンをクリックすることで、表示を切り替えることができます。

(1) [コンポーネント・ビュー] アイコン :

コンポーネント・ツリーに、コンポーネントのカテゴリごとにコンポーネントを表示します。

(2) [ハードウェア・ビュー] アイコン :

コンポーネント・ツリーに、ハードウェア・リソース階層でコンポーネントを表示します。

図 4-13 コンポーネント・ビューとハードウェア・ビューの切り替え

4.4.2　コード生成コンポーネントの追加方法

コンポーネントを追加するには、以下の 2 つの方法があります。

(a)　[コンポーネントの追加] アイコンからのコンポーネント追加

(b)　ハードウェア・リソース・ノードからのコンポーネント追加

[コンポーネントの追加] アイコンからのコンポーネント追加 (a) について説明します。

a-1. [コンポーネントの追加] アイコンをクリック、またはハードウェア・ビューに切り替えて、ハードウェア・リソース・ノードをダブルクリックします。

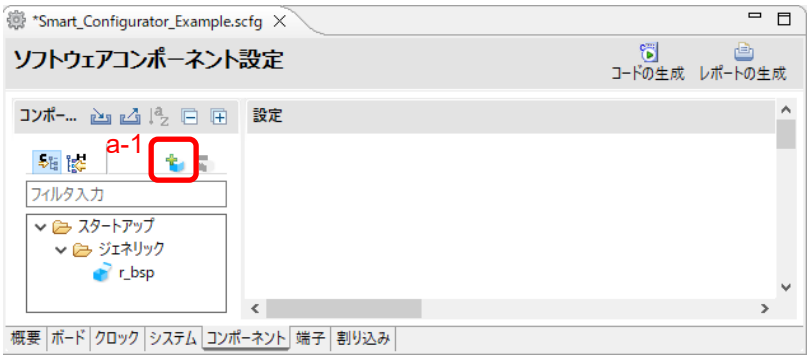

図 4-14　コンポーネントの追加

a-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] ページのリストからコンポーネントを選択します（例：A/D コンバータ）。

a-3. [タイプ] は [コード生成] であることを確認してください。

a-4. [次へ] をクリックします。

図 4-15　コード生成のコンポーネントの追加

a-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_ADC）

a-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：ADC）

a-7. [終了] をクリックします。

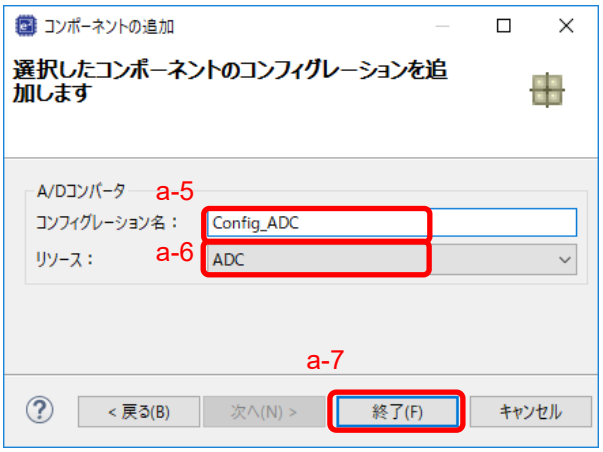

図 4-16 [コンポーネントの追加] ダイアログ

ハードウェア・リソース・ノードからのコンポーネント追加（b）について説明します。

b-1. [(ハードウェア・ビュー)] アイコンをクリックし、ツリー・ビューをハードウェア・リソース階層表示にします。

b-2. ハードウェア・リソース・ノードをダブルクリックします（例：A/D コンバータ）。

b-3. [コンポーネントの追加] ダイアログのリストに選択したハードウェア・リソース・ノードのコンポーネントが表示されます。

b-4. これ以降は、[コンポーネントの追加] アイコンからのコンポーネント追加(a-3)と同じ手順です。

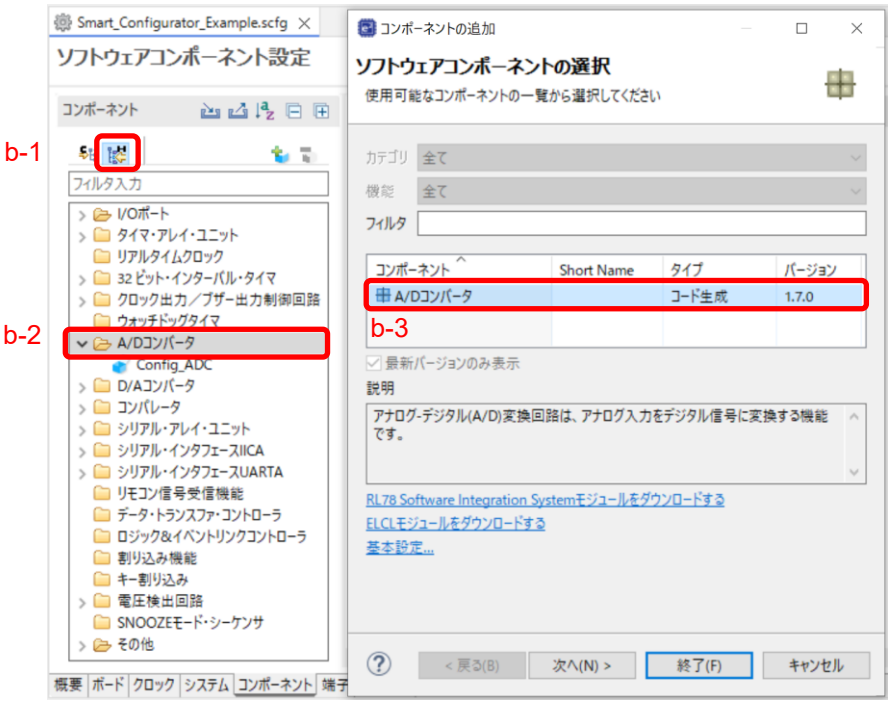

図 4-17 ハードウェア・リソース・ノードからの追加

4.4.3　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 1 項または複数項のソフトウェア・コンポーネントを選択します。
(Shift または Ctrl キーを押下しながらクリックすると複数のソフトウェア・コンポーネントを選択できます。)

(2)　[コンポーネントの削除] アイコンをクリックします。

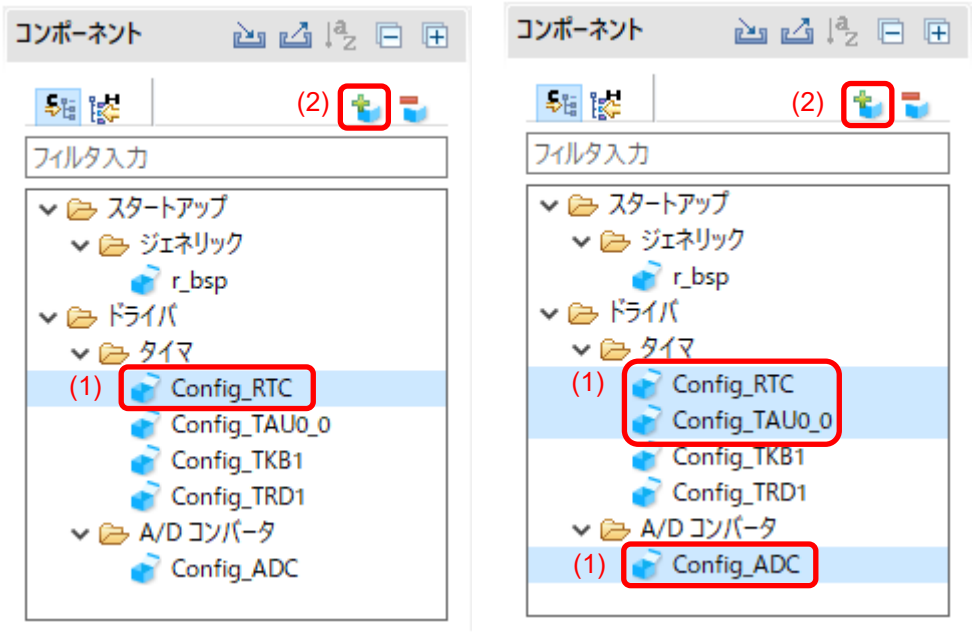

図 4-18　ソフトウェア・コンポーネントの削除

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。

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をコード生成するため、[コード生成] コードの生成 ボタンを押してください。

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_ADC）。

(2)　右側の設定パネルでドライバを設定します。以下に手順と画面の例を示します

a.　［分解能設定］で［10 ビット］を選択します。

b.　［トリガ・モード設定］で［ソフトウエア・トリガ・ノーウエイト・モード］を選択します。

c.　［A/D チャネルの選択］で［ANI0］を選択します。

d.　［変換時間］で［2112/fCLK］を選択します。

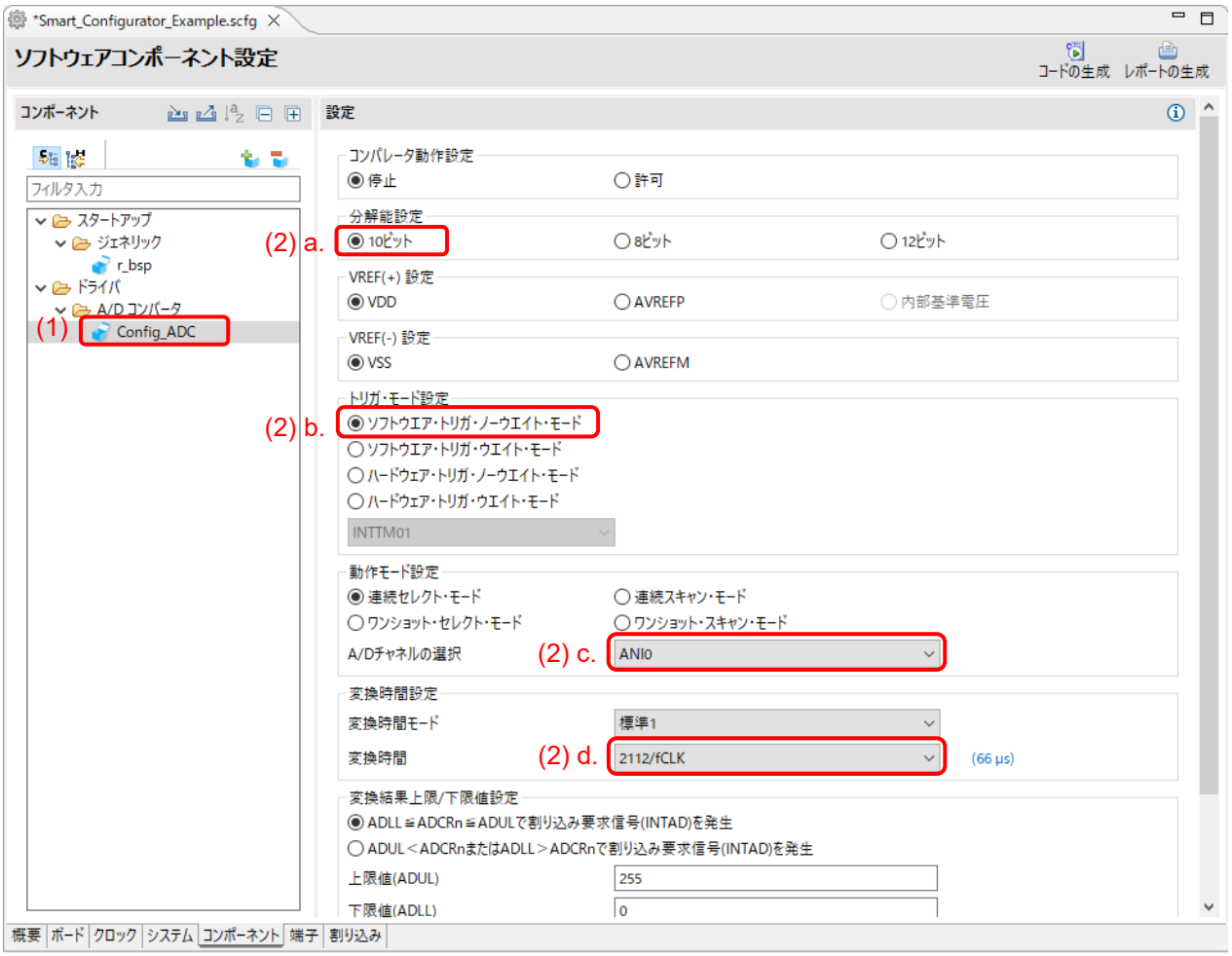

図 4-19 コンポーネントの追加

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 ］クリックすると、［ コード生成 ］コードを生成しません。

［ コード生成 ］クリックすると、［ ✓ コード生成 ］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：TAU0_1 から TAU0_3 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_TAU0_1）。

(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。

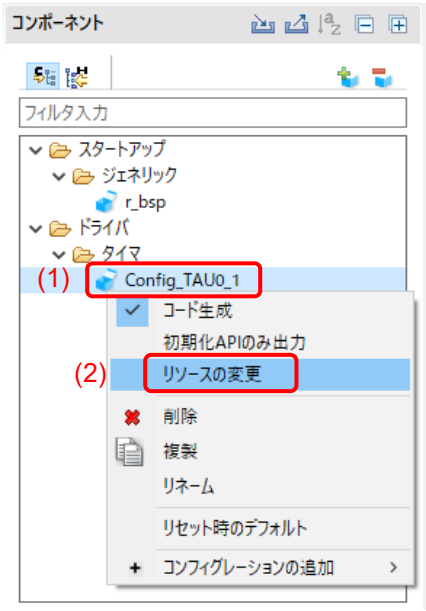

図 4-20　リソースの追加

(3)　[リソースの選択] ダイアログにある新しいリソースを選択します（例：TAU0_3）。

(4)　[次へ] ボタンが有効になるので、クリックします。

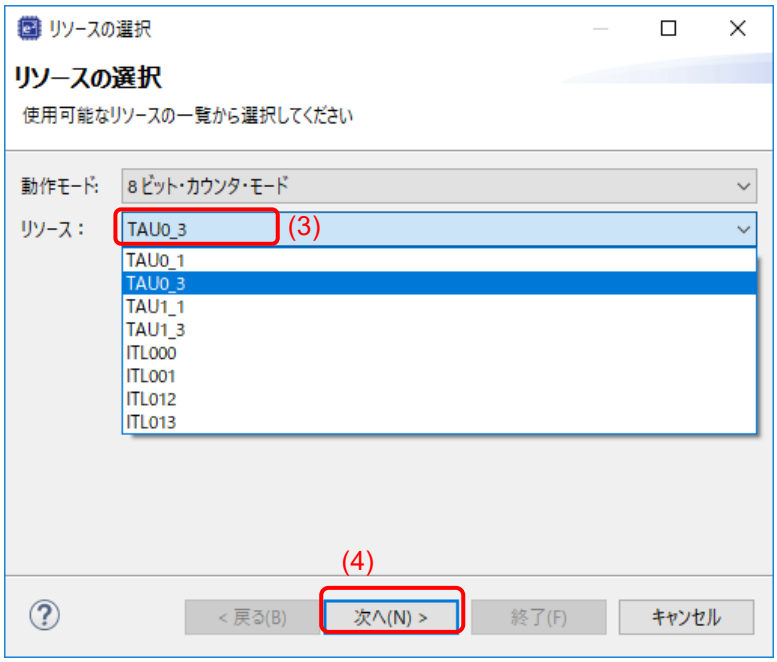

図 4-21　[コンポーネント] ページー新しいリソースの選択

(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。

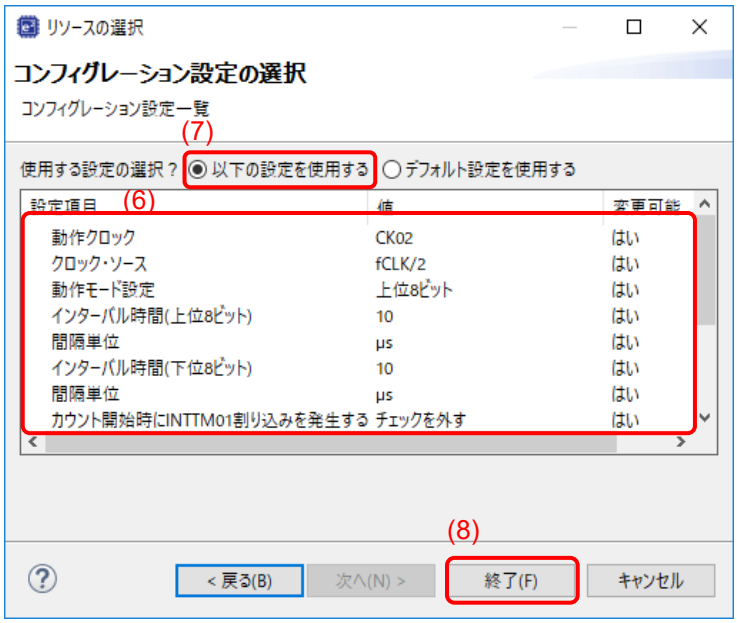

図 4-22 新しいリソース設定の確認

リソースは、自動的に更新されます（例：INTTM01 から INTTM03 へ）。

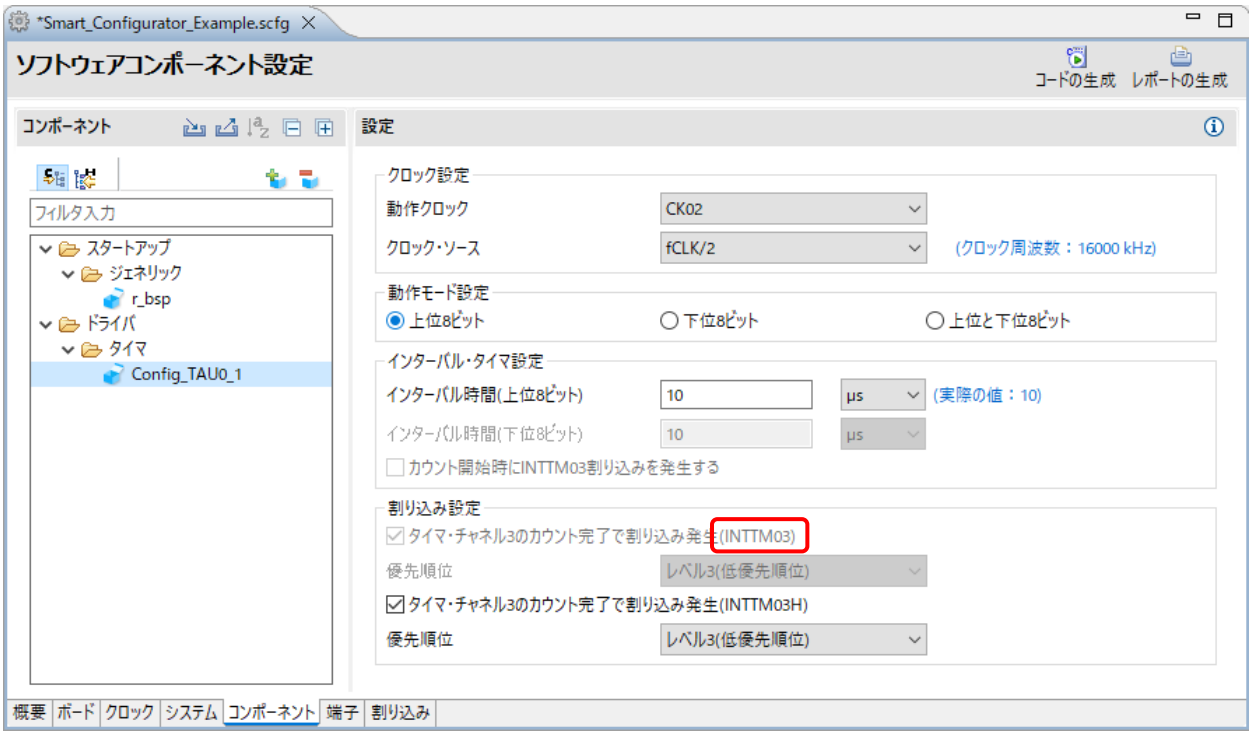

図 4-23 自動的に更新されるリソース

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_TAU0_1 から Config_TAU0_3 へ）。

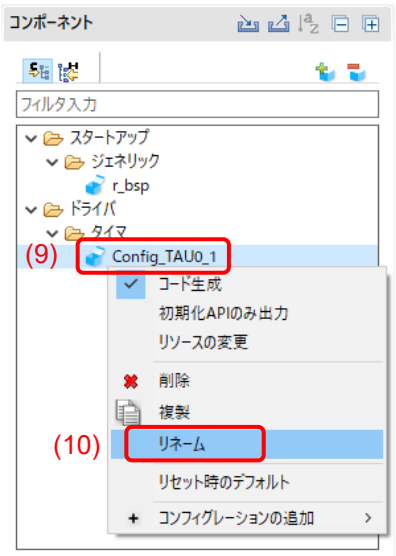

図 4-24 コンフィグレーション・リネーム

4.4.6 SNOOZE モード・シーケンサの設定

SNOOZE モード・シーケンサ（SMS）は、グラフィカル・コンフィグレータ・タイプのコンポーネントで、[ソフトウェア・コンポーネントの選択] から追加できます。

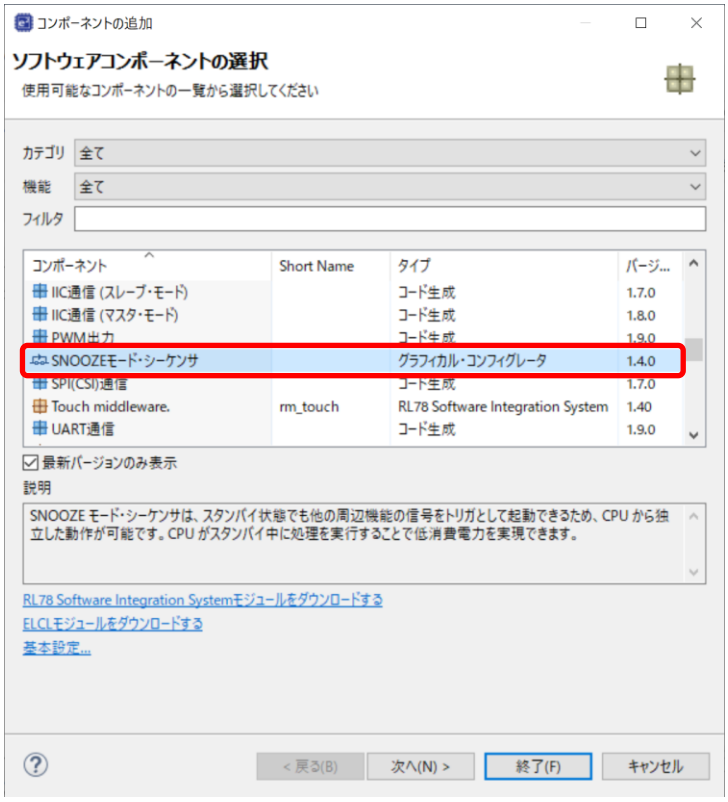

図 4-25 SNOOZE モード・シーケンサ（SMS）の追加

SNOOZE モード・シーケンサ（SMS）の GUI ルック・アンド・フィールは、以下の「図 4-26 SNOOZE モード・シーケンサ（SMS）GUI」表示となり、コード生成と比較してよりグラフィカルに表示します。ブロックをドラッグ・アンド・ドロップすることで構成できます。

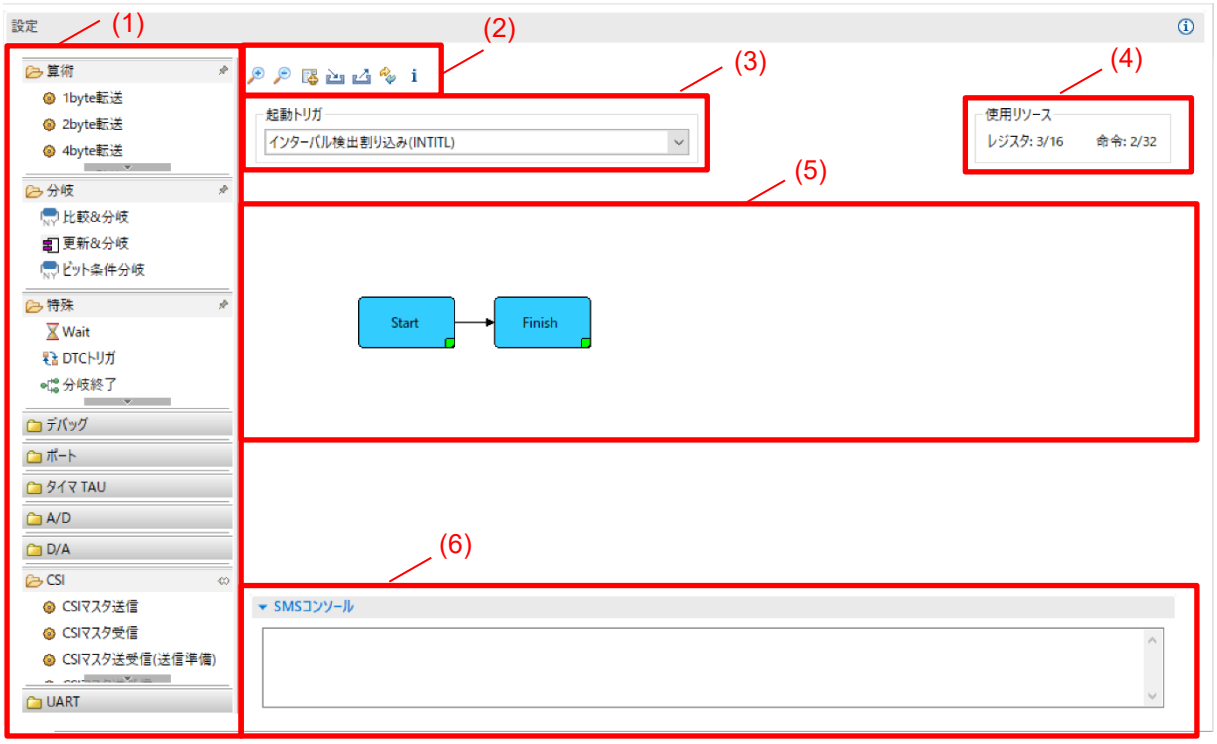

図 4-26 SNOOZE モード・シーケンサ（SMS）GUI

表 4-2　SMS GUI エリアの説明

エリア		説明
(1) SMS ブロック		SMS で使用できるブロックを表示します。 ブロックはシーケンス（機能）を形成するためのパーツで、A/D 電圧取得、比較&分岐、1byte 転送などがあります。
(2) ツールバー		キャンパスを拡大します。
		キャンパスを縮小します。
		SMS データ管理ダイアログを表示し、使用する変数などを管理します。
		SMS シーケンスをインポートします。このアイコンをクリックすると幾つかのサンプルシーケンスをご使用いただけます。
		SMS シーケンスをエクスポートします。
		SMS データファイルを更新します。
		SMS データファイルの情報を表示します。
(3) 起動トリガ選択		起動トリガを選択します。
(4) 使用リソース		キャンパスで使用しているレジスタ、命令数を表示します。
(5) キャンパス		SMS ブロックを配置して、シーケンスを作成します。
(6) コンソール		SMS で使用できないコンポーネント設定時にメッセージを表示します。

以下のように SMS ブロックを設定します。

(1) ブロック・リストからブロックを選択します（例：CSI マスタ受信）

(2) [CSI マスタ受信] ブロックを、キャンバスの Start ブロックと Finish ブロック間 のインジケーターが表示されない位置）にドラッグ＆ドロップします。

(3) [CSI マスタ受信] ブロックをダブルクリックして、CSI マスタ受信設定ダイアログを開きます。

(4) CSI マスタ受信設定ダイアログのプロパティを設定します。

(5) [データ管理] を開くと、受信データを編集できます。

(6) 設定が必要なブロックは、右下が赤く表示されます。正しく設定されると緑に変わります。

(7) 同じようにいくつかのブロックを追加して、シーケンスを作成します。

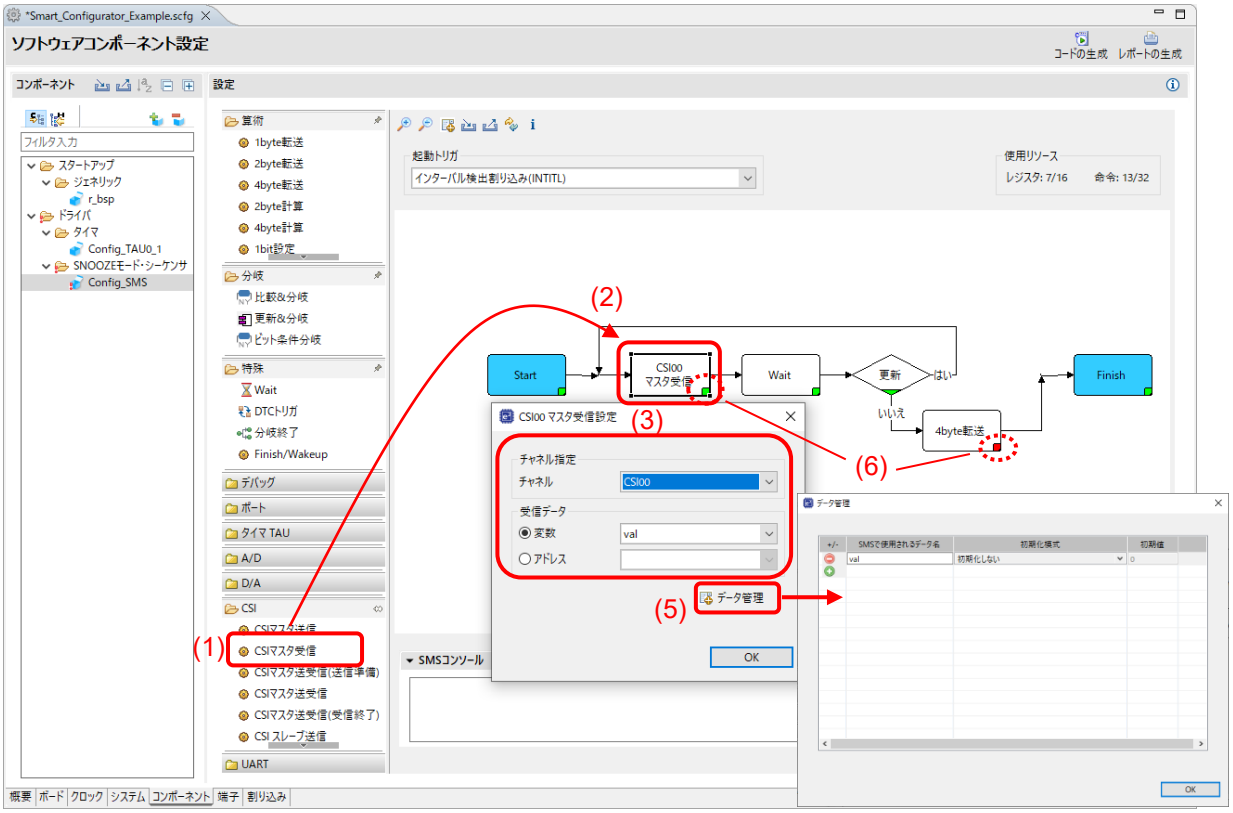

図 4-27 SMS ブロック設定

[注] SMS (Snooze Mode Sequencer) モジュールを正常にビルドするには、ルネサスエレクトロニクスホームページからから最新の SMSASM_Vxxx_setup.exe ツールをダウンロードしてインストールする必要があります。 ツールバーの ボタンをクリックして、[Renesas ツールチェーン管理] に確認します。

図 4-28 SMS アセンブラ

4.4.7　SMS データファイルの更新

以下の手順で、SMS データファイル (ブロック、シーケンス) の更新が行えます。更新することで新しいブロック、シーケンスをご使用いただけます。

(1)　[SMS データファイルを更新] ボタンをクリックして、SMS データファイルの更新を行います。

(2)　SMS データファイルの更新を確認します。

(3)　新しいバージョンが存在した場合、自動的にダウンロードして更新します。

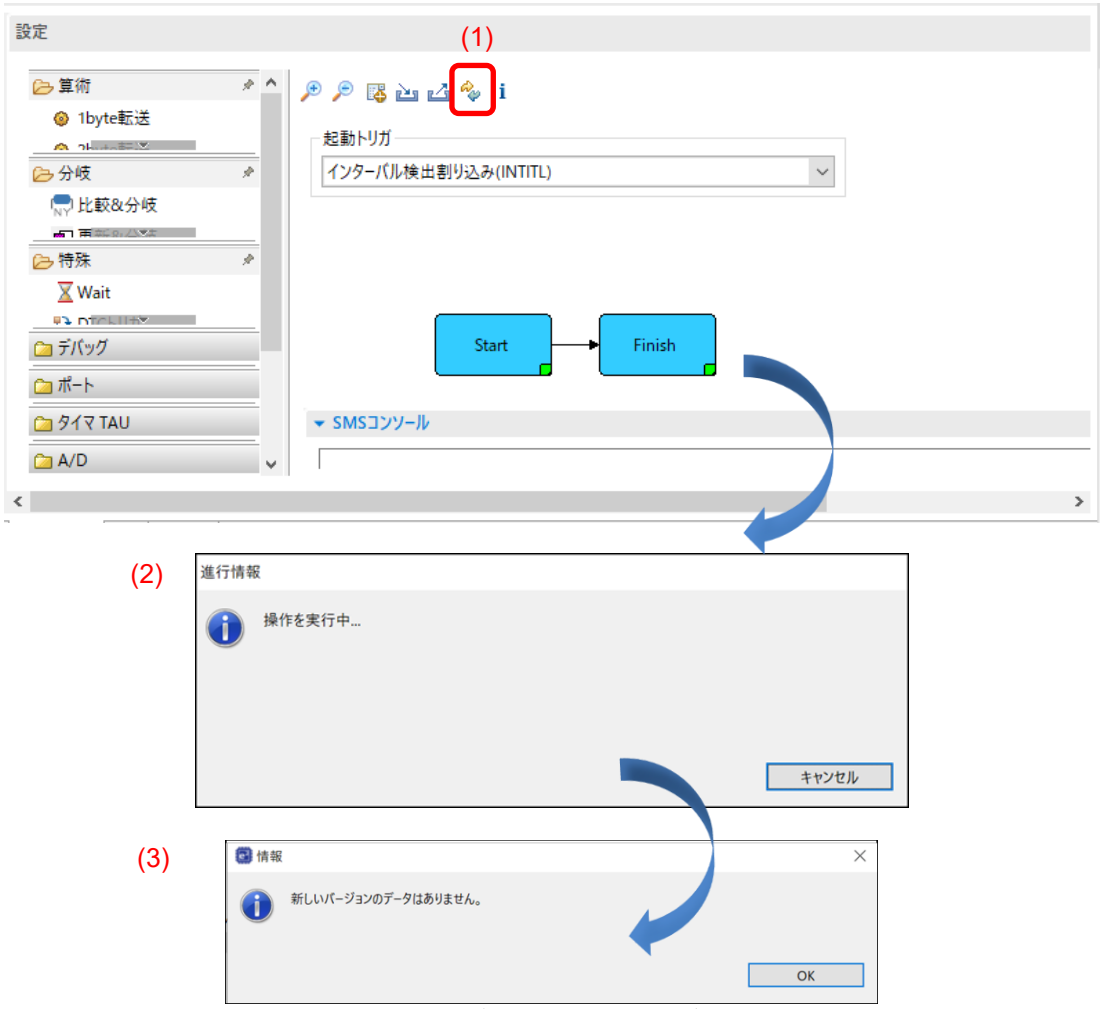

図 4-29 SMS データファイルのダウンロード

4.4.8　ELCL 固定機能モジュールのダウンロード

ELCL（ロジック＆イベント・リンク・コントローラ）のソフトウェアコンポーネントタイプは、グラフィカル・コンフィグレータです。ELCL コンポーネントには 2 種類あり、1 種類は「スレーブセレクトピン機能」、「チャタリング防止機能」などの固定機能 ELCL コンポーネントで、もう 1 種類は ELCL Flexible Circuit で、フレキシブルに ELCL 回路を作成できます。ELCL 固定機能モジュールは、[コンポーネントの追加] ダイアログから追加できます。コンポーネントリストに含まれていない ELCL 固定機能モジュールを使用したい場合は、[ELCL モジュールをダウンロードする] のリンクより、ダウンロードできます。

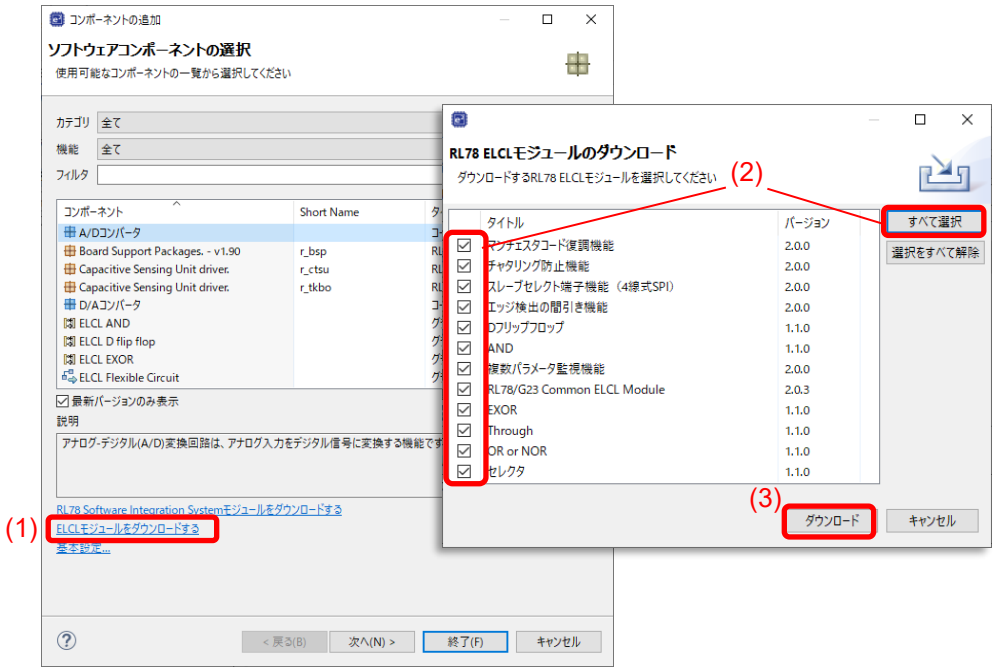

図 4-30 ELCL 固定機能モジュールのダウンロード

ダウンロードした ELCL 固定機能モジュールは、コンポーネン選択トリストに自動的に追加させます。

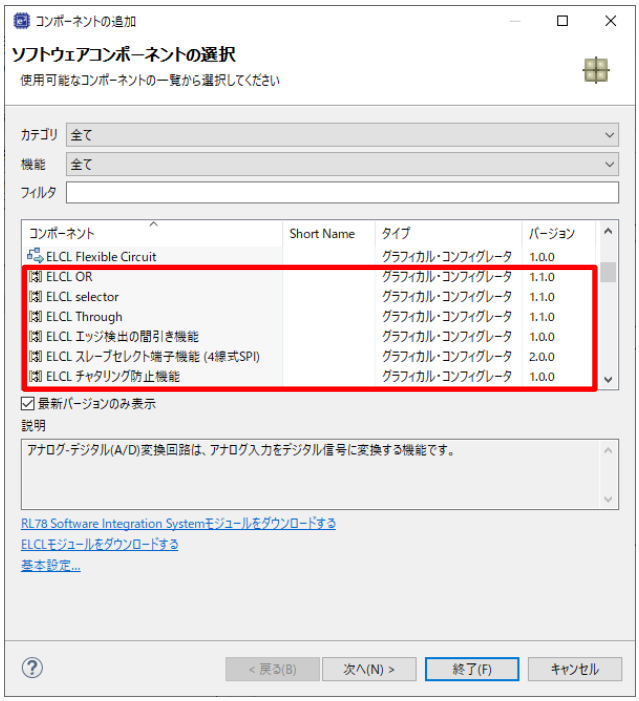

図 4-31 ELCL 固定機能モジュールの追加

4.4.9　固定機能 ELCL コンポーネントの設定

以下の手順で、固定機能 ELCL コンポーネントを設定します。

(1)　[ソフトウェア・コンポーネントの選択] から固定機能 ELCL コンポーネントを選択します。
（例：ELCL スレーブセレクト端子機能（4 線式 SPI））

(2)　[構成] パネルでドライバを構成します。

a.　[Input signal selector] ：入力信号を選択します。

b.　[Event control (link processor)] ：論理セルブロックを選択します。

c.　[Output signal selector] ：出力信号を選択します。

(3)　リンクをクリックすることで、アプリケーションノートを開くことができます。

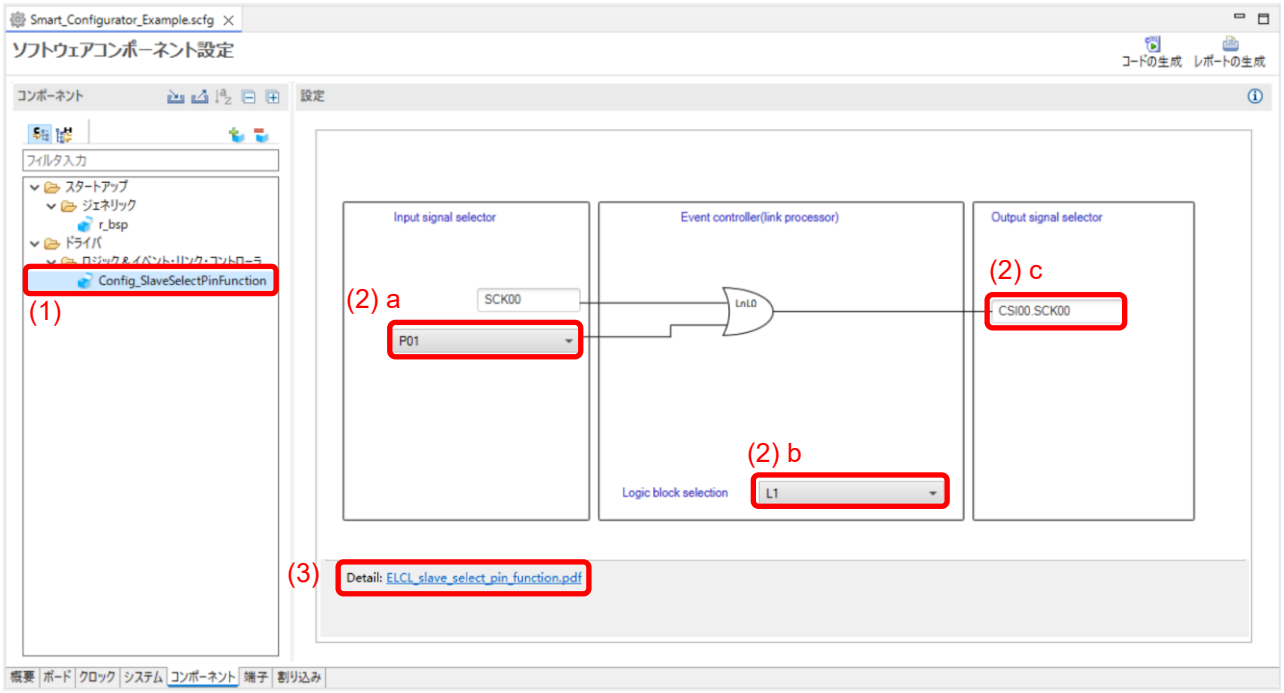

図 4-32　固定機能 ELCL コンポーネントの設定

4.4.10 ELCL Flexible Circuit の作成と編集

ELCL (ロジック&イベント・リンク・コントローラ) Flexible Circuit コンポーネントは、グラフィカル・コンフィギュレータの新しいコンポーネント・タイプで、コンポーネントリストから選択して使用することができます。

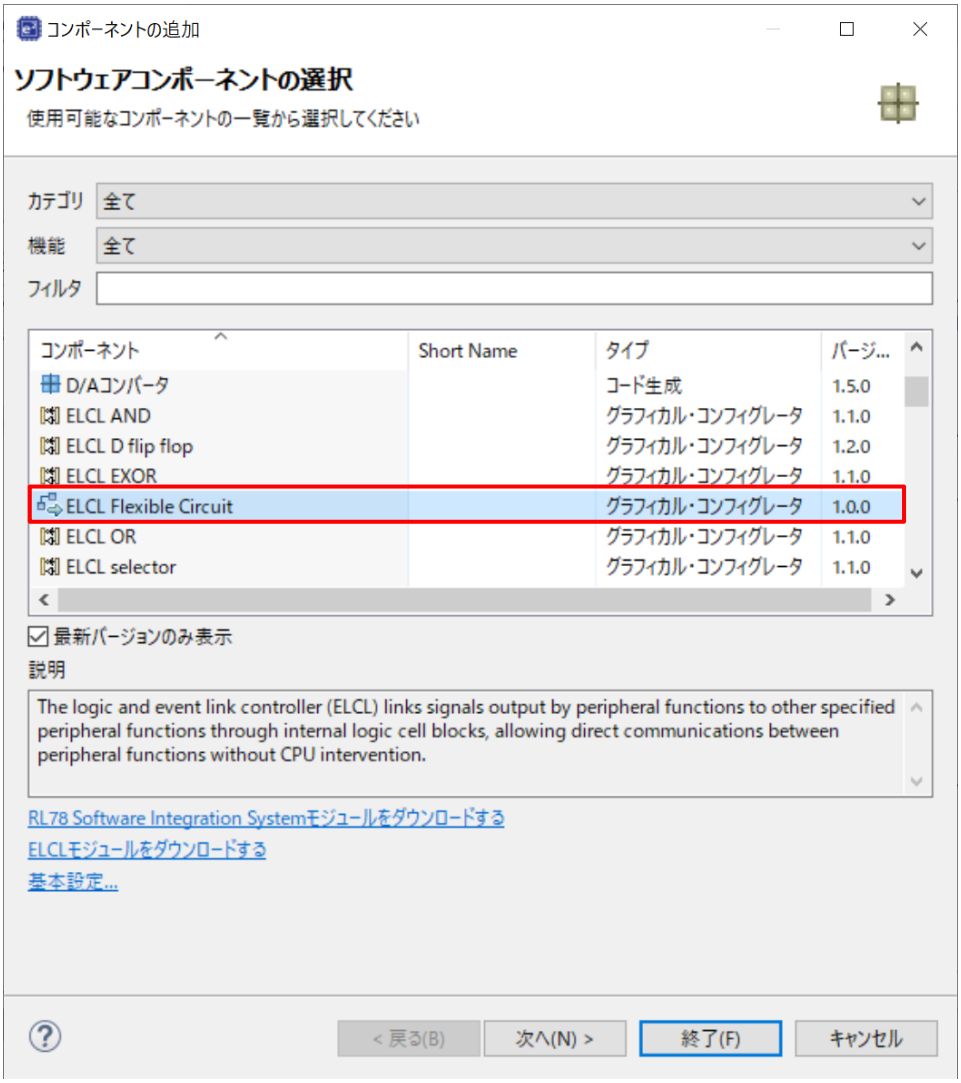

図 4-33 ELCL Flexible Circuit コンポーネントの追加

ELCL フレキシブル回路コンポーネントは、ELCL 回路の作成と編集のためのドラッグアンドドロップ操作をサポートする直感的な GUI を提供し、回路設計後に ELCL レジスタ設定を自動的に生成できます。

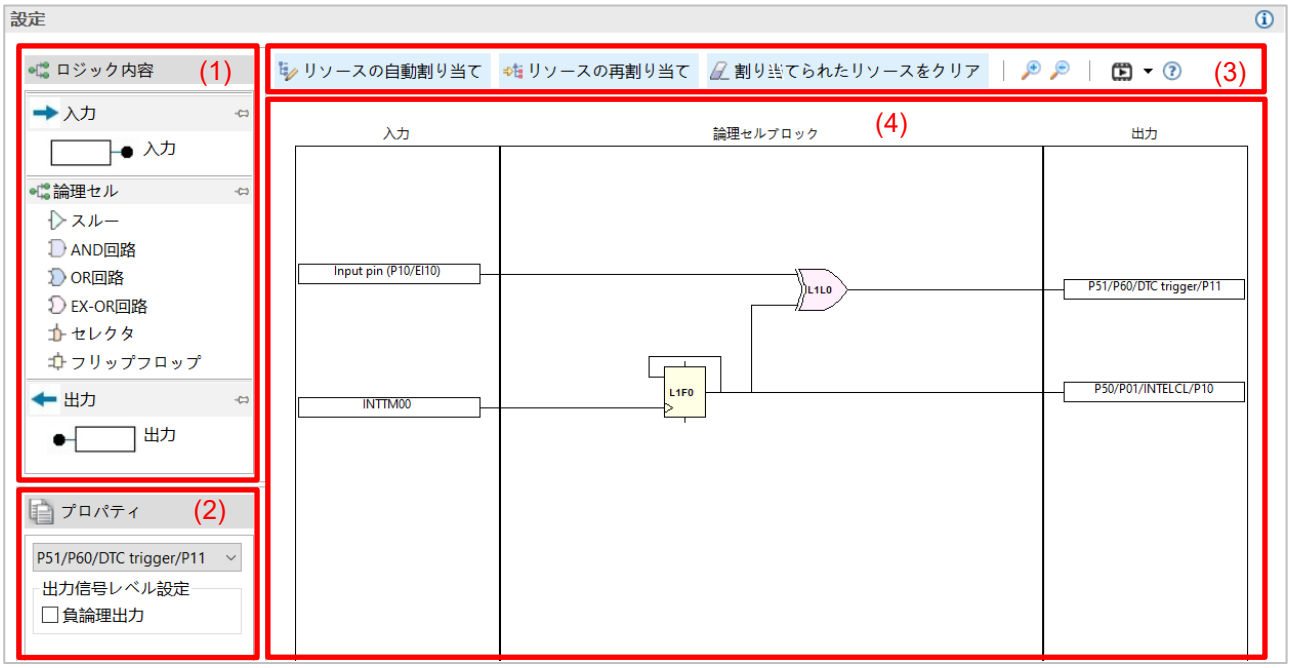

図 4-34　ELCL Flexible Circuit コンポーネント

表 4-3　ELCL Flexible Circuit GUI エリア説明

エリア		説明
(1) ELCL 要素		ELCL で使用できる要素を表示します。
(2) プロパティ		選択した ELCL 要素の設定を表示します。
(3) ツールバー	リソースの自動割り当て	論理セルブロックにリソースを自動的に割り当てます。
	リソースの再割り当て	リソースの問題を解決するために、リソースを再割り当てします。
	割り当てられたリソースをクリア	論理セルブロックに割り当てられたすべてのリソースをクリアします。
		キャンバスを拡大します。
		キャンバスを縮小します。
		Web ページの ELCL 機能の紹介に関するビデオを視聴できます。
		ヘルプを表示します。
(4) キャンバス	入力	ELCL 入力要素を配置する領域です。
	論理セルブロック	ELCL 論理セルブロック要素を配置する領域です。
	出力	ELCL 出力要素を配置する領域です。

以下の手順に従って、ELCL Flexible Circuit を作成します。

(1)　パネルから入力／論理セル／出力の ELCL 要素をキャンバスにドラッグ&ドロップします。

(2)　キャンバスで入力／論理セル／出力の ELCL 要素を選択し、プロパティを設定します。

(3)　始点を終点にドラッグ&ドロップして接続します。

(4)　[リソースの自動割り当て] ボタンをクリックすると、リソースが割り当てられていない論理セルにリソースを自動的に割り当てます。

(5)　ELCL 回路にエラーが表示される場合、[リソースを再割り当て] ボタンをクリックしてリソースエラーを解消します。

(6)　ELCL 回路作成後、[コードの生成] ボタンをクリックすると、ELCL レジスタ設定が生成されます。

【注】 手順 (1)、(2)、(3) は決まった操作ではなく、各ステップを自由に操作し ELCL 回路を作成または編集できます。

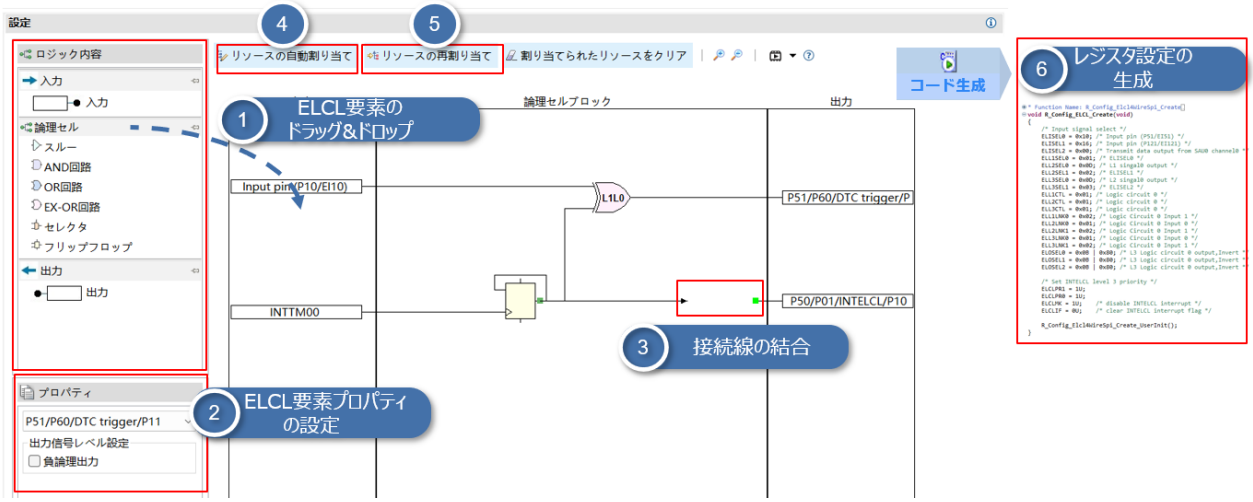

図 4-35　ELCL Flexible Circuit 作成手順

以下に、ELCL 回路を簡単に作成し、正しい設計に導くのに役立つ GUI 操作の詳細を示します。

(1)　始点を終点にドラッグ&ドロップして接続します。

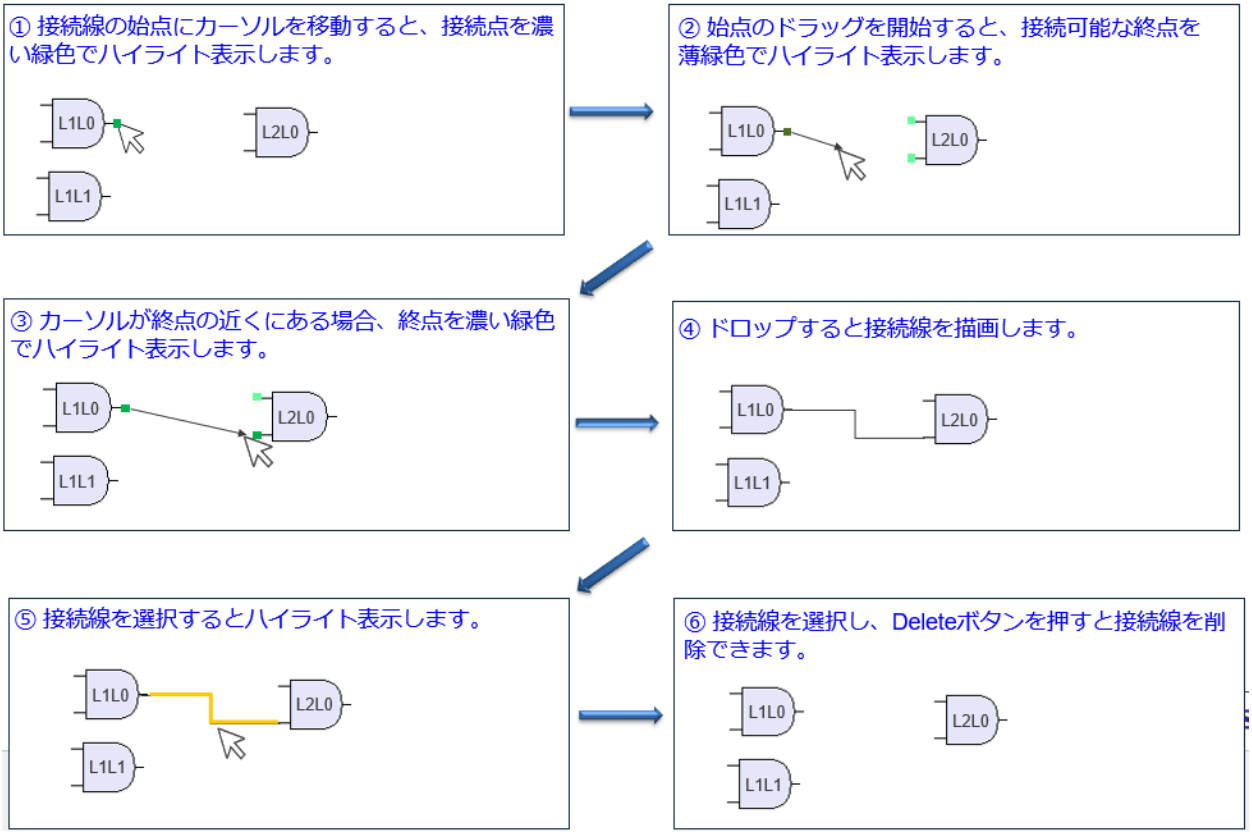

図 4-36 回路線接続操作

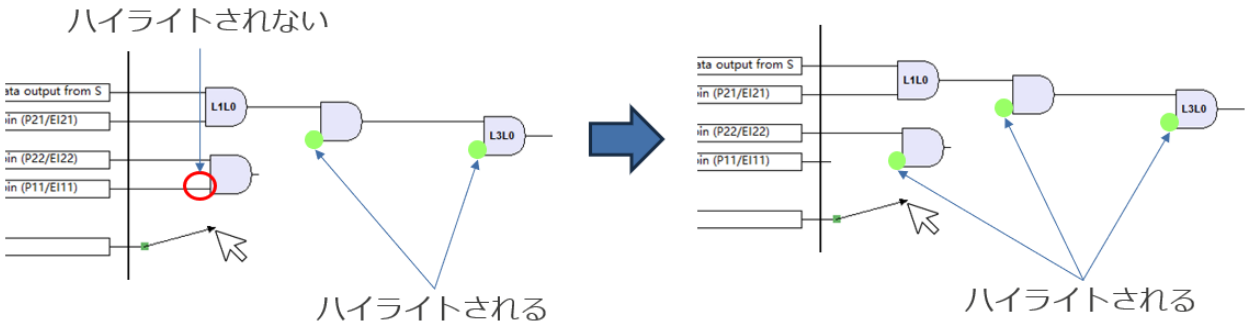

図 4-37 回路線接続時のハイライト／ノンハイライト

【注】 ポイントをドラッグした時、接続可能な終点だけを薄緑色でハイライト表示します。

(2)　ハードウェアの制限を超えた接続や設定を行った場合、各部にエラーが表示されます。
ユーザーは、示されたメッセージに基づいてエラーを解決できます。

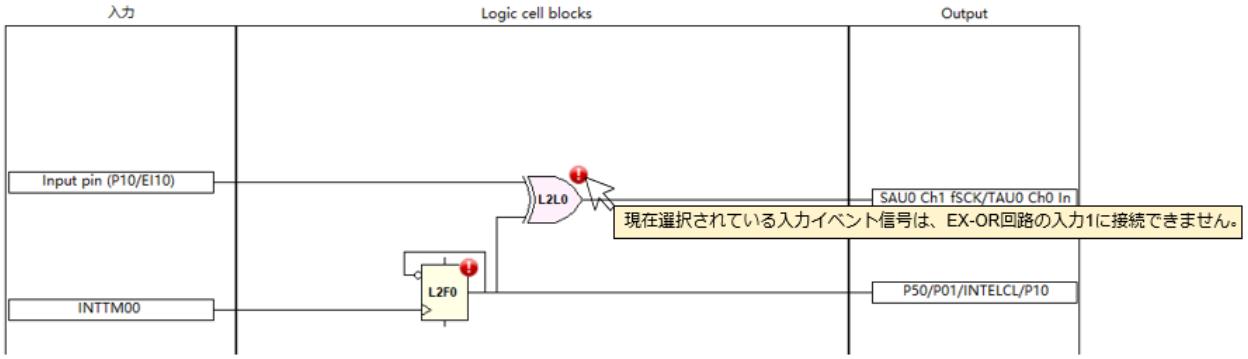

図 4-38 ELCL Flexible Circuit **エラー表示**

表 4-4　ELCL Flexible Circuit エラーメッセージ一覧

No.	エラーメッセージ
1	現在選択されている入力イベント信号は、pass-through/AND/OR/EX-OR 回路の入力 x に接続できません。
2	信号選択レジスタ xxx はすべて使用されており、割り当てられません。
3	ELL1SEL4 と ELL1SEL5 は両方とも使用されています。同じ論理セルブロック内の別のフリップフロップで使用している信号と同じ信号を使用してください。
4	このクロック設定を使用する場合は、別のフリップフロップで使用している信号と同じ信号を設定してください。
5	フリップフロップのセットとリセットは、異なる信号を選択する必要があります。
6	ELCL 回路には割り当てられていないリソースが存在するか、ライン接続が完了していないため、一部のコードが生成されないか、正しくありません。
7	入力信号に割り込み要求信号を接続した場合、出力信号に周辺機能のハードウェアトリガ以外は接続できません。
8	Logic cell block Lx celly (flip-flop *n*) は、すでに使用されています。利用可能な他のリソースを選択してください。
9	リソースの自動割り当てに失敗しました。ロジックセルの割り当てに使用できるリソースがありません。回路を修正して再実行してください。
10	リソースの自動割り当てに失敗しました。入力信号は ELISEL0～5 にのみ指定できますが、6～11 のセル入力（セット/リセット/クロック）に接続する必要があります。
11	リソースの自動割り当てに失敗しました。入力信号は INTC ですが、出力信号がトリガーイベントとして設定されていません。
12	リソースの自動割り当てに失敗しました。フリップフロップのセット/リセットは同じ入力信号を使用しています。別の入力信号に変更してください。
13	リソースの再割り当てに失敗しました。ロジックセルの割り当てに使用できるリソースがありません。回路を修正して再実行してください。
14	リソースの再割り当てに失敗しました。入力信号は ELISEL0～5 にのみ指定できますが、6～11 のセル入力（セット/リセット/クロック）に接続する必要があります。
15	リソースの再割り当てに失敗しました。入力信号は INTC ですが、出力信号がトリガーイベントとして設定されていません。
16	リソースの再割り当てに失敗しました。フリップフロップのセット/リセットは同じ入力信号を使用しています。別の入力信号に変更してください。

4.4.11 RL78 Software Integration System モジュールのダウンロード

RL78 Software Integration System モジュールは、ドライバ、ミドルウェア、アプリケーション SW のソフトウェア・コンポーネントで、コードを生成するための簡単な GUI を提供し、[コンポーネントの追加] ダイアログからダウンロードできます。

(1) [図 **4-14** コンポーネントの追加] アイコンをクリックして [コンポーネントの追加] ダイアログを開きます。

(2) [RL78 Software Integration System モジュールをダウンロードする] リンクをクリックして、使用する RL78 Software Integration System モジュールをダウンロードします。

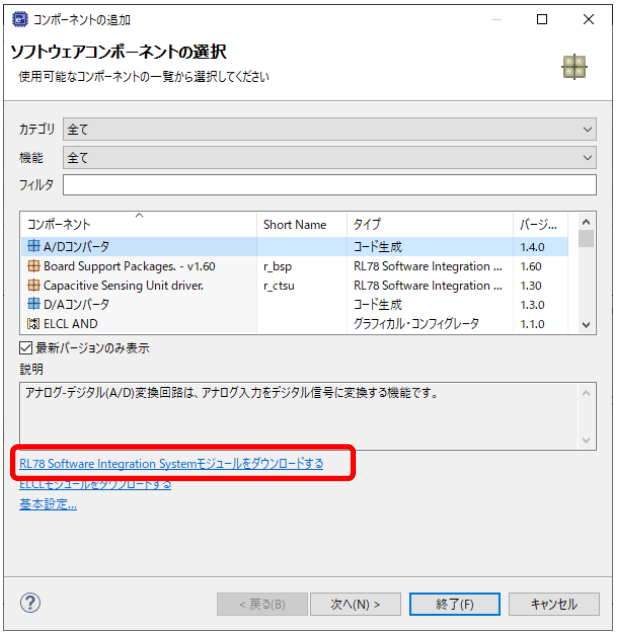

図 4-39 RL78 Software Integration System モジュールのダウンロード

【注】 ダウンロードには、My Renesas へのログインが必要です。ログインしていない場合は、次のダイアログボックスでログインを求められます。

図 4-40 My Renesas ログイン

(3) [RL78 Software Integration System モジュールのダウンロード] ダイアログで必要なモジュールを選択します。

(4) [参照] をクリックして、ダウンロードしたモジュールを保存する場所を選択します。

(5) [ダウンロード]をクリックすると、モジュールのダウンロードが開始されます。

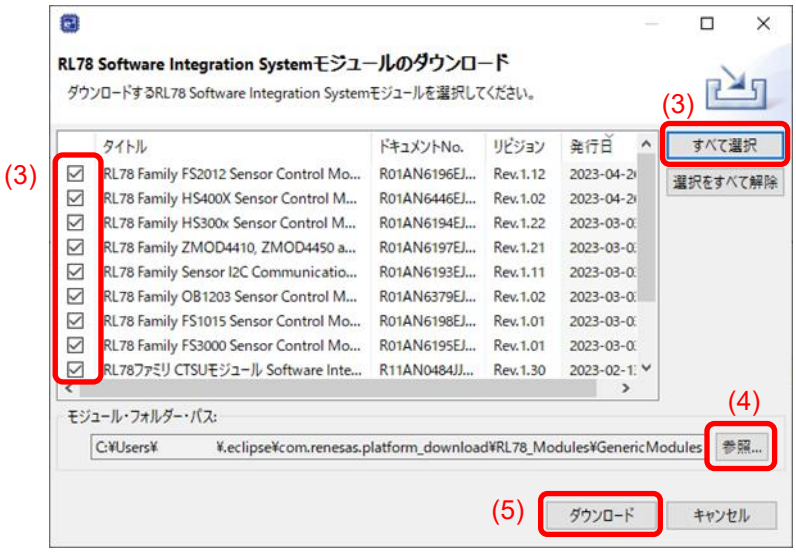

図 4-41 RL78 Software Integration System モジュールのダウンロード

4.4.12 RL78 Software Integration System モジュールの追加

RL78 Software Integration System モジュールを追加する手順は、以下の通りです。

(1) [図 **4-14** コンポーネントの追加] アイコンをクリックして [コンポーネントの追加] ダイアログを開きます。

(2) コンポーネントリストから RL78 Software Integration System タイプのコンポーネントを選択します。複数のモジュールを選択したい場合は、Ctrl キーを押下しながらクリックします。

(3) [終了] をクリックします。

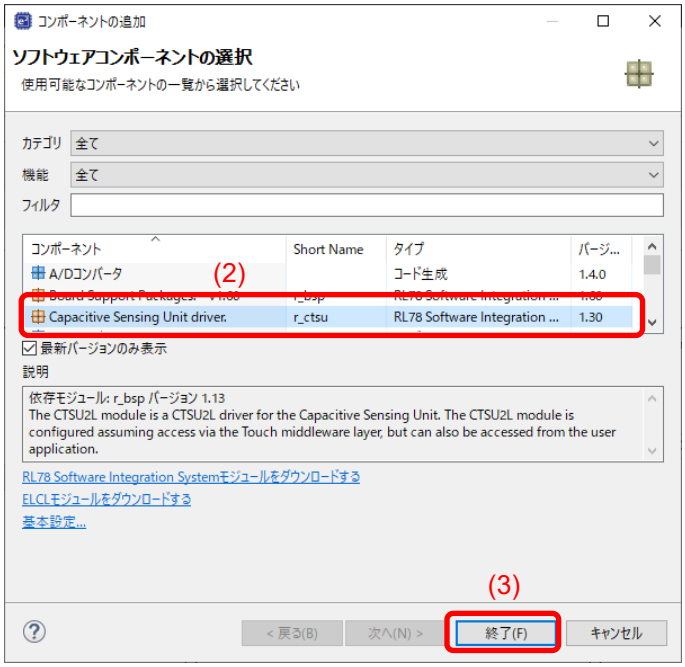

図 4-42 RL78 Software Integration System モジュールの追加

4.4.13 RL78 Software Integration System モジュールの設定

RL78 Software Integration System を使用するには、構成オプションを設定し、設定方法はコンポーネントによって異なります。

- 構成パネルで構成オプションを設定しコード生成を行うと、RL78 Software Integration System の構成ファイルに自動的に設定が反映されます。

【注】RL78 Software Integration System モジュールの構成ファイルは、r_config フォルダに生成されます。

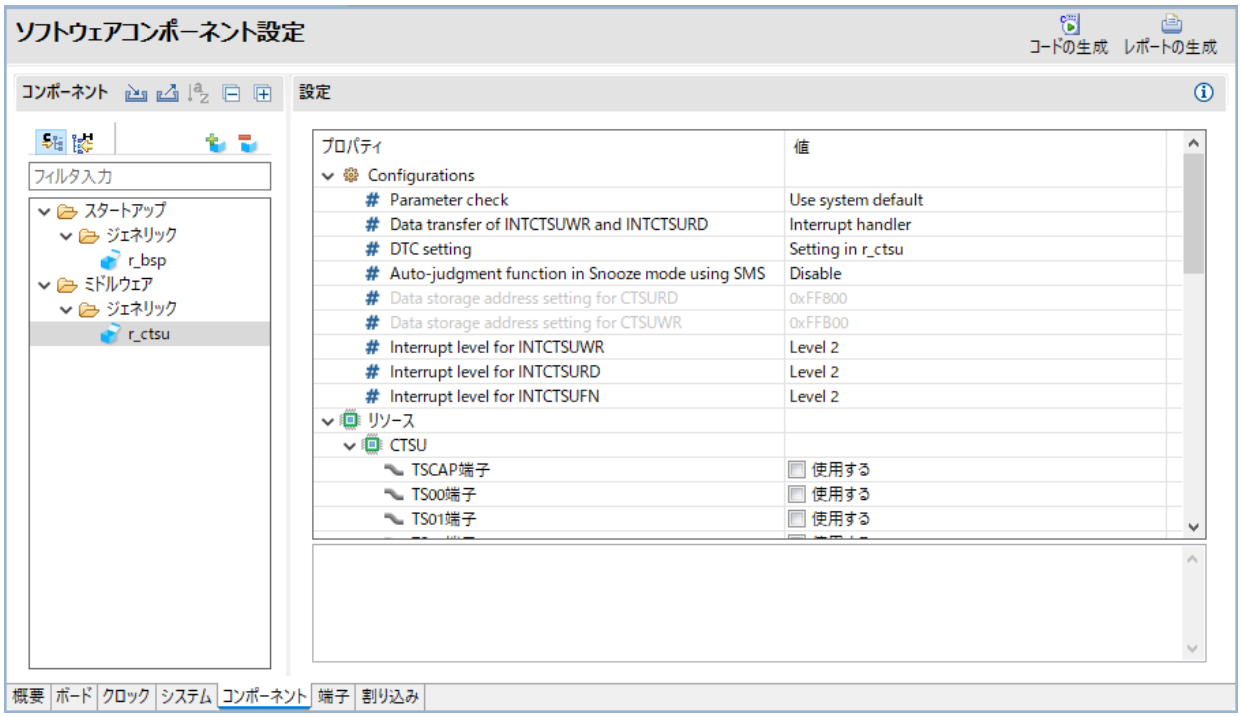

図 4-43 RL78 Software Integration System モジュールの設定

4.4.14 BSP コンフィグレーションのバージョン変更

BSP コンフィグレーションのバージョン変更は、以下の手順で行います。

(1) コンポーネント・ツリーから、バージョンを変更する r_bsp コンポーネントを右クリックします。

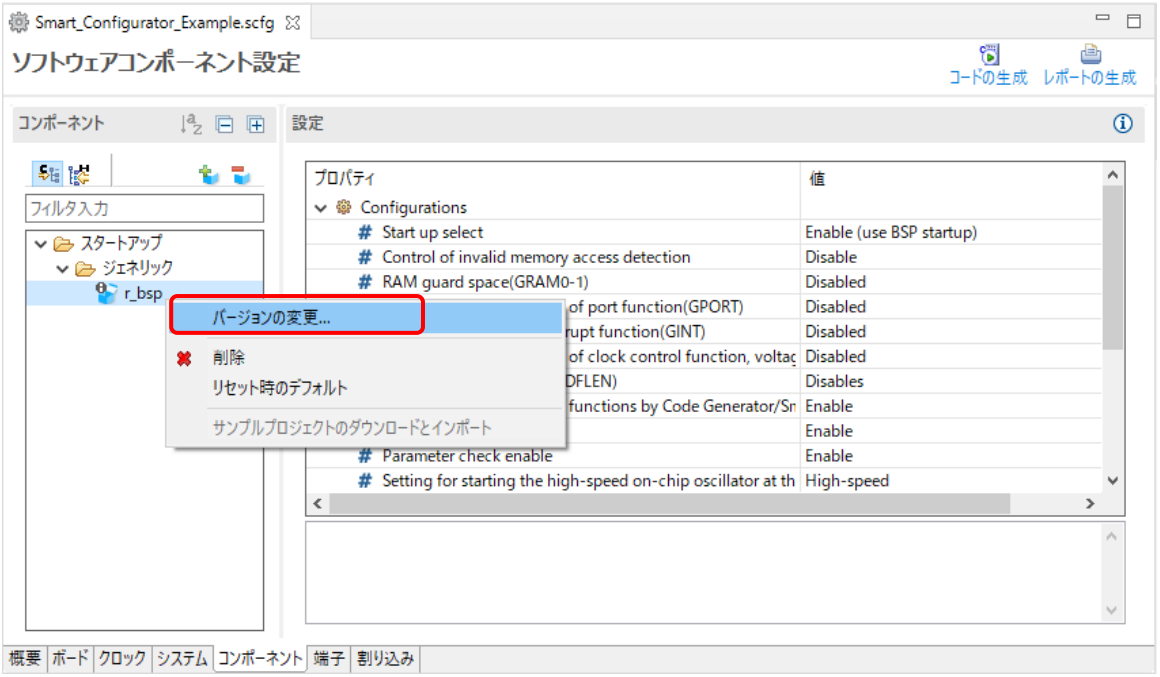

図 4-44 BSP コンフィグレーションのバージョン変更

(2) コンテキスト・メニューから [バージョンの変更...] を右クリックします。

(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。

(4) [次へ] をクリックします。

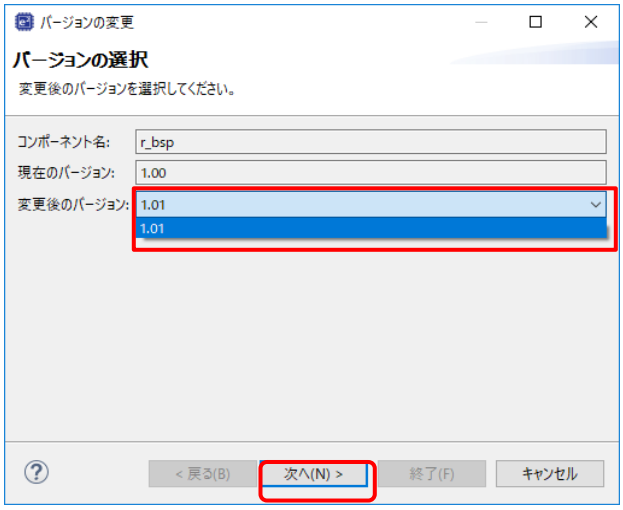

図 4-45 BSP コンポーネントのバージョン選択

(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。

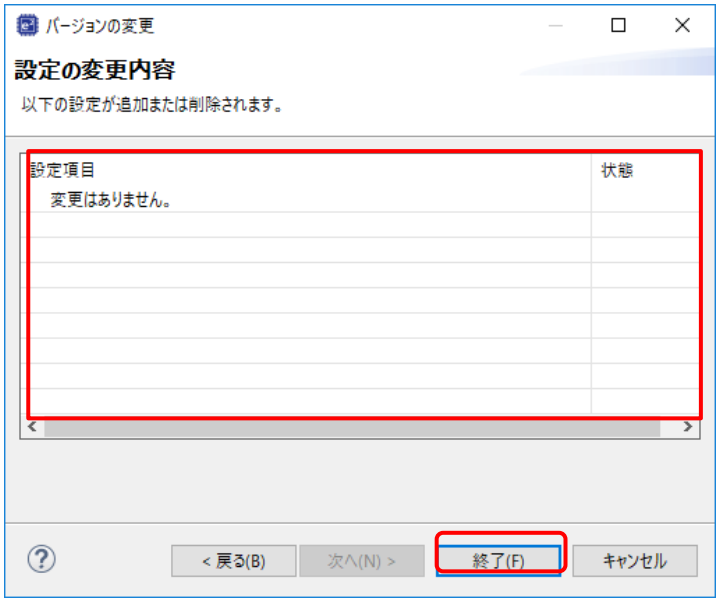

図 4-46 設定変更項目の確認

(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。

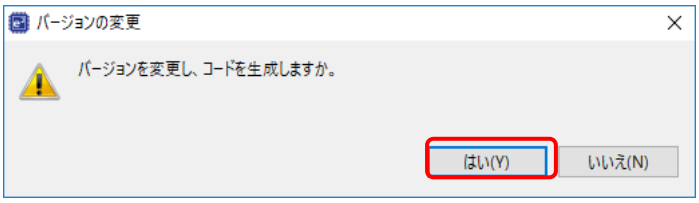

図 4-47 バージョンの変更確認

(7) BSP 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。

4.4.15 コンポーネントのコンフィグレーションのエクスポート

[コンポーネント]ページに[コンフィグレーションのエクスポート]ボタンをクリックすると、現在の設定を*.xml ファイルとしてエクスポートできます。

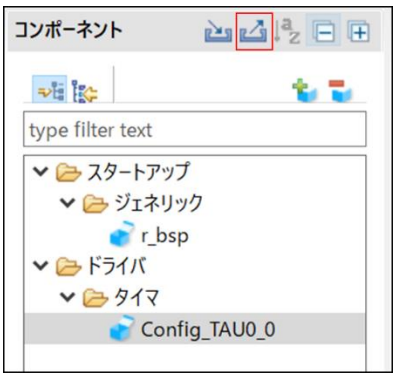

図 4-48 コンフィグレーションのエクスポート

4.4.16 コンポーネントのコンフィグレーションのインポート

[コンフィグレーションのインポート]ボタンをクリックし、エクスポートした*.xml ファイルを選択すると、*.xml ファイルの内容をインポートします。

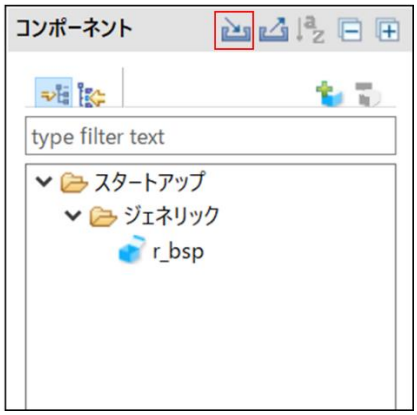

図 4-49 コンフィグレーションのインポート

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 4-15 コード生成の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。

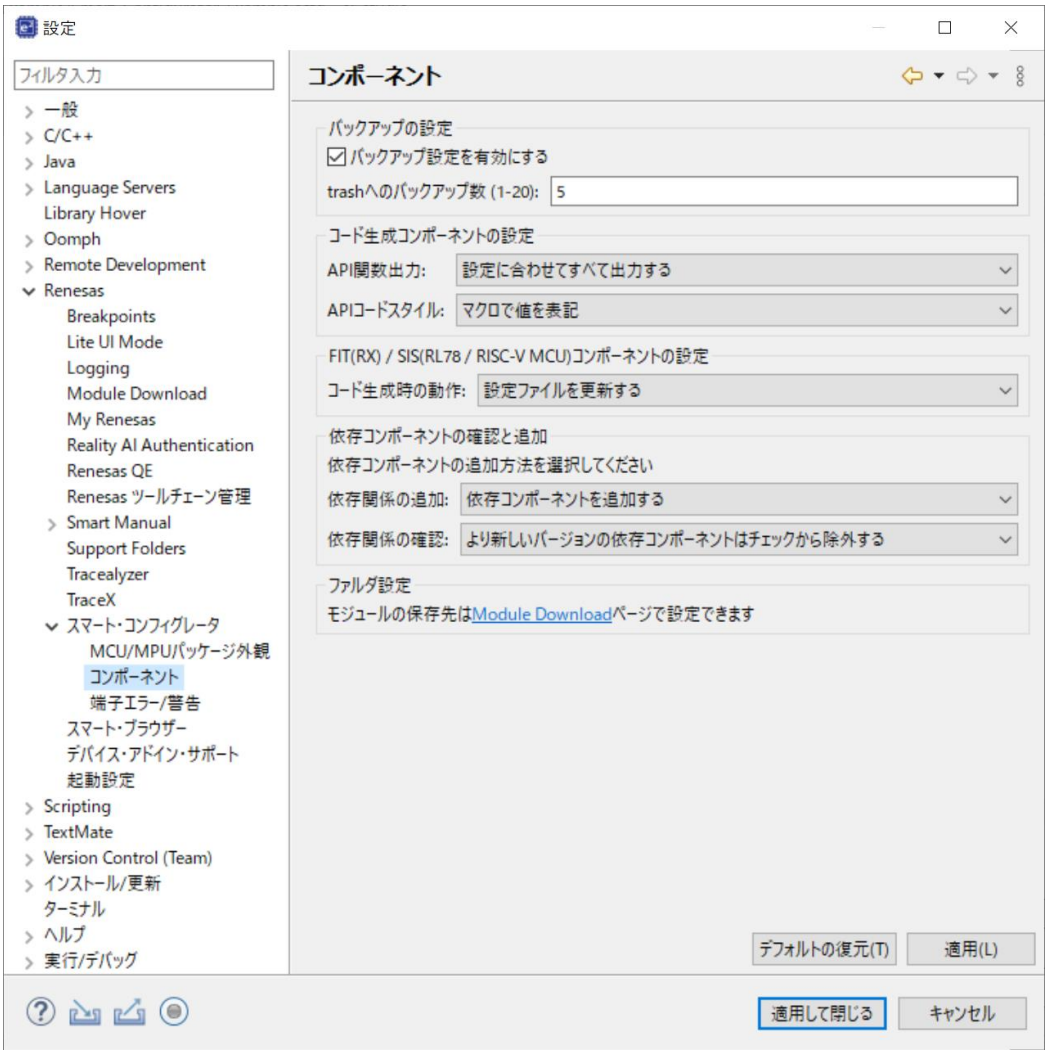

図 4-50 コンポーネントの基本設定

【注】 1. ユーザーは、[trash へのバックアップ数(1~20)] オプション（図 4-51 に示す）を設定することで、バックアップのため、トラッシュフォルダに生成したフォルダの数を制限できます。 制限を超えると、新しいフォルダが古いフォルダを置き換えます。

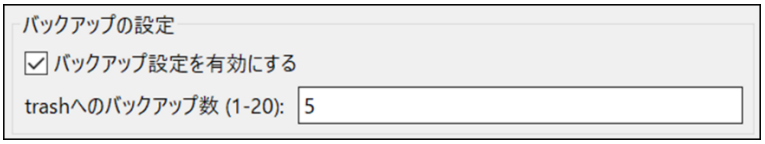

図 4-51 バックアップ数の設定

【注】 2. コード生成時の動作には、「設定ファイルを更新する」と「すべてのコンポーネントファイルを再生成する」の 2 つのオプションがあります。デフォルトの設定は、「設定ファイルを更新する」です。「設定ファイルを更新する」を選択し、コード生成する場合、スマート・コンフィグレータはプロジェクト内にファイルが存在するかどうかをチェックします。ファイルが存在する場合、そのファイルは書きされません。ただし、設定ファイル（例：xxx_config.h）はコード生成するたびに更新されます。「すべてのコンポーネントファイルを再生成する」を選択しコード生成する場合、ファイルが常に上書きされます。

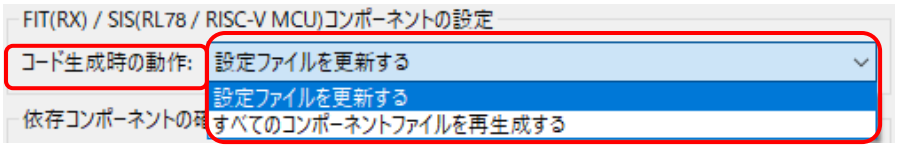

図 4-52 [コード生成時の動作:] の変更

【注】 3. 初期化 API 関数のみを生成したい場合は、[API 関数出力:] リストボックスで「初期化関数のみ出力する」に変更してください。".h "、". c "ファイルの voidR_ {ConfigurationName} _Create（void）、void R_ {ConfigurationName} _Create_UserInit（void）のみが生成されます。 デフォルトのオプション設定「設定に合わせてすべて出力する」に変更するとすべての API 関数が再度生成されます。

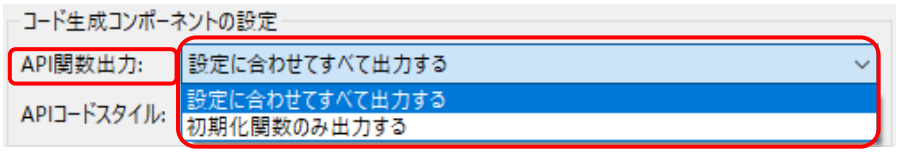

図 4-53 [API 関数出力:] の変更

e² studio 2022-10 から、初期化 API のみ出力する機能を個別の構成 (コード生成コンポーネント) に適用できるようになりました。 選択したコンポーネントを右クリックし、コンテキスト・メニューから [初期化 API のみを出力] を選択してください。

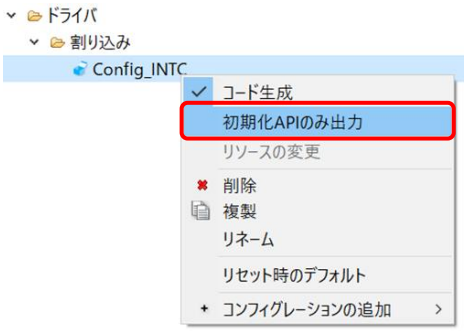

図 4-54 [初期化 API のみ出力:] の変更

【注】 4. HEX 値でコードを生成するには、下図の [マクロを使用せず即値(16 進数)で表記] オプションに変更してください。[マクロで値を表記] に戻すと、マクロ記述ですべての API が生成されます。

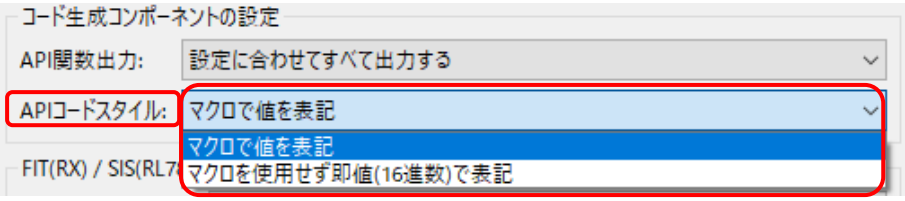

図 4-55 [API コードスタイル:] の変更

【注】 5. 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。

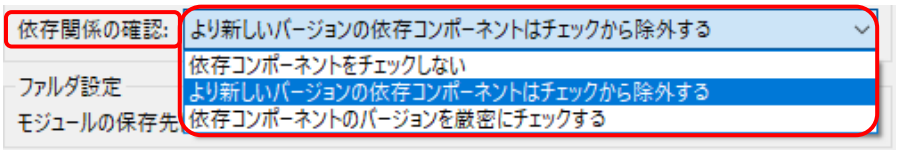

図 4-56 [依存関係の確認:] の変更

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。

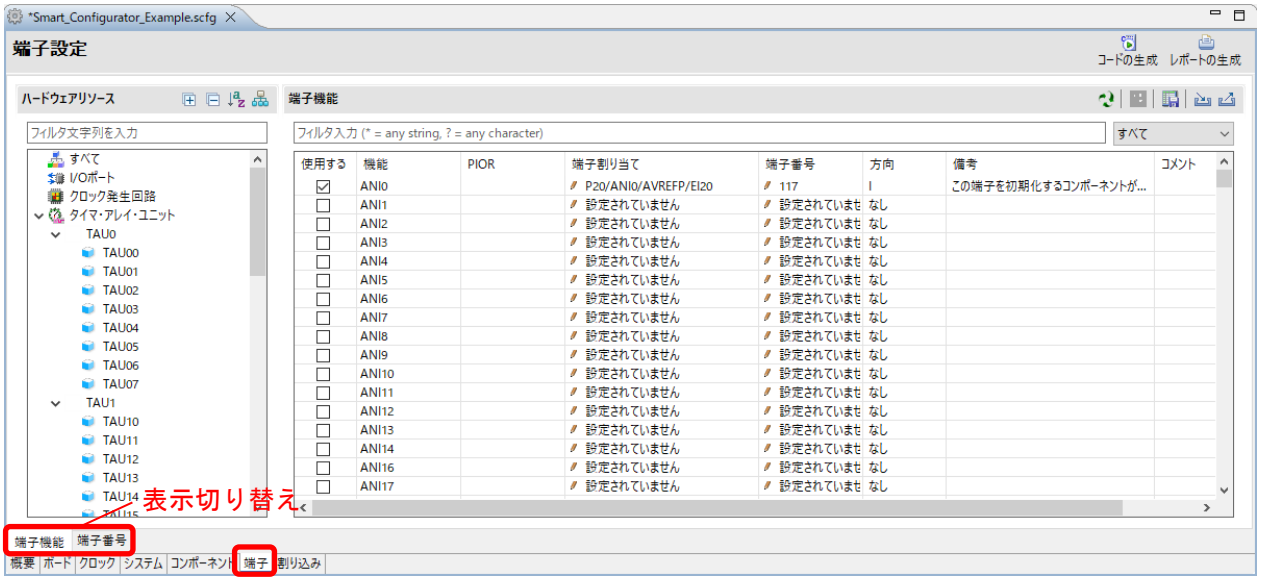

図 4-57 端子ページ（端子機能）

[ボード] ページでボードを選択すると、[ボード機能] にボードの初期端子機能情報が表示されます。また、[機能] 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。

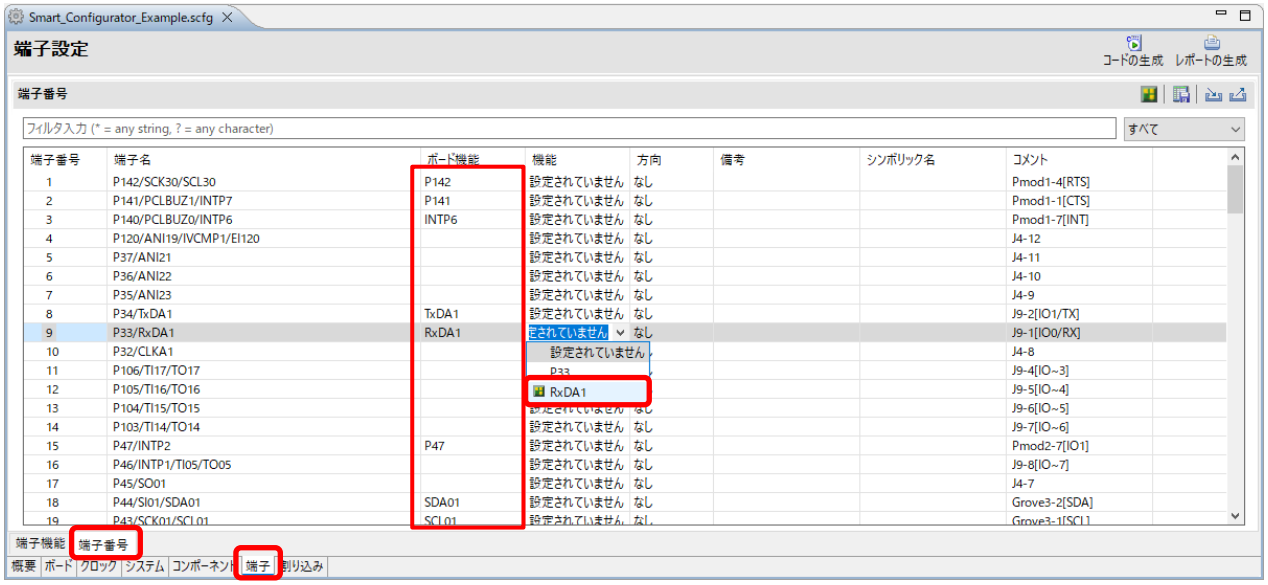

図 4-58 端子ページ（端子番号）

4.5.1　PIOR 機能による端子割り当ての変更

PIOR「フィルタ機能」は、端子機能設定の管理、端子機能設定の再構成、端子機能の競合のチェックに便利な機能です。PIOR 機能の割り当てを変更するには、以下の手順で行います。

(1)　ツールテキスト入力ボックスに「pior1」と入力し、PIOR1 に関連するすべての端子機能を表示させます。

(2)　端子割り当ての 1 つを変更すると、PIOR1 に関連するすべての端子機能割り当てが自動的に再割り当てされます。

(3)　端子エラーメッセージは、[備考] 欄と [構成問題ビュー] に表示されます。

(4)　端子エラーメッセージが表示されたら、端子割り当てを再構成する必要があります。

図 4-59 PIOR フィルタ機能

PIOR 設定コードは、bsp ファイル：\ProjectDir\src\smc_gen\r_bsp\r_config\ r_bsp_config.h に生成されます。 PIOR 設定コード値を変更したい場合は、関連するピンの割り当てを変更して、再度コードを生成してください。

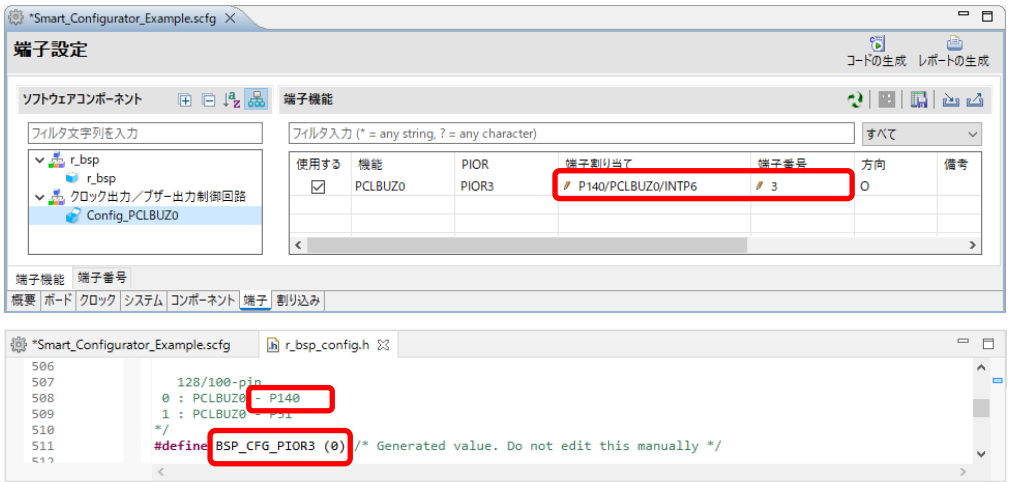

図 4-60 PIOR のコード生成

4.5.2　ソフトウェア・コンポーネントの端子配置変更

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_INTC）。

(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、端子配置を変更します（例：P46 から P56）。

(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、端子配置を変更します。クリックするごとに、機能を持つ端子が表示されます。

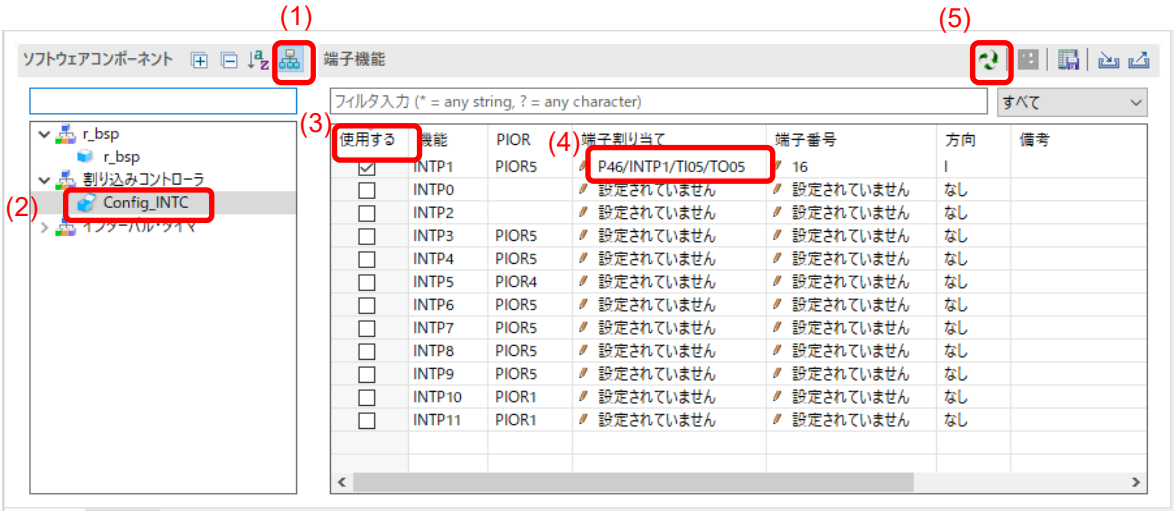

図 4-61 端子設定–［端子機能］リストの端子配置設定

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“この端子を初期化するコンポーネントがありません”という注意書きがつけられます。

4.5.3　MCU/MPU パッケージを使用した端子の設定

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設定を視覚化します。MCU/MPU パッケージビューを画像ファイルに保存し、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] ボタンをクリックするか、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。

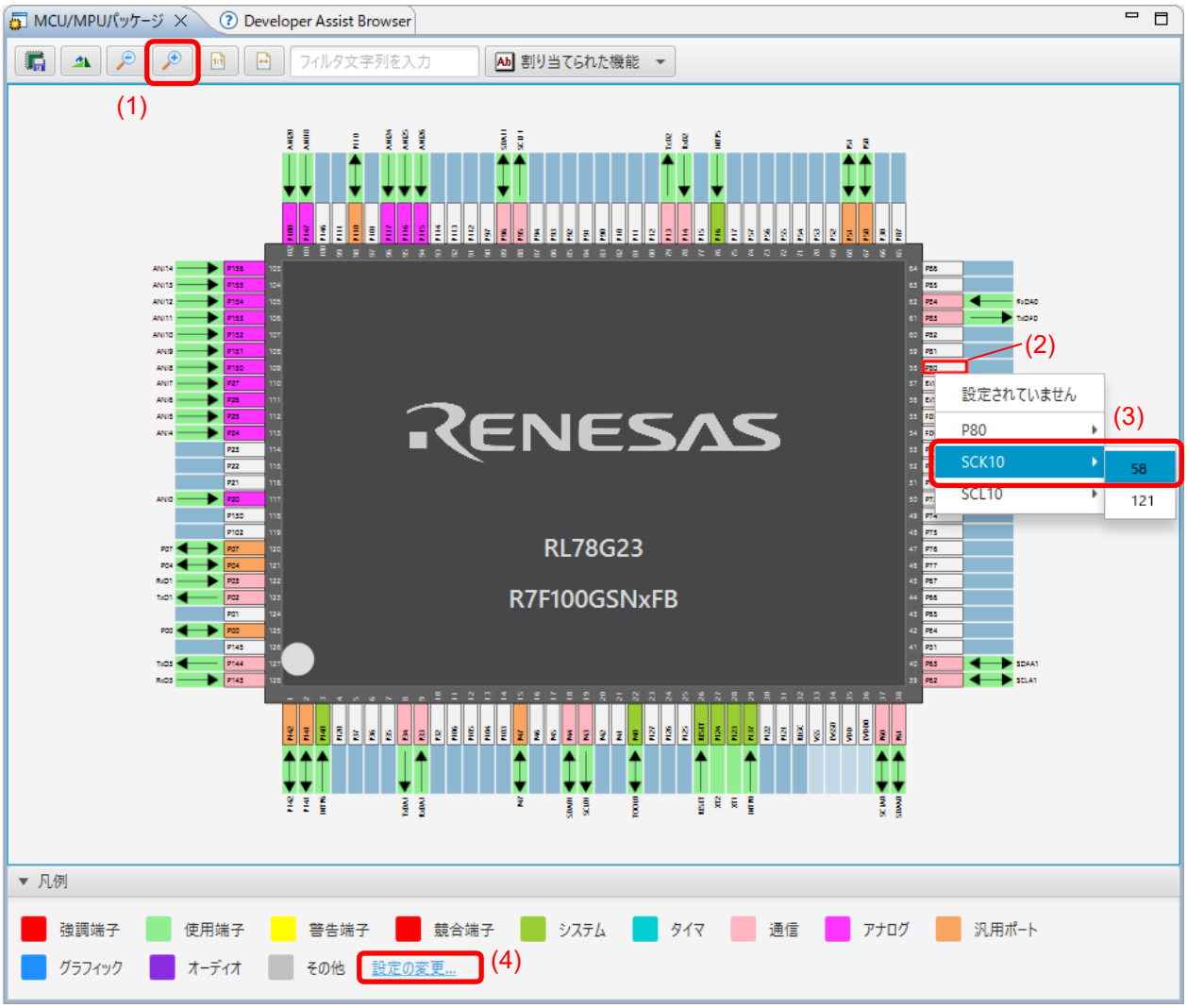

図 4-62 MCU/MPU パッケージを使用した端子設定

4.5.4　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。端子機能か端子番号に移動するには、以下の手順で移動します。

(1)　[端子機能] タブで、使用する端子機能を右クリックし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] タブに移動し、(1) で選択した端子機能の端子番号を表示します。

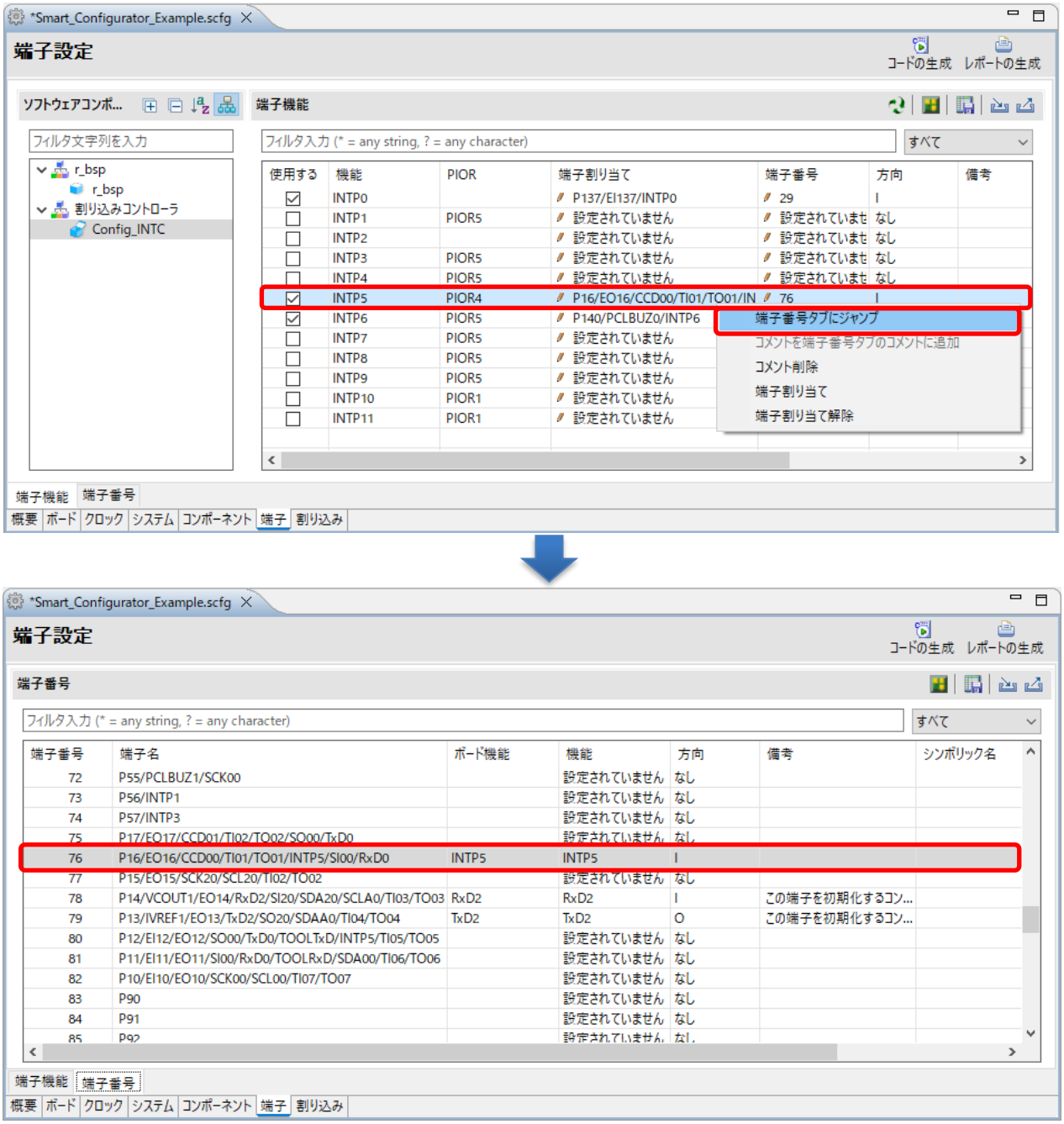

図 4-63 端子番号タブにジャンプ

4.5.5　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。

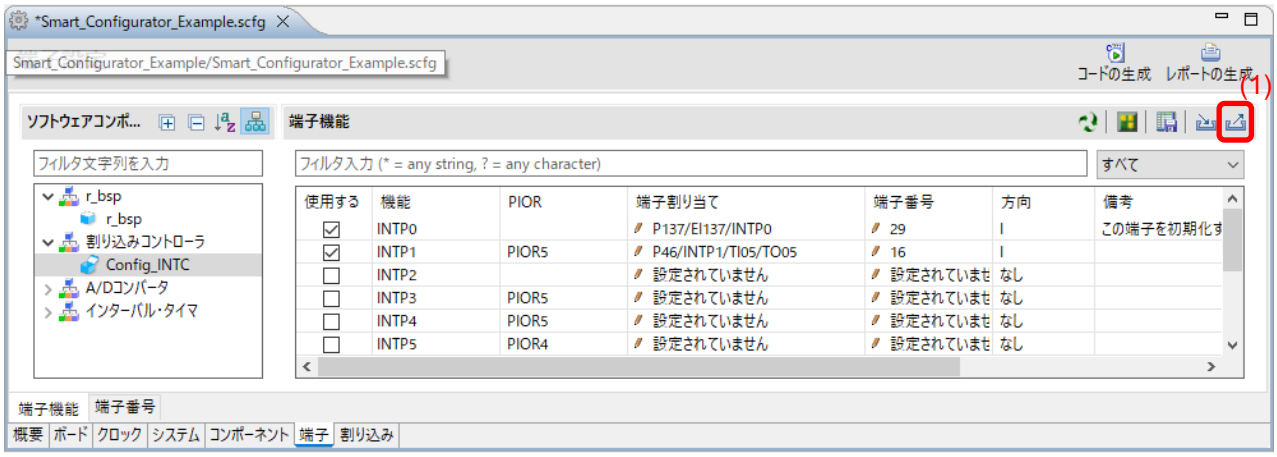

図 4-64 端子設定を XML ファイルへエクスポートする

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。

4.5.6　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。

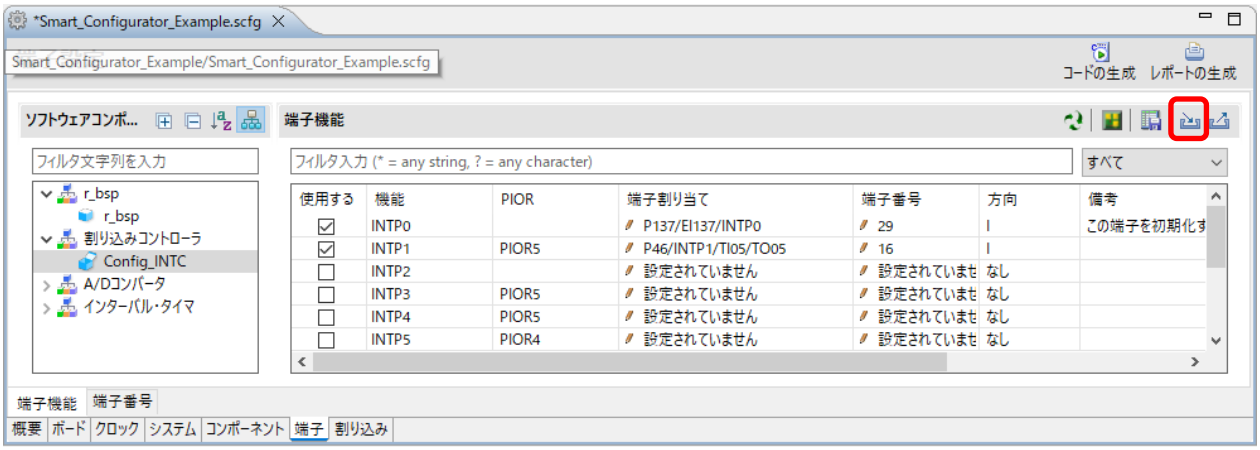

図 4-65 端子設定を XML ファイルからインポートする

【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。

4.5.7　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。端子を一括で設定するには、以下の手順で行います。

(1)　[ボード] ページで、 [カスタムユーザーボード] 以外のボード設定情報を選択します。
（4.1.2 ボード選択錯误!未找到引用源。参照）

(2)　MCU/MPU パッケージで [ボード機能] を選択します。 (ボードの初期端子設定が参照できます)

(3)　[端子設定] ページを開き、[ボードの初期端子割り当ての設定] ボタンをクリックします。

(4)　[ボードの初期端子割り当ての設定] ダイアログが開いたら [すべて選択] をクリックしてください。

(5)　[OK] ボタンをクリックします。

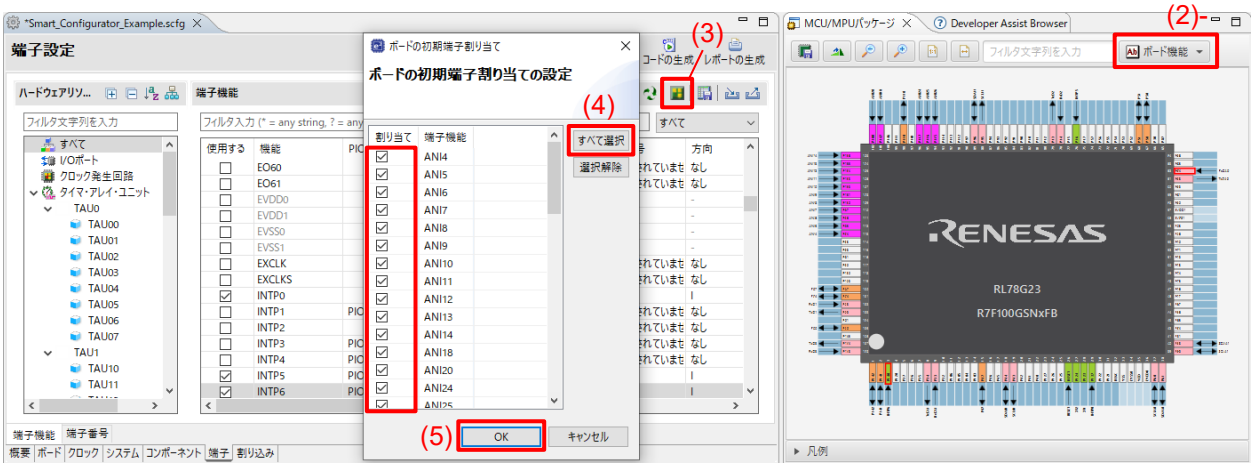

図 4-66 ボードの初期端子設定

端子設定を一度に設定しない場合は、手順 (4) で個別に設定してください。

4.5.8　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。

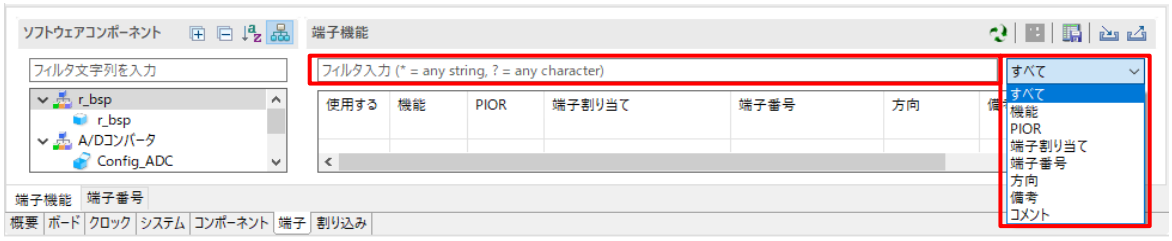

図 4-67 ［端子機能］タブのフィルタ

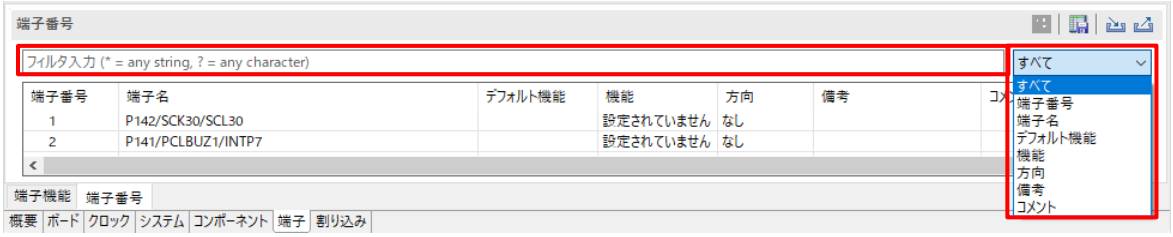

図 4-68 ［端子番号］タブのフィルタ

4.5.9　端子エラー/警告の設定

[端子エラー/警告] 設定を使用して、コンフィグレーションチェックビューの端子設定問題の表示方法を制御できます。制御を変更したい場合は、[新規コンポーネント] ダイアログで [基本設定] リンクをクリックし、[設定] ダイアログを表示します。次に、[スマート・コンフィグレータ] の [端子エラー/警告] を選択し、設定を変更します。

図 4-69 端子エラー/警告の設定

例：[ソフトウェアコンポーネント] の設定を「情報」から「エラー」に変更。

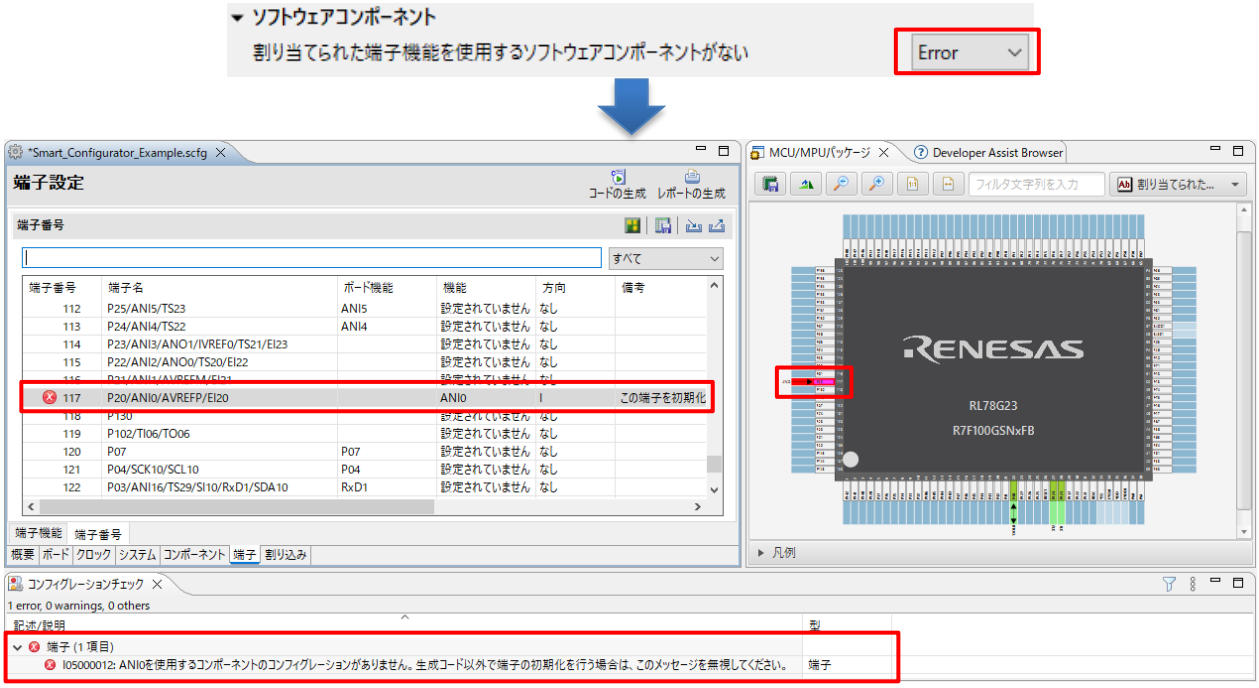

図 4-70 ソフトウェアコンポーネントのエラー

4.6　割り込み設定

[割り込み] ページには、各ベクタ番号によりすべての割り込みが表示され、[コンポーネント] ページで選択した周辺モジュールの割り込みを確認・設定できます。 コード生成タイプのコンポーネントで割り込みを使用すると、割り込みの状態が「使用中」に変わります。

(1)　使用中の割り込みのみを表示するには、[設定した割り込みの表示)] ボタンをクリックしてください。

(2)　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。

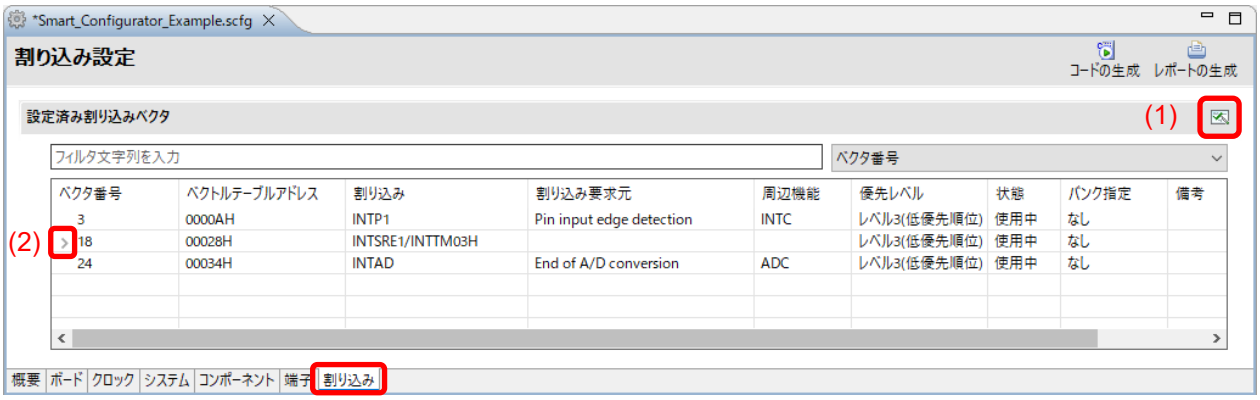

図 4-71［割り込み］ページ

4.6.1　割り込み優先レベルの設定

[割り込み] ページの割り込み優先レベルは、以下の手順で変更できます。

(1)　優先レベルを変更したい割り込みを表示します。

(2)　優先レベルセルをクリックし、ドロップダウンリストから割り込み優先レベルを設定します。

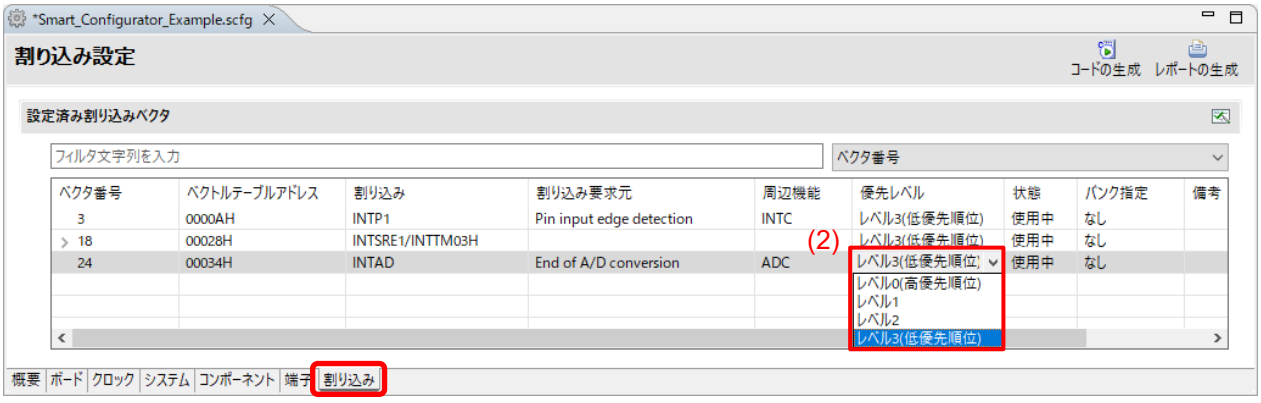

図 4-72 割り込み設定

4.6.2　割り込みバンクの設定

[割り込み] ページの割り込みバンクは、以下の手順で変更できます。

(1)　バンクを変更したい割り込みを表示します。

(2)　バンク指定セルをクリックし、ドロップダウンリストからバンク（None / 1 / 2 / 3）を設定します。

(3)　優先度の異なる複数の割り込みに対して同じバンクを指定すると警告マークが表示され、[備考] と [Configuration Problems] に警告メッセージが表示されます。必要に応じて再設定してください。

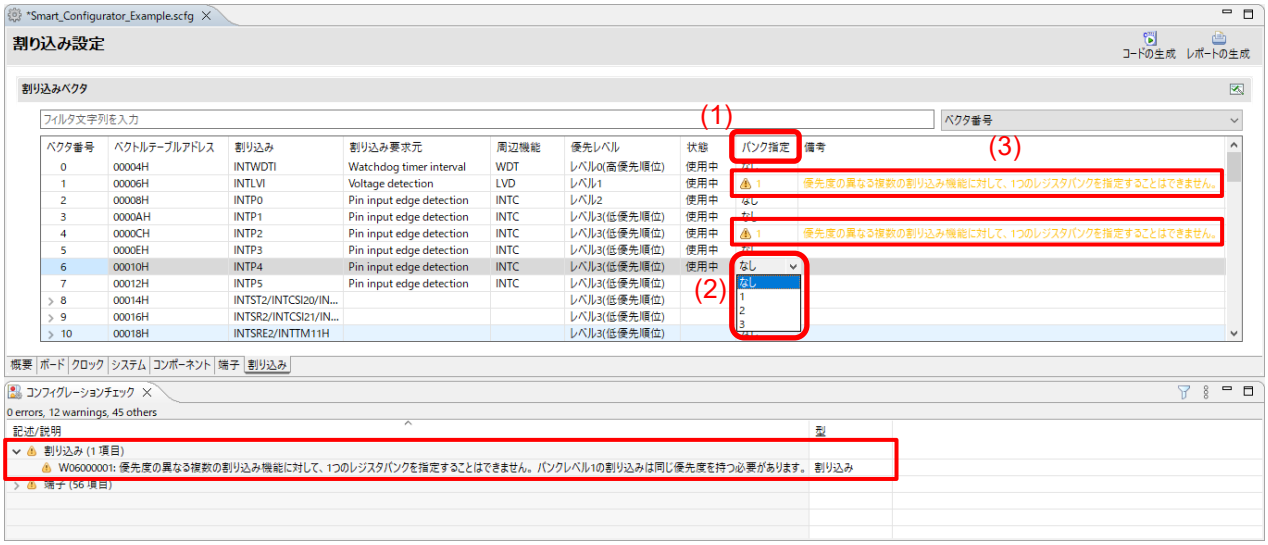

図 4-73 割り込みバンク設定

割り込みバンクの設定は、以下のようにコードに反映されます。

(1)　CCRL の場合、コンポーネントの{*ConfigurationName*} _user.c ファイルに反映されます。

```
/***********************************************************************
Pragma directive
***********************************************************************/
#pragma interrupt r_Config_INTC_intp0_interrupt(vect=INTP0, bank=RB1)
#pragma interrupt r_Config_INTC_intp1_interrupt(vect=INTP1)
/* Start user code for pragma. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
```

図 4-74 CCRL 割り込みバンクコード

(2)　LLVM の場合、<ProjectDir>\src\smc_gen\general\r_cg_interrupt_handler.h ファイルに反映されます。

```
/*
 * INT_P0 (0x8)
 */
void r_Config_INTC_intp0_interrupt(void) __attribute__ ((interrupt(bank=RB1)));
```

図 4-75 LLVM 割り込みバンクコード

実際に生成されるコード仕様は、コンパイラによって異なります。 詳細については、対応する IDE のユーザーガイドを参照してください。

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。

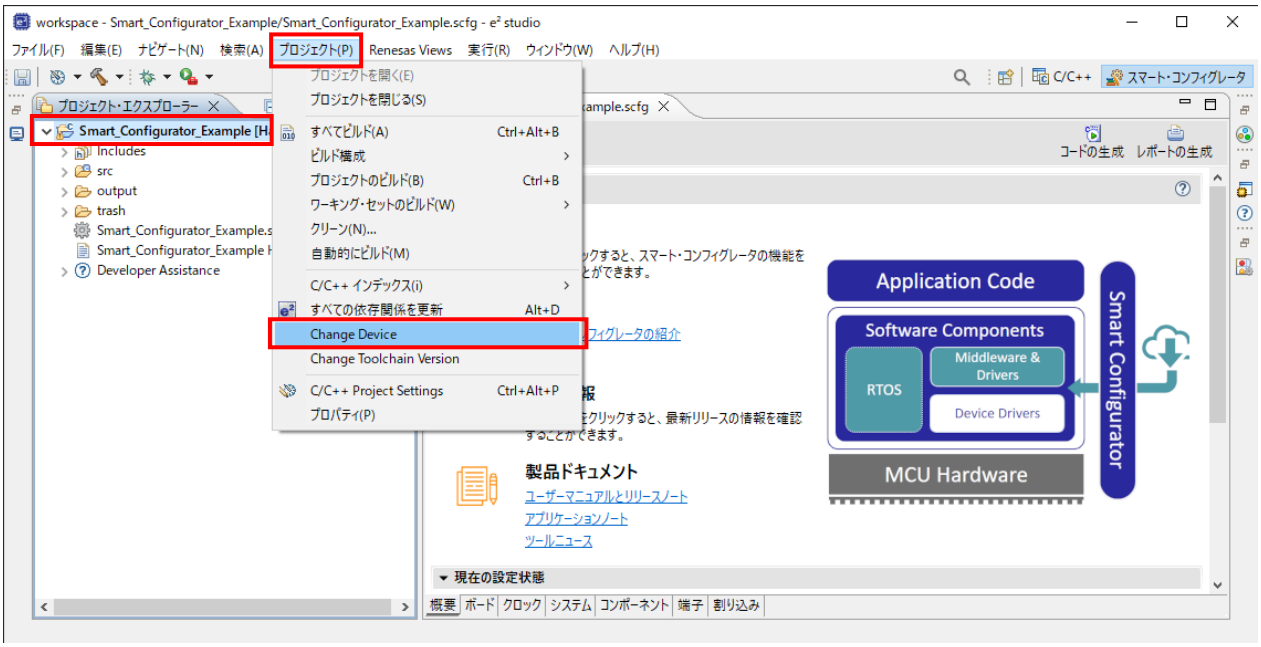

図 4-76 e² studio の [Change Device]

(2)　Target Board からボードを選択すると、デバイスは自動的に選択されます。

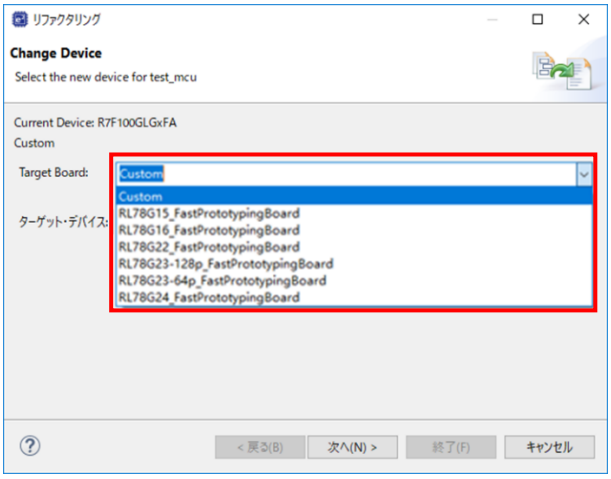

図 4-77 Target Board の選択

デバイスを直接選択したい場合は、ターゲット・デバイスからデバイスを選択します。

(例： RL78/G23 R7F1000GPGxFB を選択)

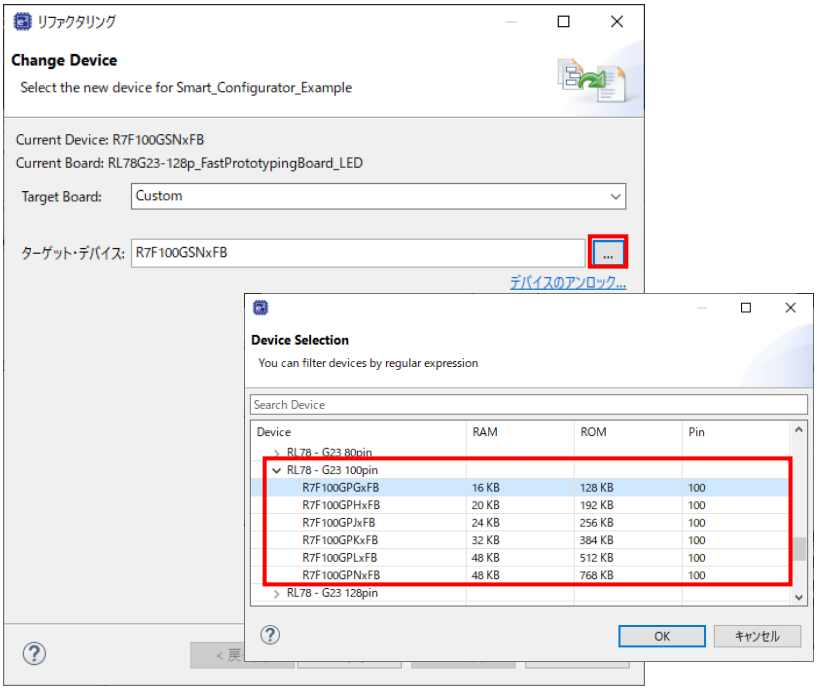

図 4-78 ターゲット・デバイスの 選択

(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。

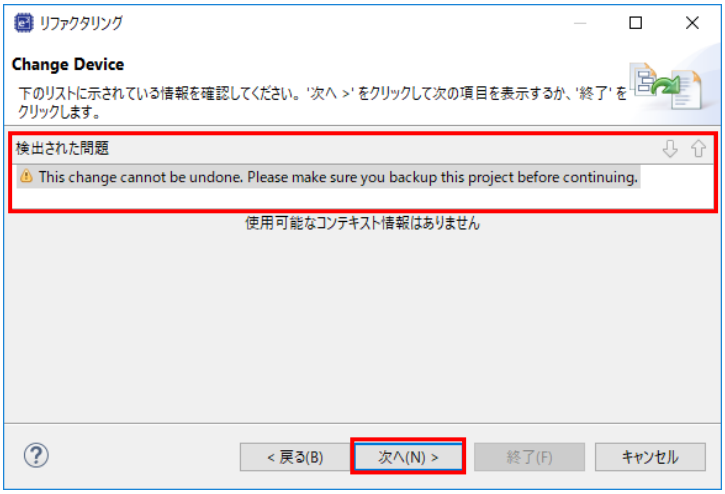

図 4-79 検出された問題

表 4-1 検出された問題のメッセージ

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。

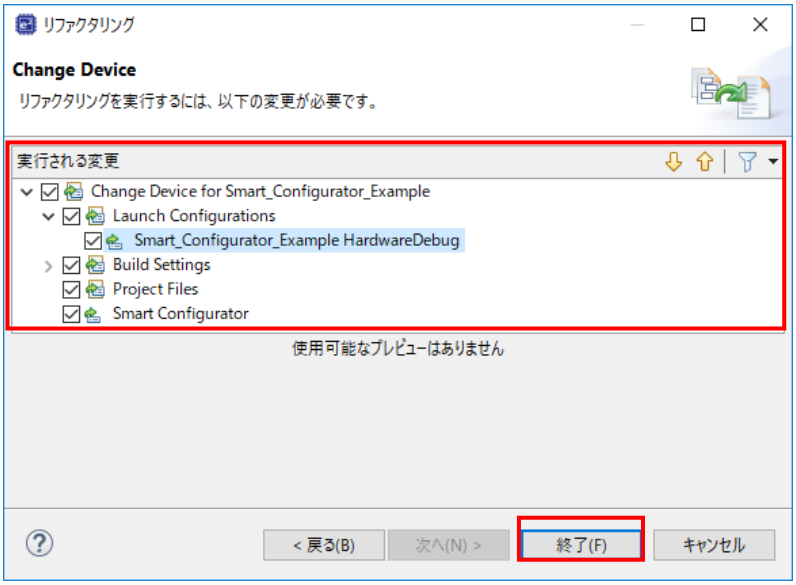

図 4-80 実行される変更項目確認

(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。

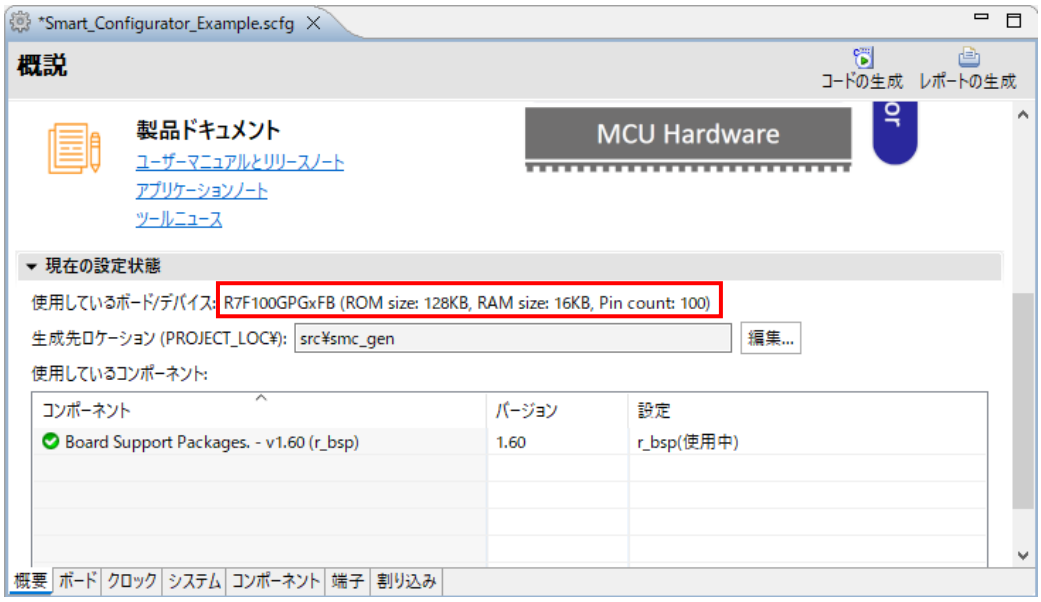

図 4-81 デバイス更新確認

(6) コンソース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

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、リソースの競合に関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。なお、競合が発生していてもコードは生成できます。

5.1　リソースの競合

同じリソース（例：S12AD1）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、ユーザーに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。

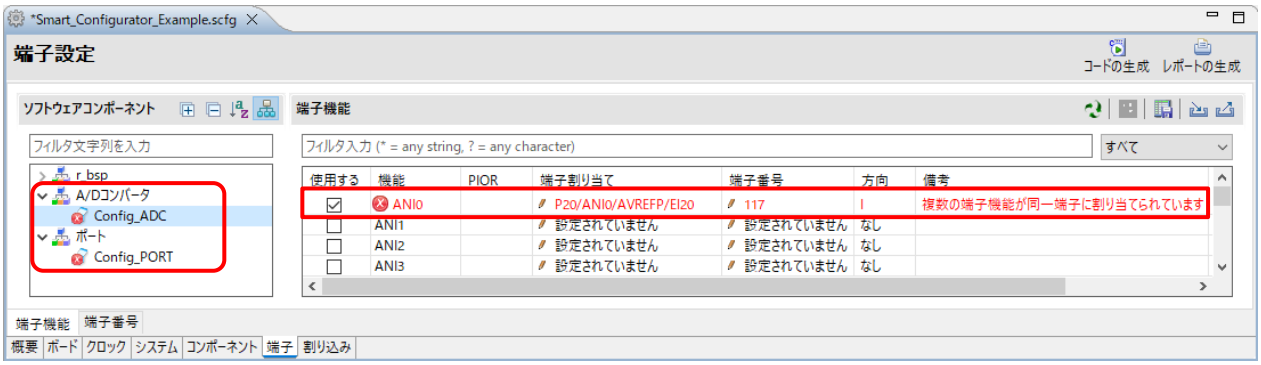

図 5-2 端子の競合

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

エラーマークが表示されているツリー・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。

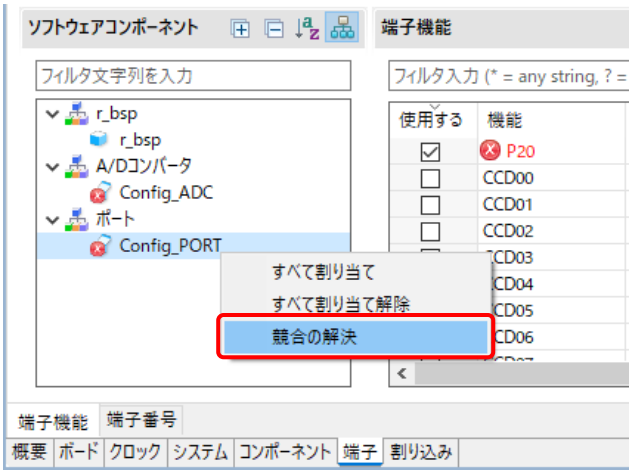

図 5-4 端子競合の解決

選択されたノードの端子機能は、他の端子に再度割り当てられます。

6.　ソースの生成

ソースの生成は、コンフィグレーションチェックビューで競合が発生していても生成できます。

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。

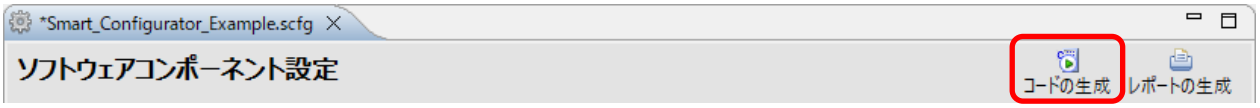

図 6-1 ソースファイルの生成

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】作成したソースファイルを sms_gen フォルダに入れると、ソースコード生成時に消去されます。

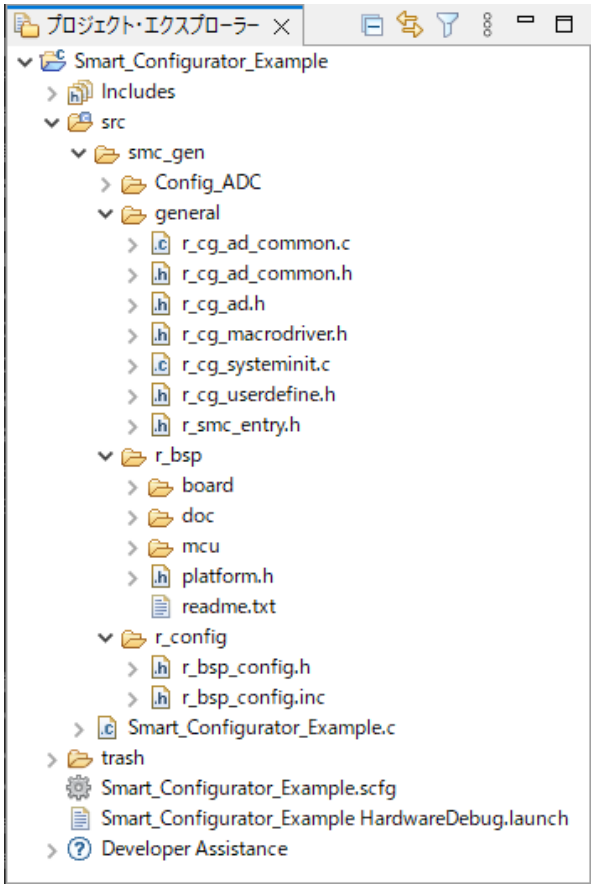

図 6-2 プロジェクト・エクスプローラー内のソースファイル

6.2　コード生成場所の変更

(1)　コードの生成場所を変更するには、[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。

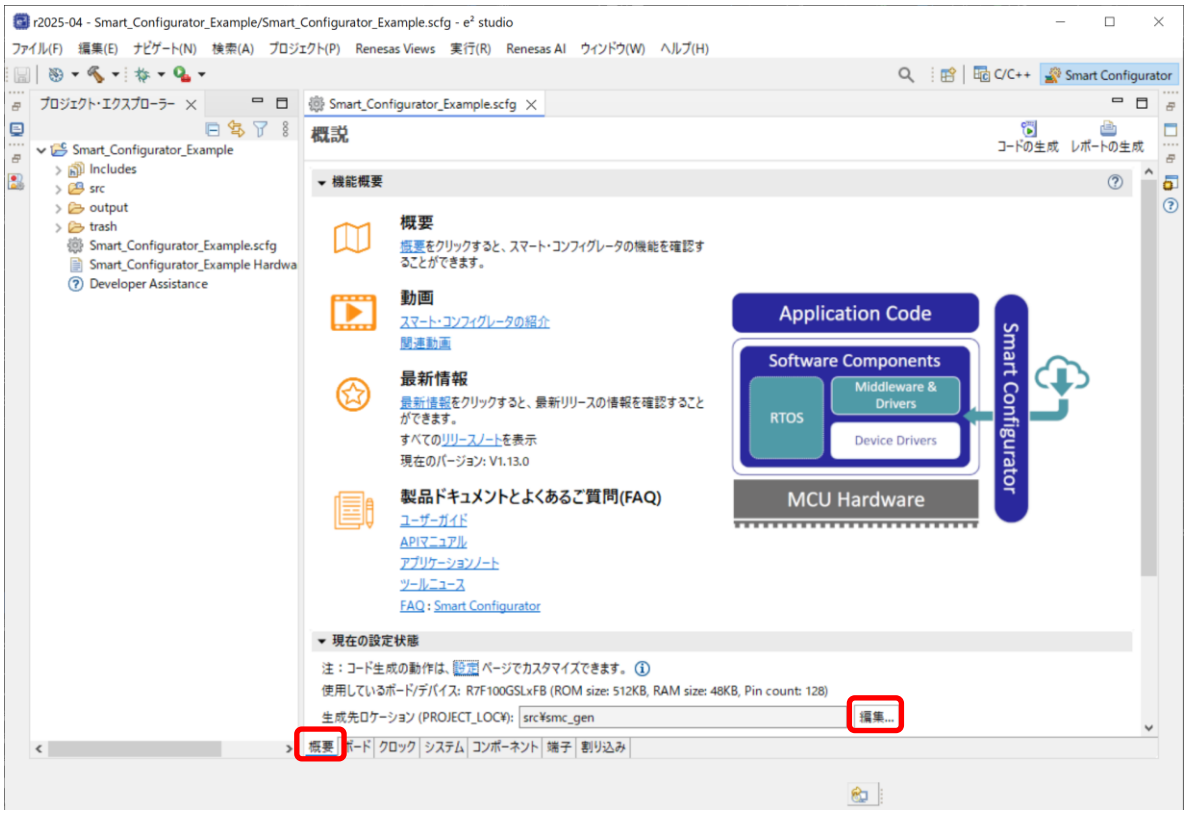

図 6-3 生成先ロケーションの編集

(2)　[フォルダの選択] ダイアログで、コード生成用の空のフォルダを選択するか、新しいフォルダを作成します。

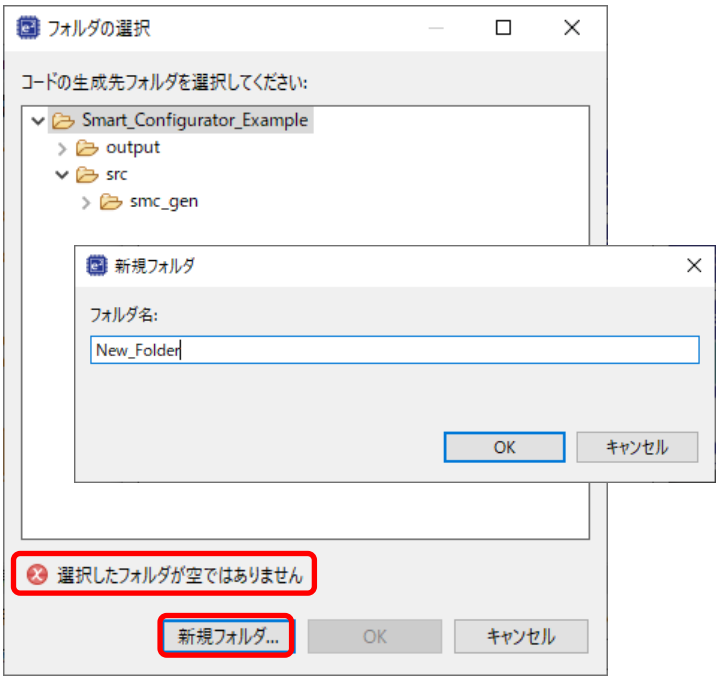

図 6-4 生成先フォルダの選択

(3) [コードの生成] ボタンをクリックすると、ソース コードは新しい場所に生成されます。
[概要] ページで、現在のコード生成先の場所を確認できます。

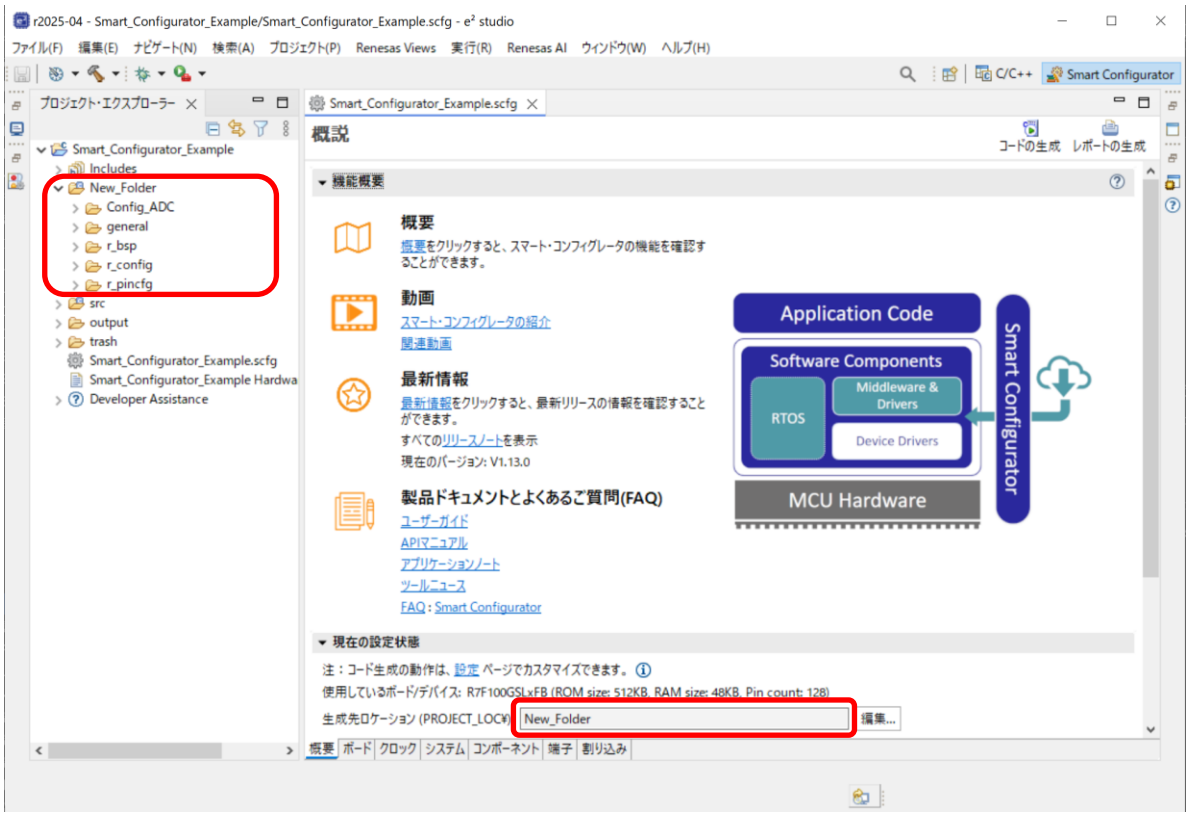

図 6-5 新しいコード生成場所に生成

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 図 6-6 生成ファイルの構成とファイル名に示します。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 RL78 Software Integration System Modules 名、“*ConfigName*”はコンポーネント設定で形成されたコンフィグレーション名、“*Project name*”は e² studio で設定されたプロジェクト名を示します。

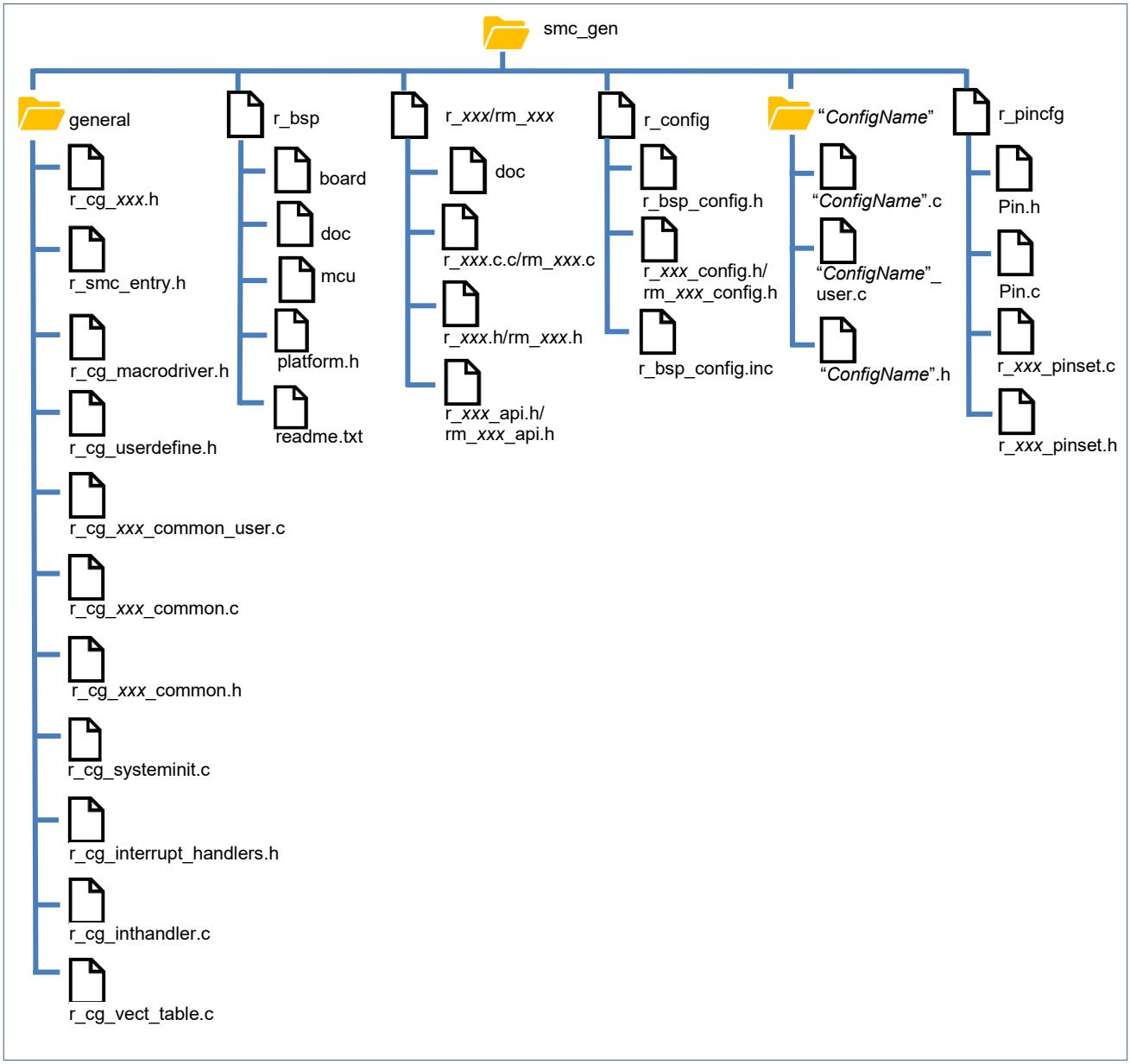

図 6-6 生成ファイルの構成とファイル名

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)	このファイルは SFR レジスタを設定するためのマクロ定義を含みます。
	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_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_cg_systeminit.c	このファイルは常時生成されます。 全コンポーネントの Create() 関数を含みます。周辺機能の初期化に使用します。
	r_cg_*xxx*_common_user.c(*1)	このファイルは使用する周辺機能の共通の割り込み API 関数を含みます。
	r_cg_*xxx*_common.c(*1)	このファイルは対応する周辺機能を使用する場合に生成されます。
	r_cg_*xxx*_common.h(*1)	このファイルは対応する周辺機能を使用する場合に生成されます。
	r_cg_interrupt_handlers.h(*2)	このファイルには、すべての割り込みルーチンの宣言が含まれています。構成が作成されていない場合、すべての割り込みルーチンがデフォルトのルーチンです。 特定の構成が作成された場合、この構成の対応する割り込みルーチンの宣言がデフォルトのルーチン宣言に置き換わります。
	r_cg_inthandler.c(*2)	このファイルには、すべてのデフォルトの割り込みルーチン定義が含まれています。
	r_cg_vect_table.c(*2)	このファイルには、すべての割り込みルーチンエントリアドレスを含む割り込みベクタテーブルが含まれています。構成が作成されていない場合、すべての割り込みルーチンエントリアドレスがデフォルトです。 特定の構成が作成された場合、この構成の対応する割り込みルーチンエントリアドレスがデフォルトのルーチンエントリアドレスに置き換わります。
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（*doc* フォルダ） 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*/ rm_*xxx*(*1)		このフォルダは、プロジェクトに追加された RL78 Software Integration System モジュール用に生成されます。 • doc フォルダ：アプリケーションノート • r_*xxx*.c/rm_*xxx*.c(Note*1)：ソースファイル • r_*xxx*.c/rm_*xxx*.h(Note*1)：ヘッダファイル • r_*xxx*_api.h/rm_*xxx*_api.h(Note*1)：すべての API とインターフェース定義のリスト
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、割り込み等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_config.inc	このファイルは常時生成されます。 構成ヘッダファイルを生成します。
	r_*xxx*_config.h/rm_*xxx*_config.h (*1)	このファイルは、プロジェクトに追加されるすべての RL78 Software Integration System ドライバ／ミドルウェアの構成ヘッダファイルです。
r_pincfg	Pin.h	このファイルは常時生成されます。 端子のシンボルをサポートするために生成され、smc_entry.h に含まれています。
	Pin.c	このファイルは常時生成されます。 [端子] ページで有効になっている端子設定が生成されます。PIOR を設定する必要のない端子設定を生成するだけです。
	r_*xxx*_pinset.c	このファイルは、RL78 Software Integration System の端子設定ソースファイルです。
	r_*xxx*_pinset.h	このファイルは、RL78 Software Integration System の端子設定ヘッダファイルです。
{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。 このフォルダ内の API 関数には、*ConfigName*（設定名）を含んだ名称がつけられています。
	{ConfigName}.c	このファイルは、ドライバを初期化する関数（R_ConfigName_Create）、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。

【注】 *1. *xxx* は周辺機能名を示します。
*2. LLVM ツールチェーンで生成されるファイルです。

6.4　クロック設定

[クロック] ページで選択したクロック・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（パッケージ、スタックサイズなど）も含みます。

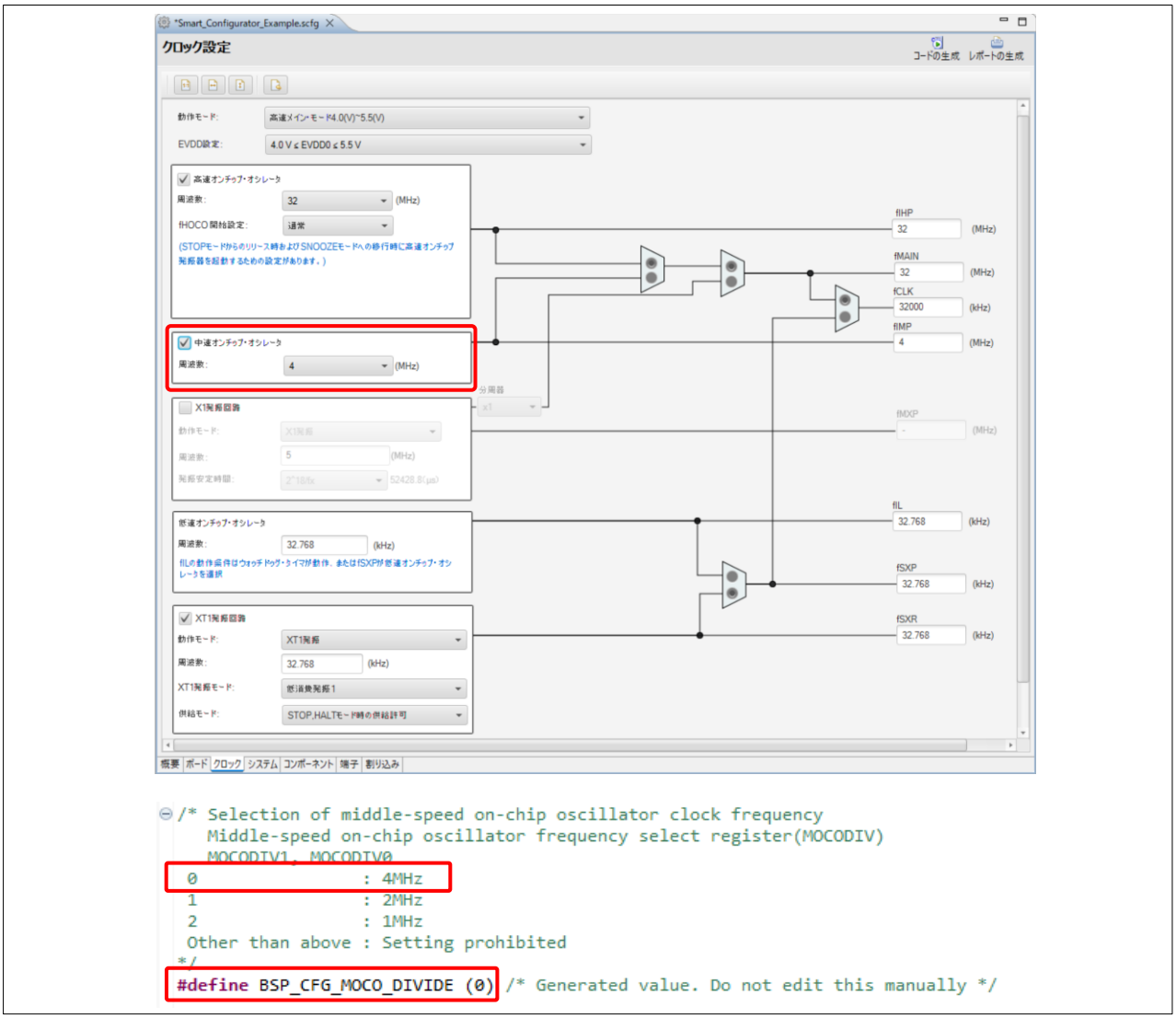

図 6-7 メインクロックをクロック・ソースとして選択した場合のクロック設定

フォルダ	ファイル	マクロ/関数	説明
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。*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・コンフィグレータによってマクロが生成されます（例：パッケージタイプ）。マクロの詳細は、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。

【注】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（8 章を参照）。

6.5　端子設定

[端子] ページの設定は、コンポーネントにより下記に示す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。

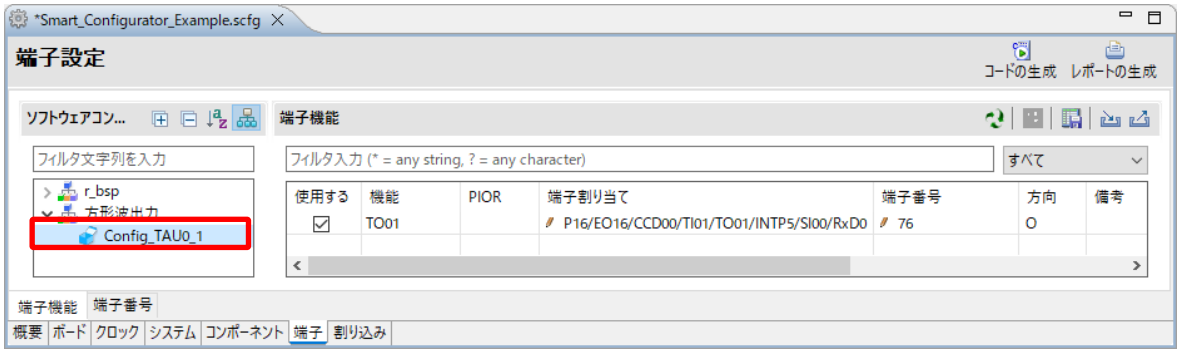

図 6-8 Config_TAU0_0 の端子設定

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する端子を API 関数が初期化します。main() 関数を実行する前に、r_cg_systeminit はこの関数を呼び出します。

2)　RL78 Software Integration System コンポーネントの端子初期化

端子機能は\src\smc_gen\r_pincfg*{ConfigName}*_pinset.c の R_*{PeripheralName}*_PinSetInit で初期化されます。このファイルの API 関数は、アプリケーションコードからユーザーによって呼び出されます。

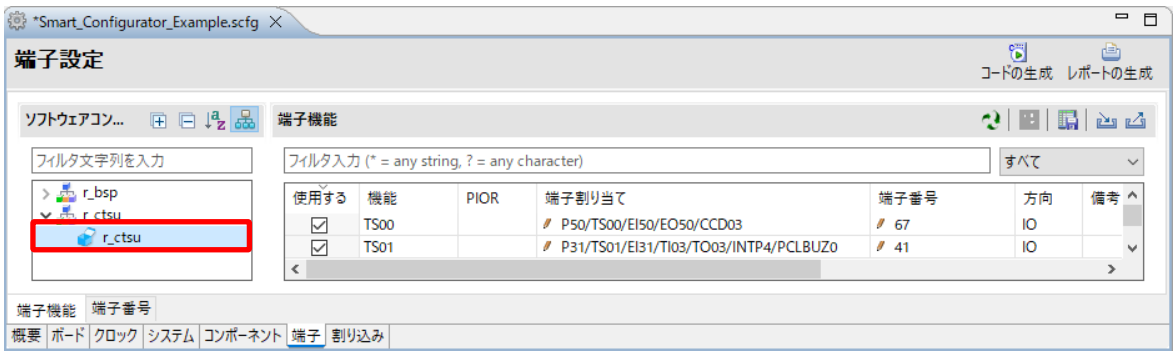

図 6-9 r_ctsu の端子設定

フォルダ	**ファイル**	**関数**	**ドライバ**	**説明**
r_pincfg	*{ConfigName}*_pinset.c	R_*{PeripheralName}*_PinSetInit	RL78 Software Integration System	このドライバが使用する端子を API 関数が初期化します。ユーザーは、main() 関数でこの関数を呼び出す必要があります。

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。

割り込み関数は、ファイル\src\smc_gen*{ConfigName}**{ConfigName}*.c の R_ConfigName_Create で初期化されます。

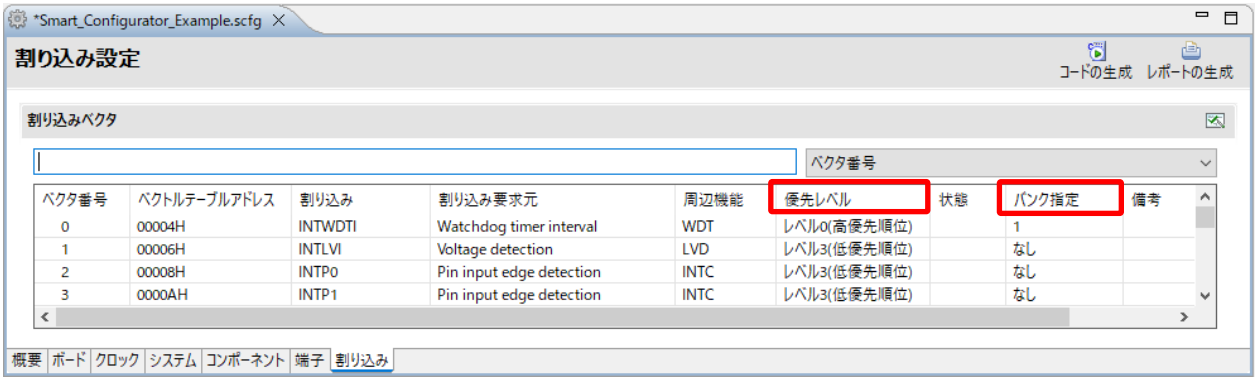

図 6-10 割り込み設定

項目	フォルダ	ファイル	ドライバ	説明
Priority	***{ConfigName}***	*{ConfigName}*.c	Code Generator	優先レベルは、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_systeminit はこの関数を呼び出します。
Bank (CCRL Project)	***{ConfigName}***	*{ConfigName}*_user.c	Code Generator	割り込みを次のように宣言します。 #pragma interrupt "Interrupt API Name"(vect="Interrupt Name", bank=RBbankNumber) 図 4-74 CCRL 割り込みバンクコードを参照してください。
Bank (LLVM Project)	***{\projectDIR\src\smc_gen\general}***	r_cg_interrupt_handler.h		割り込みを次のように宣言します。 Void "Interrupt API Name" (void) __attribute__ ((interrupt(bank=RBbankNumber))); 図 4-75 LLVM 割り込みバンクコードを参照してください。

7. ユーザープログラムの生成

ここでは、スマート・コンフィグレータが出力したソースファイルへのカスタムコード追加方法について説明します。

7.1 コード生成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 xxx. Do not edit comment generated here */

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

［コード生成］の場合、指定した周辺機能ごとに 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、A/D コンバータ（リソース ADC）の場合、xxx は“ADC”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、「*.c」ファイルの先頭と最後および「*.h」ファイルの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 ADC の例（Config_ADC_user.c）を示します。

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

/***********************************************************************
Pragma directive
***********************************************************************/
#pragma interrupt r_Config_ADC_interrupt(vect=INTAD)
/* Start user code for pragma. 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_ADC_Create_UserInit
* Description  : This function adds user code after initializing the AD converter.
* Arguments    : None
* Return Value : None
***********************************************************************************************/
void R_Config_ADC_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_ADC_interrupt
* Description  : This function is INTAD interrupt service routine.
* Arguments    : None
* Return Value : None
***********************************************************************************************/
static void __near r_Config_ADC_interrupt(void)
{
    /* Start user code for r_Config_ADC_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 */
```

7.2　ユーザーアプリケーションコードの使用方法

RL78 Software Integration System Modules 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。

(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。

RL78 Software Integration System Modules の場合は、r_*xxx*.h です。
コード生成の場合は、自動的に r_smc_entry.h に追加されます。

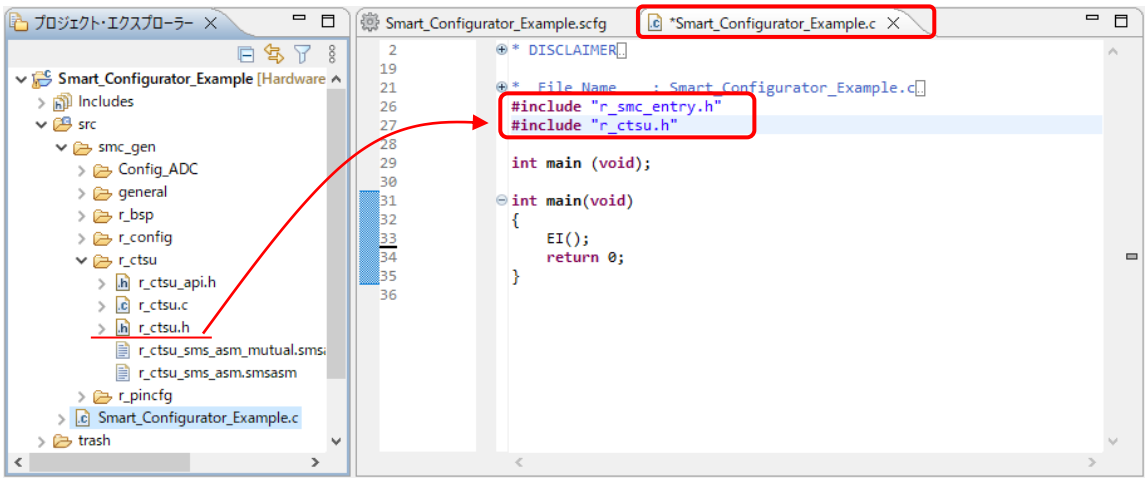

図 7-1 ヘッダファイルの追加

(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_systeminit.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) と停止 (R_*ConfigName*_Stop) です。

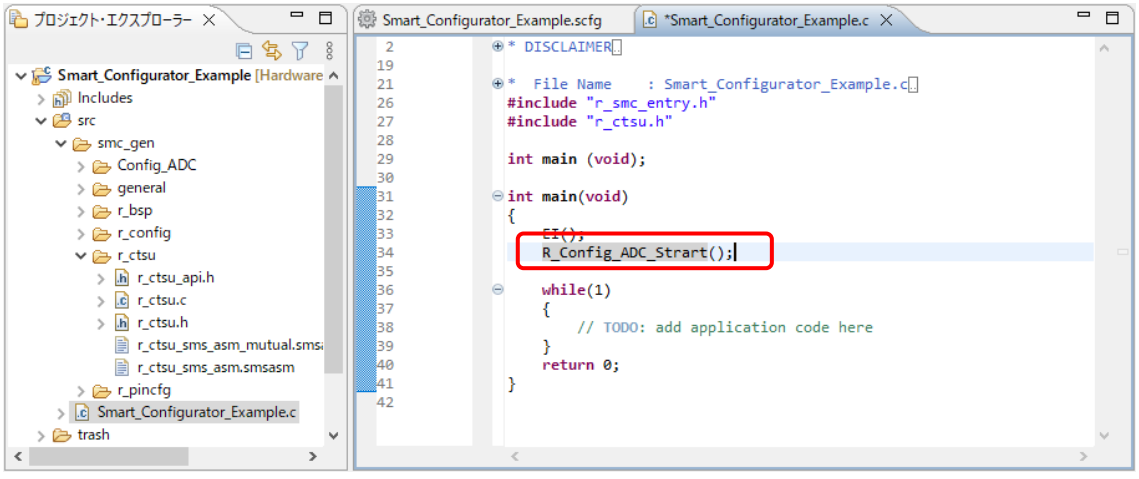

図 7-2 コード生成関数コール

RL78 Software Integration System Modules の場合は、対応するアプリケーションノートの「API 機能」の章に記載されている例を参照してください。

詳細については、「13. 参考ドキュメント」 のスマート・コンフィグレータのアプリケーション例を参照してください。

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、以下の場所にソースコードをバックアップする機能があります。

<ProjectDir>\trash\<Date-and-Time>

［コード生成］ 

ボタンをクリックしてコード生成を行うと、スマート・コンフィグレータはコード生成前のソースのバックアップを作成します。<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。

9.　レポートの生成

スマート・コンフィグレータは、ユーザー設定のレポートを提供します。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（PDF または txt 形式）

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。

図 9-1 全設定内容レポート出力（PDF または txt 形式）

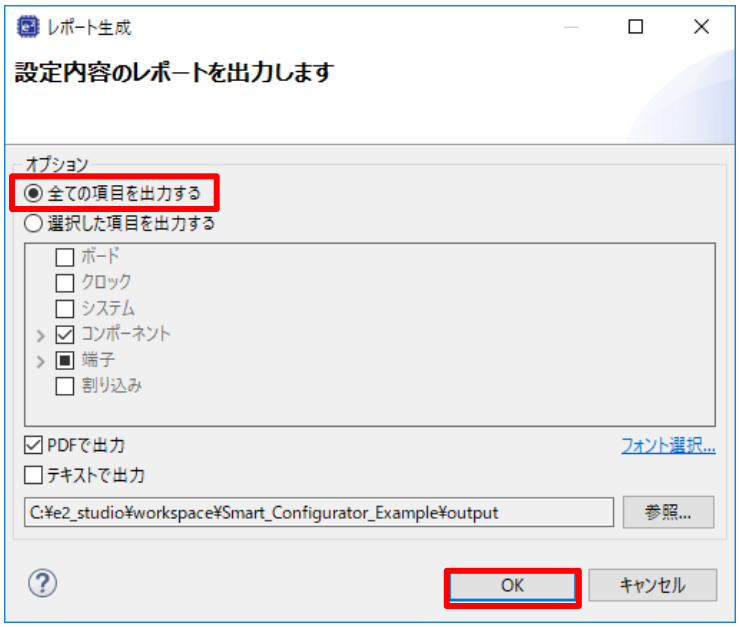

図 9-2 レポート出力ダイアログ

9.2　端子機能リスト、端子番号リストの設定内容

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、表示中の端子リスト（端子機能リストまたは端子番号リスト）の設定内容を出力します。

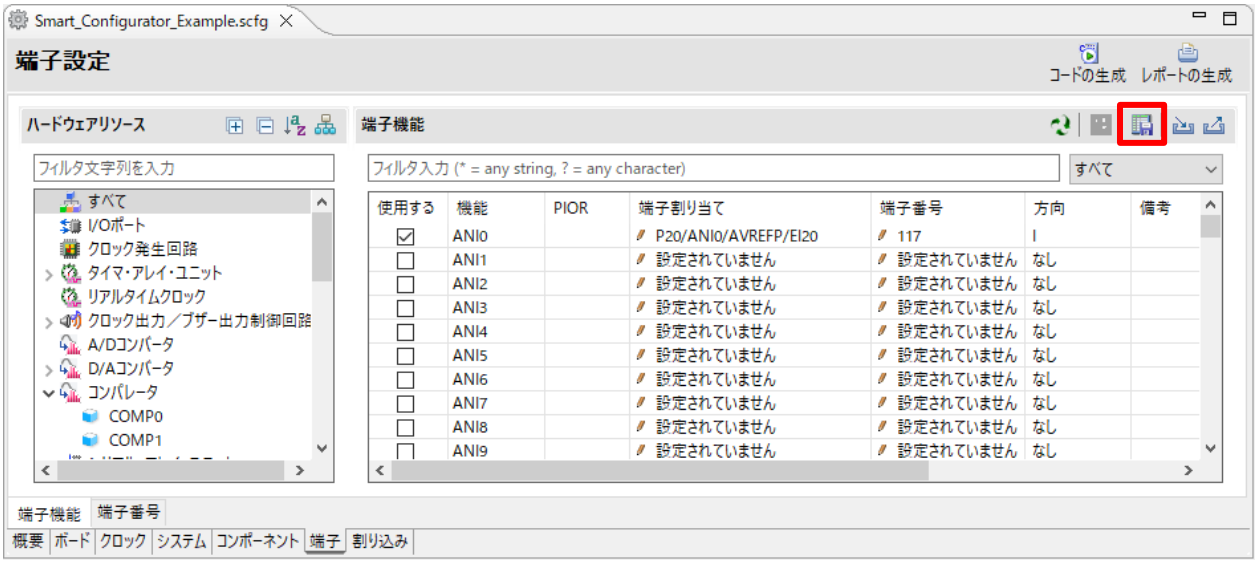

図 9-3 端子機能リスト、端子番号リスト出力（csv 形式）

9.3　MCU/MPU パッケージ図

[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。

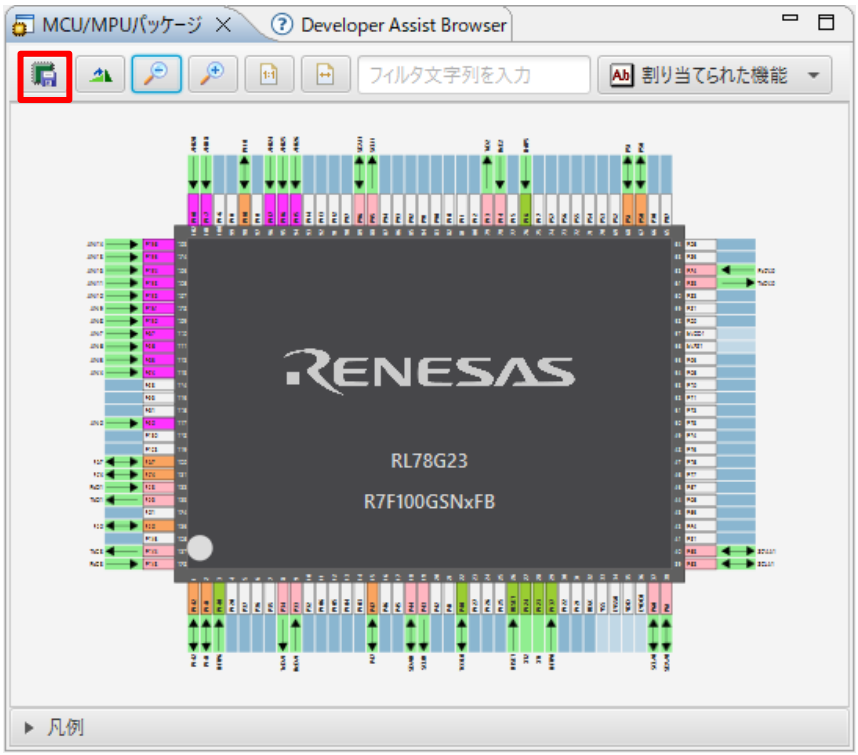

図 9-4 MCU/MPU パッケージ図出力（png 形式）

10. Developer Assistance

Developer Assistance 機能は、プロジェクト作成時にスマート・コンフィグレータを選択すると、プロジェクト・ツリーに [Developer Assistance] という仮想ルート・ノードを提供します。

コード生成された API は [Developer Assistance] ツリーに表示され、API を選択すると [Developer Assist Browser] ビューに API 情報をナビゲートします。コーディング中に API テンプレート・ノードを C/C++エディタにドラッグ&ドロップすることもできます。

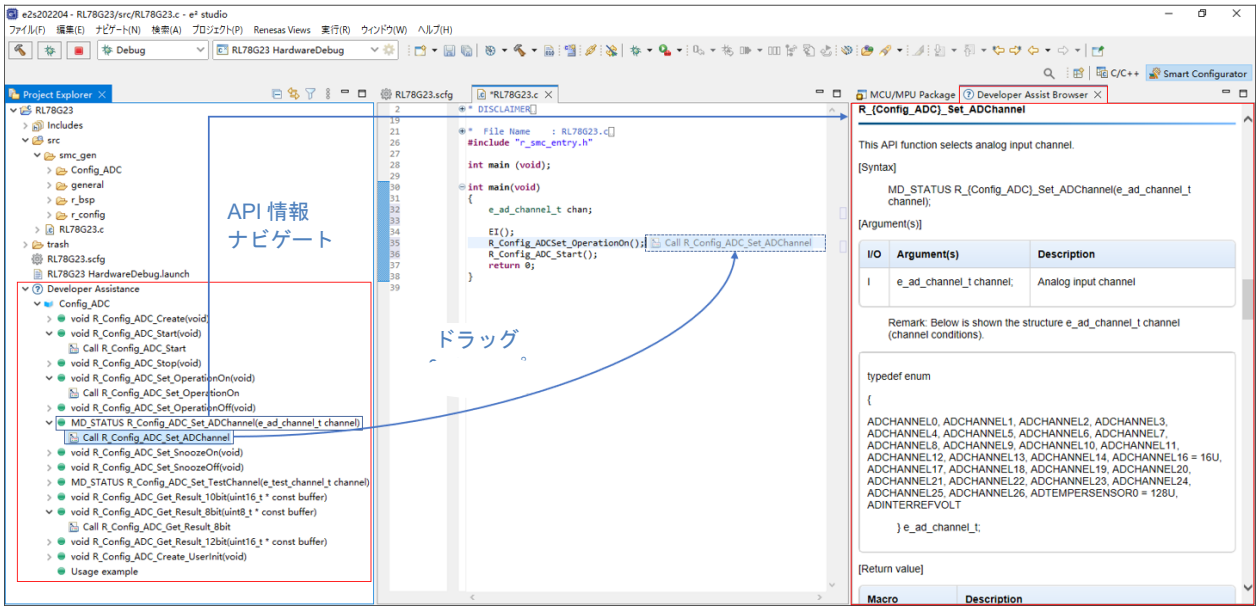

図 10-1 API 情報のナビゲートと API のドラッグ&ドロップ

[Developer Assist Browser] ビューでテキストを選択し、[Copy] コンテキスト・メニューを使用して、コード生成コンポーネント API 使用例のコードスニペットを C/C++エディタに貼り付けることができます。

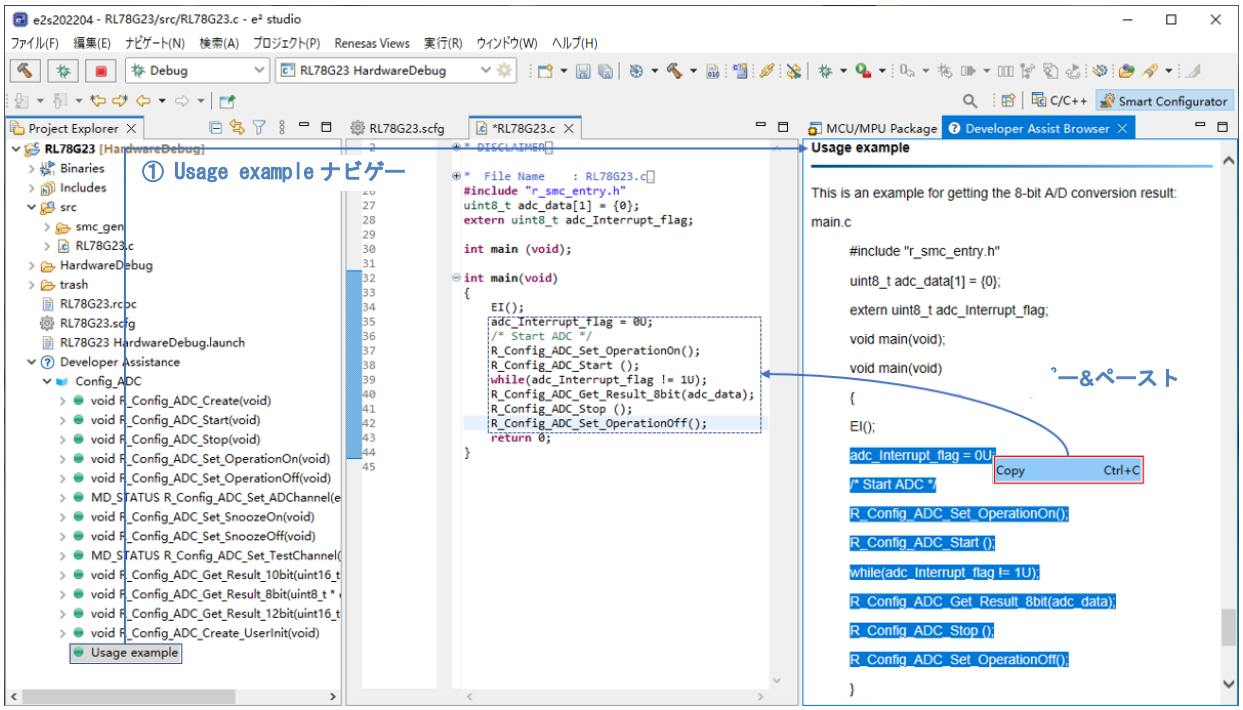

図 10-2 Usage example のコピー&ペースト

11. ユーザーコード保護機能

e² studio 2023-01 Smart Configurator for RL78 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 11-1 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します。

11.1 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 11-1 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 11-1 ユーザーコード保護機能の指定タグ

11.2 ユーザーコード保護機能の使用例

図 11-2 に示すように、図 11-1 の指定タグを使用し、A/D コンバータモジュールの 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、A/D コンバータの GUI 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。

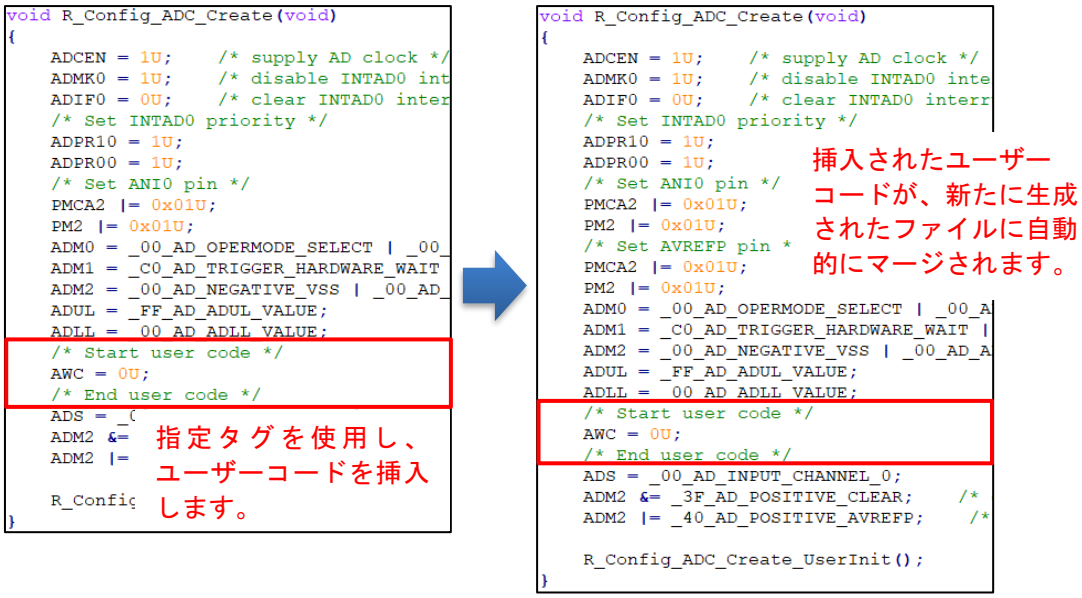

図 11-2 ユーザーコードの保護機能

11.3 競合発生時の対応方法

11.3.1 競合の発生条件

GUI の設定変更やスマート・コンフィグレータのバージョンアップにより、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、生成コードに競合が発生します。

競合が発生した場合、図 11-3 生成コードのメッセージのようにコンソールに競合メッセージが表示されます。

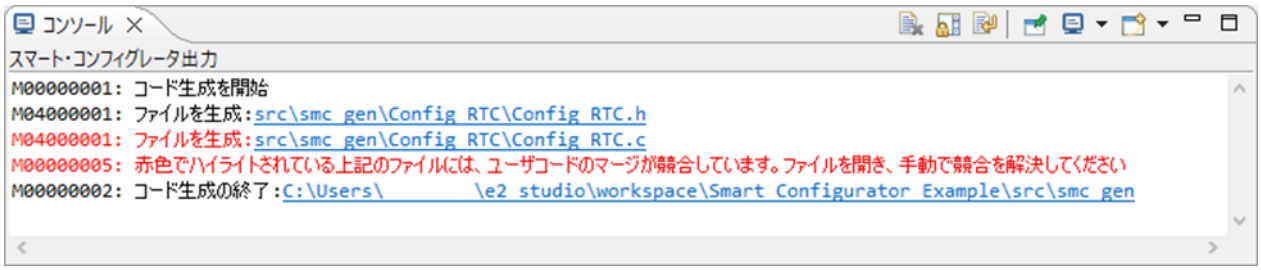

図 11-3 生成コードのメッセージ

ユーザーは、コンソール メッセージに競合ファイルをクリックし、[ファイル比較]ビューを開き、次の章 11.3.2 競合の解決方法ように競合を解決できます。

11.3.2 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) コンソールメッセージの競合ファイルをクリックし、[File Compare] ビューを開きます。(図 11-4)

(2) 図 11-4 のように、矢印 [>] をクリックし左側パネルのコードを右側パネルにコピーします。

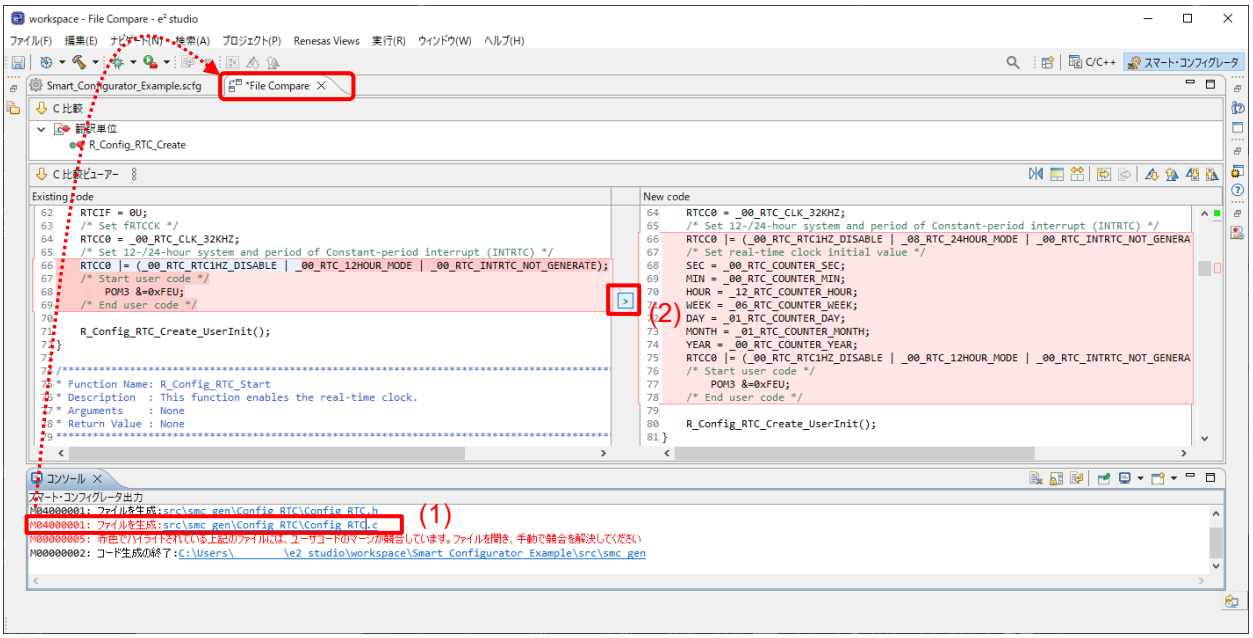

図 11-4 生成コードの競合解決前

(3) 図 11-56 のように、適切な位置にコードを追加し、不要なコードを削除します。

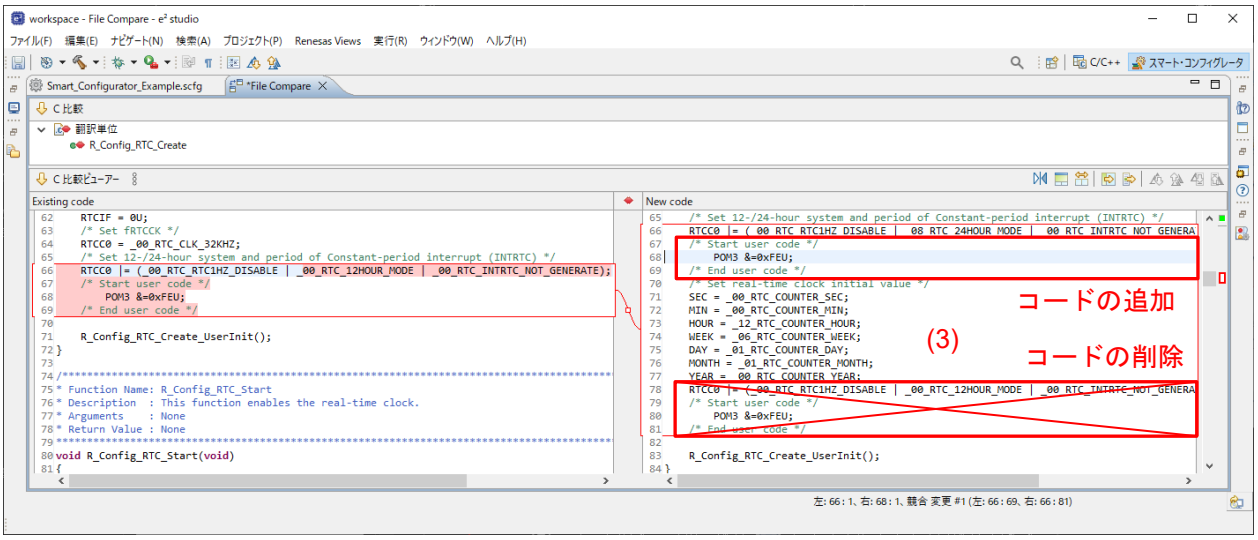

図 11-56 生成コードの競合解決後

(4)　コードのマージ後、保存 アイコンをクリックし保存します。

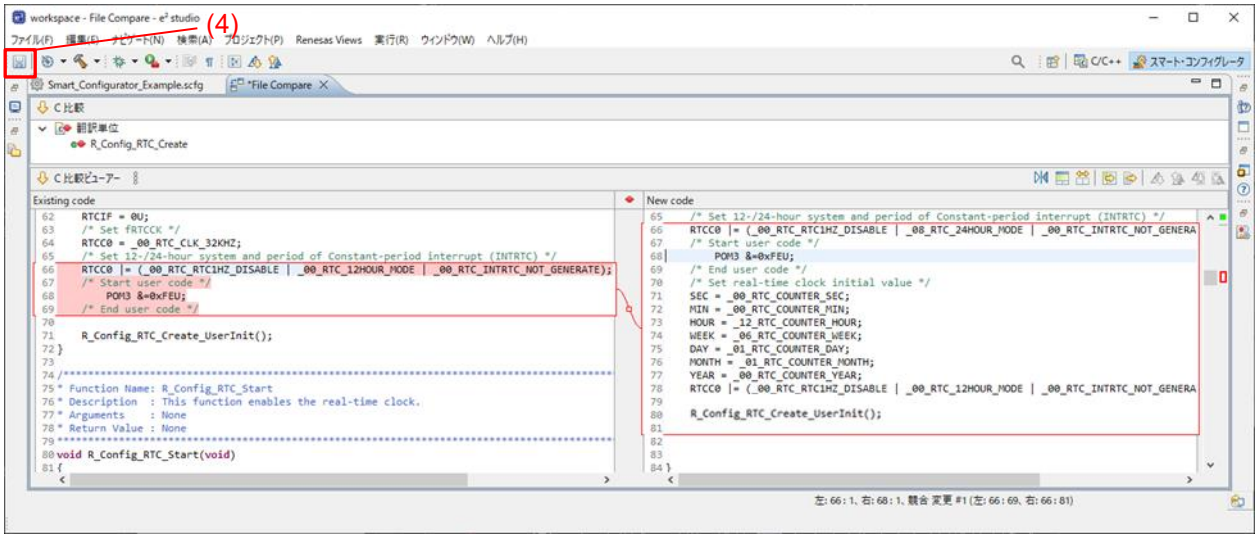

図 11-7 コードの削除と保存

左側パネルのコードを右側パネルにコピーするか、右側パネルのコードを直接編集することで、競合を手動で解決することもできます。

【注】 競合が解決された後も、競合メッセージをクリックすると、[ファイル比較]ビューを開くことができます。

12. ヘルプ

スマート・コンフィグレータの詳細については、e² studio メニューのヘルプを参照してください。 ヘルプをメニューから選択すると、ヘルプの目次が表示されます。

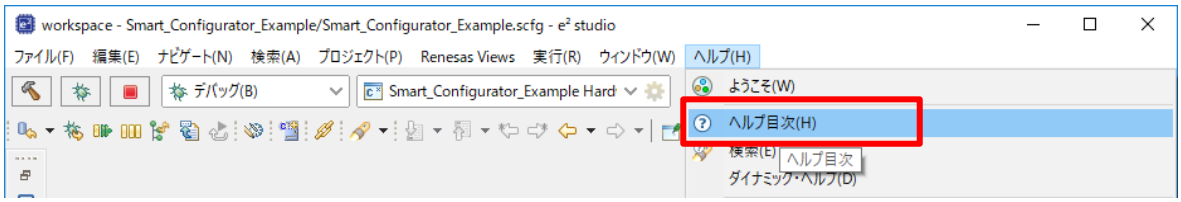

図 12-1 ヘルプ表示

ヘルプは 、[概要] ページの アイコンからも参照できます。

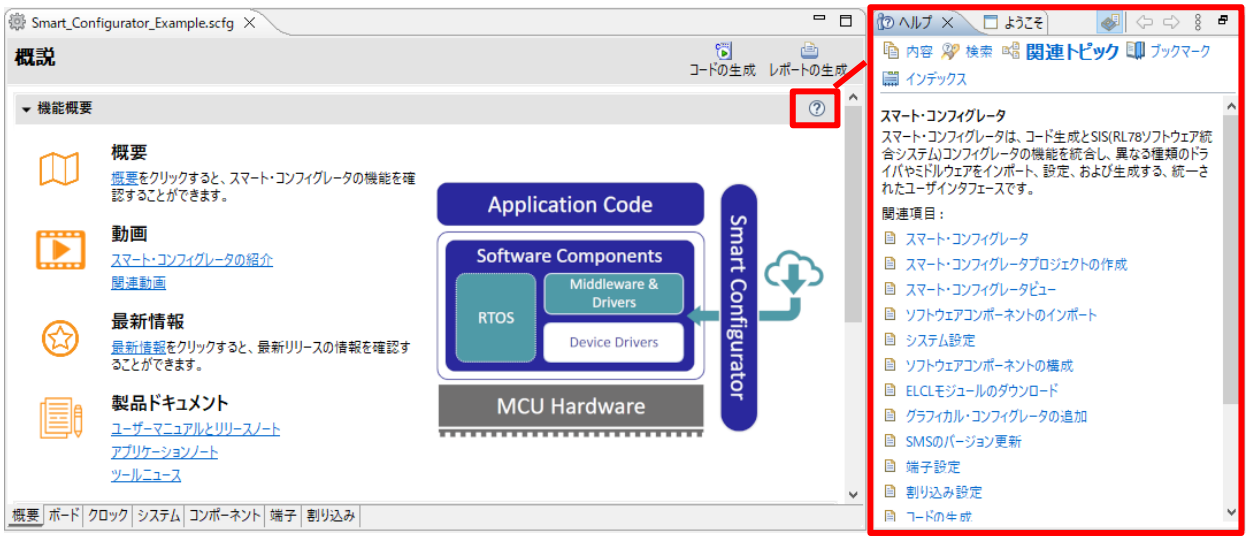

図 12-2 クイックスタート

どちらの方法でも同じヘルプを参照できます。

13. 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-RL コンパイラ ユーザーズマニュアル (R20UT3123)

スマート・コンフィグレータ ユーザーズマニュアル RL78 API リファレンス (R20UT4852)

（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：SMS & ELCL】

(最新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。

改訂記録

Rev.	セクション	改訂内容
1.00	-	新規作成
1.01	第 2 章 プロジェクトの作成	2.1 プロジェクトの作成：図 2-5,2-7 の変更
	第 3 章 スマート・コンフィグレータ レータの操作方法	3.4 ウィンドウ：図 3-3,3-4,3-5 の変更
		3.4.6 Developer Assist Browser：3.4.6 章の追加
	第 10 章 Developer Assistance	10.Developer Assistance：10 章の追加
1.02	要旨	対象デバイスの URL 更新
	第 4 章 周辺機能の設定	4.1.1 デバイス選択：図 4-2,表 4-1 及び表 4-1 の注を削除
		4.1.2 ボード選択：図 4-4,表 4-2 及び表 4-2 の注を削除
		4.4.12 BSP コンフィグレーションのバージョン変更の注を削除
		4.4.13　コンポーネントの基本設定：図 13-1 コンポーネントの基本設定の変更
		4.4.13　コンポーネントの基本設定：注 1,2 の変更
		4.4.13　コンポーネントの基本設定：注 3 の追加
		4.6.2　割り込みバンクの設定：手順(3)の変更、図 4-57　割り込みバンク設定の変更
	第 12 章 参考ドキュメント	SMS & ELCL アプリケーションノート：参考先を削除
1.03	第 2 章 プロジェクトの作成	e² studio プロジェクト作成ウィザード 更新
	第 3 章 スマート・コンフィグレータの操作	3.4.3 MCU/MPC パッケージビュー 更新
	第 4 章 周辺機能の設定	4.1.2 ボード選択 更新
		4.4.3 ソフトウェア・コンポーネントの削除 複数コンポーネントの指定方法を追加
		4.4.10 RL78 Software Integration System モジュールのダウンロード 更新
		4.4.11 RL78 Software Integration System モジュールの追加 追加
		4.4.12 RL78 Software Integration System モジュールの設定 更新
		4.5 端子設定 図 4-50, 図 4-51 更新
		4.5.3 MCU/MPU パッケージビューを使用した端子の設定 更新
		4.5.4 端子機能から端子番号の表示 追加
		4.5.9 端子エラー/端子警告の設定 追加
		4.7 MCU マイグレーション機能 更新
	第 6 章 ソースの生成	6.2 コード生成場所の変更 追加
		6.3 生成ファイルの構成とファイル名 更新
	第 7 章 ユーザープログラムの作成	7.2 ユーザーアプリケーションコードの使用方法 追加
	第 11 章 ユーザーコード保護機能	11. ユーザーコード保護機能 追加
1.04	第 4 章 周辺機能の設定	4.4.8 ELCL 固定機能モジュールのダウンロード 更新
		4.4.9 固定機能 ELCL コンポーネントの設定 更新
		4.4.10 ELCL Flexible Circuit の作成と編集 追加
	第 6 章 ソースの生成	6.3 生成ファイルの構成とファイル名
1.05	第 4 章 周辺機能の設定	4.4.10 ELCL Flexible Circuit の作成と編集 更新

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. 静電気対策
   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。
2. 電源投入時の処置
   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. 電源オフ時における入力信号
   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。
4. 未使用端子の処理
   未使用端子は、「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。
5. クロックについて
   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。
6. 入力端子の印加波形
   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V_{IL}（Max.）から V_{IH}（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}（Max.）から V_{IH}（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。
7. リザーブアドレス（予約領域）のアクセス禁止
   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、将来の拡張機能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。
8. 製品間の相違について
   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、フラッシュメモリ、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、ノイズ幅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。