RL78 ソフトウェア置き換えガイド

RL78/G13 サンプルコードの移植 CC-RL (CS+, e2 studio)

要旨

本アプリケーションノートでは、RL78/G13 の周辺機能用サンプルコードを他の RL78 製品用に置き換える方法について説明します。

対象デバイス

RL78 ファミリ

本アプリケーションノートを他のマイコンへ適用する場合、そのマイコンの仕様にあわせて変更し、十分評価してください。

目次

1. 概要

1.1 移植対象

RL78/G13 の周辺機能用サンプルコードは、通常、コード生成ツール（CG）が生成するプログラム（デバイス・ドライバ）を使用しています。本アプリケーションノートでは、CG が生成したデバイス・ドライバを他の RL78 製品用に置き換える手順について説明します。

ただし、移植先の RL78 製品が同じ周辺機能を搭載していない場合は、デバイス・ドライバの置き換えができません。置き換えができない場合は、移植先の RL78 製品用プログラムを新規に開発してください。

表 1-1 に移植先対象の RL78 製品を示します。

表 1-1　移植先対象の RL78 製品

RL78/G1x	RL78/G11, RL78/G12, RL78/G14, RL78/G13A, RL78/G1A, RL78/G1C, RL78/G1E, RL78/G1F, RL78/G1G, RL78/G1H
RL78/H1x	RL78/H1D
RL78/I1x	RL78/I1A, RL78/I1B, RL78/I1C, RL78/I1D, RL78/I1E
RL78/L1x	RL78/L12, RL78/L13, RL78L1A, RL78/L1C

表 1-2　移植可否の目安

移植先の RL78 製品	移植の可／不可
移植元と同じ周辺機能が搭載されていない。	不可
移植元と同じ周辺機能があり、 移植元と同じチャネルがある。	可
移植元と同じ周辺機能があるが、 移植元と同じチャネルがない。	チャネル変更できれば可
移植元と動作電圧が異なる。	動作電圧に応じた設定に変更できれば可[注]
移植元と動作周波数が異なる。	動作周波数に応じた設定に変更できれば可[注]

注. 移植先の RL78 製品の電気的特性を満たすように回路設計してください。

1.2　サンプルコードの構成

RL78 製品のサンプルコードは、コード生成ツール（CG）が生成するファイル（CG 生成ファイル）とそれ以外のファイル（非 CG 生成ファイル）で構成されています。CG 生成ファイル内には、プロジェクトの対象 RL78 製品に合わせた周辺機能の設定コードがあります。

図 1-1 にサンプルコードのプロジェクト例を示し、表 1-3 に CG 生成ファイルおよび関数の概要を示します。図 1-2 にデバイスの CPU リセット発生から main 関数までの処理イメージを示します。

図 1-1　サンプルコードのプロジェクト例

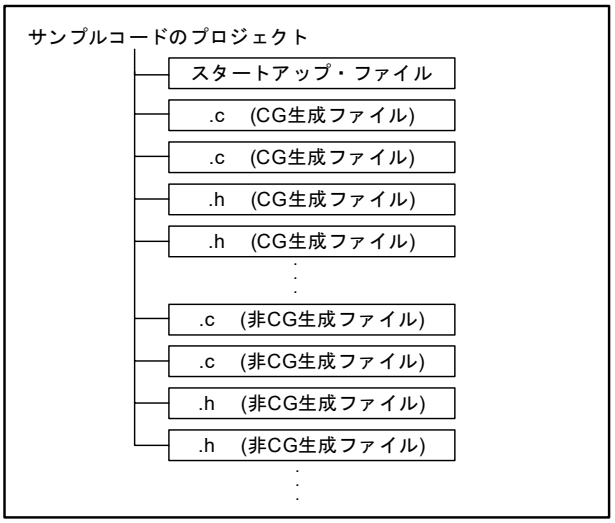


表 1-3　CG 生成ファイルおよび関数の概要

ファイル名 注 1	関数名 注 1	内容
r_main.c	main R_MAIN_Userinit	main 関数 main 関数実行前の処理
r_systeminit.c	hdwinit R_Systeminit	R_Systeminit の呼び出し 各周辺機能の初期化設定の呼び出し
r_cg_macrodriver.h	-	CG 生成ファイル内で共通で使用するマクロ定義および Typedef 定義
r_cg_userdefine.h	-	ユーザ定義のマクロ、Typedef 等
r_cg_xxx.c 注 2	R_XXX_Create R_XXX_Start 注 3 R_XXX_Stop 注 3	周辺機能の初期化設定 周辺機能の動作開始 周辺機能の動作停止
r_cg_xxx_user.c	r_xxx_interrupt	周辺機能の割り込み関数
r_cg_xxx.h	-	周辺機能用マクロ定義および Typedef 定義

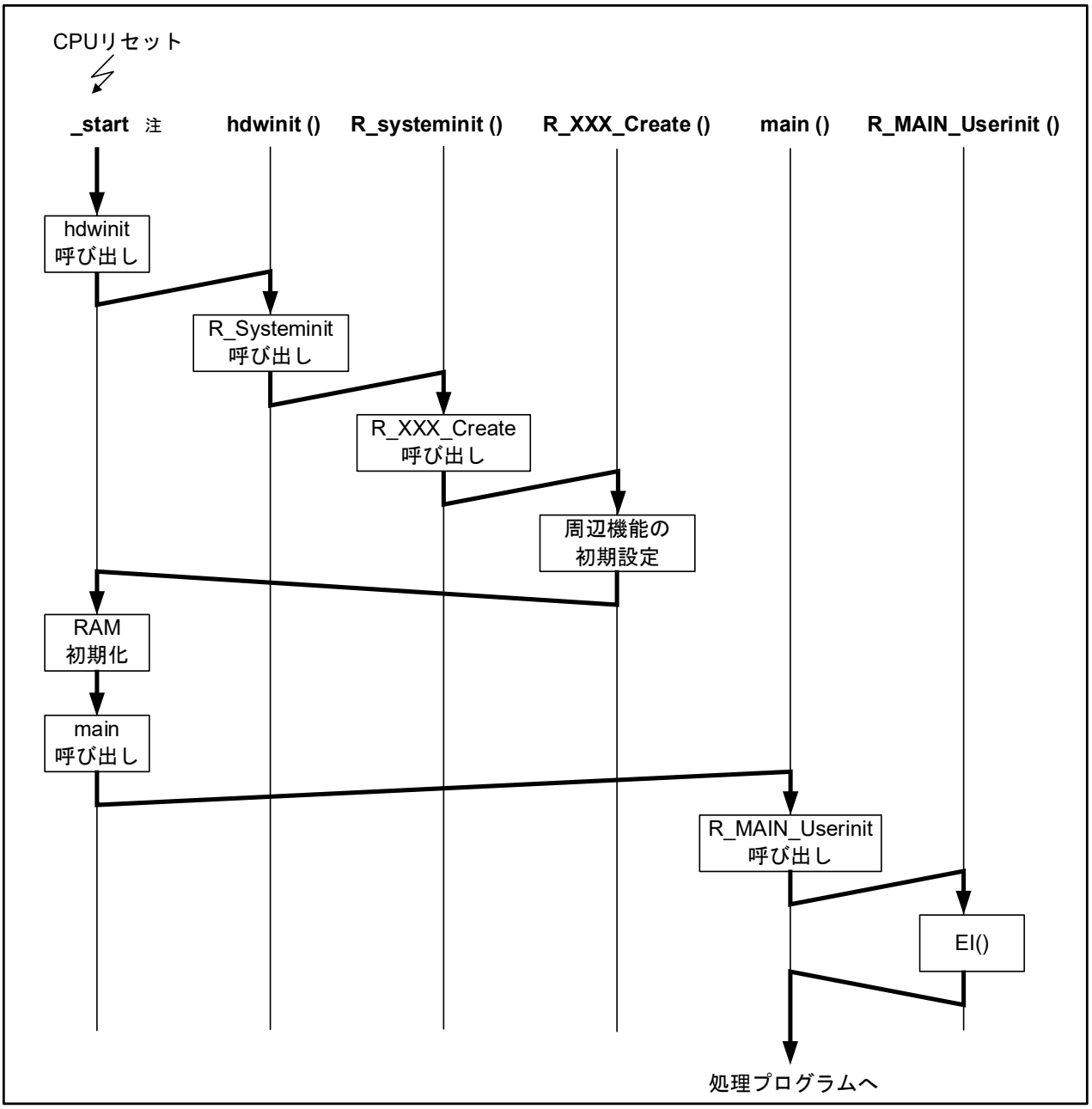
注 1. xxx, XXX には、周辺機能名を表す省略文字が入ります。

注 2. 周辺機能により、「関数名」欄に記載されていない関数が生成される場合があります。

注 3. 周辺機能により、生成される関数名の最後が Start、Stop でない場合があります。

注意. サンプルコードにより、表 1-3 に記載の CG 生成ファイルおよび関数を使用していない場合があります。

図 1-2　CPU リセット発生から main 関数までの処理イメージ



注.　cstart.asm 内のスタートアップ処理

cstrat.asm は、CS+ for CC または e^2 studio で C プロジェクトを作成時に自動で生成されます。

1.3　移植方法

サンプルコードを置き換える場合、移植先 RL78 製品用サンプルコードのプロジェクトに対してデバイス変更を実施し、移植先 RL78 製品に応じたリソース変更を実施します。

サンプルコードによっては、CG 生成コードを部分的にコメントアウトする等の変更しているものがあります。移植先のプロジェクトをリビルドする時に同じ変更を加える必要があるため、移植元の RL78 製品用プロジェクトを新規に作成し、CG 生成コードが変更されているかを確認してください。また、移植先の RL78 製品では、移植元のプロジェクトと同じリソースを使用できない場合があります。その場合は、リソース変更に影響を受けるコードを変更する必要があります。

図 1-3 に移植のイメージを示し、図 1-4 に移植フローの概要を示します。

図 1-3　移植のイメージ

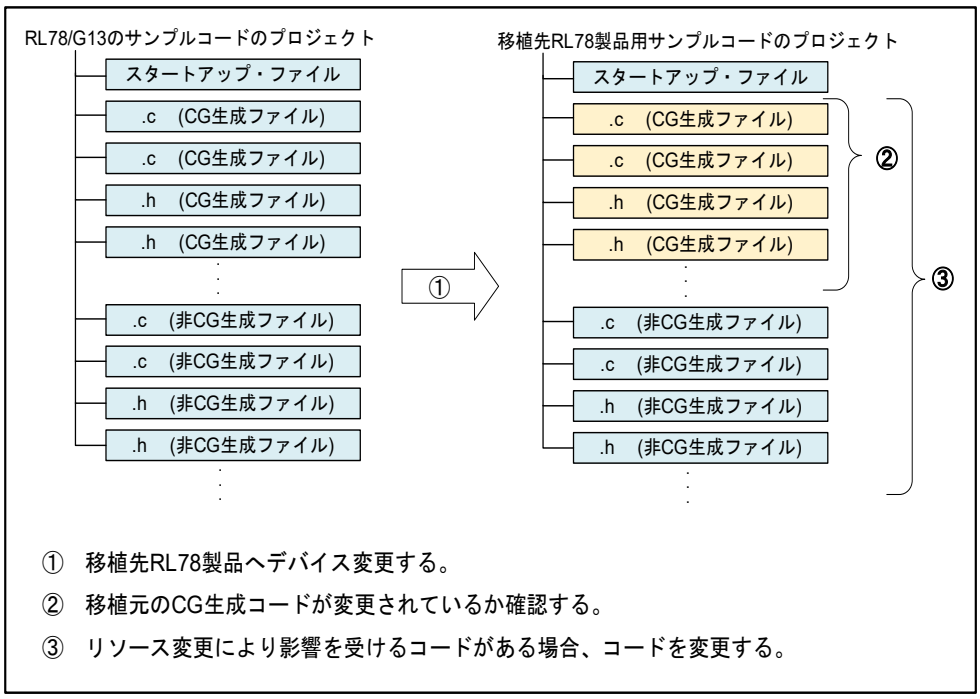
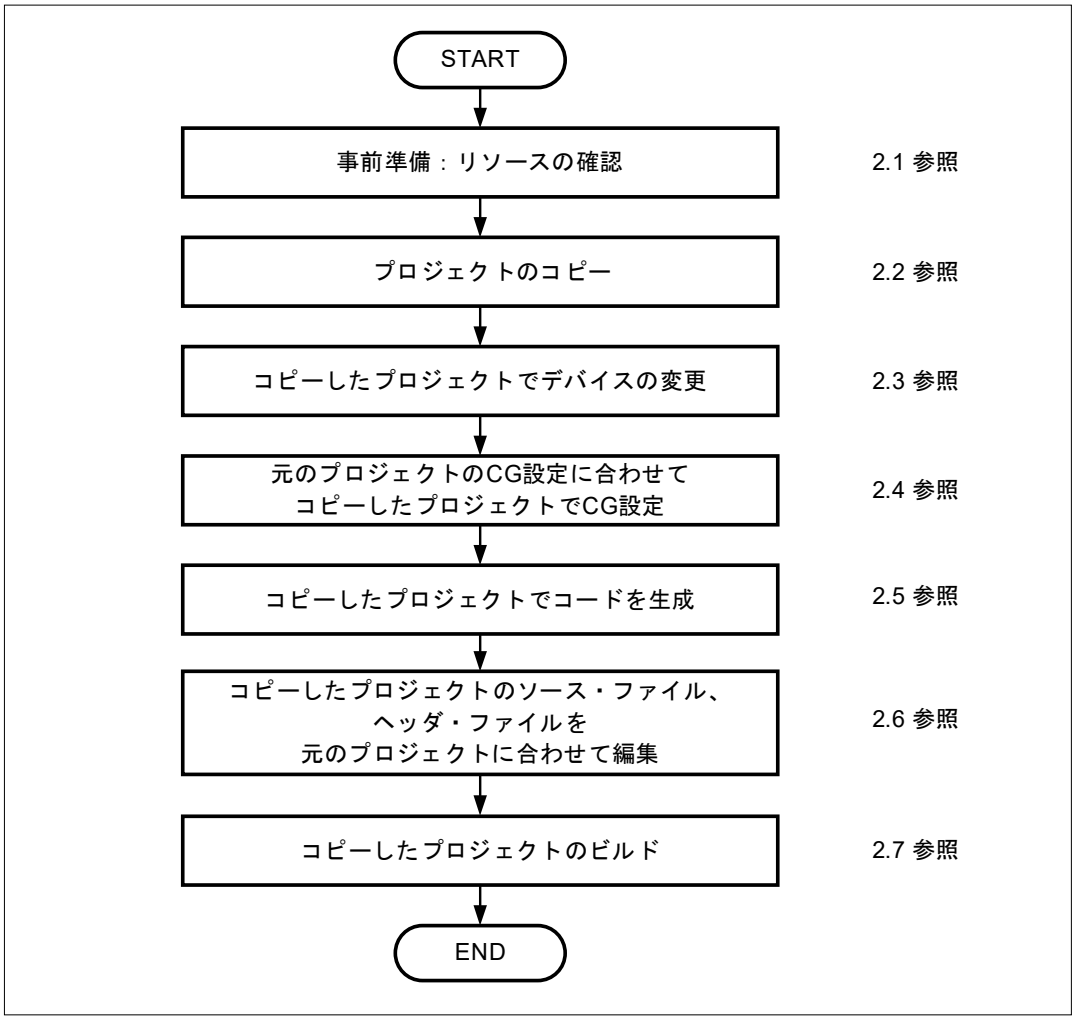


図 1-4　移植フローの概要



備考．コピーしたプロジェクト = 移植プロジェクト

2.　移植手順

図 1-4 に示した移植フローの各手順について説明します。

2.1　事前準備

サンプルコードで使用しているポート端子と周辺機能をサンプルコードに添付されているドキュメントで確認します。次に、移植先 RL78 製品のユーザーズマニュアルを利用して移植方法を検討します。

表 2-1 に周辺機能の確認内容を示し、表 2-2 に使用端子の確認内容を示します。

表 2-1　周辺機能の確認内容

移植先の RL78 製品	移植方法
移植元と同じチャネルがある。	同じチャネルを使用する。
移植元と同じチャネルがない。	別のチャネルに変更する。

表 2-2　使用端子の確認内容

移植先の RL78 製品	移植方法
移植元と同じポート端子があり、 同じ兼用機能が割り付けられている。	同じ端子を使用する。
移植元と同じポート端子があるが、 同じ兼用機能が割り付けられていない。	兼用機能を使用せず、ポート機能を使用する場合： 　同じポート端子を利用する。 兼用機能を使用する場合： 　同じ兼用機能がある端子に変更する。
移植元と同じポート端子がない。	兼用機能を使用せず、ポート機能を使用する場合： 　別のポート端子に変更する。 兼用機能を使用する場合： 　同じ兼用機能がある端子に変更する。

2.2　プロジェクトのコピー

サンプルコードのプロジェクトを任意のフォルダにコピーします。

以降、元のサンプルコードのプロジェクトを「元プロジェクト」、コピーしたプロジェクトを「移植プロジェクト」とします。

2.3　移植プロジェクトのデバイス変更

移植プロジェクトで、対象デバイスを変更します。

CS+ for CC の移植プロジェクトを使用する場合は 2.3.1 を参照し、e^2 studio の移植プロジェクトを使用する場合は 2.3.2 を参照してください。

2.3.1　CS+ for CC の移植プロジェクト

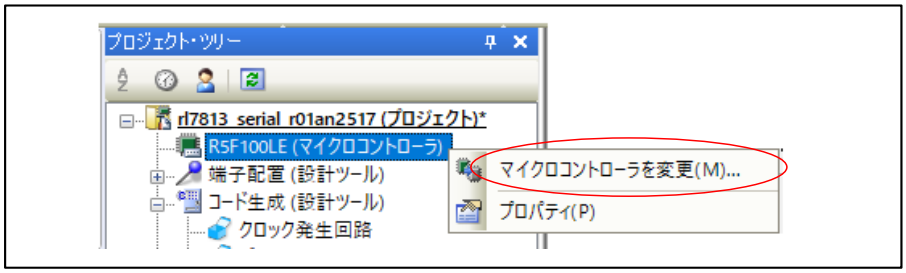
[手順]

(1)　CS+ for CC を起動し、移植プロジェクトの mtpj ファイルを読み込みます。

備考. サンプルコード作成時のツールと現在使用しているツールのバージョンが異なる場合、バージョンが異なることを知らせる警告メッセージが出ます。

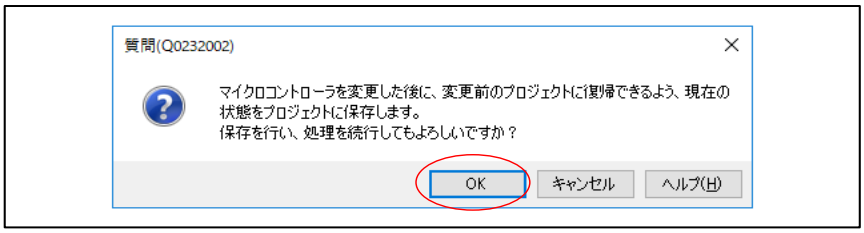
(2)　CS+ for CC のプロジェクト・ツリーで、マイクロコントローラ名を右クリックし、「マイクロコントローラを変更」をクリックします。

図 2-1　「マイクロコントローラを変更」メニュー



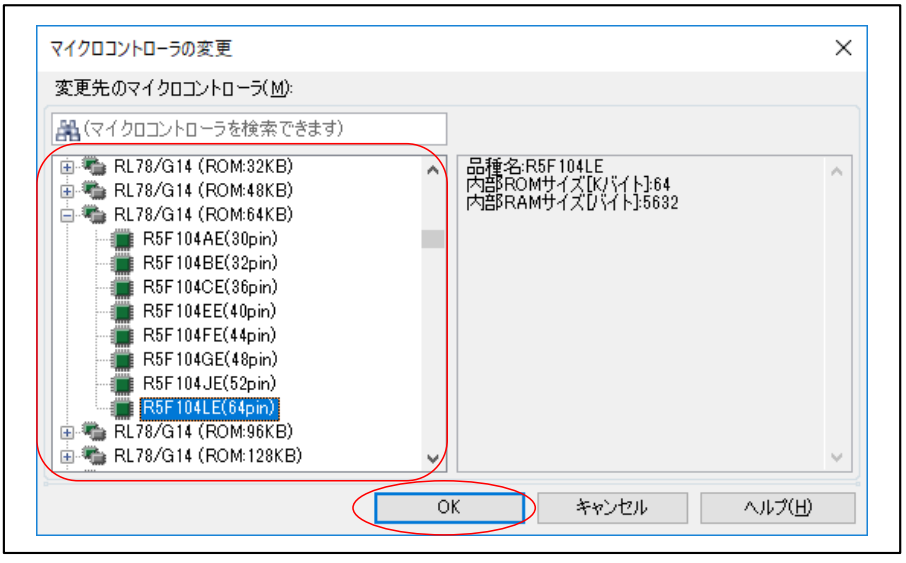
(3)　質問ダイアログで、「OK」をクリックします。

図 2-2　質問ダイアログ



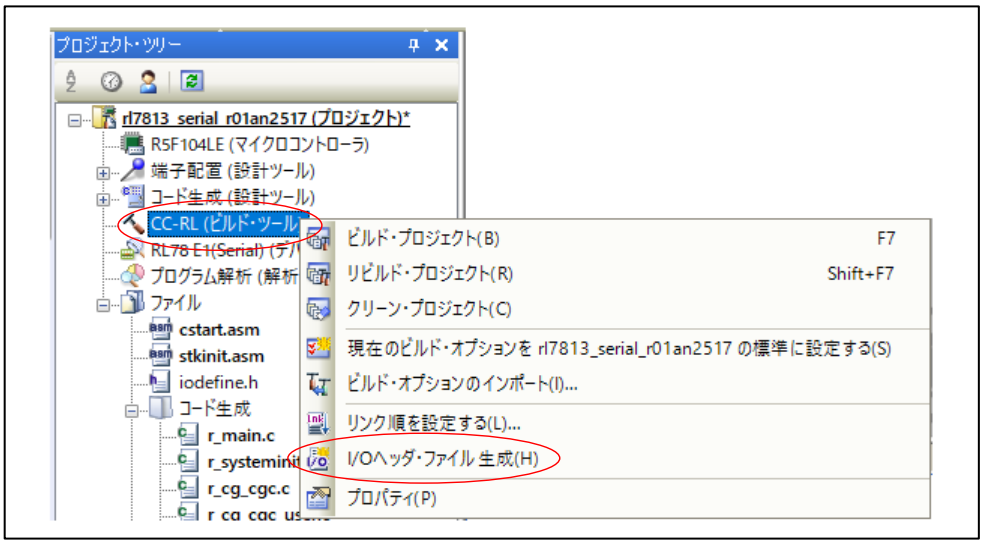
(4)　マイクロコントローラの変更ダイアログで、移植先のデバイスを選択し、「OK」をクリックします。

図 2-3　マイクロコントローラの変更



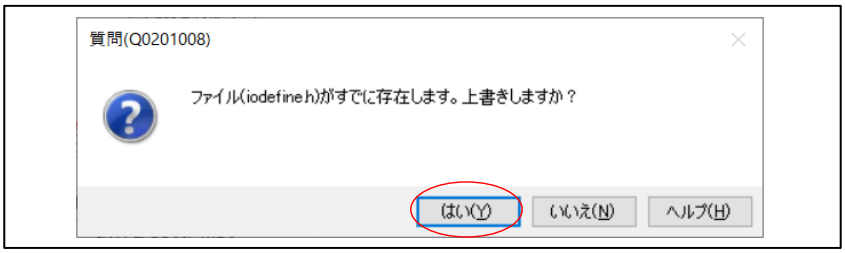
(5) 変更後のデバイス用の I/O ヘッダ・ファイルを生成します。「CC-RL（ビルド・ツール）」を右クリックし「I/O ヘッダ・ファイル生成」をクリックします。

図 2-4　I/O ヘッダ・ファイル生成メニュー



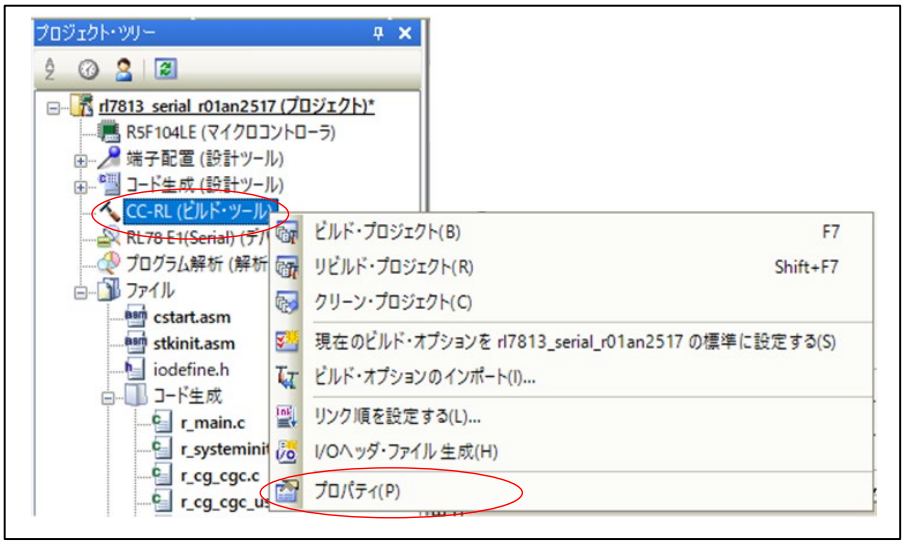
(6) 質問ダイアログで、「はい」をクリックします。

図 2-5　質問ダイアログ



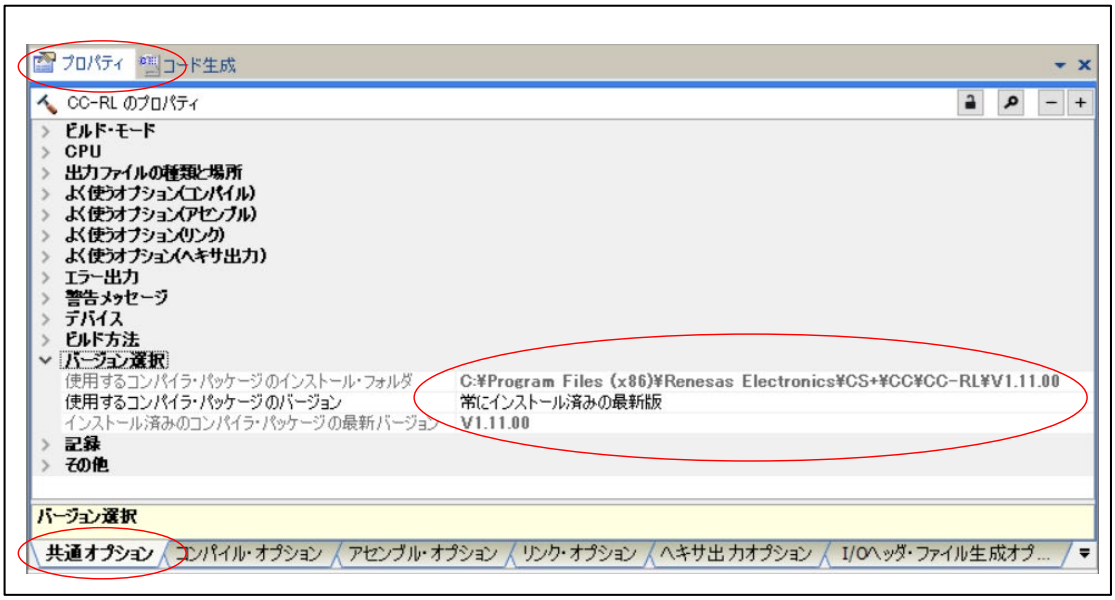
(7) CC-RL コンパイラのバージョンを指定します。「CC-RL（ビルド・ツール）」を右クリックし「プロパティ」をクリックします。

図 2-6　CC-RL プロパティメニュー



(8)　現在使用しているインストール済の CC-RL コンパイラのバージョンを指定します。

図 2-7　CC-RL プロパティ－バージョン選択

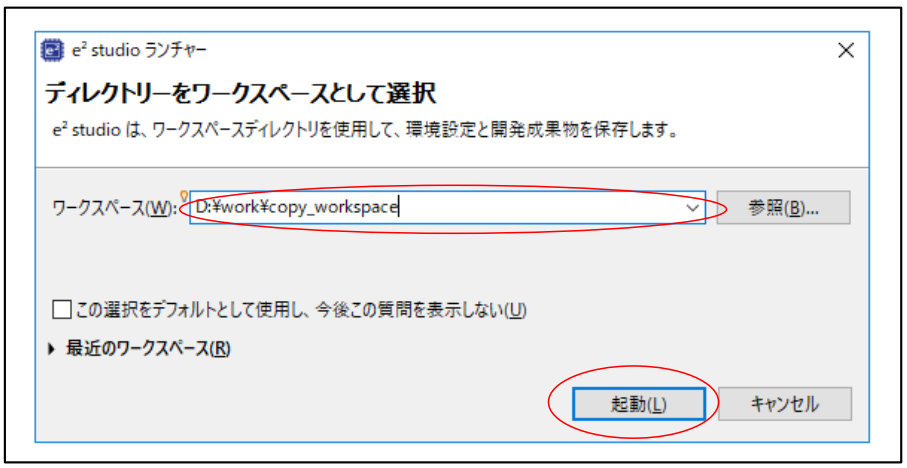


2.3.2　e² studio の移植プロジェクト

[手順]

(1)　e² studio を起動し、e² studio ランチャーのワークスペースで任意のフォルダを指定し、「起動」をクリックします。

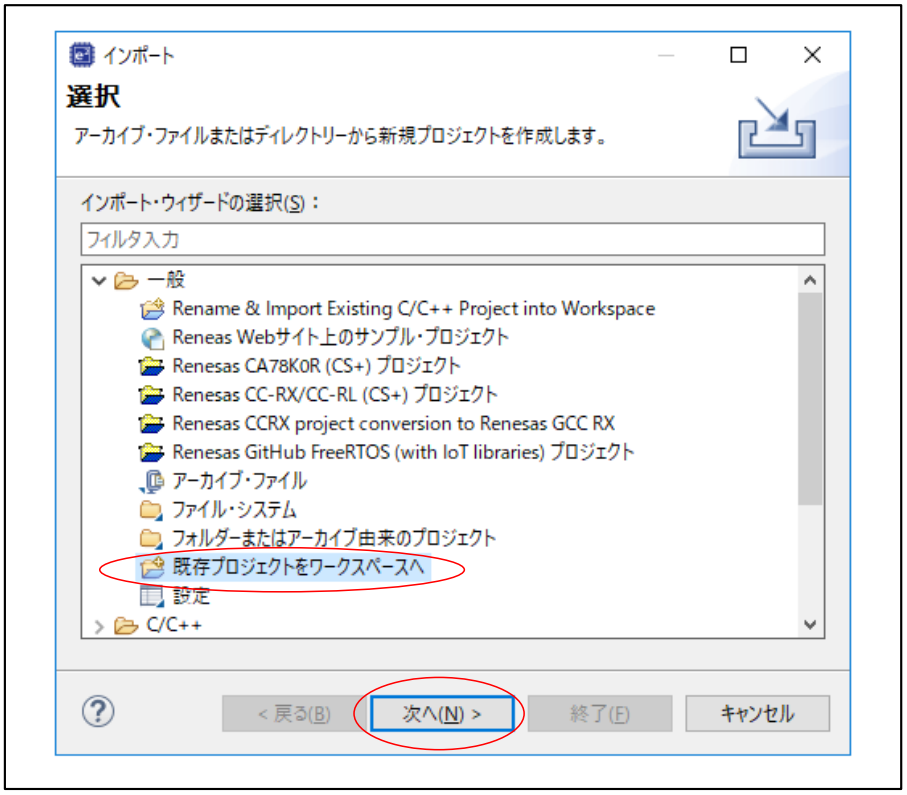
図 2-8　e² studio ランチャー



(2)　e² studio の「ファイル」－「インポート」メニューを選択します。

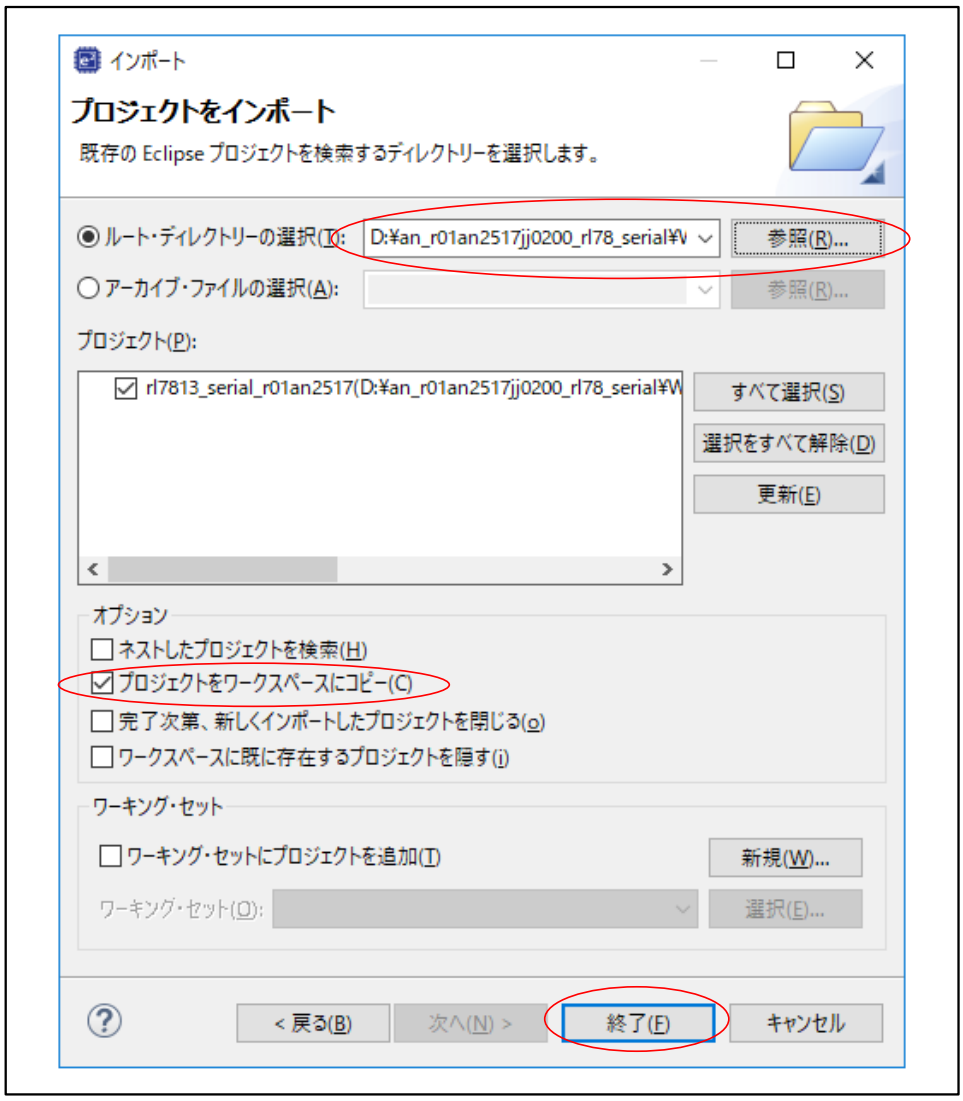
(3)　インポートダイアログで、「一般」－「既存プロジェクトをワークスペースへ」を選択し、「次へ」をクリックします。

図 2-9　インポート(1/2)



(4) インポートダイアログで、「ルート・ディレクトリーの選択」に移植プロジェクトが存在するフォルダを指定後、オプションの「プロジェクトをワークスペースにコピー」にチェックをし、「終了」をクリックします。

図 2-10　インポート(2/2)



(5) 移植プロジェクトをインポート後、e2 studio の画面右下に図 2-11 のメッセージが表示される、または、e2 studio の問題ビューに 図 2-12 のエラーが表示される場合は、(6)の処理を実行してください。メッセージや問題ビューが表示されない場合は、(7)へ進んでください。

図 2-11　プロジェクト更新を促すメッセージ

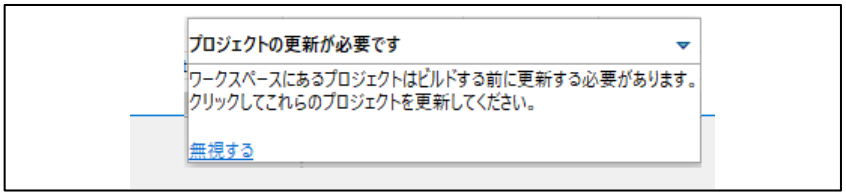
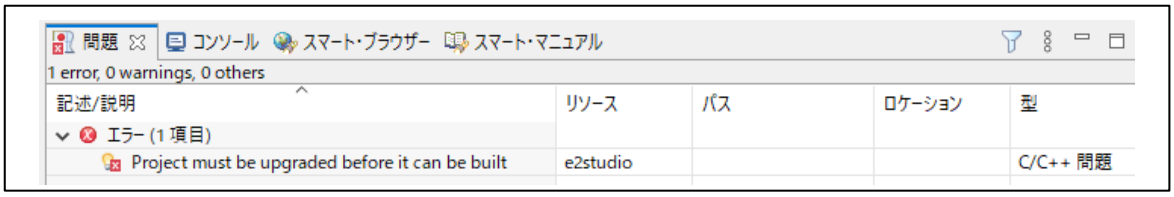


図 2-12　問題ビュー



(6) サンプルコード作成時と現在使用している e2 studio のバージョンが異なる場合、現在使用しているバージョンに合わせるため移植プロジェクトを更新する必要があります。プロジェクト更新を促すメッセージをクリックします。（図 2-13）
続いて表示される古い e2 studio プロジェクトの更新ダイアログで、プロジェクトにチェックをし、「終了」をクリックします。（図 2-14）

図 2-13　プロジェクト更新を促すメッセージのクリック

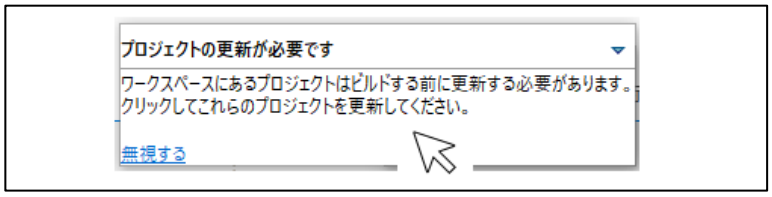
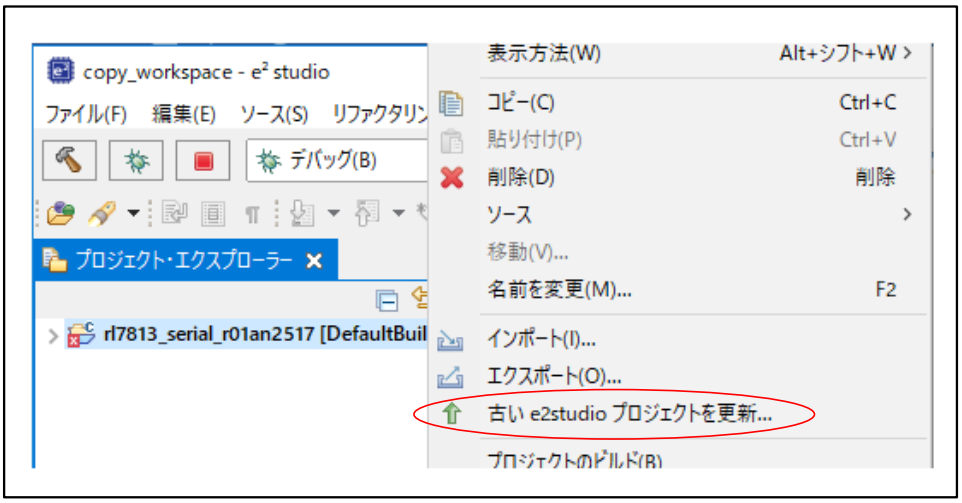


図 2-14　古い e2 studio プロジェクトの更新

備考. 移植プロジェクトを更新せずに、プロジェクトの更新を促すメッセージを閉じてしまった場合には、次の手順でプロジェクトを更新してください。

e2 studio のプロジェクト・エクスプローラでプロジェクト名を右クリックし、「古い e2studio プロジェクトを更新」をクリックします。(図 2-15)

図 2-15　古い e2 studio プロジェクトを更新メニュー

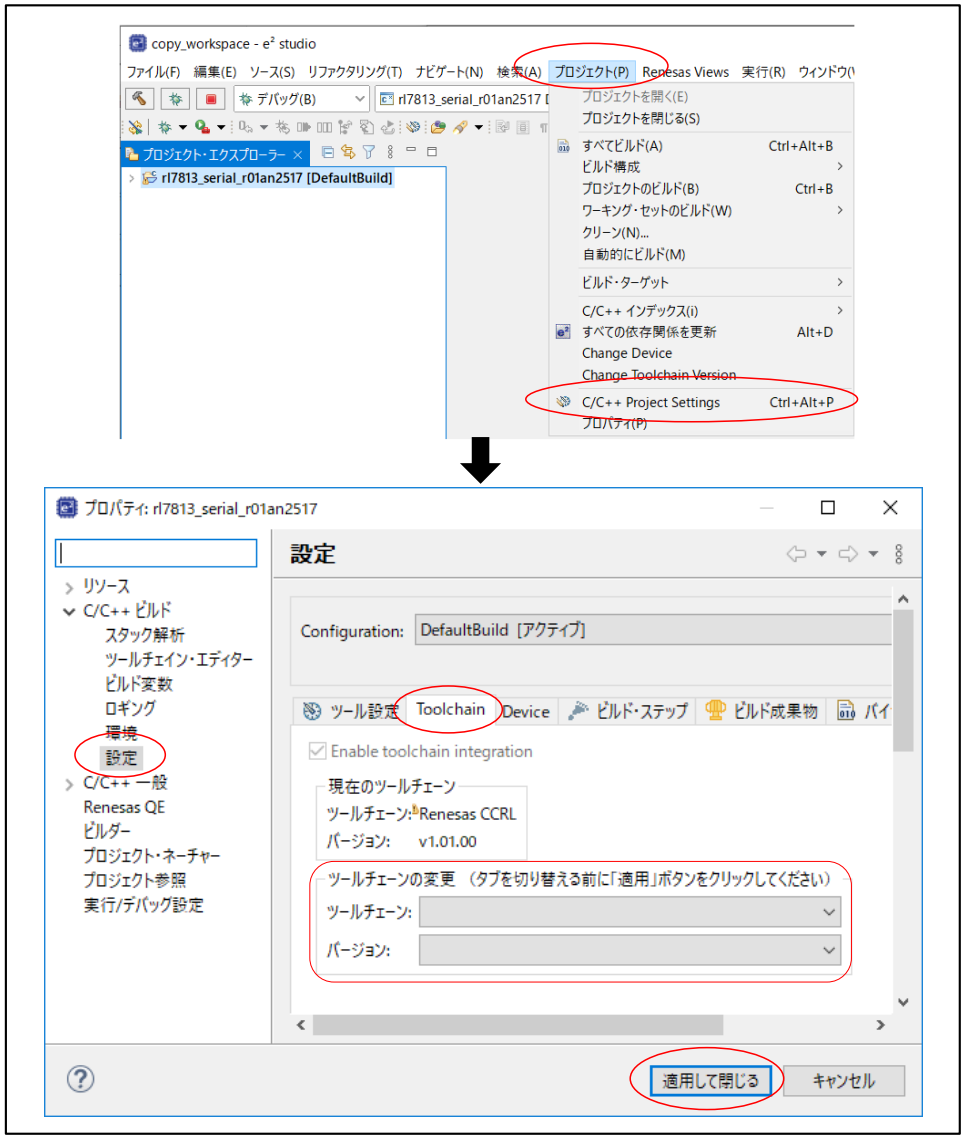


続いて表示される古い e² studio プロジェクトの更新ダイアログで、プロジェクトにチェックをし、「終了」をクリックします。

図 2-16　古い e2 studio プロジェクトの更新

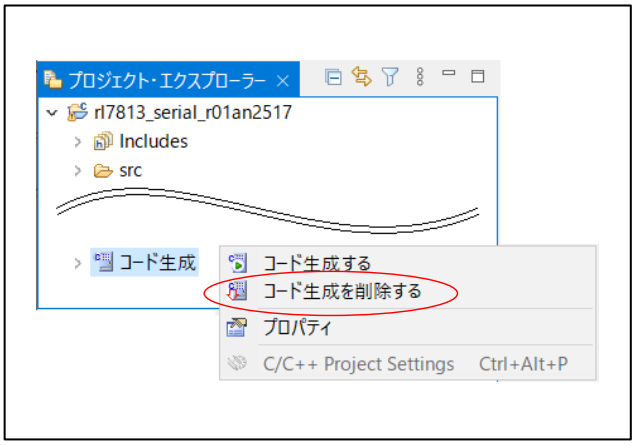
(7) 移植プロジェクトをインポート後、サンプルコード作成時と現在使用している CC-RL コンパイラのバージョンが異なる場合、現在使用しているバージョンに合わせるためツールチェーン設定を更新する必要があります。e² studio のプロジェクト・エクスプローラでプロジェクト名をクリックで選択し、「プロジェクト」－「C/C++ Project Settings」メニューをクリックします。プロパティ・ダイアログの「C/C++ ビルド」－「設定」の Toolchain タブで、現在使用しているバージョンを指定し、「適用して閉じる」をクリックします。

図 2-17　プロパティの Toolchain 設定



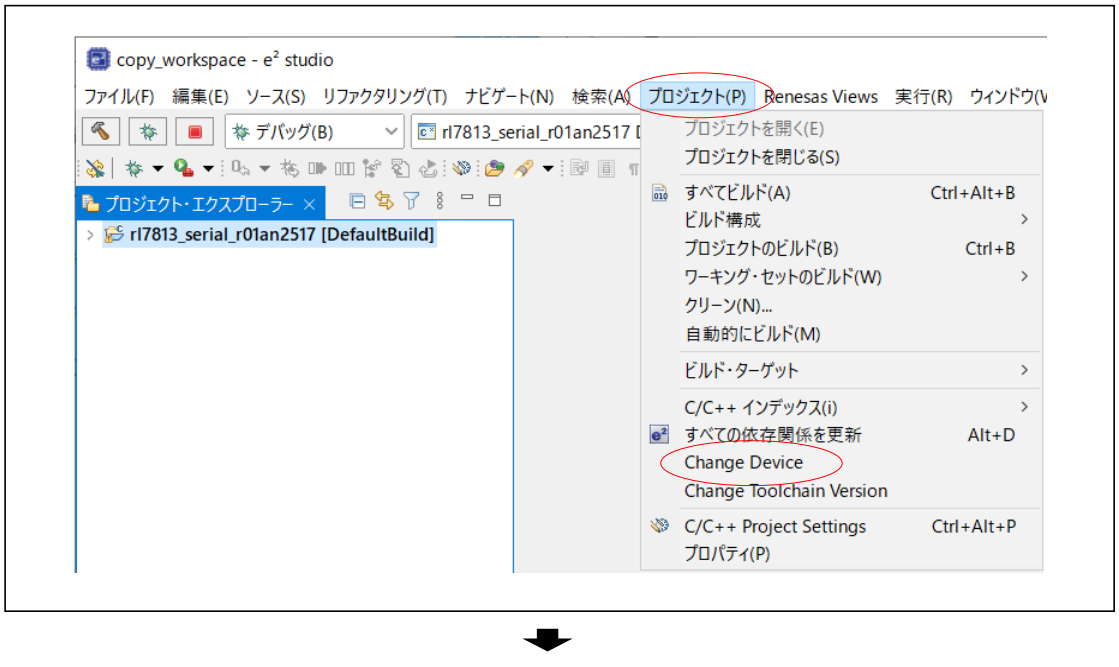
(8) e² studio のプロジェクト・エクスプローラで「コード生成」を右クリックし、メニュー「コード生成を削除する」をクリックします。

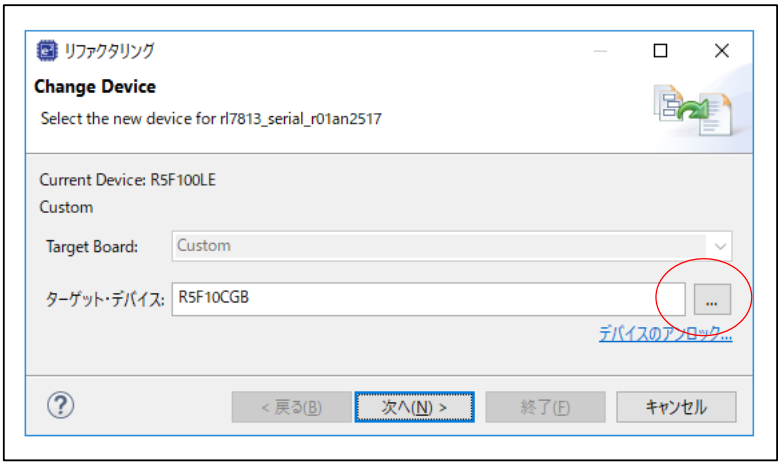
図 2-18　コード生成の削除

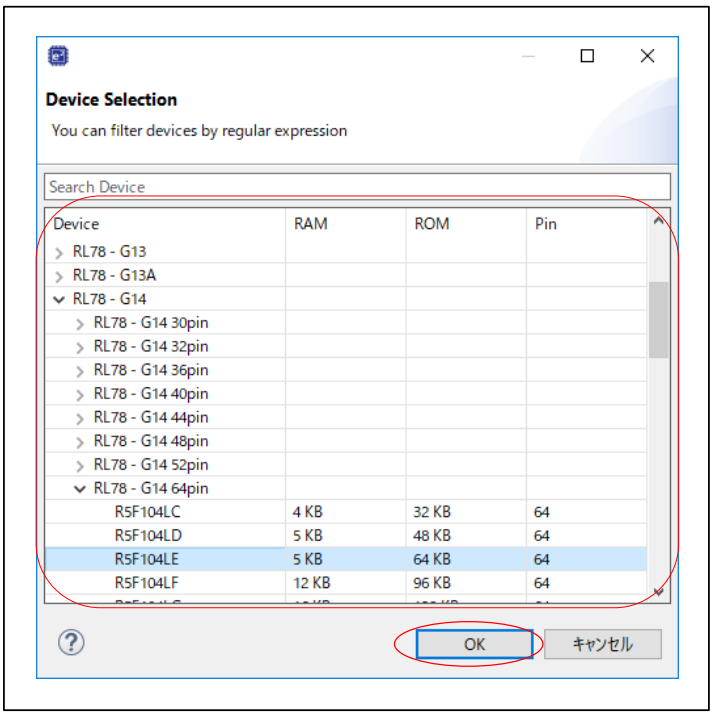


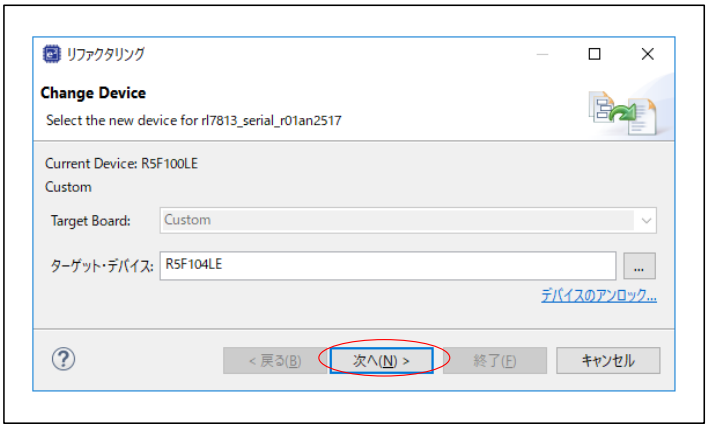
(9) e² studio のプロジェクト・エクスプローラでプロジェクト名をクリックで選択し、「プロジェクト」−「Change Device」をクリックします。Device Selection ダイアログで移植先のデバイスを選択し、「OK」をクリックします。続いて表示されるダイアログで「次へ」をクリックし、最後のダイアログで「終了」をクリックします。

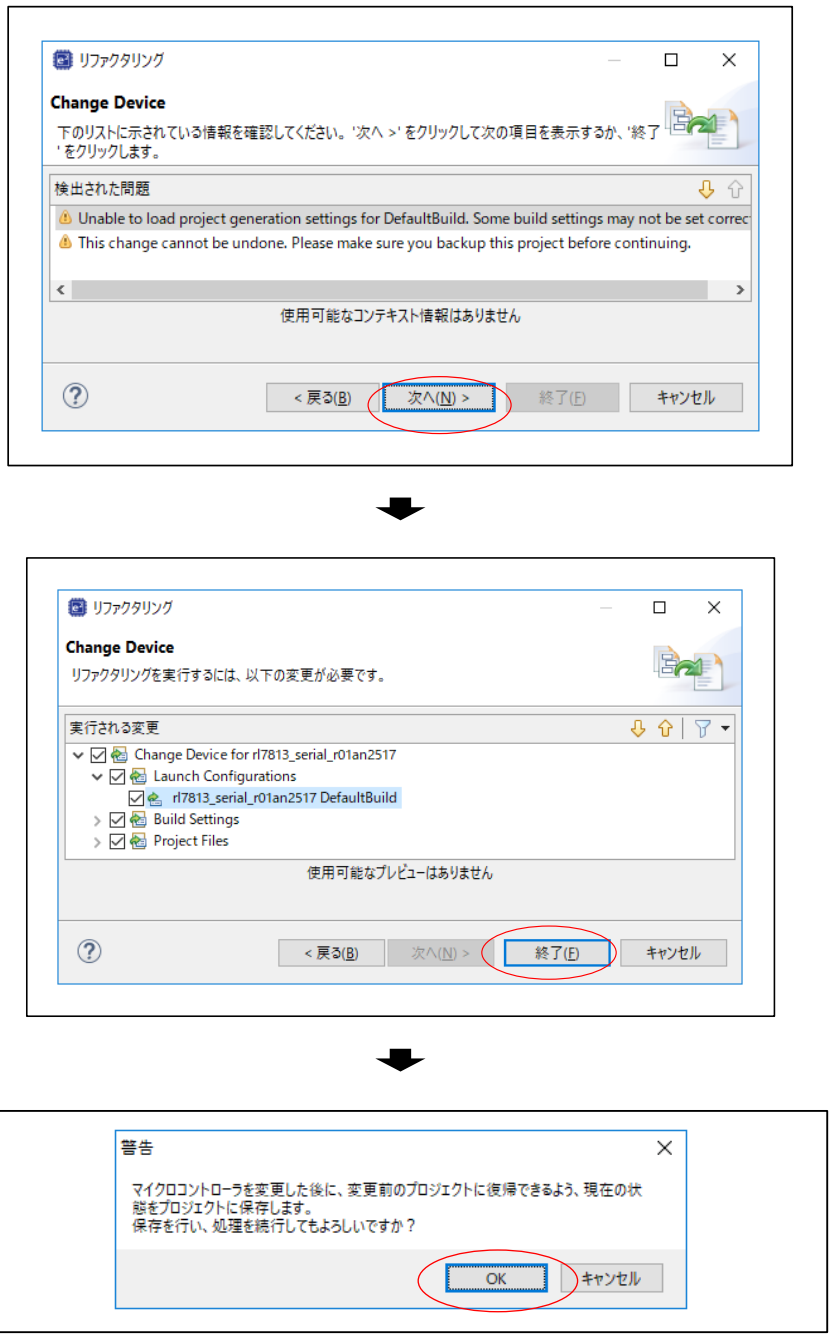
図 2-19　デバイス変更




リファクタリング
Change Device
Select the new device for rl7813_serial_r01an2517
Current Device: R5F100LE
Custom
Target Board:
Custom
ターゲット・デバイス:
R5F10CGB
...
デバイスのアンロック...
< 戻る(B)
次へ(N) >
終了(F)
キャンセル


Device Selection
You can filter devices by regular expression
Search Device
Device
RAM
ROM
Pin
RL78 - G13
RL78 - G13A
RL78 - G14
RL78 - G14 30pin
RL78 - G14 32pin
RL78 - G14 36pin
RL78 - G14 40pin
RL78 - G14 44pin
RL78 - G14 48pin
RL78 - G14 52pin
RL78 - G14 64pin
R5F104LC 4 KB 32 KB 64
R5F104LD 5 KB 48 KB 64
R5F104LE 5 KB 64 KB 64
R5F104LF 12 KB 96 KB 64
OK
キャンセル


リファクタリング
Change Device
Select the new device for rl7813_serial_r01an2517
Current Device: R5F100LE
Custom
Target Board:
Custom
ターゲット・デバイス:
R5F104LE
...
デバイスのアンロック...
< 戻る(B)
次へ(N) >
終了(F)
キャンセル



(10) 変更後のデバイス用プロジェクトにコード生成を追加します。ツール・バー上の「新規コード生成の作成」ボタンをクリックします。クリック後、プロジェクト・ツリーに「コード生成」ノードが追加されます。

図 2-20　新規コード生成の作成ボタン

2.4　移植プロジェクトでのコード生成設定

周辺機能設定に関わるコードを、移植先デバイス用の設定コードに自動で差し替えるために、元プロジェクトのコード生成ツール（CG）の設定に合わせて、移植プロジェクトで CG 設定をします。

元プロジェクト用にもう一つの CS+ for CC または e^2 studio を起動し、移植プロジェクトと元プロジェクトを並べて作業します。

図 2-21　CG 設定作業のイメージ画面

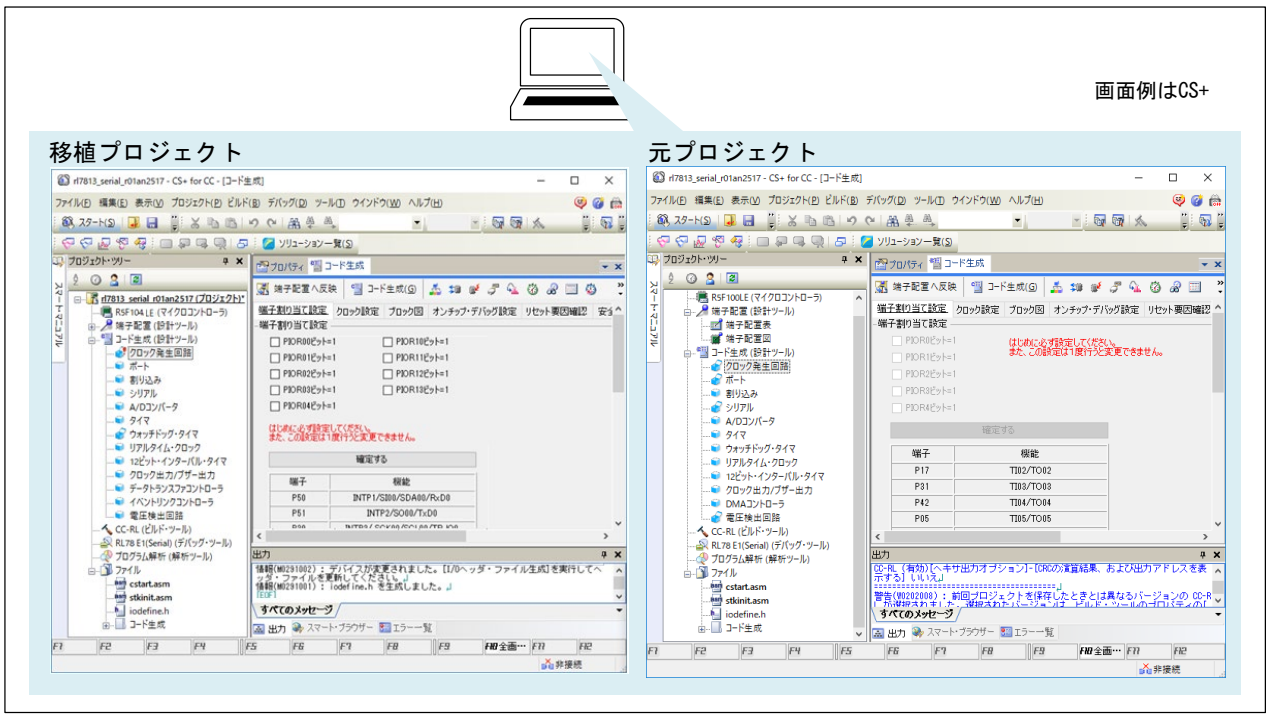
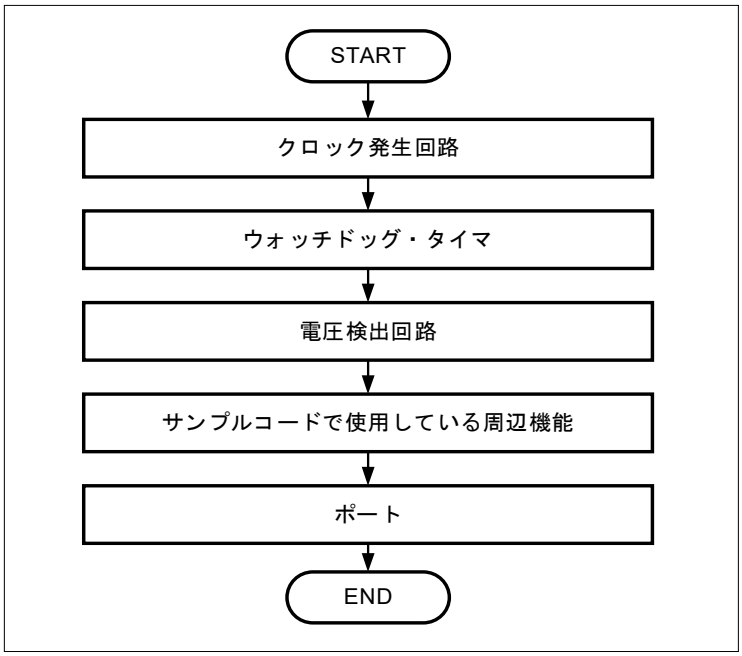


図 2-22　CG の設定順



以降の手順では、説明に CS+ for CC を使用しています。e2 studio でも設定手順は同じです。
また、元プロジェクトと移植プロジェクトで、設定項目の有無または選択肢に差がある場合があります。差がある場合には、移植プロジェクトで下記のように設定します。

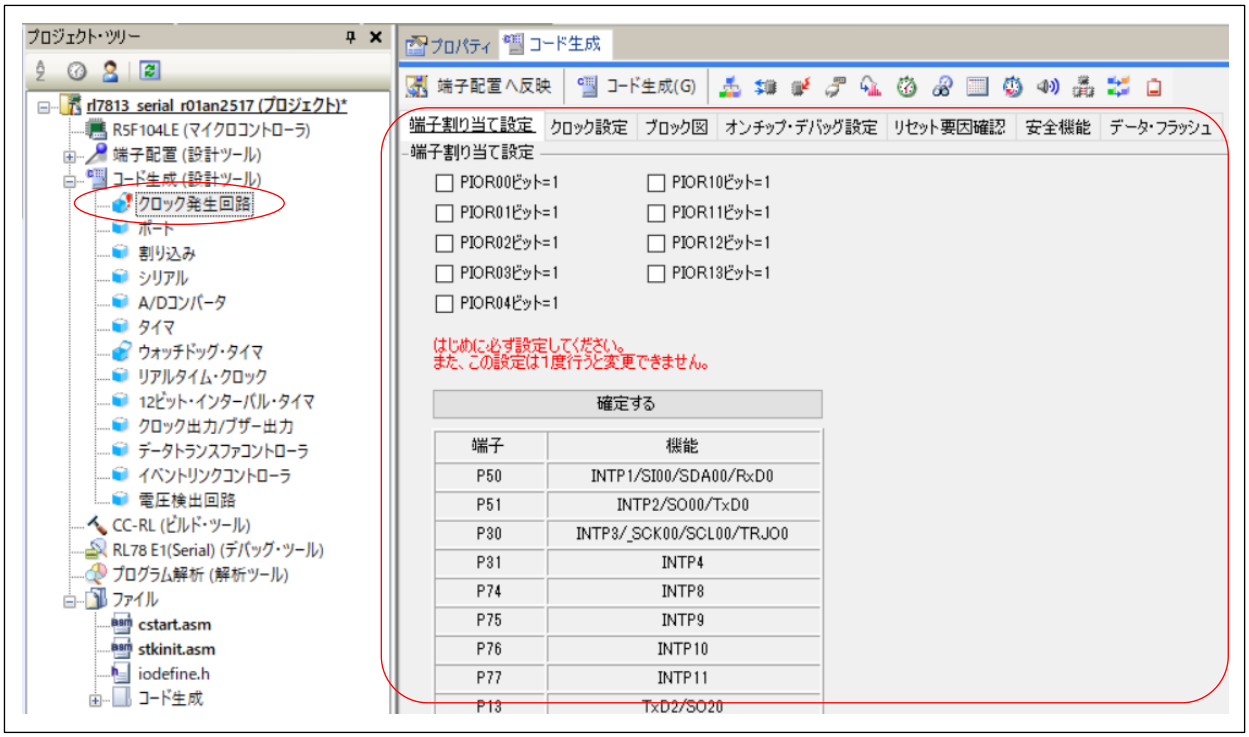
- 元プロジェクトにあるが、移植プロジェクトに設定項目がない場合
  →　移植プロジェクトでは、何も設定しません。
- 元プロジェクトになく、移植プロジェクトに設定項目がある場合
  →　移植プロジェクトでは、初期設定のまま、またはご使用の環境に合わせて設定します。
- 元プロジェクトと、移植プロジェクトに同じ設定項目はあるが選択肢が異なる場合
  →　移植プロジェクトでは、初期設定のまま、またはご使用の環境に合わせて設定します。

[手順]

(1) 新たに CS+ for CC を起動し、元プロジェクトを読み込みます。移植プロジェクトを読み込んだ CS+ for CC と、元プロジェクトを読み込んだ CS+ for CC の 2 つが起動している状態になります。

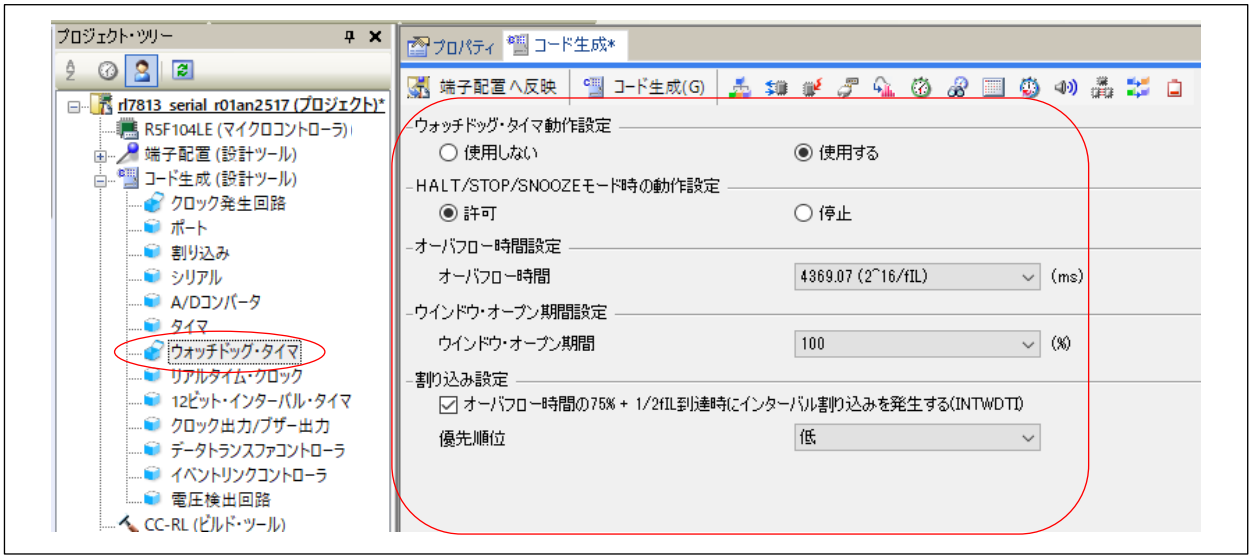
(2) 「クロック発生回路」を設定します。両プロジェクトのコード生成ノード下の「クロック発生回路」をダブルクリックします。元プロジェクトの設定に合わせて、移植プロジェクトで設定します。
「端子割り当て設定」、「クロック設定」、「オンチップ・デバッグ設定」、「リセット要因確認」、「安全機能」、「データ・フラッシュ」のすべてのタブを設定します。

図 2-23　「クロック発生回路」の画面



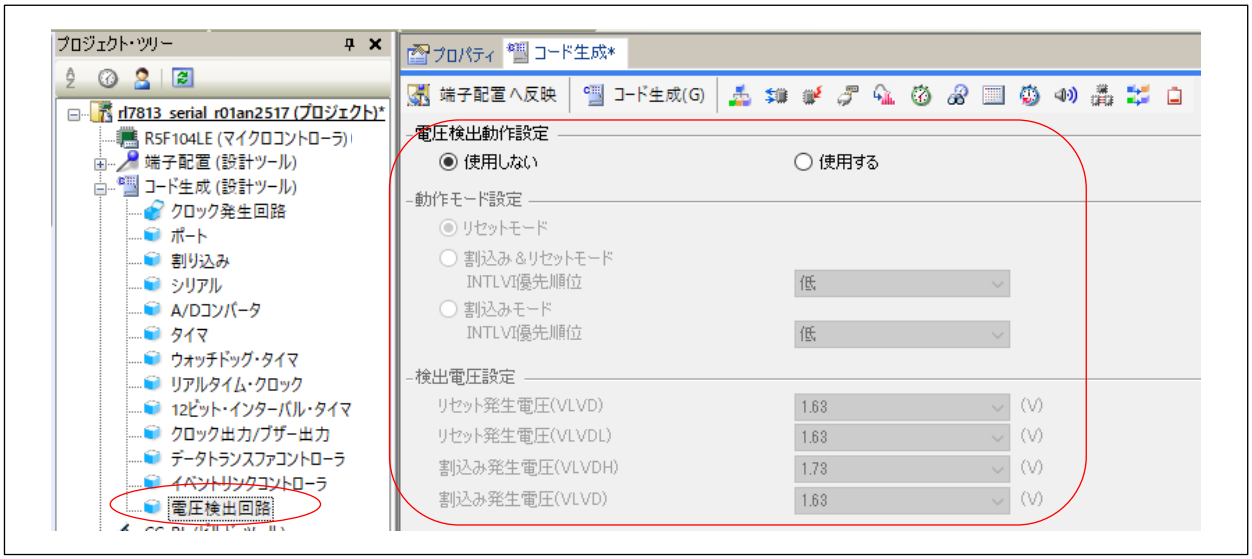
(3) 「ウォッチドッグ・タイマ」を設定します。両プロジェクトのコード生成ノード下の「ウォッチドッグ・タイマ」をダブルクリックします。元プロジェクトの設定に合わせて、移植プロジェクトで設定します。

図 2-24　「ウォッチドッグ・タイマ」の画面



(4) 「電圧検出回路」を設定します。両プロジェクトのコード生成ノード下の「電圧検出回路」をダブルクリックします。元プロジェクトの設定に合わせて、移植プロジェクトで設定します。

図 2-25　「電圧検出回路」の画面



(5) 元プロジェクトで設定されている周辺機能を、移植プロジェクトで設定します。設定の有無はアイコンで知ることができます。元プロジェクトの「コード生成（設計ツール）」に関連づけられているアイコンを確認し、設定のある周辺機能を、元プロジェクトの設定に合わせて移植プロジェクトで設定します。

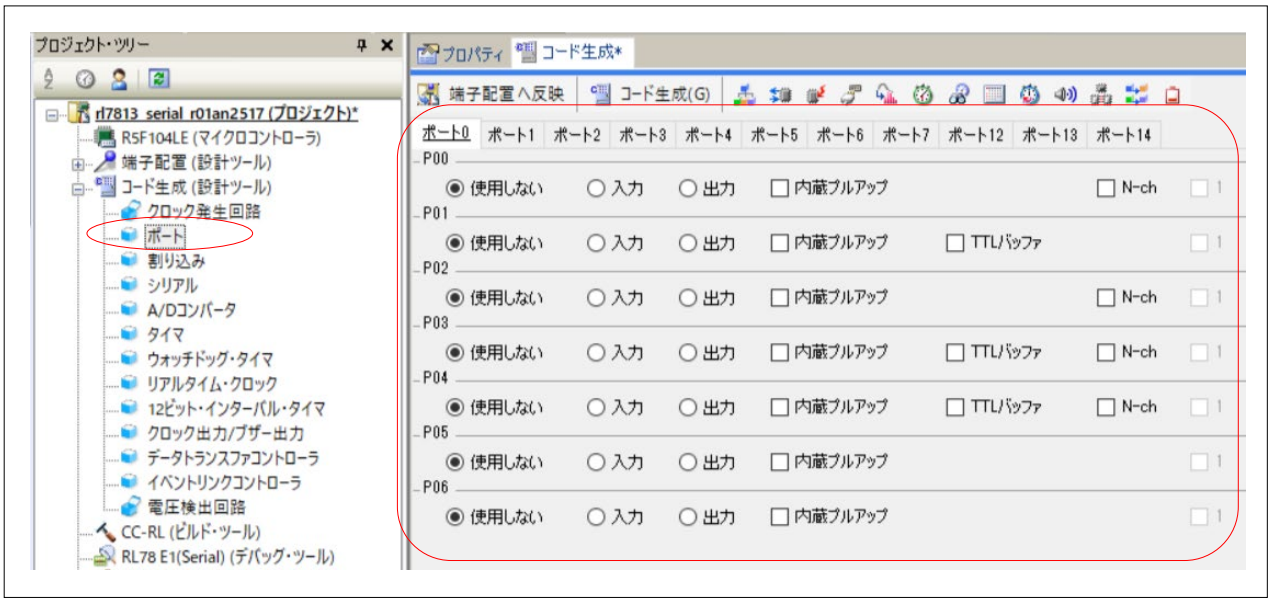
：周辺機能が設定されている

：周辺機能が設定されていない

(6) 「ポート」を設定します。両プロジェクトのコード生成ノード下の「ポート」をダブルクリックします。元プロジェクトの設定に合わせて、移植プロジェクトで設定します。
「ポート」での I/O ポート設定は、表 2-2 を参照してください。

注意. 未使用のポートは、端子処理などを適切に行い、電気的特性を満たすように設計してください。また、未使用の入力専用ポートは個別に抵抗を介して VDD 又は VSS に接続してください。

図 2-26　「ポート」の画面



(7) 移植プロジェクトで、設定内容を保存します。

CS+ for CC：「ファイル」ー「プロジェクトの保存」メニューを選択

e² studio：「ファイル」ー「すべて保管」メニューを選択

2.5　移植プロジェクトでのコード生成

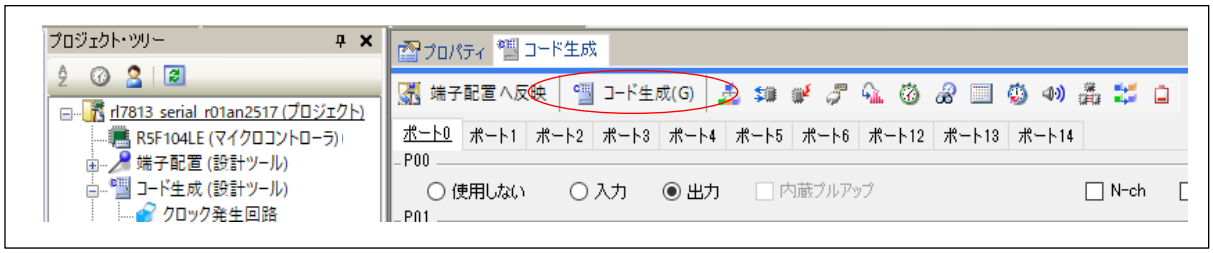
移植プロジェクトで、コード生成ツール（CG）の設定に合わせてコードを生成します。

2.5.1　CS+ for CC の移植プロジェクト

[手順]

(1) 移植プロジェクトで、「コード生成」をクリックします。既存の CG 生成の同名ファイルに、変更後のデバイスに合わせた周辺機能の設定コードが上書きされます。

図 2-27　「コード生成」ボタン



2.5.2　e² studio の移植プロジェクト

e² studio では、2.3.2 e² studio の移植プロジェクト で対象デバイスを変更時に「generate」フォルダ内に変更後のデバイス用にスタートアップファイル等が生成されています。また、コード生成ツール（CG）は、コード生成時に「src」フォルダ内にファイルを生成します。

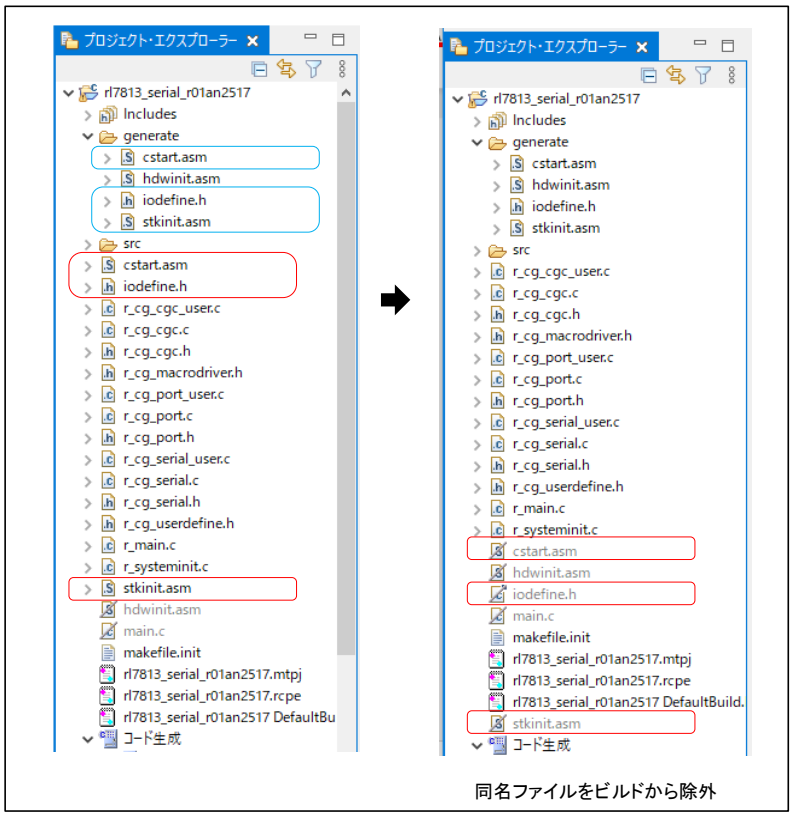
コード生成前に、下記の手順が必要となります。

- 元プロジェクトから引き継いだビルド対象ファイルのうち、「generate」フォルダ内のファイルと同名のファイルをビルド対象から除外します。
- CG 生成ファイルを「src」フォルダ内に移動します。

[手順]

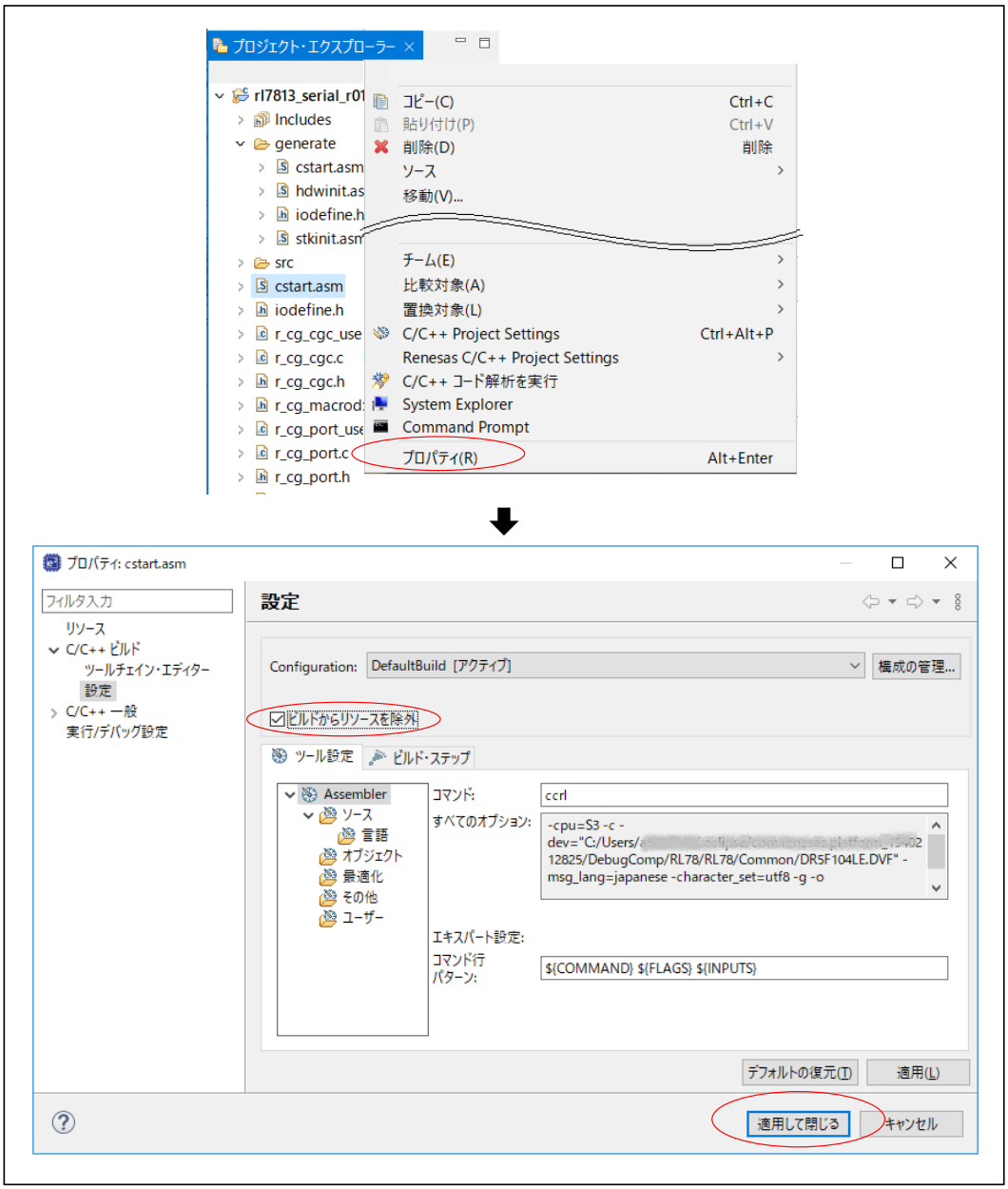
(1) 2.3.2 e² studio の移植プロジェクト で生成された「generate」フォルダ内のファイルと、移植プロジェクトに既に登録されている同名のファイルをビルド対象外にします。
同名ファイルを右クリックし、「プロパティ」メニューをクリックします。プロパティ・ダイアログの「C/C++ ビルド」－「設定」で、「ビルドからリソースを除外」にチェックし、「適用して閉じる」をクリックします。すべての同名ファイルにこの操作を行います。

図 2-28　同名ファイル除外



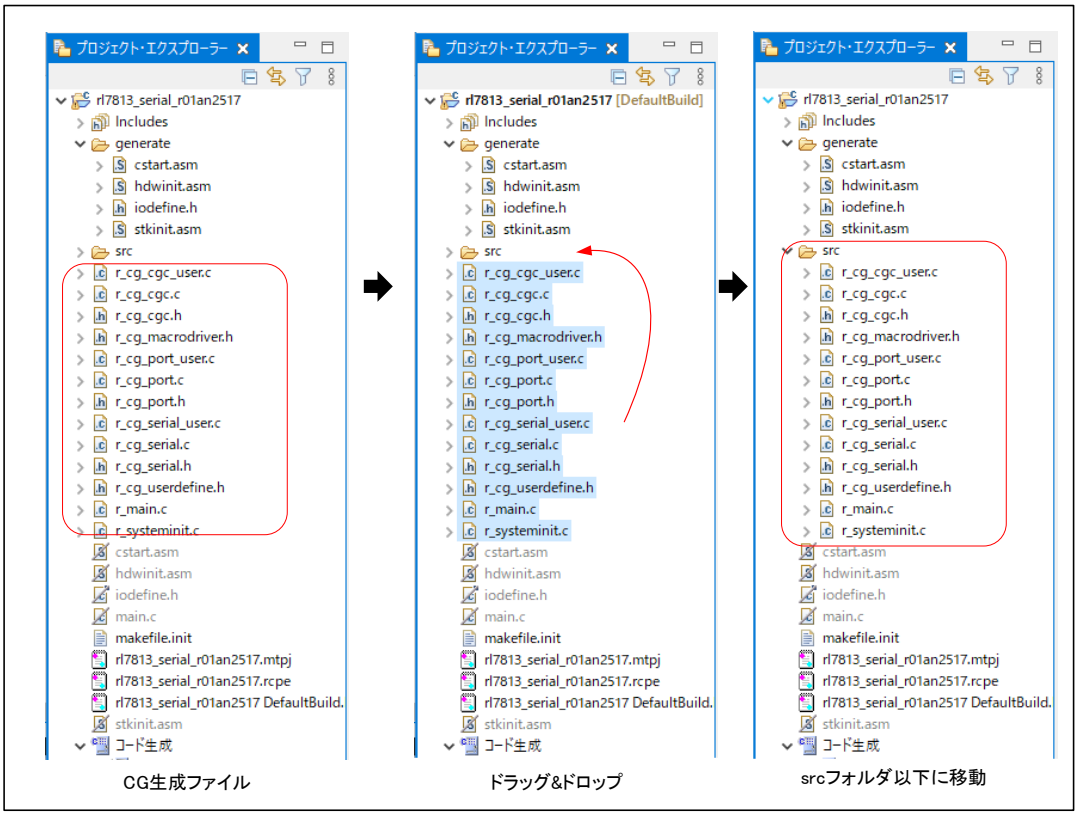
備考.　「generate」フォルダ内の「hdwinit.asm」は、手順(3)でコード生成により自動でビルド対象外となります。手順(3)を実施後、「hdwinit.asm」がビルド対象外とならなかった場合、図 2-29 に示す操作でビルド対象外にしてください。

図 2-29　ビルドからリソースを除外



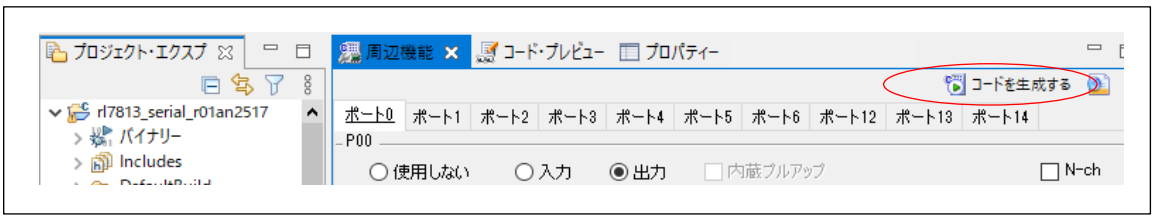
(2) コード生成ツール（CG）が生成するファイルが、プロジェクトの「src」フォルダ内に登録されているか確認します。表 1-3　CG 生成ファイルおよび関数の概要 に示す CG 生成ファイルが「src」フォルダ内に無い場合は、ファイルをドラッグ＆ドロップで「src」フォルダ内に移動します。また、プロジェクトに「src」フォルダが無い場合には、「src」フォルダを作成後に CG 生成ファイルを移動します。

図 2-30　CG 生成ファイルの移動



(3) 移植プロジェクトで、「コードを生成する」をクリックします。「src」フォルダ内の既存の CG 生成の同名ファイルに、変更後のデバイスに合わせた周辺機能の設定コードが上書きされます。

図 2-31　「コードを生成する」ボタン



2.6　移植プロジェクトでのファイル編集

サンプルコードによって、CG 生成のコードをそのまま使用せず、コードをコメント化するなど変更しているものがあります。しかし、「2.5 移植プロジェクトでのコード生成」において、移植プロジェクトでコードを生成することによりコードが上書きされ、元プロジェクトでの CG 生成コードに対する変更が消去されてしまいます。そのため、元プロジェクトでの CG 生成ファイルに対するコードの変更を確認し、変更に合わせて移植プロジェクトの CG 生成コードを編集します。

また、サンプルコードの移植時に、使用するリソース（周辺機能のチャネルや I/O ポート）を変更した場合は、CG 生成コード以外についてもコード編集が必要になります。

＜編集が必要な個所＞

- 元プロジェクトで、CG 生成コードをコメントアウト化するなど変更している個所

  →移植プロジェクトで、元プロジェクトと同様の変更をする。

- 移植プロジェクトで、使用する周辺機能のチャネルを変更した場合、周辺機能の動作許可／停止などの CG 生成の API 関数を呼び出している個所

  →移植プロジェクトでは、変更後のチャネル用 API 関数を呼び出しに変更する。ただし、API 関数本体は、「2.5 移植プロジェクトでのコード生成」で変更後のチャネルに置き換えられているので編集不要です。

- 移植プロジェクトで、使用する I/O ポートを変更した場合、I/O ポートの入出力値を扱っているコード

  →移植プロジェクトでは、変更後の I/O ポート名に変更する。

[手順]

(1) 元プロジェクトと移植プロジェクトの同名ファイルをエディタで開き、コードについて差分を確認し元プロジェクトの削除・変更を移植プロジェクトに反映させます。

(2) すべてのファイルについて、(1)を行います。

(3) 移植プロジェクトにあって、元プロジェクトにないファイルがあった場合は、ビルド対象から外します。

CS+ for CC：ファイル名を右クリック→「プロジェクトから外す」を選択

e^2 studio：ファイル名を右クリック→「プロパティ」を選択

プロパティ・ダイアログの「C/C++ビルド」で「ビルドからリソースを除外」をチェック

図 2-32 に CG 生成コードの変更例を示し、図 2-33 に周辺機能のチャネル変更に伴うコード編集例を示し、図 2-34　に I/O ポート変更に伴うコード編集例を示します。

図 2-32　CG 生成コードの変更例

r_systeminit.c 内の関数

移行プロジェクト：CG生成のコード

```
/***********************************************
* Function Name: R_Systeminit
* Description  : This function initializes every
* Arguments    : None
* Return Value : None
***********************************************
void R_Systeminit(void)
{
    PIOR = 0x00U;
    R_CGC_Create();
    R_PORT_Create();
    R_ADC_Create();
    R_IT_Create();
    R_INTC_Create();

    IAWCTL = 0x00U;
}
```

元プロジェクト：一部をコメントアウトしている

```
/***********************************************
* Function Name: R_Systeminit
* Description  : This function initializes every
* Arguments    : None
* Return Value : None
***********************************************
void R_Systeminit(void)
{
    PIOR = 0x00U;
    R_CGC_Create();
    R_PORT_Create();
//    R_ADC_Create();
    R_IT_Create();
    R_INTC_Create();

    IAWCTL = 0x00U;
}
```

元プロジェクトに合わせて、”R_ADC_Create();”をコメントアウトする

```
void R_Systeminit(void)
{
    PIOR = 0x00U;
    R_CGC_Create();
    R_PORT_Create();
//    R_ADC_Create();
    R_IT_Create();
    R_INTC_Create();

    IAWCTL = 0x00U;
}
```

図 2-33　周辺機能のチャネル変更に伴うコード編集例

r_main.c 内の関数

移行プロジェクト：UART1の使用に変更

API関数本体は、コード生成により書き換えられるが、
周辺機能のAPI関数を呼び出している個所はコード生成
により書き換わらない

```
/***********************************************************
* Function Name: main
* Description  : This function implements main function.
* Arguments    : None
* Return Value : None
***********************************************************/
void main(void)
{
    R_MAN_UserInit();
    /* Start user code. Do not edit comment generated here
    {
        volatile MD_STATUS    status = 0U;        /* UART

        /* UART0 receive buffer setting */
        status = R_UART0_Receive(&g_uart0_rx_buffer, 1U);

        /* Start the UART0 Tx/Rx operation */
        R_UART0_Start();

        /* Main loop */
        while (1U)
        {
```

使用チャネルに合わせてAPI関数名を変更する

```
void main(void)
{
    R_MAN_UserInit();
    /* Start user code. Do not edit comment generated here
    {
        volatile MD_STATUS    status = 0U;        /* UART

        /* UART0 receive buffer setting */
        status = R_UART1_Receive(&g_uart0_rx_buffer, 1U);

        /* Start the UART0 Tx/Rx operation */
        R_UART1_Start();

        /* Main loop */
        while (1U)
        {
            HALT();     /* Wait for UART Rx interrupt */
```

元プロジェクト：UART0を使用

```
/***********************************************************
* Function Name: main
* Description  : This function implements main function.
* Arguments    : None
* Return Value : None
***********************************************************/
void main(void)
{
    R_MAN_UserInit();
    /* Start user code. Do not edit comment generated here
    {
        volatile MD_STATUS    status = 0U;        /* UART

        /* UART0 receive buffer setting */
        status = R_UART0_Receive(&g_uart0_rx_buffer, 1U);

        /* Start the UART0 Tx/Rx operation */
        R_UART0_Start();

        /* Main loop */
        while (1U)
        {
```

図 2-34　I/O ポート変更に伴うコード編集例

r_main.c 内の関数

移行プロジェクト：LED表示用にP10 ,P11 を使用に変更

自動では書き換わらない

```
/***********************************************
* Function Name: main
* Description  : This function implements main func
* Arguments    : None
* Return Value : None
***********************************************
void main(void)
{
    R_MAIN_UserInit();
    /* Start user code. Do not edit comment generat
    {

        P5_bit.no2 = 0U;    /* LED setting of the H
        P5_bit.no3 = 0U;    /* P5.2-P5.3:0 */
```

使用I/Oポートに合わせてコードを変更する

```
void main(void)
{
    R_MAIN_UserInit();
    /* Start user code. Do not edit comment ge
    {

        P0_bit.no0 = 0U;    /* LED setting of
        P0_bit.no1 = 0U;    /* P5.2-P5.3:0 */
```

元プロジェクト：LED表示用にP52,P53を使用

```
/***********************************************
* Function Name: main
* Description  : This function implements main func
* Arguments    : None
* Return Value : None
***********************************************
void main(void)
{
    R_MAIN_UserInit();
    /* Start user code. Do not edit comment generat
    {

        P5_bit.no2 = 0U;    /* LED setting of the H
        P5_bit.no3 = 0U;    /* P5.2-P5.3:0 */
```

2.7　移植プロジェクトのビルド

デバイス変更および元プロジェクトに合わせてコードを編集した移植プロジェクトをビルドします。

[手順]

(1)　移植プロジェクトをビルドします。

CS+ for CC：「ビルド」→「リビルド・プロジェクト」メニューを選択

e^2 studio：「プロジェクト」→「プロジェクトのビルド」メニューを選択

注意. エラーが出た場合は、エラーメッセージを元に対応してください。

3. 移植例

この章では、サンプルコード移植の具体例を説明します。

以下に移植例に使用するアプリケーションノートを示します。

移植例 1

項目	内容
アプリケーションノート（サンプルコード）	RL78/G13 シリアル・アレイ・ユニット 3 線シリアル I/O（マスタ送受信）CC-RL Rev2.01 (R01AN2547JJ0201)
移植先デバイス	RL78/G12
統合開発環境	ルネサス エレクトロニクス製　CS+ for CC V8.07.00
C コンパイラ	ルネサス エレクトロニクス製　CC-RL V1.11.00
移植時の作業概要	I/O ポートの変更を伴う移植例

移植例 2

項目	内容
アプリケーションノート（サンプルコード）	RL78/G13 シリアル・インタフェース IICA（マスタ送受信）Rev.2.01 (R01AN2759JJ0201)
移植先デバイス	RL78/G14
統合開発環境	ルネサス エレクトロニクス製　e^2 studio V2022-01 (22.1.0)
C コンパイラ	ルネサス エレクトロニクス製　CC-RL V1.11.00
移植時の作業概要	コード生成ツールが生成したファイルの編集を伴う移植例

3.1　移植例 1

アプリケーションノート「RL78/G13 シリアル・アレイ・ユニット 3 線シリアル I/O（マスタ送受信）CC-RL」R01AN2547JJ0201（以下、RL78/G13 用 AN）で説明されている RL78/G13 用サンプルコードを RL78/G12 用サンプルコードに移植する手順を示します。

3.1.1　事前準備

RL78/G13 用 AN の「2. 動作確認条件」、「表 1.1 使用する周辺機能と用途」、「4.2 使用端子一覧」および「4.1 ハードウェア構成例」を参照し使用しているリソースを確認します。
RL78/G12 へ移植時の構成例を以下に示します。

- 動作確認条件

差分を赤字で示します。RL78/G12 は 32MHz 動作できないため、動作周波数を 24MHz に変更します。動作周波数の変更による各種レジスタ設定値の変更はコード生成（CG）が対応するため、使用するサンプルコードの動作については影響ありません。

表 3-1　動作確認条件（移植例 1）

項目	RL78/G13	RL78/G12
使用マイコン	RL78/G13（R5F100LEA）	RL78/G12（R5F1026A）
動作周波数	● 高速オンチップオシレータ（HOCO）クロック：32MHz ● CPU／周辺ハードウエア・クロック：32MHz	● 高速オンチップオシレータ（HOCO）クロック：24MHz ● CPU／周辺ハードウエア・クロック：24MHz
動作電圧	5.0V（2.9V～5.5V で動作可能） LVD 動作（V_{LVD}）：リセット・モード 2.81V (2.76V～2.87V)	5.0V（2.9V～5.5V で動作可能） LVD 動作（V_{LVD}）：リセット・モード 2.81V (2.76V～2.87V)

- 使用する周辺機能と用途

RL78/G13 用 AN で使用している周辺機能およびチャネルをそのまま RL78/G12 でも使用します。

表 3-2　使用する周辺機能（移植例 1）

RL78/G13

周辺機能	用途
シリアル・アレイ・ユニット 0 チャネル 0	CSI00 のマスタ送受信動作をする
タイマ・アレイ・ユニット 0 チャネル 0	インターバル・タイマ動作

RL78/G12

周辺機能	用途
シリアル・アレイ・ユニット 0 チャネル 0	CSI00 のマスタ送受信動作をする
タイマ・アレイ・ユニット 0 チャネル 0	インターバル・タイマ動作

● 使用端子

差分を赤字で示します。RL78/G12（R5F1026A）には P00 端子が存在しません。そのため、P00 を P41 に変更します。

表 3-3　使用端子（移植例 1）

RL78/G13

端子名	入出力	内容
P10/SCK00/SCL00	出力	シリアル・クロック出力用端子
P11/SI00/RxD0/TOOL RxD/SDA00	入力	データ受信用端子
P12/SO00/TxD0/TOOLTxD	出力	データ送信用端子
P00/ANI17/TI00/TxD1	入力	BUSY 信号検出用端子

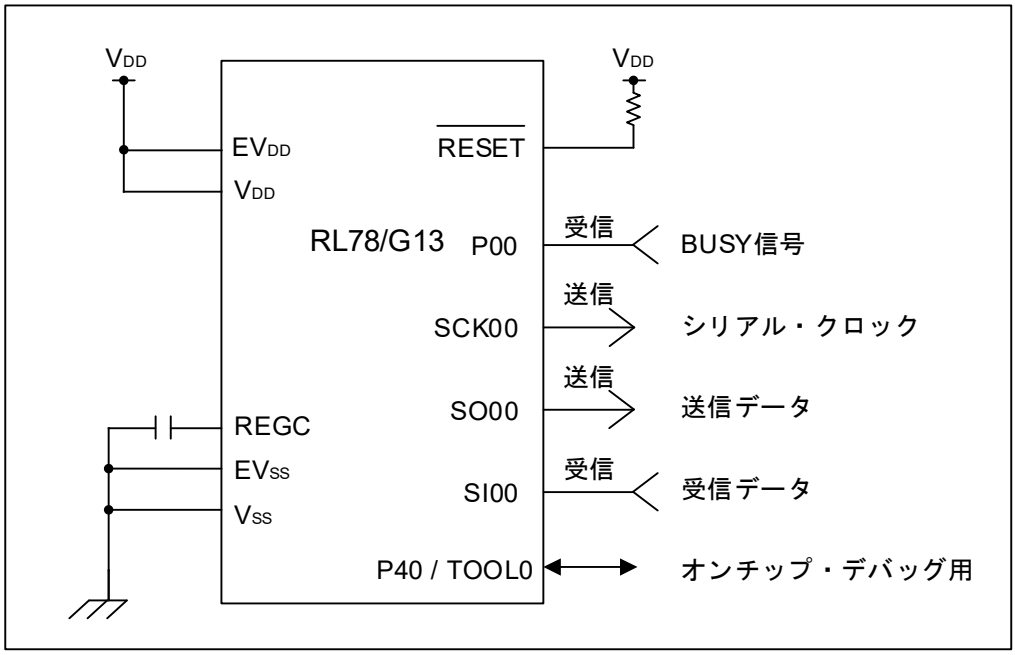
RL78/G12

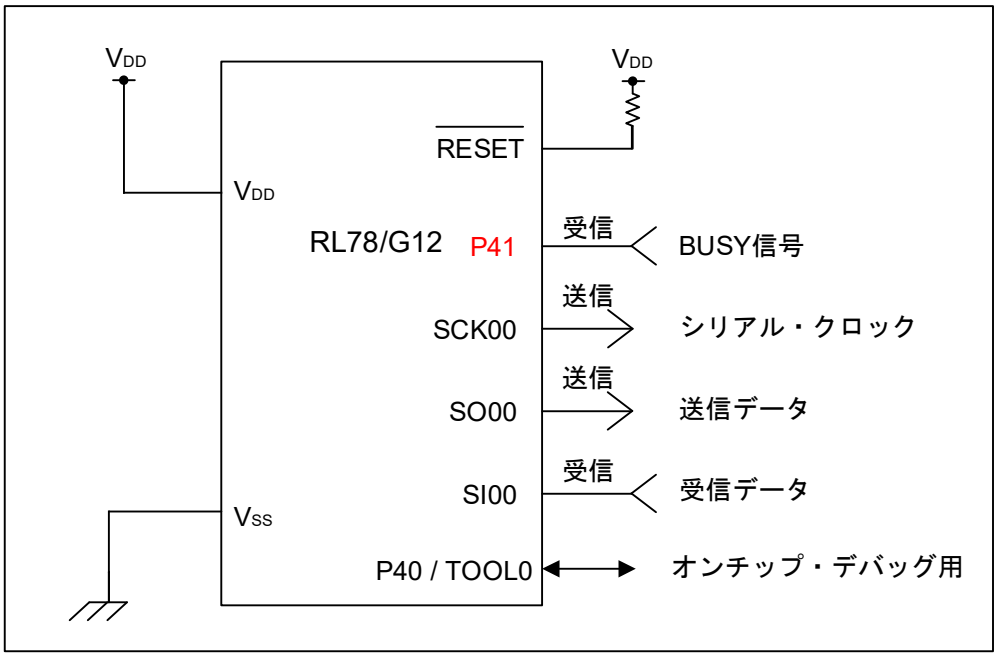
端子名	入出力	内容
P10/ANI16/PCLBUZ0/SCK00/SCL00	出力	シリアル・クロック出力用端子
P11/ANI17/SI00/RxD0/SDA00 /TOOLRxD	入力	データ受信用端子
P12/ANI18/SO00/TxD0/TOOLTxD	出力	データ送信用端子
P41/ANI22/SO01/SDA01/TI02/TO02/INTP1	入力	BUSY 信号検出用端子

● ハードウェア構成

差分を赤字で示します。RL78/G12（R5F1026A）には P00 端子が存在しません。そのため、P00 を P41 に変更します。

図 3-1　ハードウェア構成例（移植例 1）





注意 1　この回路イメージは接続の概要を示す為に簡略化しています。実際に回路を作成される場合は、端子処理などを適切に行い、電気的特性を満たすように設計してください（入力専用ポートは個別に抵抗を介して VDD 又は VSS に接続して下さい）。

2　EVSS で始まる名前の端子がある場合には VSS に、EVDD で始まる名前の端子がある場合には VDD にそれぞれ接続してください。

3　VDD は LVD にて設定したリセット解除電圧（VLVD）以上にしてください。

3.1.2　プロジェクトのコピー

「2.2 プロジェクトのコピー」を参照し、移植プロジェクトを作成します。

3.1.3　移植プロジェクトのデバイス変更

「2.3.1 CS+ for CC の移植プロジェクト」を参照し、移植プロジェクトのデバイスを変更します。

3.1.4　移植プロジェクトでのコード生成設定

周辺機能設定に関わるコードを、移植先デバイス用の設定コードに差し替えるために、元プロジェクトのコード生成ツール（CG）の設定に合わせて、移植プロジェクトで CG 設定をします。

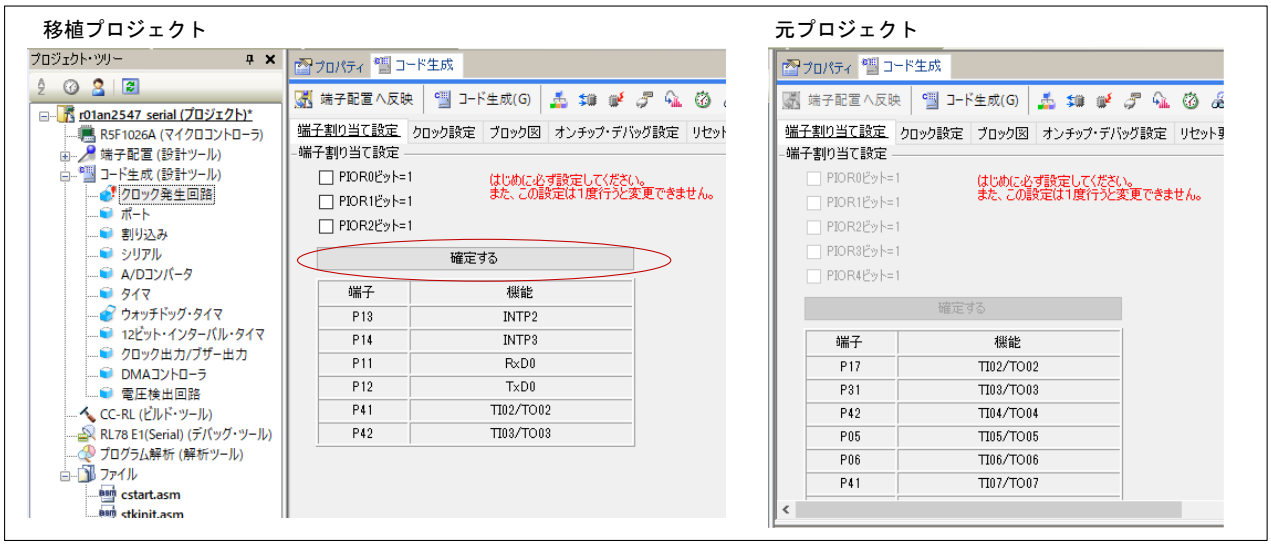
元プロジェクト用にもう一つの CS+ for CC を起動し、移植プロジェクトと元プロジェクトを並べて作業します。

(1) クロック発生回路

- 端子割り当て設定

  今回の移植では、機能端子のリダイレクションは使用しないので、設定は何も変更せずに「確定する」をクリックします。

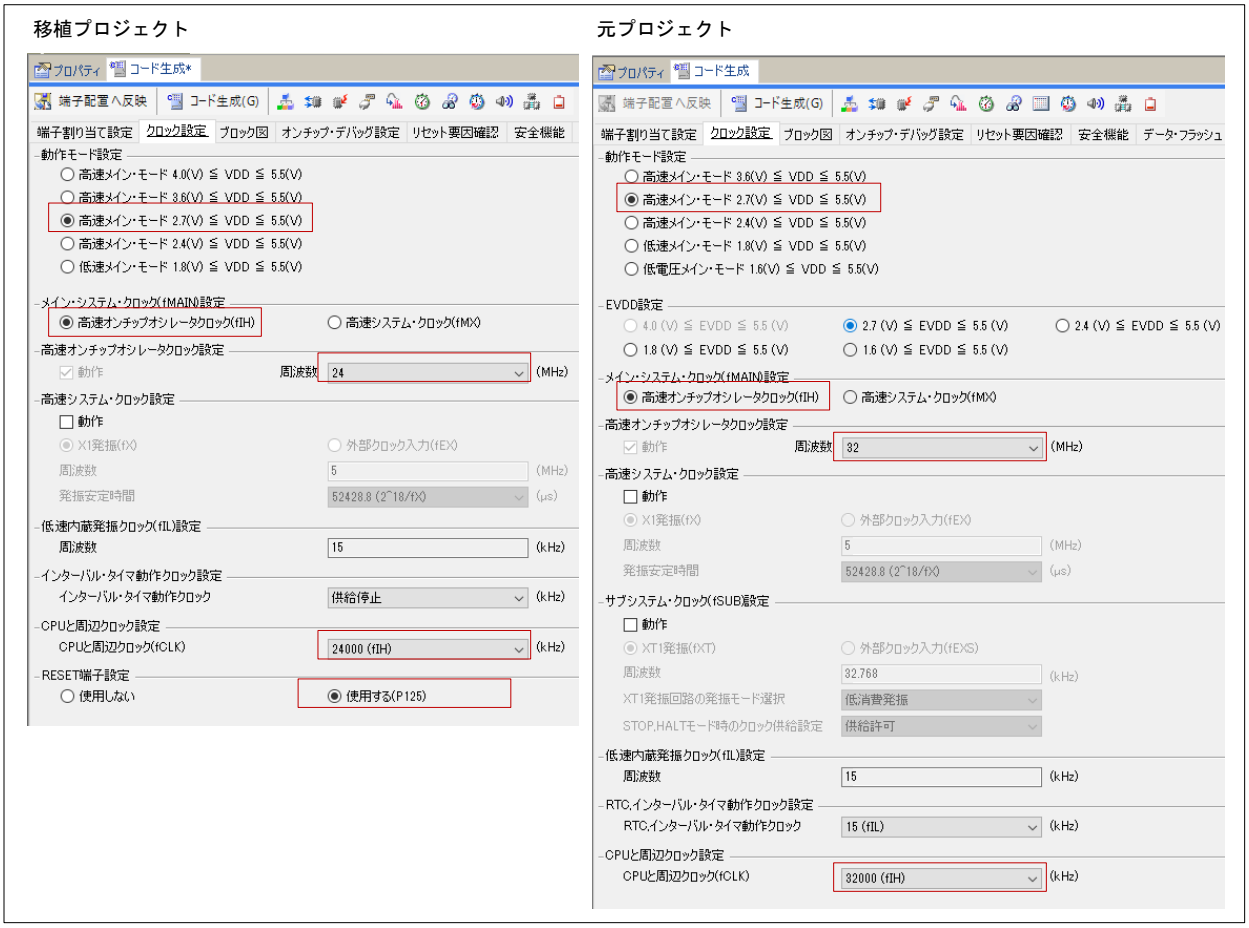
図 3-2　端子割り当て設定（移植例 1）



● クロック設定

元プロジェクトと移植プロジェクトで、いくつかの設定項目が異なります。今回の移植では、表 3-1 動作確認条件（移植例 1）に合わせて、下記のように設定します。

図 3-3　クロック設定（移植例 1）

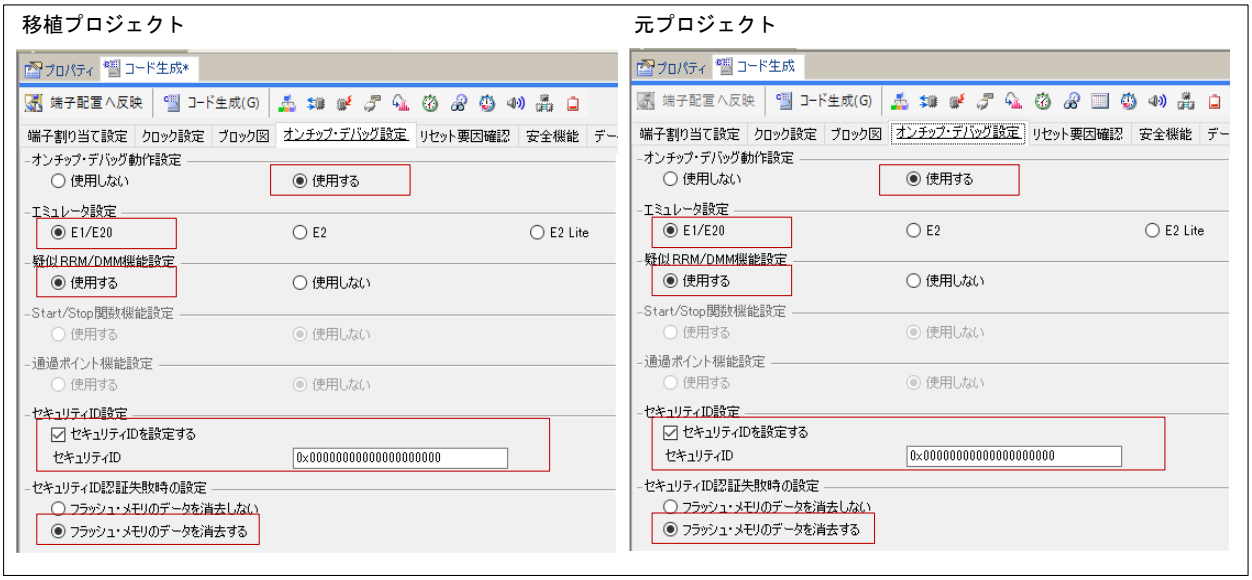


- オンチップ・デバッグ設定

  元プロジェクトと同じ設定をします。

  備考．ご使用のエミュレータに合わせて、エミュレータ設定が変更可能です。

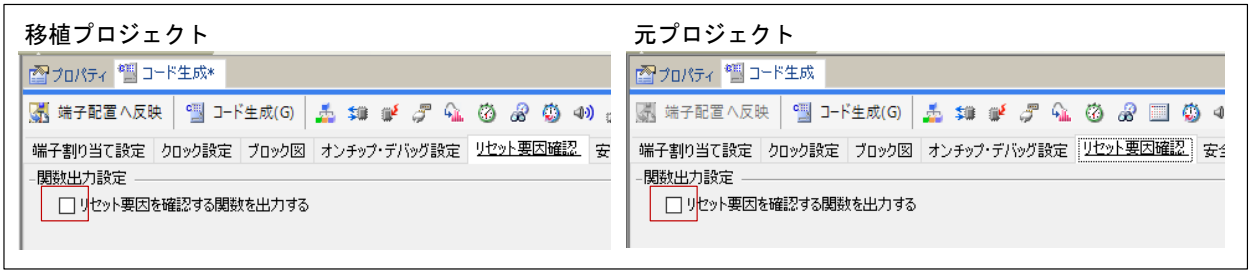
図 3-4　オンチップ・デバッグ設定（移植例 1）



- リセット要因確認

  元プロジェクトと同じ設定をします。

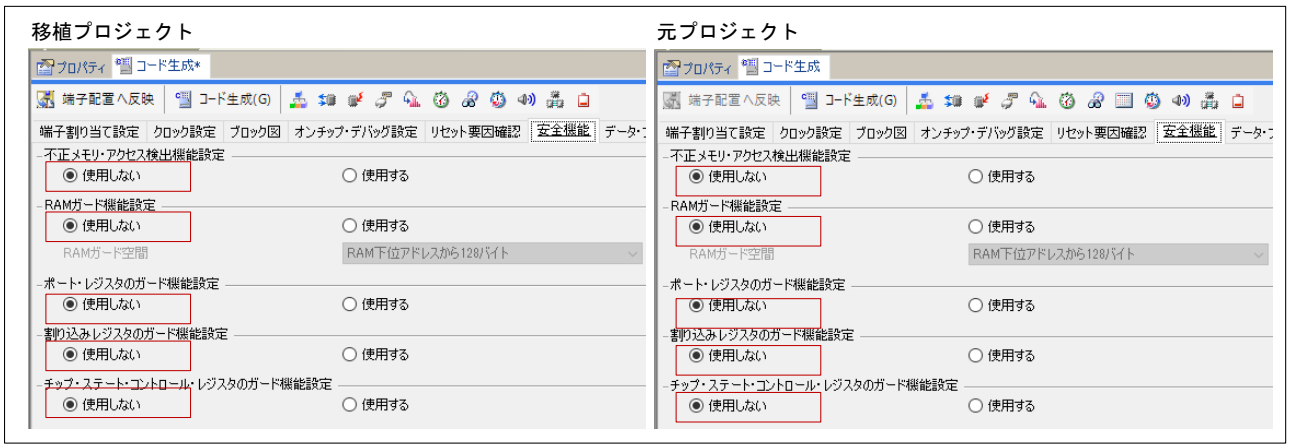
図 3-5　リセット要因確認（移植例 1）



- 安全機能

  元プロジェクトと同じ設定をします。

図 3-6　安全機能（移植例 1）



- データ・フラッシュ

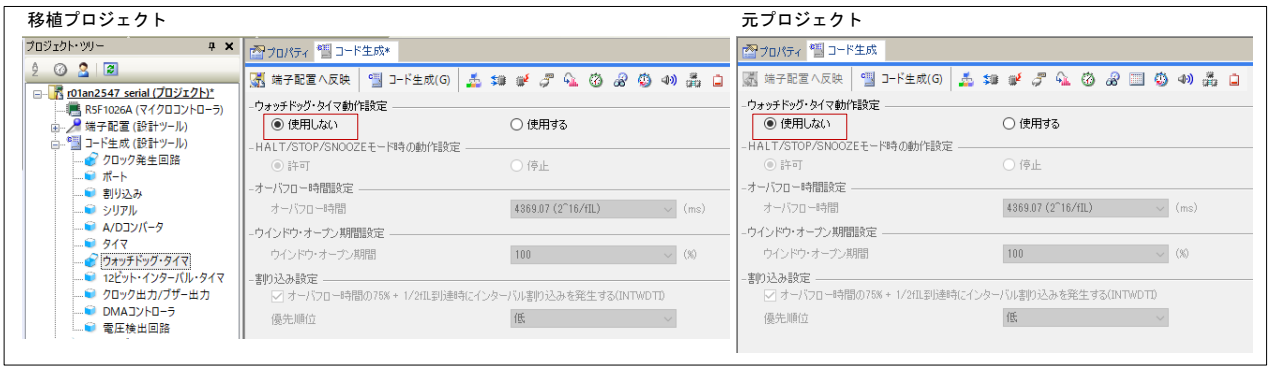
  元プロジェクトと同じ設定をします。

図 3-7　データ・フラッシュ（移植例 1）

(2) ウォッチドッグ・タイマ

元プロジェクトと同じ設定をします。

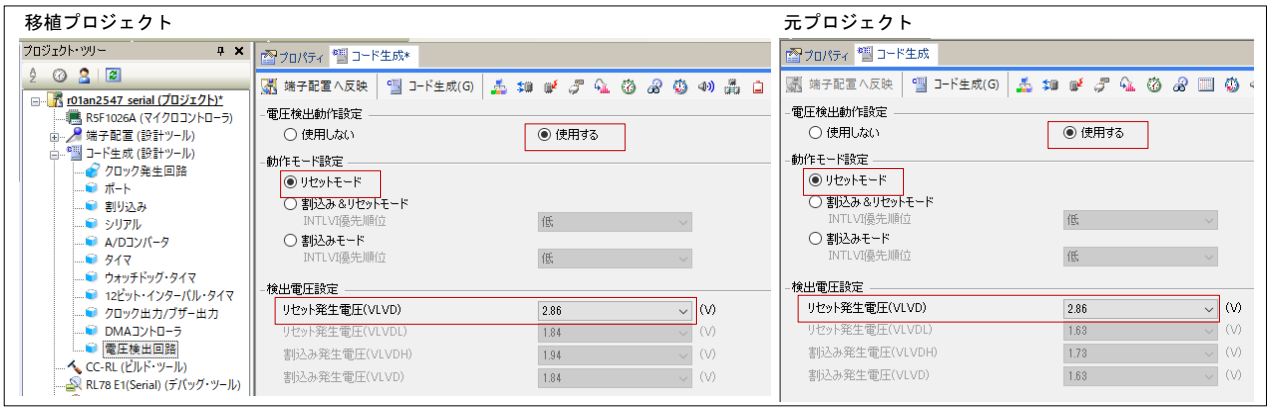
図 3-8　ウォッチドッグ・タイマ（移植例 1）



(3) 電圧検出回路

元プロジェクトと同じ設定をします。

図 3-9　電圧検出回路（移植例 1）



(4) サンプルコードで使用している周辺機能

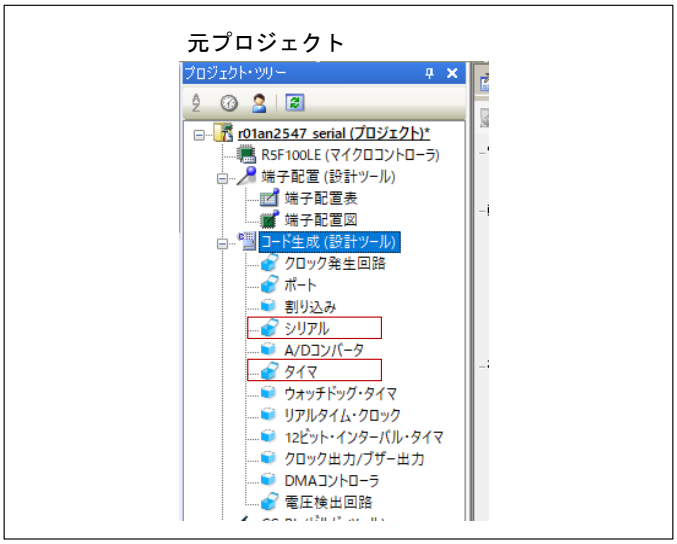
元プロジェクトの「コード生成（設計ツール）」に関連づけられているアイコンを確認し、設定のある周辺機能を元プロジェクトの設定に合わせて移植プロジェクトで設定します。

本 RL78/G13 用 AN では、「シリアル」と「タイマ」が設定されています。なお、「ポート」は(5)で設定します。

：周辺機能が設定されている

：周辺機能が設定されていない

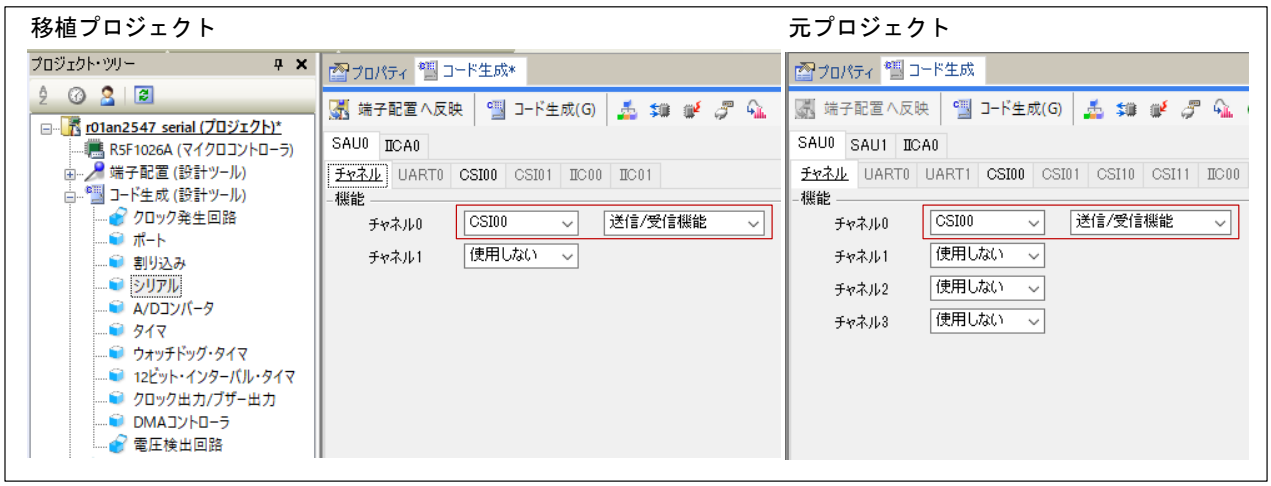
図 3-10　設定のある周辺機能（移植例 1）

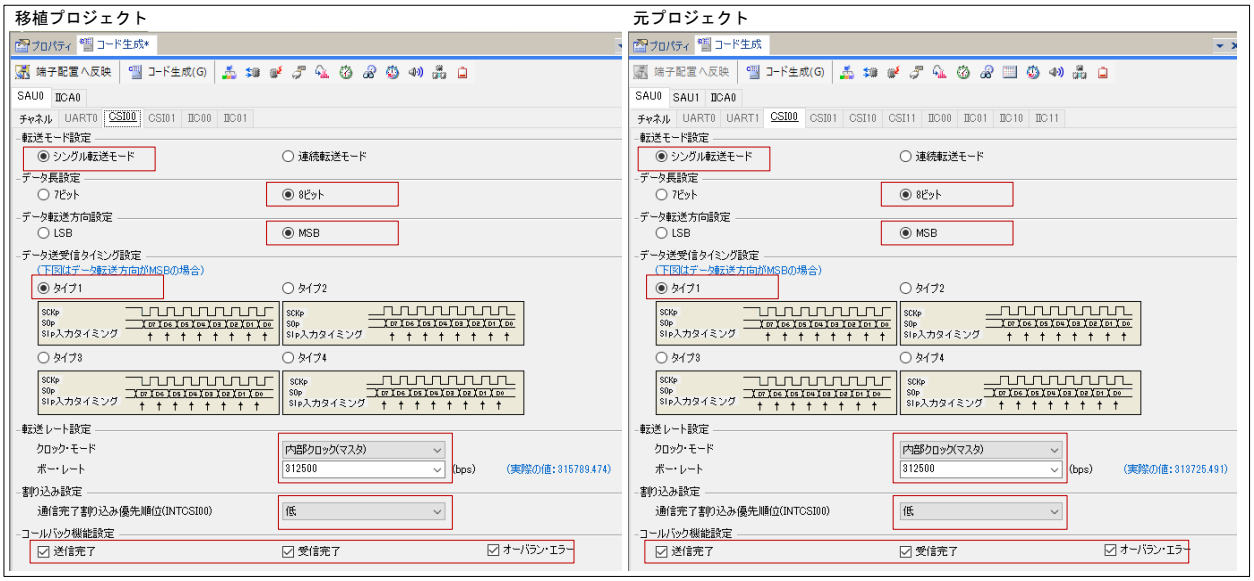


- シリアル

  元プロジェクトと同じ設定をします。

図 3-11　シリアルーCSI00（移植例 1）

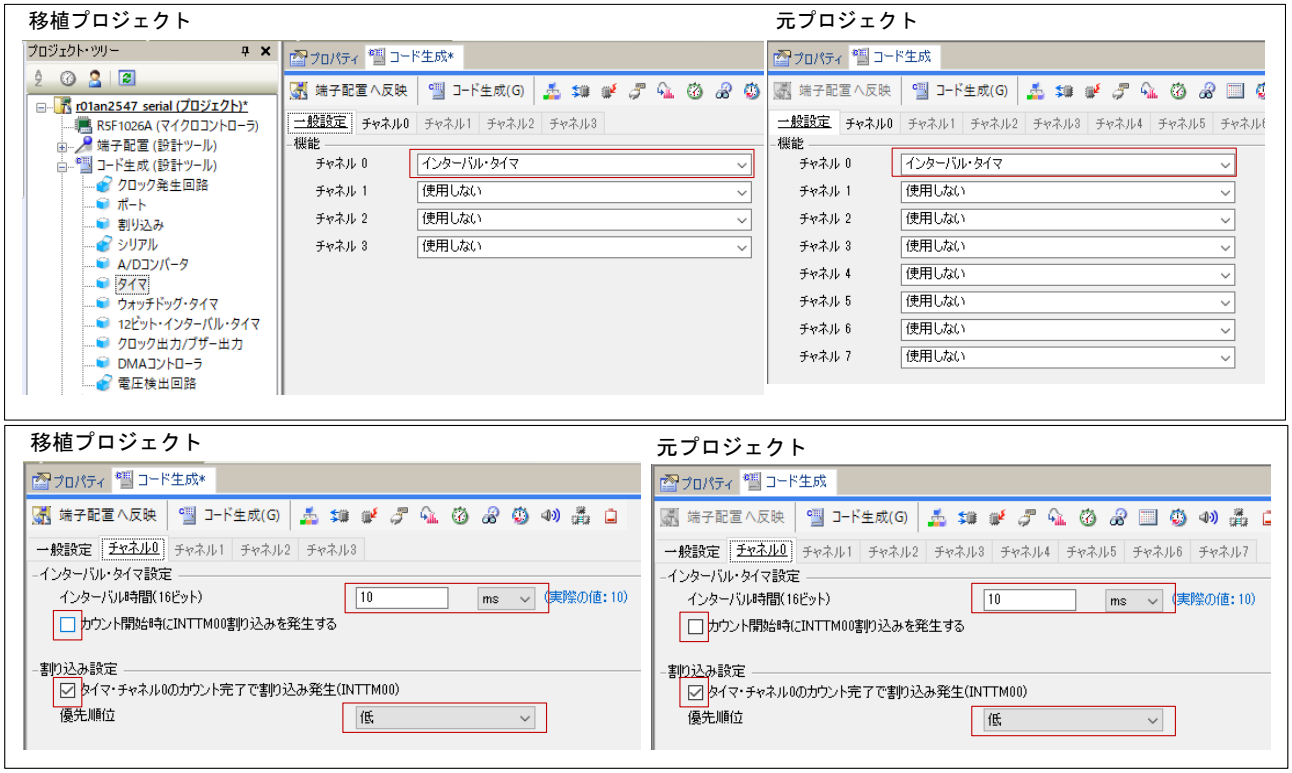




- タイマ

元プロジェクトと同じ設定をします。

図 3-12　タイマーチャネル 0（移植例 1）

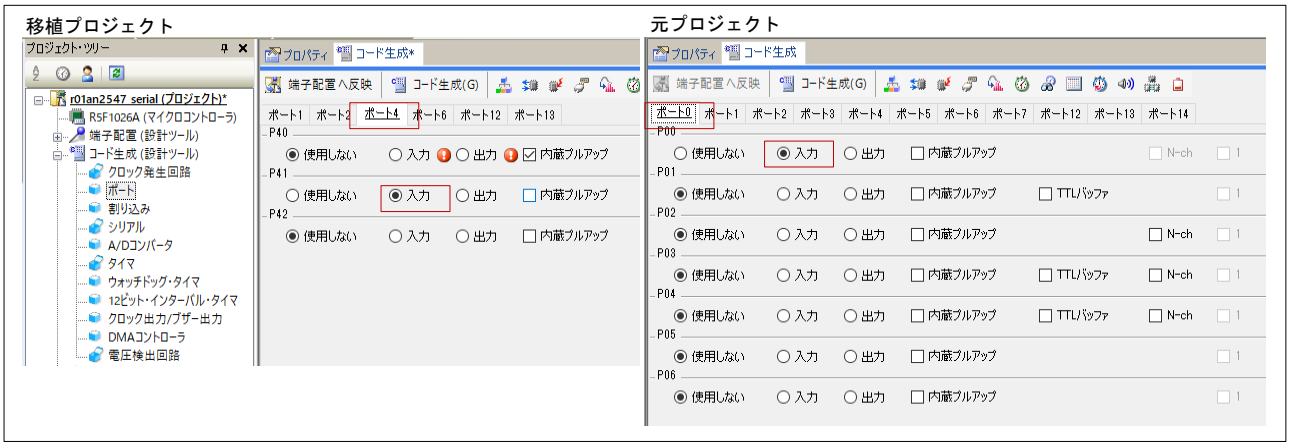


(5) ポート

元プロジェクトでは、BUSY 信号検出用端子に P00 を使用していますが、移植プロジェクトでは P41 を使用します。P41 を入力ポートに設定します。

注意. 未使用のポートは、端子処理などを適切に行い、電気的特性を満たすように設計してください。
また、未使用の入力専用ポートは個別に抵抗を介して VDD 又は VSS に接続してください。

図 3-13　ポート（移植例 1）



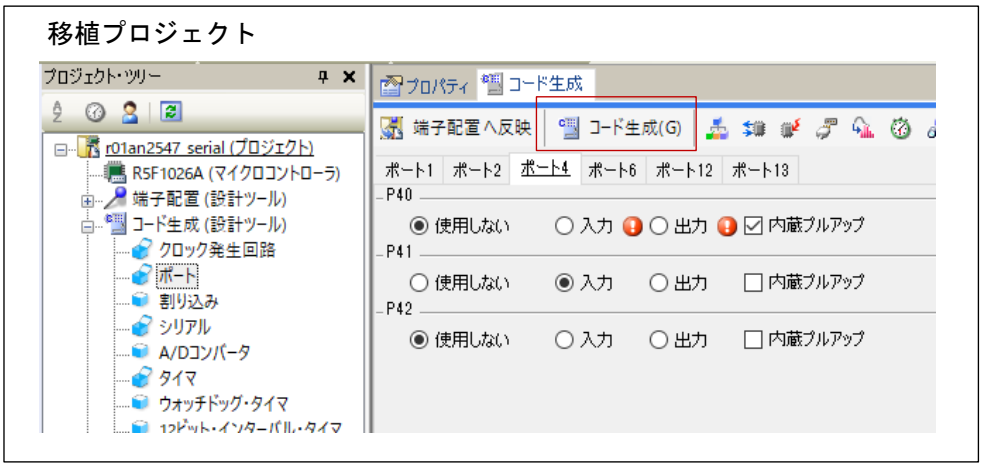
(6) 設定の保存

移植プロジェクトで、設定内容を保存します。CS+の「ファイル」－「プロジェクトの保存」メニューを選択します。

3.1.5　移植プロジェクトでのコード生成

移植プロジェクトで、コード生成ツール（CG）の設定に合わせてコードを生成します。「コード生成」をクリックします。

図 3-14　コード生成（移植例 1）



3.1.6　移植プロジェクトでのファイル編集

元プロジェクトで、CG 生成コードをコメントするなどの変更がされているかを確認し、変更箇所があれば移植プロジェクトに同様の変更をします。元プロジェクトと移植プロジェクトの同名ファイルをエディタで開き、コードについて差分を確認し元プロジェクトの削除・変更を移植プロジェクトに反映させます。

また、元プロジェクトでは BUSY 信号検出用端子に P00 を使用していますが、移植プロジェクトでは P41 に変更しているため、BUSY 信号検出用端子に対するコードを変更します。

図 3-15　確認対象ファイル（移植例 1）

備考. 移植プロジェクトでコード生成後に「コード生成」ノードが追加される場合がありますが、サンプルコードに影響ありません。

表 3-4　確認対象ファイルと編集の要否（移植例 1）

移植プロジェクト ファイル名	編集の要否 〇：必要 －：不要	差分内容
r_main.c	〇	BUSY 信号検出用端子操作のコード
r_systeminit.c	－	差分なし。コメント化などの変更なし。
r_cg_cgc.c	－	関数 R_CGC_Create 内のコード差分は、変更前と変更後のデバイス仕様の差分によるため、編集不要。
r_cg_cgc_user.c	－	差分なし。コメント化などの変更なし。
r_cg_port.c	－	関数 R_PORT_Create 内のコード差分は、BUSY 信号検出用端子の P00 から P41 への変更による差分のため、編集不要。 未使用ポートについての注意：実際に回路を作成される場合は、端子処理などを適切に行い、電気的特性を満たすように設計してください（入力専用ポートは個別に抵抗を介して VDD 又は VSS に接続して下さい）。
r_cg_port_user.c	－	差分なし。コメント化などの変更なし。
r_cg_serial.c	－	関数 R_CSI00_Create 内のコード差分は、変更前と変更後のデバイス仕様の差分によるため、編集不要。
r_cg_serial_user.c	－	差分なし。コメント化などの変更なし。
r_cg_microdriver.h	－	差分なし。コメント化などの変更なし。
r_cg_userdefine.h	－	差分なし。コメント化などの変更なし。
r_cg_cgc.h	－	define、type 定義の差分は、変更前と変更後のデバイス仕様の差分によるため、編集不要。
r_cg_port.h	－	define 定義の差分は、変更前と変更後のデバイス仕様の差分によるため、編集不要。
r_cg_serial.h	－	define 定義の差分は、変更前と変更後のデバイス仕様の差分によるため、編集不要。
r_cg_timer.c	－	関数 R_TAU0_Create 内のコード差分は、変更前と変更後のデバイス仕様の差分によるため、編集不要。
r_cg_timer_user.c	－	差分なし。コメント化などの変更なし。
r_cg_timer.h	－	define 定義の差分は、変更前と変更後のデバイス仕様の差分によるため、編集不要。

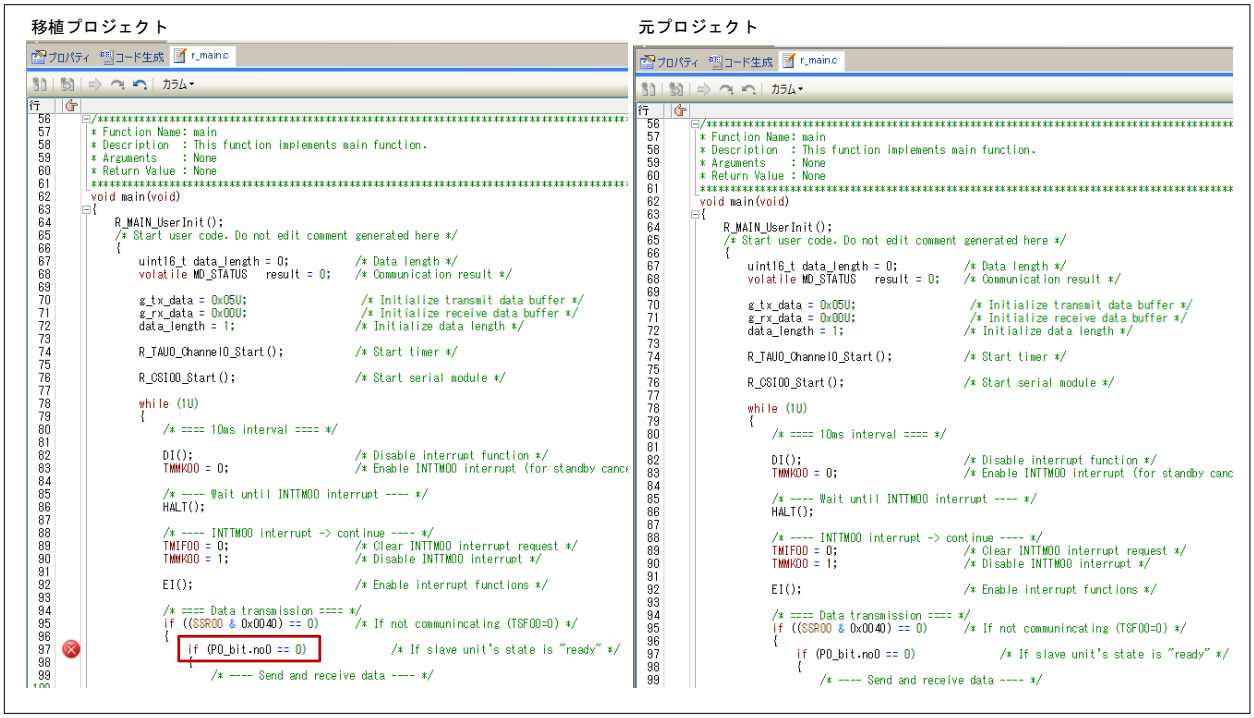
(a)　r_main.c

97 行目の BUSY 信号検出用端子の P00 に対するコードを P41 へのコードに変更します。

変更前：P0_bit.no0

変更後：P4_bit.no1

図 3-16　r_main.c（移植例 1）



移植プロジェクト：変更後

```
/* ==== Data transmission ==== */
if ((SSR00 & 0x0040) == 0)      /* If not communincating (TSF00=0) */
{
    if (P4_bit.no1 == 0)            /* If slave unit's state is "ready" */
    {
        /* ---- Send and receive data ---- */
```

3.1.7　移植プロジェクトのビルド

デバイス変更および元プロジェクトに合わせてコードを編集した移植プロジェクトをビルドします。

CS+の「ビルド」→「リビルド・プロジェクト」メニューを選択します。

3.2　移植例 2

アプリケーションノート「RL78/G13 シリアル・インタフェース IICA（マスタ送受信）CC-RL」R01AN2759JJ0201（以下、RL78/G13 用 AN）で説明されている RL78/G13 用サンプルコードを RL78/G14 用サンプルコードに移植する手順を示します。

3.2.1　事前準備

RL78/G13 用 AN の「2. 動作確認条件」、「表 1.1 使用する周辺機能と用途」、「4.2 使用端子一覧」、「4.1 ハードウェア構成例」および「5.5 関数一覧」を参照し使用しているリソースを確認します。

RL78/G14 へ移植時の構成例を以下に示します。

- 動作確認条件

差分はありません。動作周波数、動作電圧は変更せずに移植できます。

表 3-5　動作確認条件（移植例 2）

項目	RL78/G13	RL78/G14
使用マイコン	RL78/G13（R5F100LEA）	RL78/G14（R5F104MLA）
動作周波数	● 高速オンチップオシレータ（HOCO）クロック：32MHz ● CPU／周辺ハードウエア・クロック：32MHz	● 高速オンチップオシレータ（HOCO）クロック：32MHz ● CPU／周辺ハードウエア・クロック：32MHz
動作電圧	5.0V（2.7V～5.5V で動作可能） LVD 動作（V_{LVD}）：リセット・モード 2.81V (2.76V～2.87V)	5.0V（2.7V～5.5V で動作可能） LVD 動作（V_{LVD}）：リセット・モード 2.81V (2.76V～2.87V)

- 使用する周辺機能と用途

RL78/G13 用 AN で使用している周辺機能をそのまま RL78/G14 でも使用します。

表 3-6　使用する周辺機能（移植例 2）

RL78/G13

周辺機能	用途
シリアル・インタフェース IICA0	シングルマスタ・システムでのIIC 通信を行う。（SCLA0 端子と SDAA0 端子を使用）
12 ビット・インターバル・タイマ	1msインターバル計測
タイマ・アレイ・ユニット 0 チャネル 2	最大2msのインターバル

RL78/G14

周辺機能	用途
シリアル・インタフェース IICA0	シングルマスタ・システムでのIIC 通信を行う。（SCLA0 端子と SDAA0 端子を使用）
12 ビット・インターバル・タイマ	1msインターバル計測
タイマ・アレイ・ユニット 0 チャネル 2	最大2msのインターバル

- 使用端子

RL78/G13 用 AN で使用している端子をそのまま RL78/G14 で使用します。

表 3-7　使用端子（移植例 2）

RL78/G13

端子名	入出力	内容
P60/SCLA0	入出力	IICA0 のシリアル・クロック入出力端子
P61/SDAA0	入出力	IICA0 のシリアル・データ送受信用端子
P62	出力	ステータス表示 LED ドライブ信号
P63	出力	エラー表示 LED ドライブ信号
P137	入力	動作開始指定用 SW 入力信号

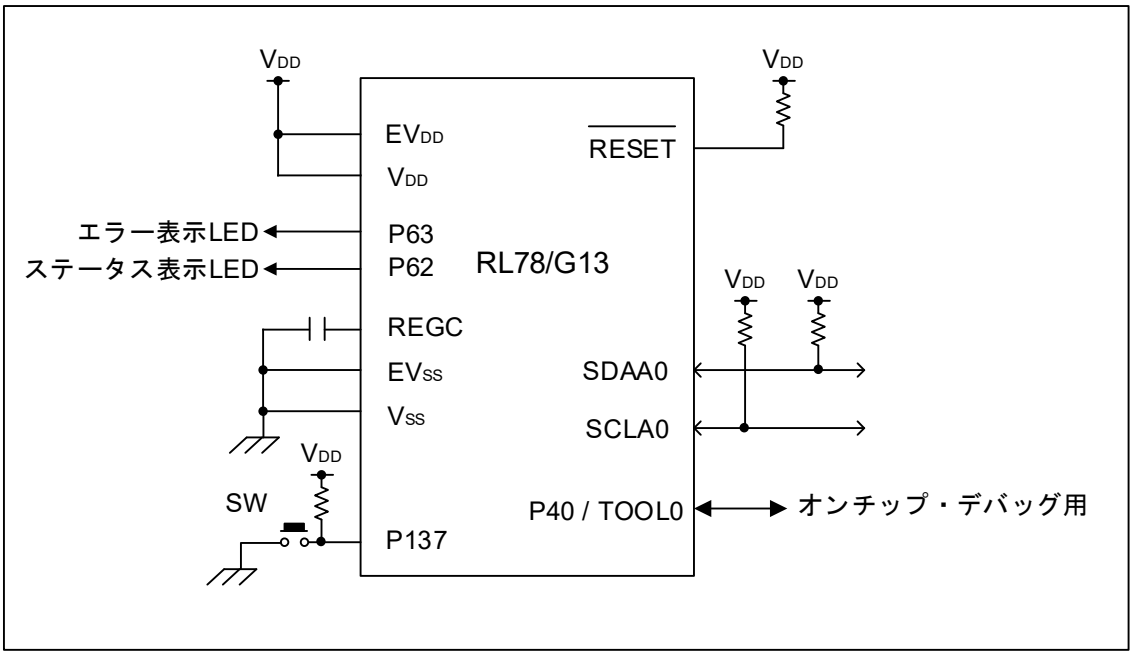
RL78/G14

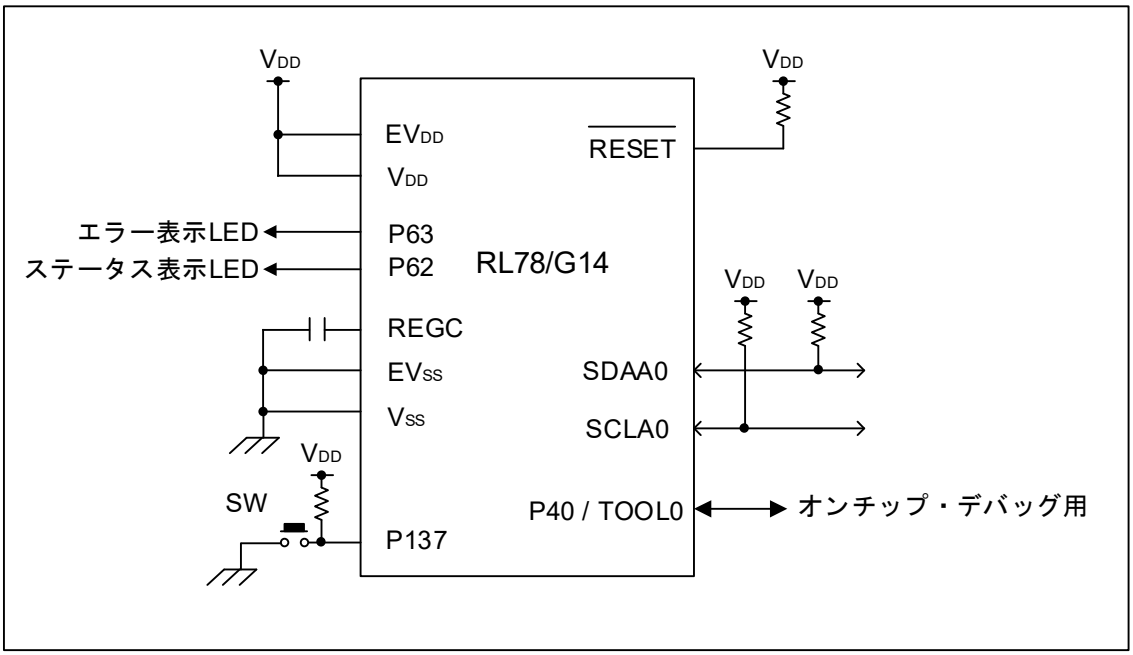
端子名	入出力	内容
P60/SCLA0	入出力	IICA0 のシリアル・クロック入出力端子
P61/SDAA0	入出力	IICA0 のシリアル・データ送受信用端子
P62	出力	ステータス表示 LED ドライブ信号
P63	出力	エラー表示 LED ドライブ信号
P137	入力	動作開始指定用 SW 入力信号

● ハードウェア構成

RL78/G13 用 AN で使用しているハードウェア構成をそのまま RL78/G14 でも使用します。

図 3-17　ハードウェア構成例（移植例 2）





注意 1　この回路イメージは接続の概要を示す為に簡略化しています。実際に回路を作成される場合は、端子処理などを適切に行い、電気的特性を満たすように設計してください（入力専用ポートは個別に抵抗を介して VDD 又は VSS に接続して下さい）。

2　EVSS で始まる名前の端子がある場合には VSS に、EVDD で始まる名前の端子がある場合には VDD にそれぞれ接続してください。

3　VDD は LVD にて設定したリセット解除電圧（VLVD）以上にしてください。

3.2.2　プロジェクトのコピー

「2.2 プロジェクトのコピー」を参照し、移植プロジェクトを作成します。

3.2.3　移植プロジェクトのデバイス変更

「2.3.2 e^2 studio の移植プロジェクト」を参照し、移植プロジェクトのデバイスを変更します。

3.2.4　移植プロジェクトでのコード生成設定

周辺機能設定に関わるコードを、移植先デバイス用の設定コードに自動で差し替えるために、元プロジェクトのコード生成ツール（CG）の設定に合わせて、移植プロジェクトで CG 設定をします。

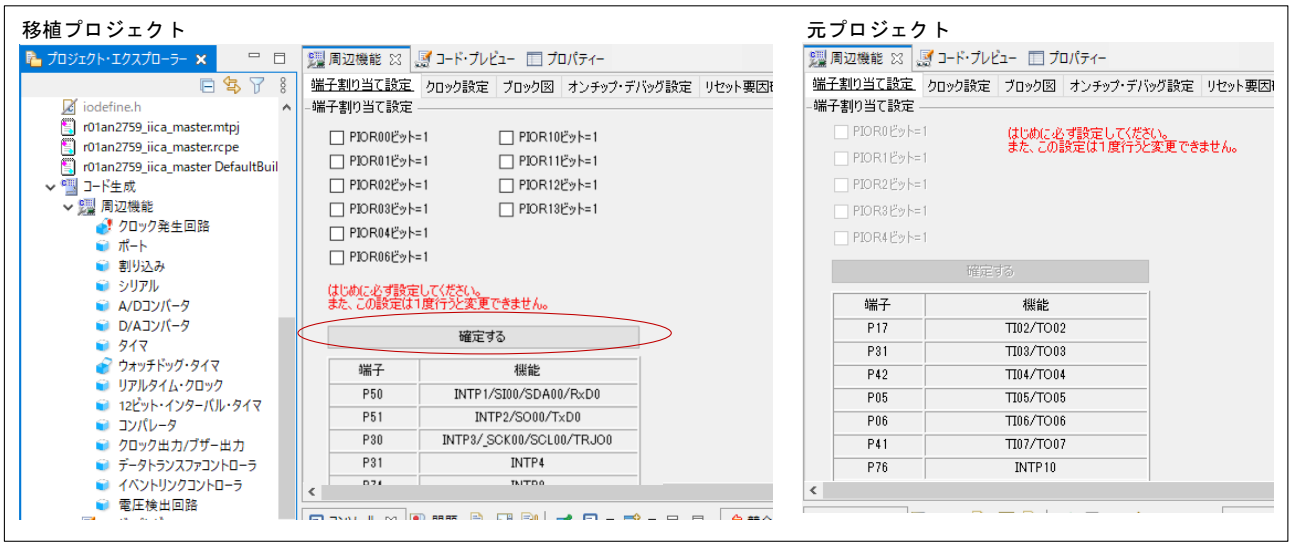
元プロジェクト用にもう一つの e^2 studio を起動し、移植プロジェクトと元プロジェクトを並べて作業します。

(1) クロック発生回路

- 端子割り当て設定

  今回の移植では、機能端子のリダイレクションは使用しないので、設定は何も変更せずに「確定する」をクリックします。

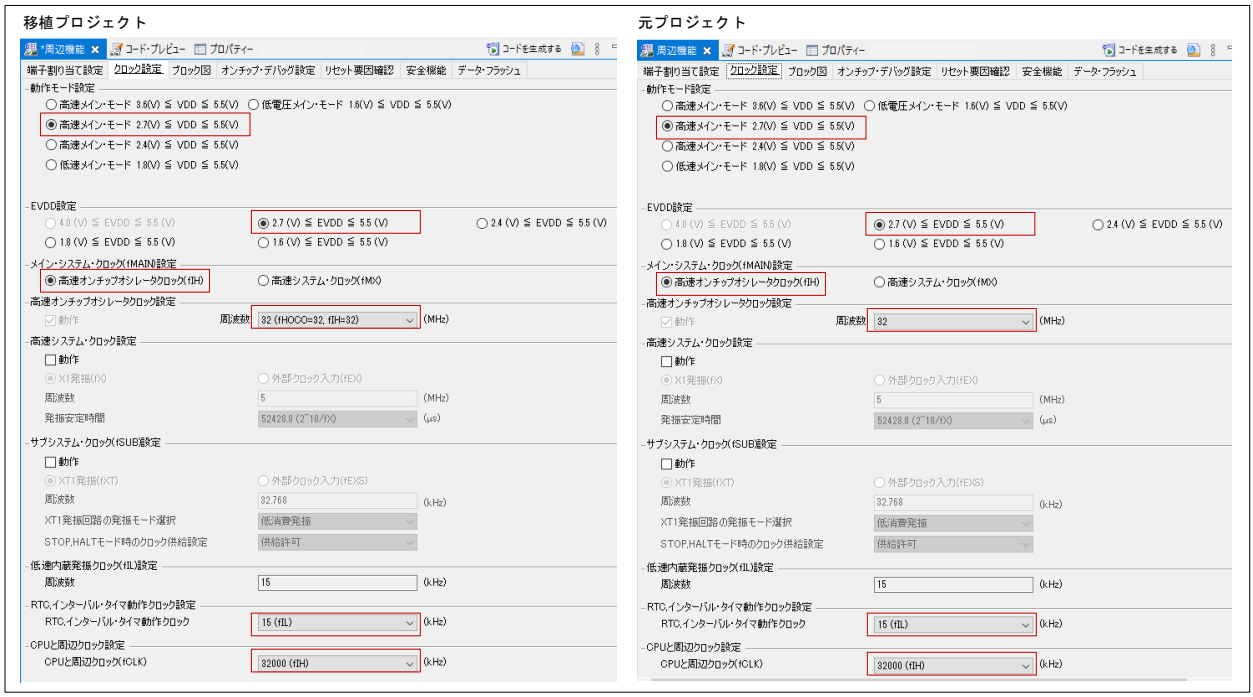
図 3-18　端子割り当て設定（移植例 2）



● クロック設定

元プロジェクトと同じ設定をします。

図 3-19　クロック設定（移植例 2）

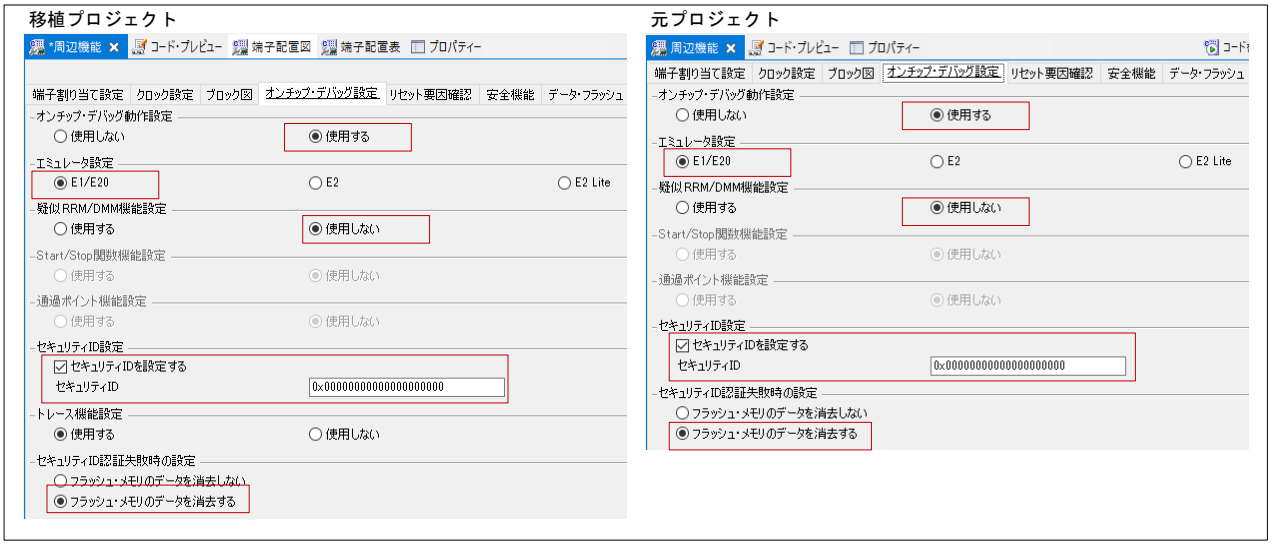


● オンチップ・デバッグ設定

元プロジェクトと同じ設定をします。移植プロジェクトのみにある項目は、初期設定のままにします。

備考．ご使用のエミュレータに合わせて、エミュレータ設定が変更可能です。

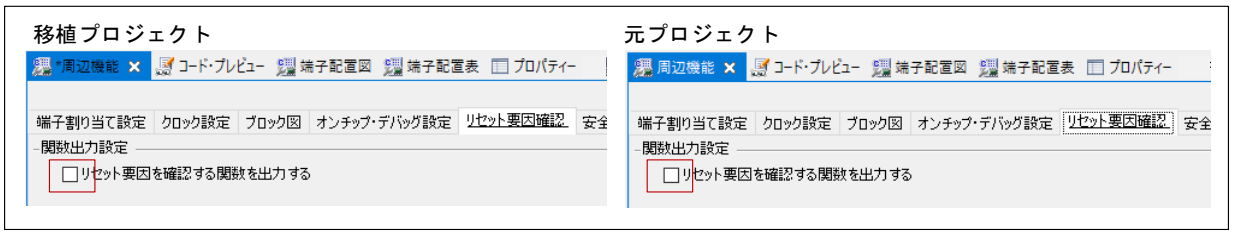
図 3-20　オンチップ・デバッグ設定（移植例 2）



- リセット要因確認

  元プロジェクトと同じ設定をします。

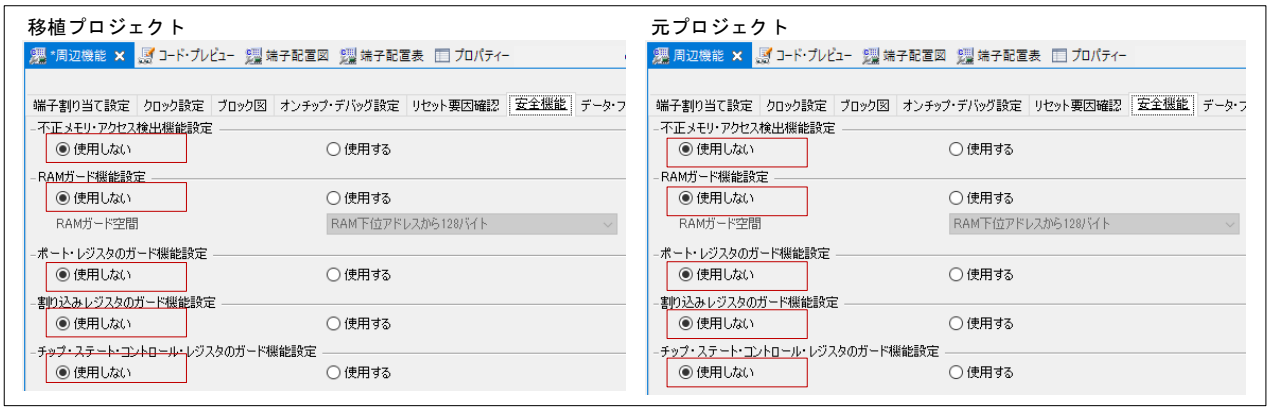
図 3-21　リセット要因確認（移植例 2）



- 安全機能

  元プロジェクトと同じ設定をします。

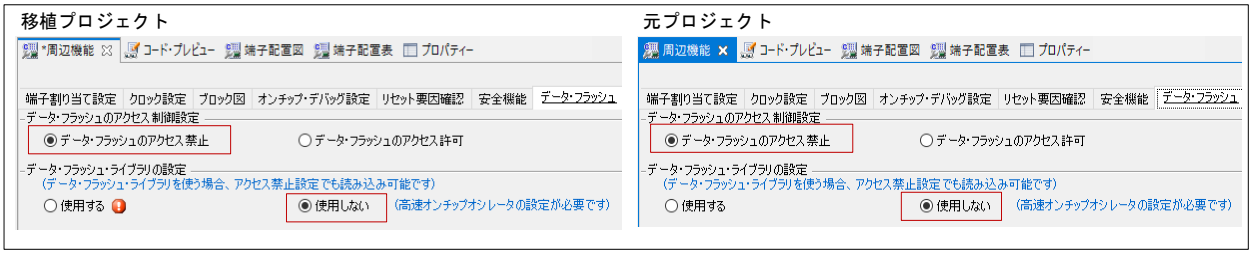
図 3-22　安全機能（移植例 2）



- データ・フラッシュ

  元プロジェクトと同じ設定をします。

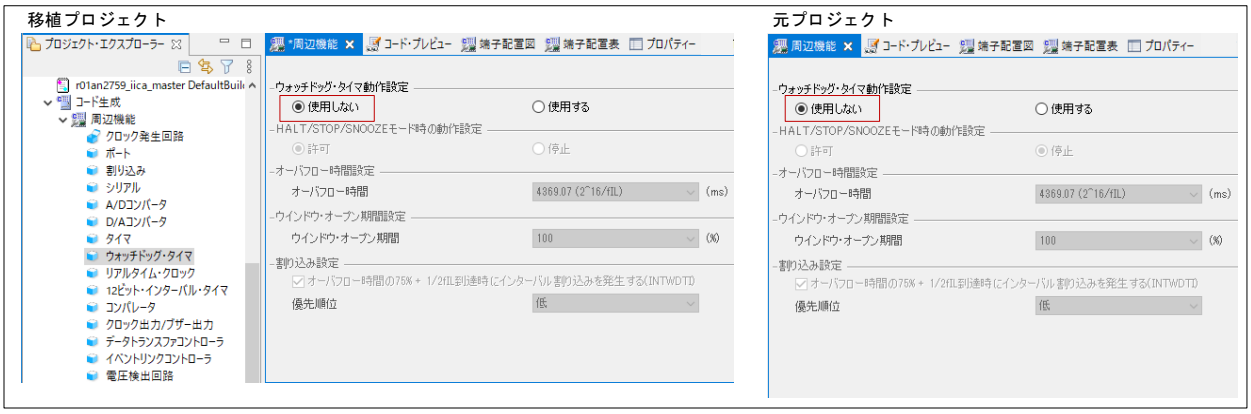
図 3-23　データ・フラッシュ（移植例 2）



(2) ウォッチドッグ・タイマ

元プロジェクトと同じ設定をします。

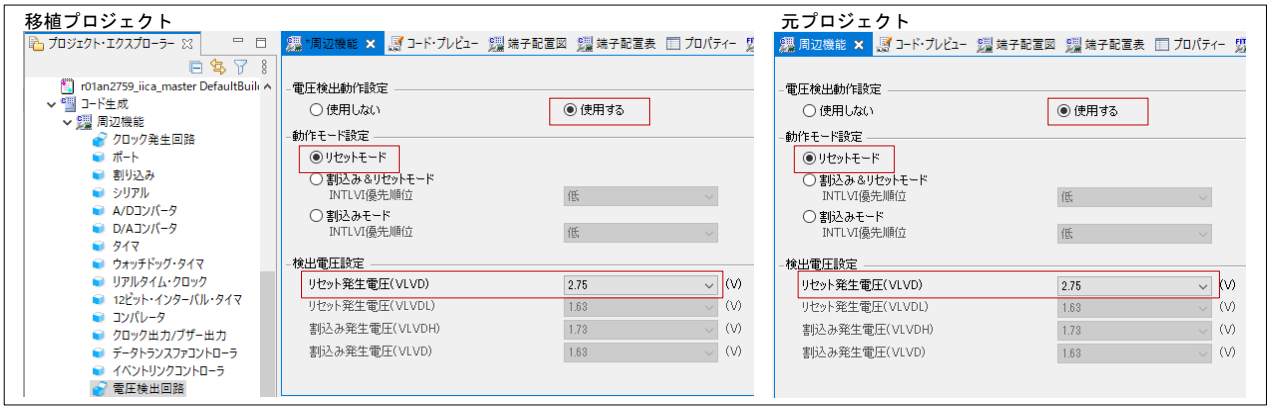
図 3-24　ウォッチドッグ・タイマ（移植例 2）



(3) 電圧検出回路

元プロジェクトと同じ設定をします。

図 3-25　電圧検出回路（移植例 2）



(4) サンプルコードで使用している周辺機能

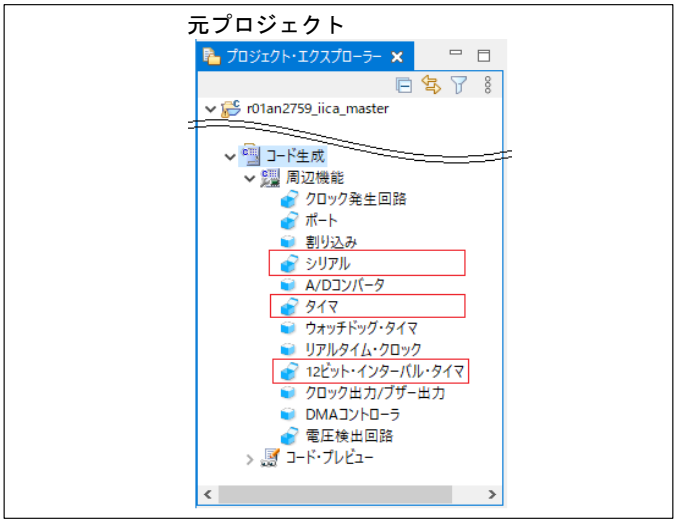
元プロジェクトの「コード生成（設計ツール）」に関連づけられているアイコンを確認し、設定のある周辺機能を元プロジェクトの設定に合わせて移植プロジェクトで設定します。

本 RL78/G13 用 AN では、「シリアル」、「タイマ」および「12 ビット・インターバル・タイマ」が設定されています。なお、「ポート」は(5)で設定します。

：周辺機能が設定されている

：周辺機能が設定されていない

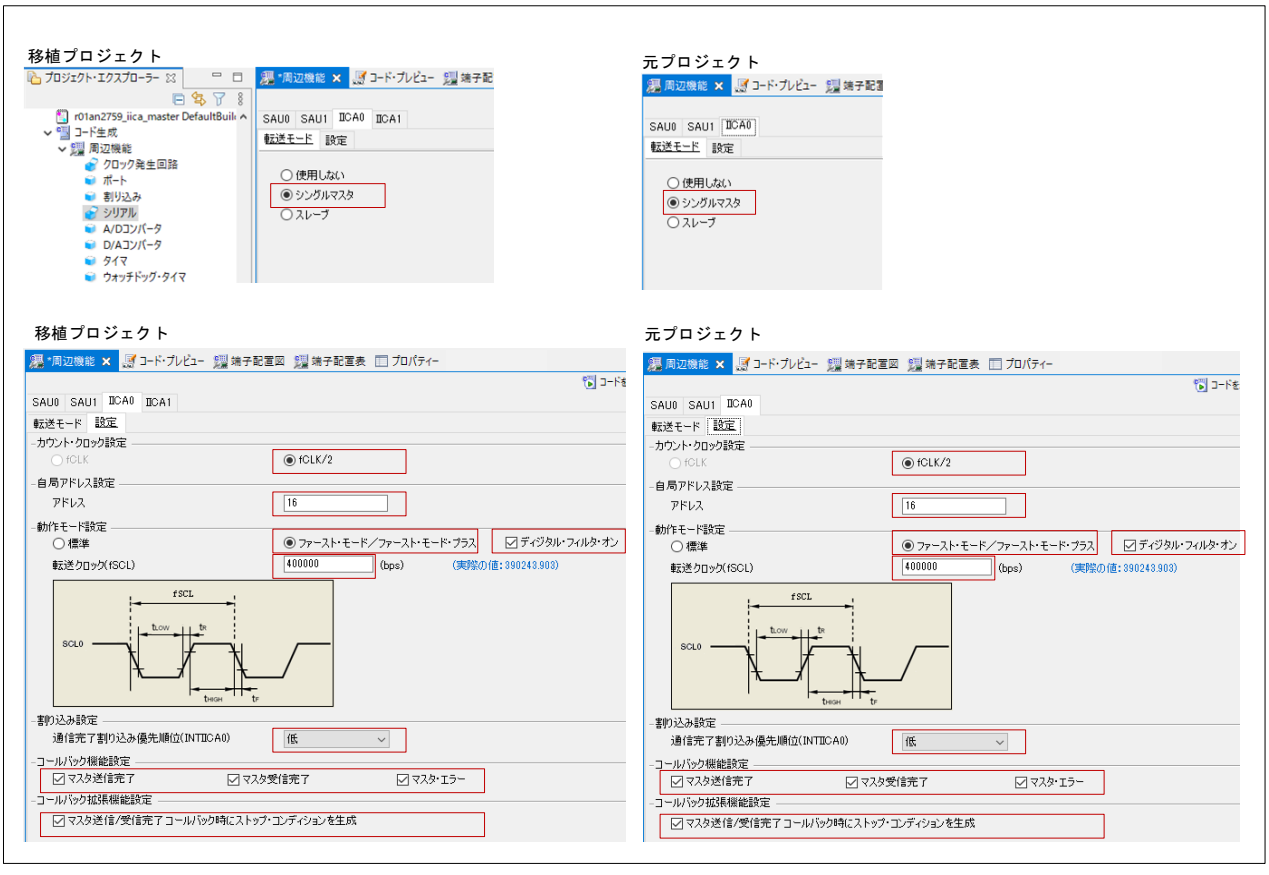
図 3-26　設定のある周辺機能（移植例 2）



- シリアル

  元プロジェクトと同じ設定をします。

図 3-27　シリアルーIICA0（移植例 2）

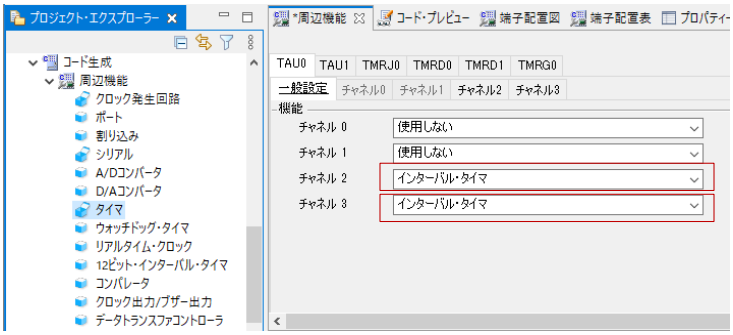


- タイマ

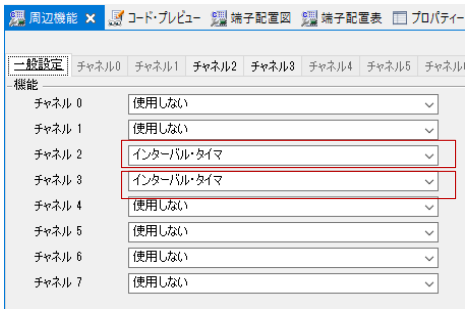
  元プロジェクトと同じ設定をします。

図 3-28　タイマーチャネル 2、チャネル 3（移植例 2）

移植プロジェクト

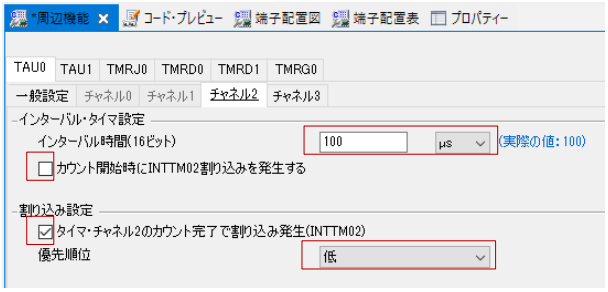


元プロジェクト

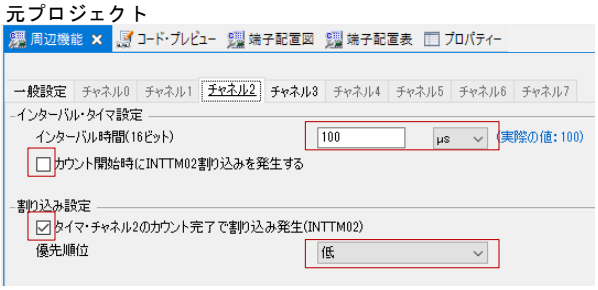


チャネル2

移植プロジェクト

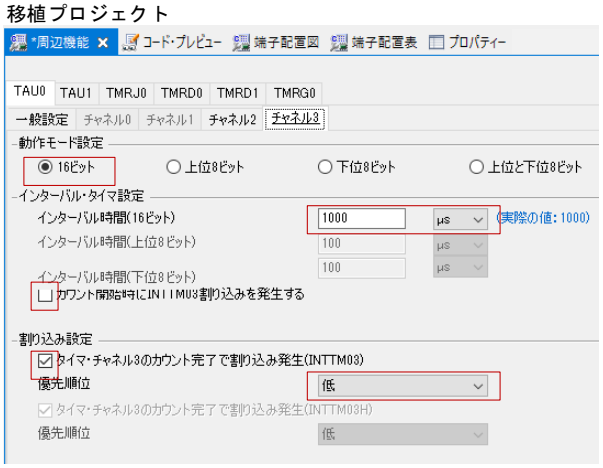


元プロジェクト

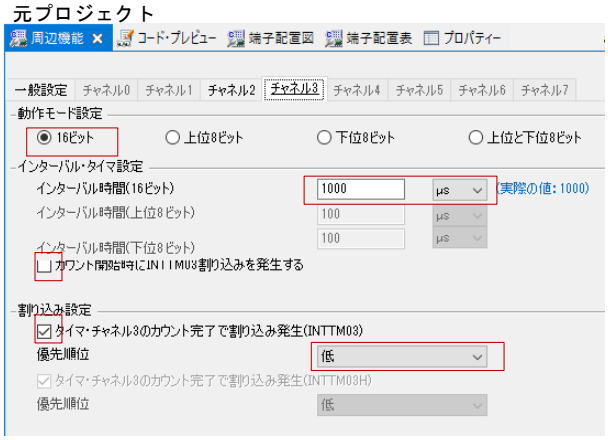


チャネル3

移植プロジェクト



元プロジェクト



- 12 ビット・インターバル・タイマ

  元プロジェクトと同じ設定をします。

- 図 3-29　12 ビット・インターバル・タイマ（移植例 2）

(5) ポート

元プロジェクトと同じ設定をします。

注意. 未使用のポートは、端子処理などを適切に行い、電気的特性を満たすように設計してください。
また、未使用の入力専用ポートは個別に抵抗を介して VDD 又は VSS に接続してください。

図 3-30　ポート（移植例 2）

(6) 設定の保存

移植プロジェクトで、設定内容を保存します。e^2 studio の「ファイル」－「すべて保管」メニューを選択します。

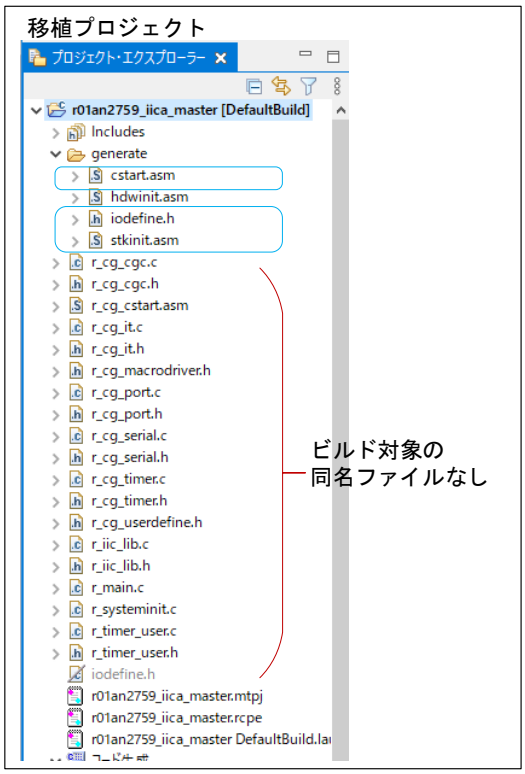
3.2.5　移植プロジェクトでのコード生成

コード生成前に、「generate」フォルダ内のファイルと同名ファイルをビルド対象外にします。またコード生成（CG）生成ファイルを「src」フォルダ内に移動します。
その後、移植プロジェクトで、コード生成ツール（CG）の設定に合わせてコードを生成します。

(1)　「generate」フォルダ内のファイルと同名ファイルの確認

元プロジェクトから引き継いだビルド対象ファイルに、「generate」フォルダ内のファイルと同名のファイルはないため、ビルド対象外にするファイルはありません。

図 3-31　同名ファイルの確認（移植例 2）



(2)　CG 生成ファイルを「src」フォルダ内へ移動

表 1-3　CG 生成ファイルおよび関数の概要 に示す CG 生成ファイルが「src」フォルダ内に登録されているか確認します。
元プロジェクトでは「src」ファイルが存在しないため、「src」ファイルを新規に作成します。プロジェクト名を右クリックし、「新規」－「フォルダー」メニューをクリックし、新規フォルダーダイアログで「フォルダ名」に「src」を入力し、「終了」をクリックします。
「src」フォルダ作成後、CG 生成ファイルをドラッグ＆ドロップで「src」フォルダ内に移動します。

図 3-32　「src」フォルダの作成（移植例 2）

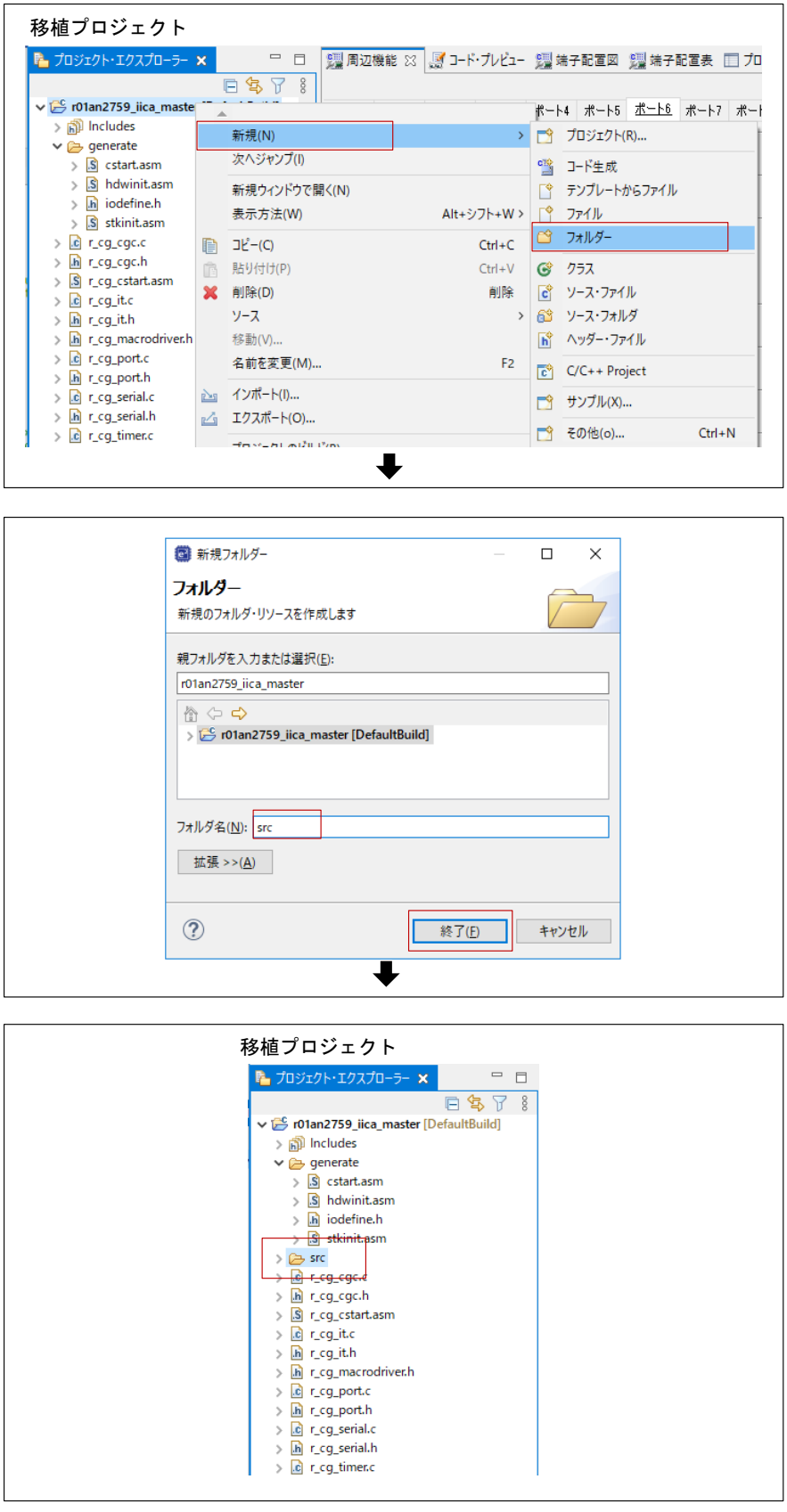
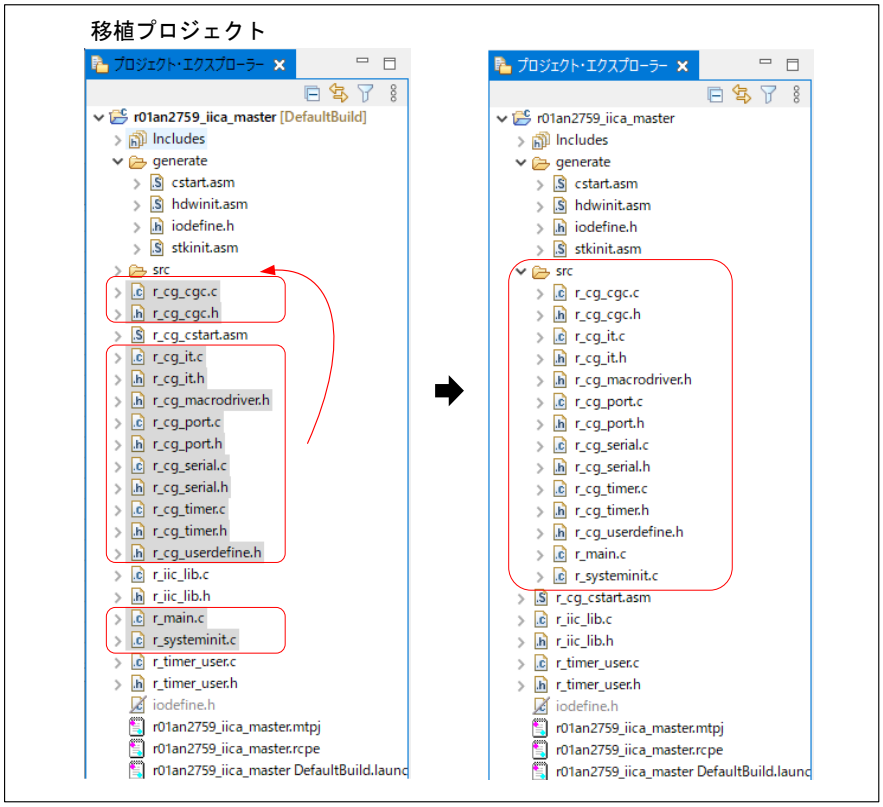


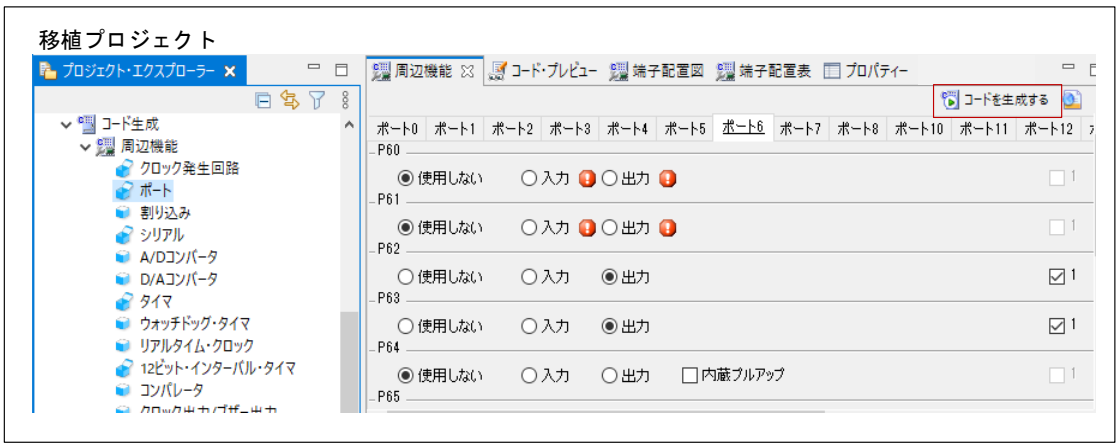
図 3-33　CG 生成ファイルを「src」フォルダ内へ移動（移植例 2）



(3)　コードを生成

移植プロジェクトで、コード生成ツール（CG）の設定に合わせてコードを生成します。「コードを生成する」をクリックします。

図 3-34　コード生成（移植例 2）



3.2.6　移植プロジェクトでのファイル編集

元プロジェクトで、CG 生成コードをコメントするなどの変更がされているかを確認し、変更箇所があれば移植プロジェクトに同様の変更をします。元プロジェクトと移植プロジェクトの同名ファイルをエディタで開き、コードについて差分を確認し元プロジェクトの削除・変更を移植プロジェクトに反映させます。

図 3-35　確認対象ファイル（移植例 2）

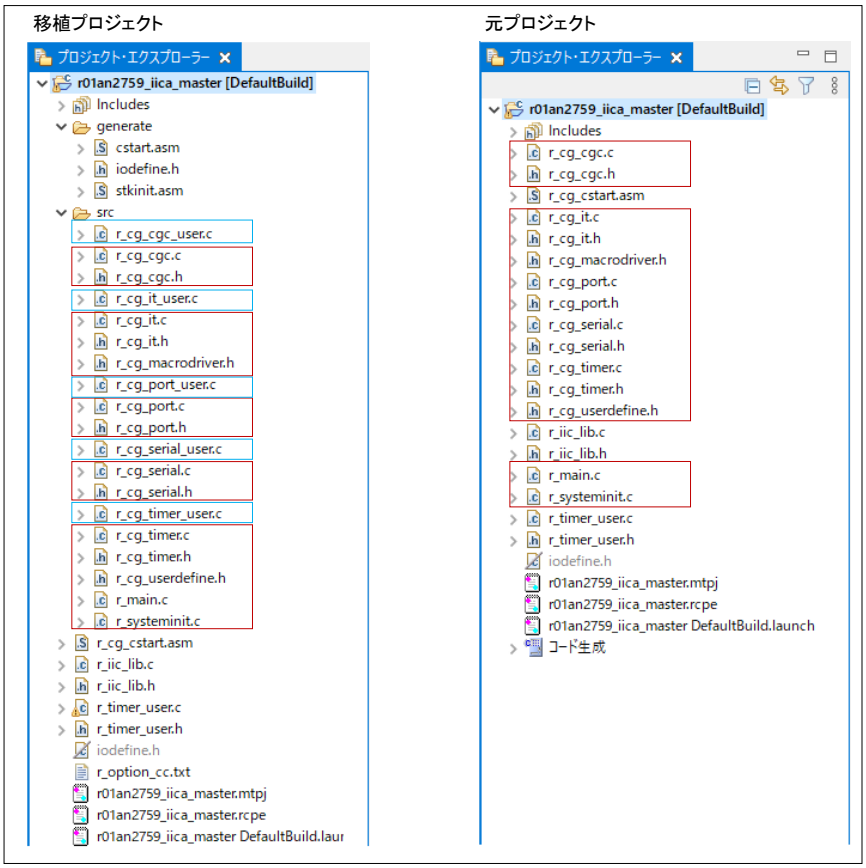


表 3-8　確認対象ファイルと編集の要否（移植例 2）（1/2）

移植プロジェクト ファイル名	編集の要否 ○：必要 －：不要	差分内容
r_cg_cgc_user.c	ビルド対象外にする	移植プロジェクトの CG で生成されたが、元プロジェクトで使用されていないファイルのためビルド対象外。
r_cg_cgc.c	－	差分なし。コメント化などの変更なし。
r_cg_cgc.h	－	差分なし。コメント化などの変更なし。
r_cg_it_user.c	ビルド対象外にする	移植プロジェクトの CG で生成されたが、元プロジェクトで使用されていないファイルのためビルド対象外。
r_cg_it.c	○	元プロジェクトでは削除されている関数を削除。
r_cg_it.h	○	元プロジェクトでは削除されている関数のプロトタイプ宣言を削除。
r_cg_microdriver.h	－	差分なし。コメント化などの変更なし。
r_cg_port_user.c	ビルド対象外にする	移植プロジェクトの CG で生成されたが、元プロジェクトで使用されていないファイルのためビルド対象外。

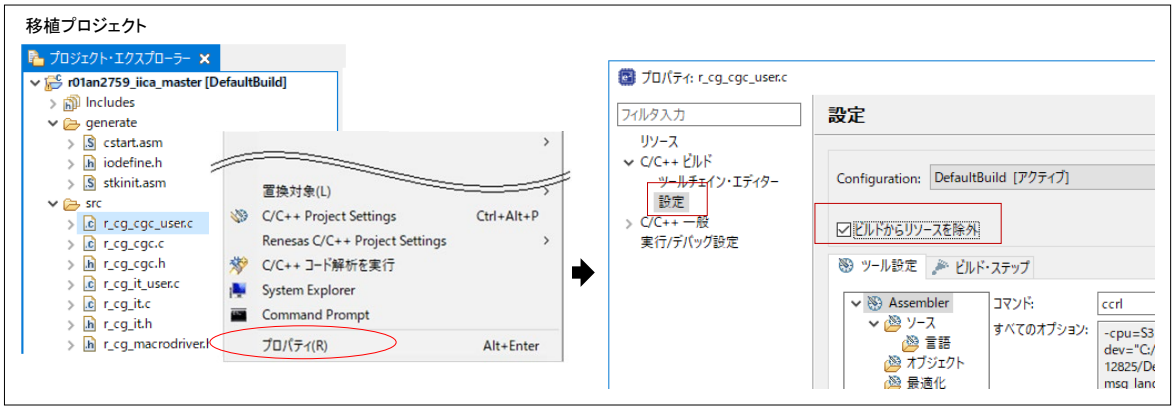
表 3-9　確認対象ファイルと編集の要否（移植例 2）（2/2）

移植プロジェクト ファイル名	編集の要否 ○：必要 －：不要	差分内容
r_cg_port.c	－	関数 R_PORT_Create 内のコード差分は、変更前と変更後のデバイス仕様の差分によるため、編集不要。 未使用ポートについての注意：実際に回路を作成される場合は、端子処理などを適切に行い、電気的特性を満たすように設計してください（入力専用ポートは個別に抵抗を介して VDD 又は VSS に接続して下さい）。
r_cg_port.h	－	define 定義の差分は、変更前と変更後のデバイス仕様の差分によるため、編集不要。
r_cg_serial_user.c	ビルド対象外にする	移植プロジェクトの CG で生成されたが、元プロジェクトで使用されていないファイルのためビルド対象外。
r_cg_serial.c	○	元プロジェクトでは削除されている変数、関数を削除。 また、生成コードを編集。
r_cg_serial.h	○	元プロジェクトでは削除されている関数のプロトタイプ宣言を削除。また、define 定義の変更。
r_cg_timer_user.c	ビルド対象外にする	移植プロジェクトの CG で生成されたが、元プロジェクトで使用されていないファイルのためビルド対象外。
r_cg_timer.c	○	関数 R_TAU0_Create 内のコード差分は、変更前と変更後のデバイス仕様の差分によるため、編集不要。 元プロジェクトでは削除されている関数を削除。
r_cg_timer.h	○	元プロジェクトでは削除されている関数のプロトタイプ宣言を削除。 define 定義の差分は、変更前と変更後のデバイス仕様の差分によるため、編集不要。
r_cg_userdefine.h	－	差分なし。コメント化などの変更なし。
r_main.c	－	差分なし。コメント化などの変更なし。
r_systeminit.c	－	関数 R_Systeminit 内のコード差分は、変更前と変更後のデバイス仕様の差分によるため、編集不要。

(a)　r_cg_cgc_user.c、r_cg_it_user.c、r_cg_port_user.c、r_cg_serial_user.c、r_cg_timer_user.c

ファイルを右クリックし、「プロパティ」メニューをクリックします。プロパティ・ダイアログの「C/C++ ビルド」－「設定」で、「ビルドからリソースを除外」にチェックし、「適用して閉じる」をクリックします。

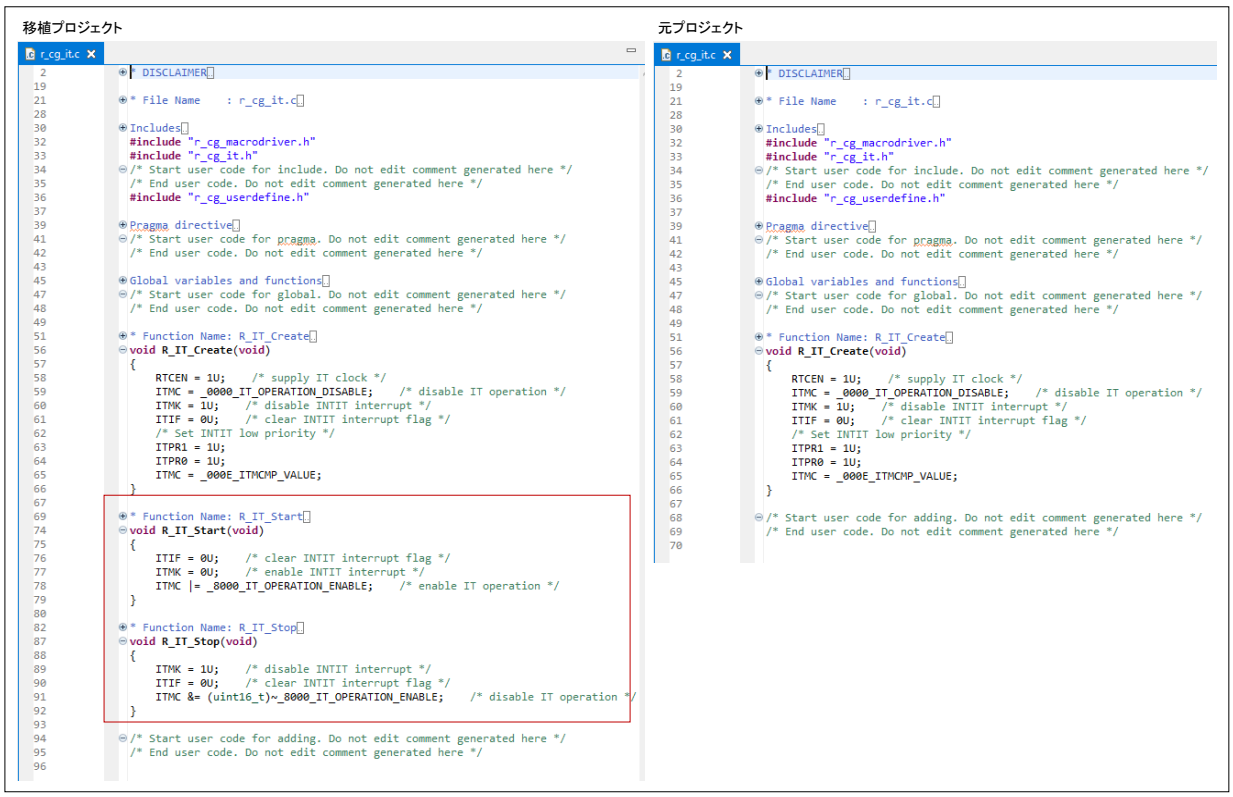
図 3-36　r_cg_cgc_user.c（移植例 2）



(b)　r_cg_it.c

関数 R_IT_Start、R_IT_Stop を削除します。

図 3-37　r_cg_it.c（移植例 2）



(c)　r_cg_it.h

r_cg_it.c 内で削除した関数 R_IT_Start、R_IT_Stop のプロトタイプ宣言を削除します。

図 3-38　r_cg_it.h（移植例 2）

移植プロジェクト

r_cg_it.h

```
 2  * DISCLAIMER
19
21  * File Name    : r_cg_it.h
28
29  #ifndef IT_H
30  #define IT_H
31
33  Macro definitions (Register bit)
35  /*
36      Interval timer control register (ITMC)
37  */
38  /* Interval timer operation enable/disable specification (RINTE) */
39  #define _0000_IT_OPERATION_DISABLE     (0x0000U)    /* disable interv
40  #define _8000_IT_OPERATION_ENABLE      (0x8000U)    /* enable interva
41
43  Macro definitions
45  /* Interval timer compare value (ITMCMP11 - 0) */
46  #define _000E_ITMCMP_VALUE             (0x000EU)
47
49  Typedef definitions
51
53  Global functions
55  void R_IT_Create(void);
56  void R_IT_Start(void);
57  void R_IT_Stop(void);
58
59  /* Start user code for function. Do not edit comment generated here
60  /* End user code. Do not edit comment generated here */
61  #endif
62
```

元プロジェクト

r_cg_it.h

```
 2  * DISCLAIMER
19
21  * File Name    : r_cg_it.h
28
29  #ifndef IT_H
30  #define IT_H
31
33  Macro definitions (Register bit)
35  /*
36      Interval timer control register (ITMC)
37  */
38  /* Interval timer operation enable/disable specification (RINTE) */
39  #define _0000_IT_OPERATION_DISABLE     (0x0000U)    /* disable interv
40  #define _8000_IT_OPERATION_ENABLE      (0x8000U)    /* enable interva
41
43  Macro definitions
45  /* Interval timer compare value (ITMCMP11 - 0) */
46  #define _000E_ITMCMP_VALUE             (0x000EU)
47
49  Typedef definitions
51
53  Global functions
55  void R_IT_Create(void);
56
57  /* Start user code for function. Do not edit comment generated here
58  /* End user code. Do not edit comment generated here */
59  #endif
60
```

(d)　r_cg_serial.c

変数をすべて削除します。（図 3-39）

関数 R_IICA0_Stop、 R_IICA0_StopCondtition、 R_IICA0_Master_Send、 R_IICA0_Master_Receive を削除します。（図 3-40）

関数 R_IICA0_Create のコードを元プロジェクトに合わせて編集します。（図 3-41）

- SCLA0、SDAA0 端子の設定コード
- SVA0 レジスタへの設定値（スレーブとして使用する場合の自局アドレス）

図 3-39　r_cg_serial.c ー変数削除（移植例 2）

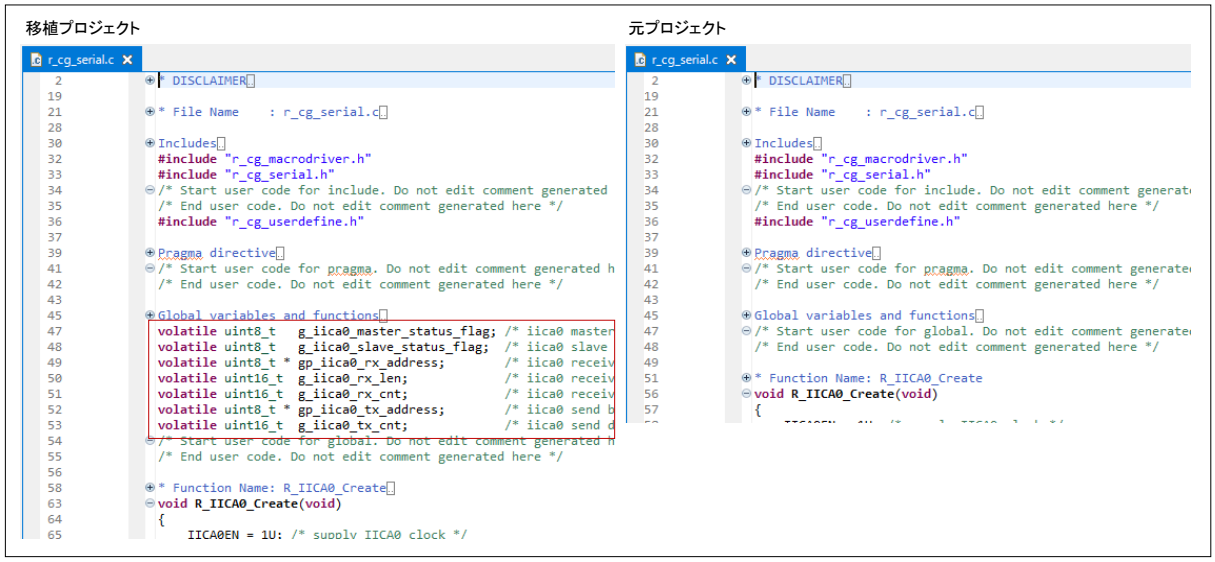


図 3-40　r_cg_serial.c ー関数削除（移植例 2）

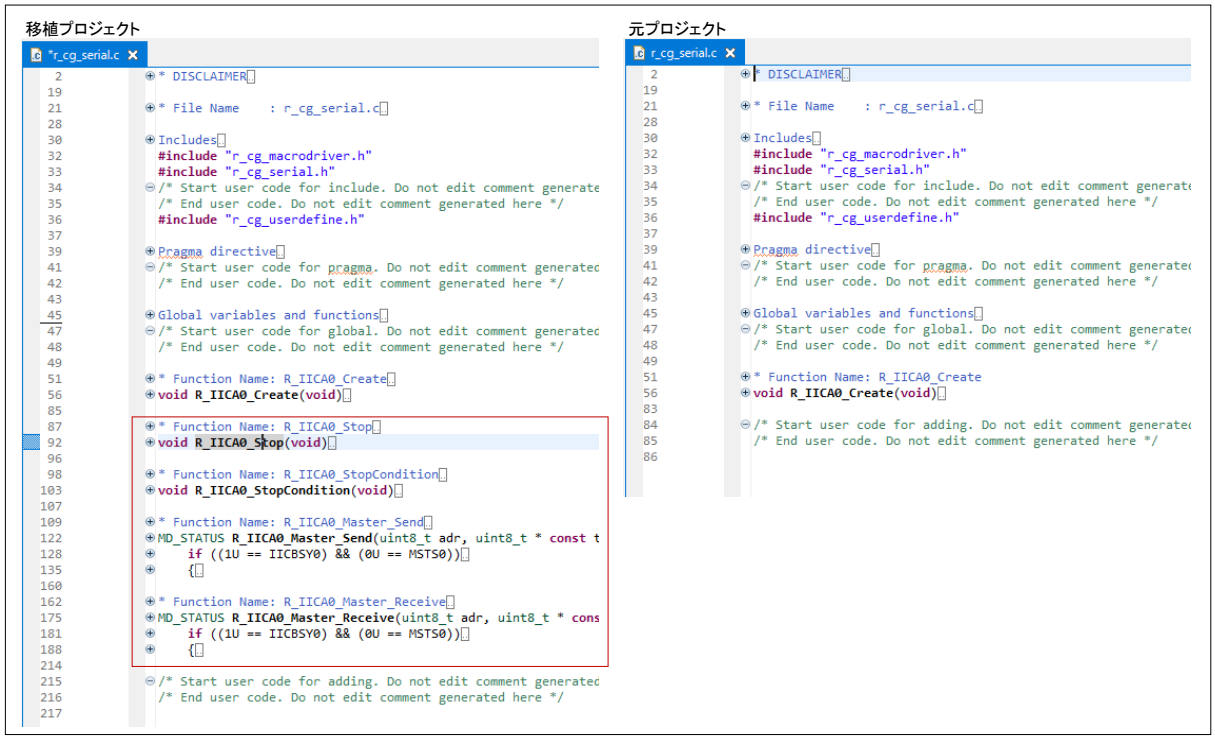


図 3-41　r_cg_serial.c ーコード編集（移植例 2）

移植プロジェクト

```
51      * Function Name: R_IICA0_Create
56      void R_IICA0_Create(void)
57      {
58          IICA0EN = 1U; /* supply IICA0 clock */
59          IICE0 = 0U; /* disable IICA0 operation */
60          IICAMK0 = 1U; /* disable INTIICA0 interrupt */
61          IICAIF0 = 0U; /* clear INTIICA0 interrupt flag */
62          /* Set INTIICA0 low priority */
63          IICAPR10 = 1U;
64          IICAPR00 = 1U;
65          /* Set SCLA0, SDAA0 pin */
66          P6 &= 0xFCU;
67          PM6 |= 0x03U;
68          SMC0 = 1U;
69          IICWL0 = _15_IICA0_IICWL_VALUE;
70          IICWH0 = _14_IICA0_IICWH_VALUE;
71          DFC0 = 1U; /* digital filter on */
72          IICCTL01 |= _01_IICA_fCLK_HALF;
73          SVA0 = _10_IICA0_MASTERADDRESS;
74          STCEN0 = 1U;
75          IICRSV0 = 1U;
76          SPIE0 = 0U;
77          WTIM0 = 1U;
78          ACKE0 = 1U;
79          IICAMK0 = 0U;
80          IICE0 = 1U;
81          LREL0 = 1U;
82          /* Set SCLA0, SDAA0 pin */
83          PM6 &= 0xFCU;
84      }
85
```

元プロジェクト

```
51      * Function Name: R_IICA0_Create
56      void R_IICA0_Create(void)
57      {
58          IICA0EN = 1U; /* supply IICA0 clock */
59          IICE0 = 0U; /* disable IICA0 operation */
60          IICAMK0 = 1U; /* disable INTIICA0 interrupt */
61          IICAIF0 = 0U; /* clear INTIICA0 interrupt flag */
62          /* Set INTIICA0 low priority */
63          IICAPR10 = 1U;
64          IICAPR00 = 1U;
65          SMC0 = 1U;
66          IICWL0 = _15_IICA0_IICWL_VALUE;
67          IICWH0 = _14_IICA0_IICWH_VALUE;
68          DFC0 = 1U; /* digital filter on */
69          IICCTL01 |= _01_IICA_fCLK_HALF;
70          SVA0 = _50_IICA0_MASTERADDRESS;
71          STCEN0 = 1U;
72          IICRSV0 = 1U;
73          SPIE0 = 0U;
74          WTIM0 = 1U;
75          ACKE0 = 1U;
76          IICAMK0 = 0U;
77          IICE0 = 1U;
78          LREL0 = 1U;
79          /* Set SCLA0, SDAA0 pin */
80          P6 &= 0xFCU;
81          PM6 &= 0xFCU;
82      }
83
84      /* Start user code for adding. Do not edit comment gene
85      /* End user code. Do not edit comment generated here */
```

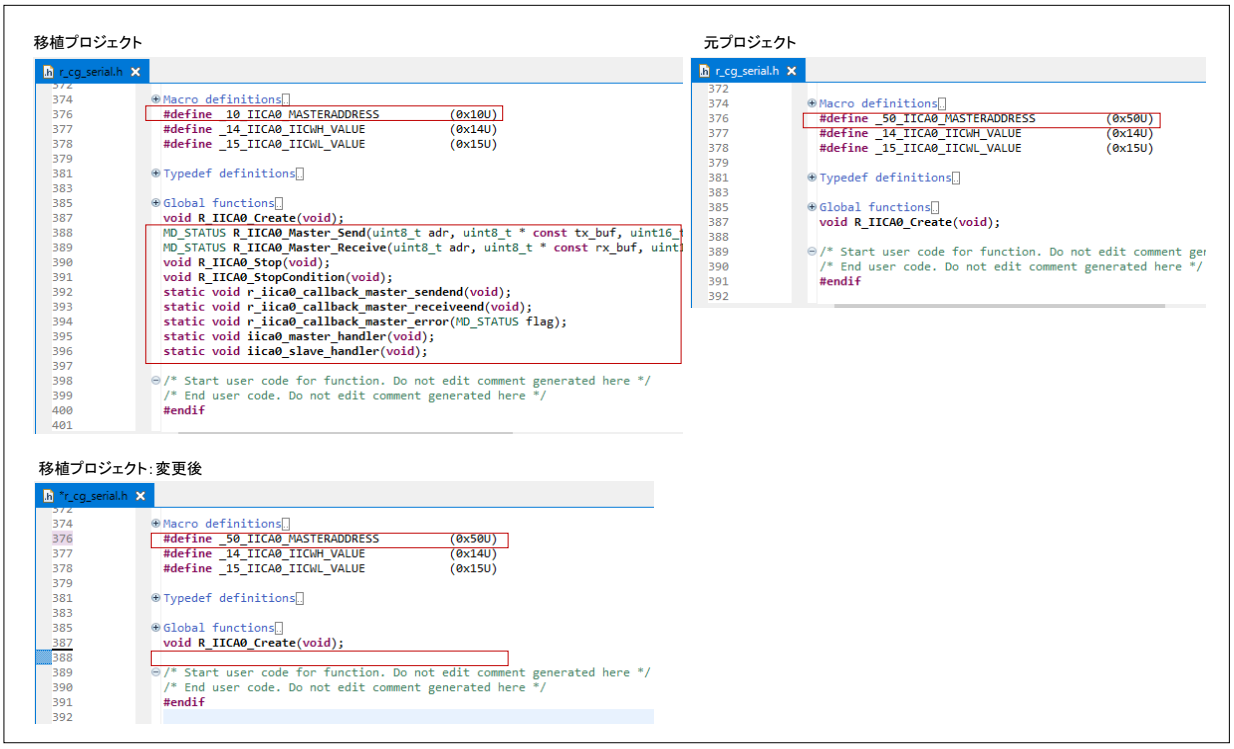
移植プロジェクト:変更後

```
51      * Function Name: R_IICA0_Create
56      void R_IICA0_Create(void)
57      {
58          IICA0EN = 1U; /* supply IICA0 clock */
59          IICE0 = 0U; /* disable IICA0 operation */
60          IICAMK0 = 1U; /* disable INTIICA0 interrupt */
61          IICAIF0 = 0U; /* clear INTIICA0 interrupt flag */
62          /* Set INTIICA0 low priority */
63          IICAPR10 = 1U;
64          IICAPR00 = 1U;
65          SMC0 = 1U;
66          IICWL0 = _15_IICA0_IICWL_VALUE;
67          IICWH0 = _14_IICA0_IICWH_VALUE;
68          DFC0 = 1U; /* digital filter on */
69          IICCTL01 |= _01_IICA_fCLK_HALF;
70          SVA0 = _50_IICA0_MASTERADDRESS;
71          STCEN0 = 1U;
72          IICRSV0 = 1U;
73          SPIE0 = 0U;
74          WTIM0 = 1U;
75          ACKE0 = 1U;
76          IICAMK0 = 0U;
77          IICE0 = 1U;
78          LREL0 = 1U;
79          /* Set SCLA0, SDAA0 pin */
80          P6 &= 0xFCU;
81          PM6 &= 0xFCU;
82      }
```

(e)　r_cg_serial.h

- define 定義を変更します。
- r_cg_serial.c 内で削除した関数 R_IICA0_Stop、 R_IICA0_StopCondtition、 R_IICA0_Master_Send、 R_IICA0_Master_Receive のプロトタイプ宣言を削除します。

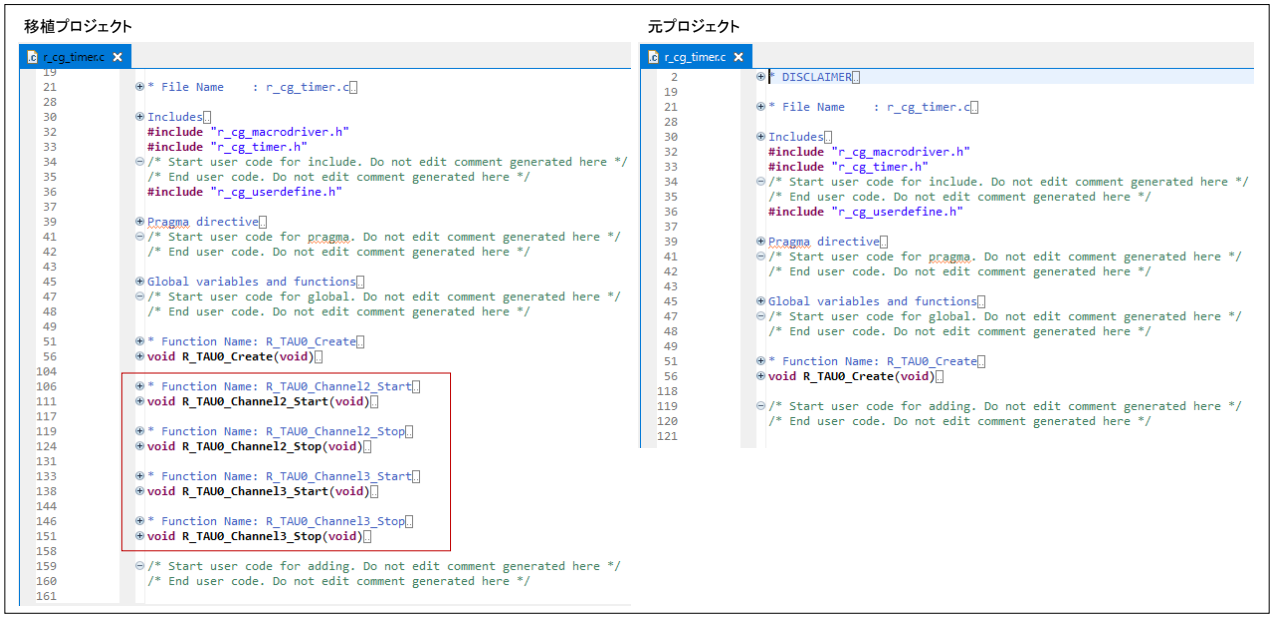
図 3-42　r_cg_serial.h（移植例 2）



(f)　r_cg_timer.c

関数 R_TAU0_Channel2_Strat、R_TAU0_Channel2_Stop、R_TAU0_Channel3_Strat、R_TAU0_Channel3_Stop を削除します。

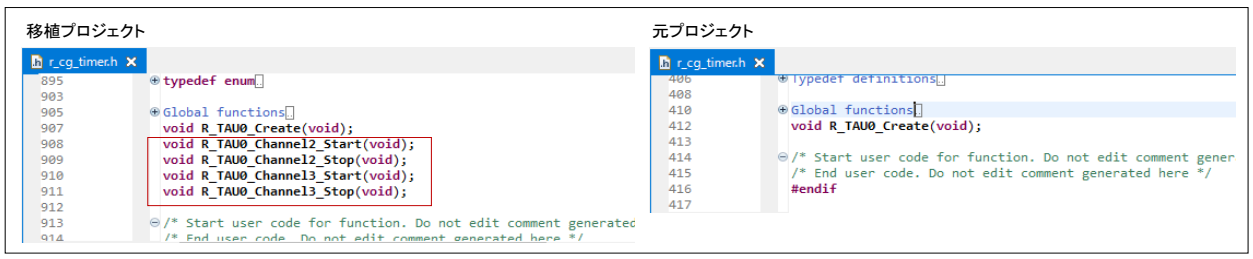
図 3-43　r_cg_timer.c（移植例 2）



(g)　r_cg_timer.h

r_cg_timer.c 内で削除した関数関数 R_TAU0_Channel2_Strat、R_TAU0_Channel2_Stop、R_TAU0_Channel3_Strat、R_TAU0_Channel3_Stop のプロトタイプ宣言を削除します。

図 3-44　r_cg_timer.h（移植例 2）



移植プロジェクトで、編集内容を保存します。e^2 studio の「ファイル」－「すべて保管」メニューを選択します。

3.2.7　移植プロジェクトのビルド

デバイス変更および元プロジェクトに合わせてコードを編集した移植プロジェクトをビルドします。

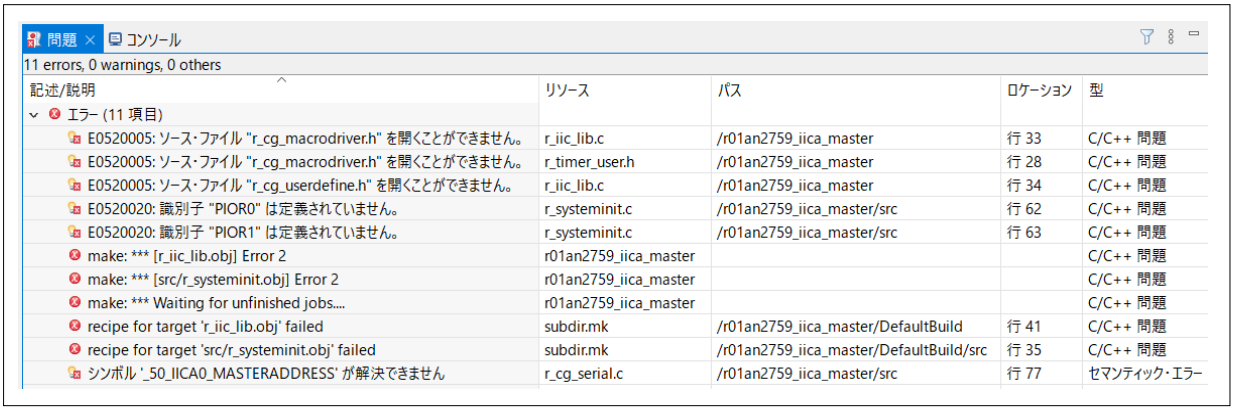
e² studio の「プロジェクト」→「プロジェクトのビルド」メニューを選択します。

本移植例では、ビルド時にエラーが発生するので、エラーメッセージをもとにエラーをとるための対応をします。

(1)　エラー処置 1

「src」フォルダにパスが設定されていないためエラーが発生しています。コンパイラ・オプションで「src」フォルダをパスに追加します。

図 3-45　ビルド・エラー1（移植例 2）

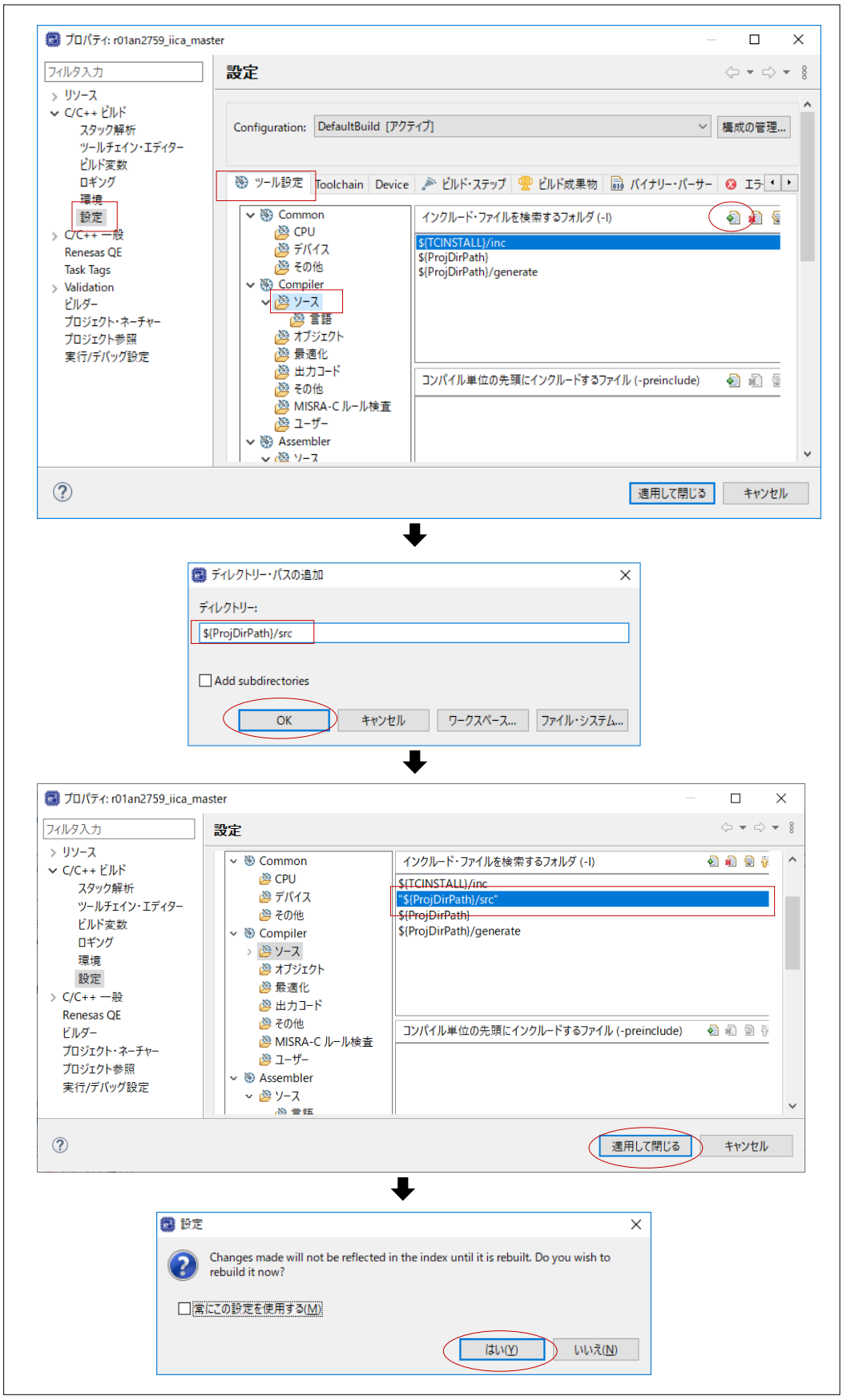


- インクルード・パスの追加

  e² studio の「プロジェクト」→「C/C++ Project Settings」メニューを選択します。プロパティ・ダイアログで「C/C++ビルド」－「設定」を選択し、「ツール設定」タブの「ソース」でパスを追加します。パスを追加後、「プロジェクト」→「プロジェクトのビルド」メニューを選択し、再度ビルドします。

  追加するパス：${ProjDirPath}/src

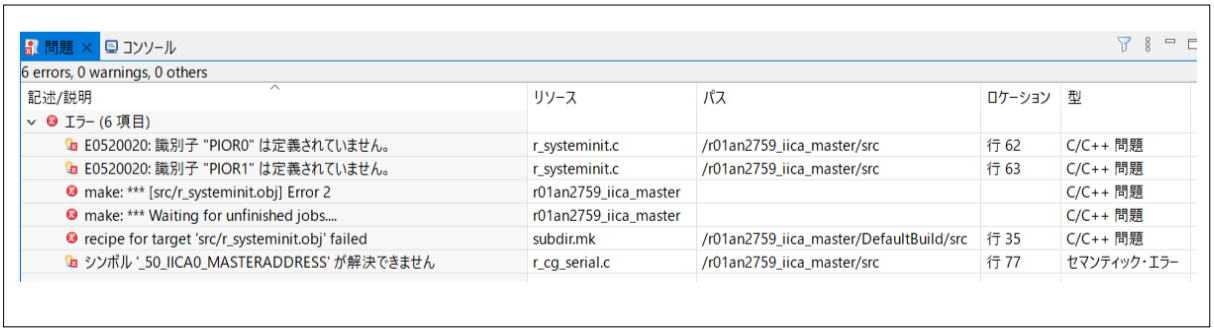
図 3-46　インクルード・パスの追加（移植例 2）



(2)　エラー処置 2

デバイス変更時に「generate」フォルダ内に生成された iodefine.h（RL78/G14 用）ではなく、変更前のプロジェクト・フォルダ直下の iodefine.h（RL78/G13 用）がインクルードされているためにエラーが発生しています。「generate」フォルダ内の iodefine.h をインクルードするようにコンパイラ・オプションで「generate」フォルダを先頭に移動します。

図 3-47　ビルド・エラー2（移植例 2）

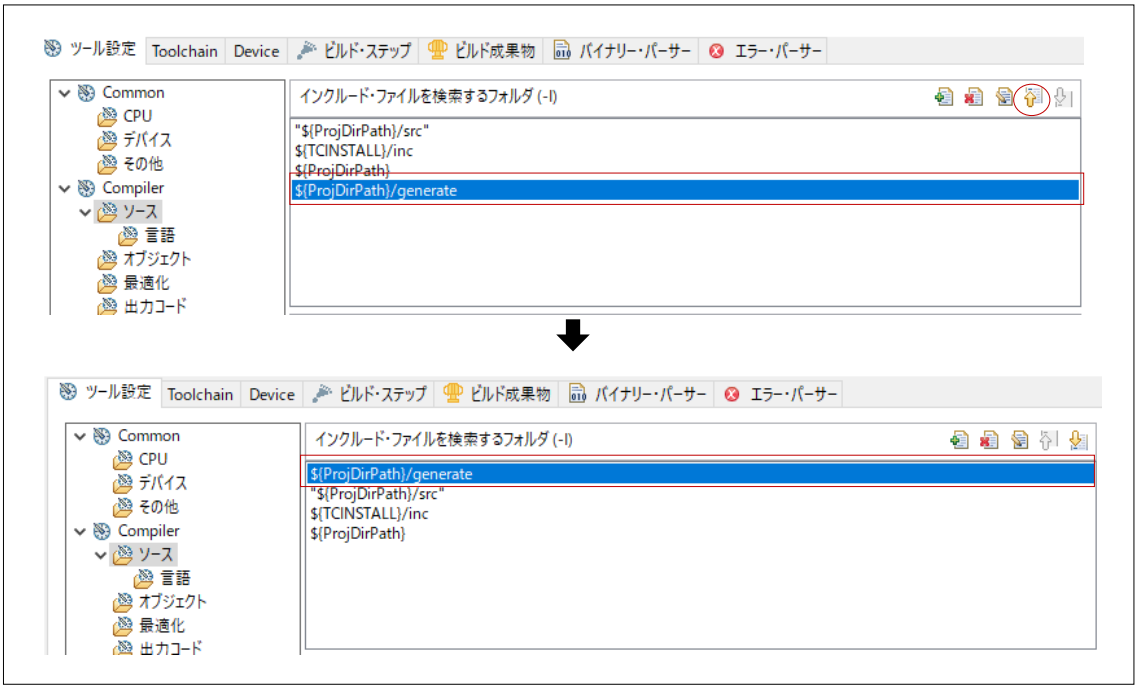


- インクルード・パスの順番変更

e2 studio の「プロジェクト」→「C/C++ Project Settings」メニューを選択します。プロパティ・ダイアログで「C/C++ビルド」ー「設定」を選択し、「ツール設定」タブの「ソース」でパスの順番を変更します。

パスの順番を変更後、「プロジェクト」→「プロジェクトのビルド」メニューを選択し、再度ビルドします。

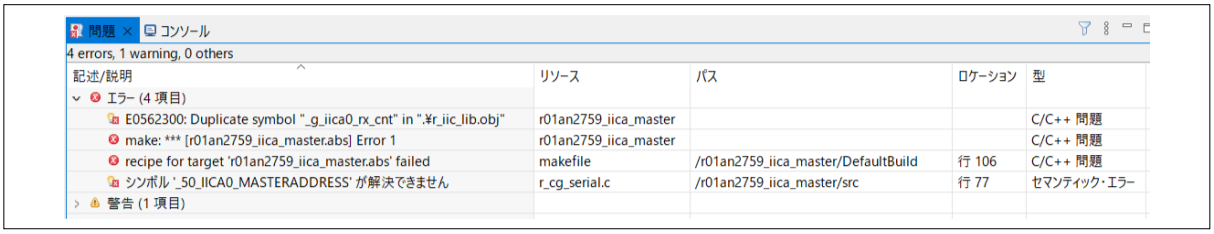
図 3-48　インクルード・パスの順番変更（移植例 2）



(3)　エラー処置 3

シンボル「_exit」が 2 重定義されているためにエラーが発生しています。「_exit」が定義されているファイルを確認し処置をします。

図 3-49　ビルド・エラー3（移植例 2）



● 「_exit」が定義されているファイルの特定

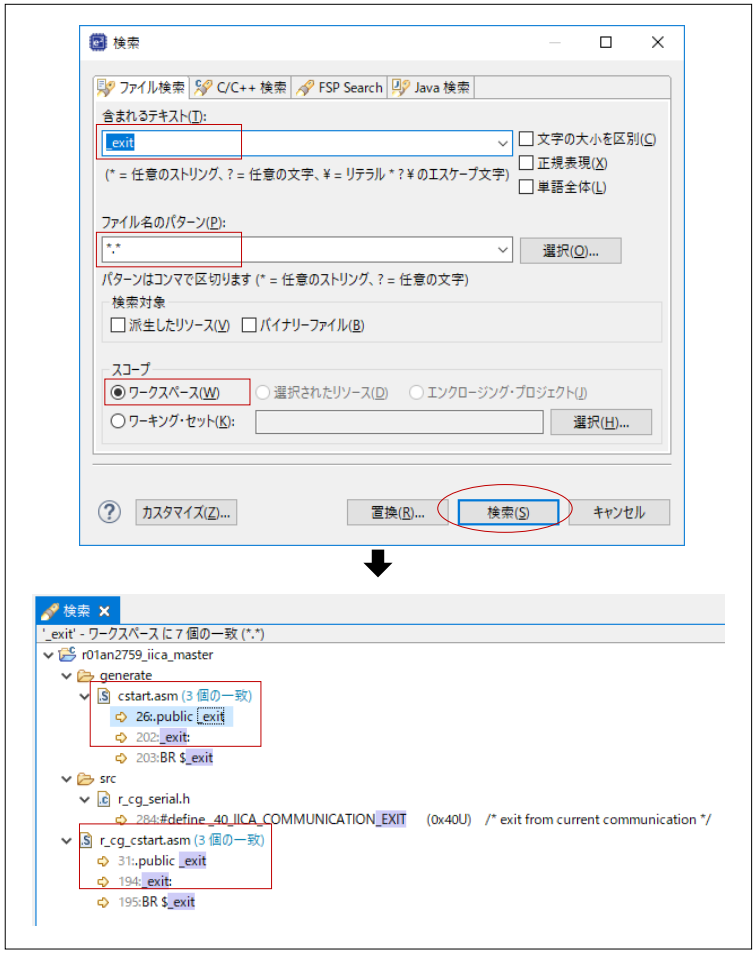
e² studio の「検索」－「検索」メニューを選択します。検索ダイアログで「_exit」の記述がある箇所を検索します。

含まれるテキスト：_exit

ファイル名のパターン：*.*

スコープ：ワークスペース

図 3-50　検索ダイアログ（移植例 2）

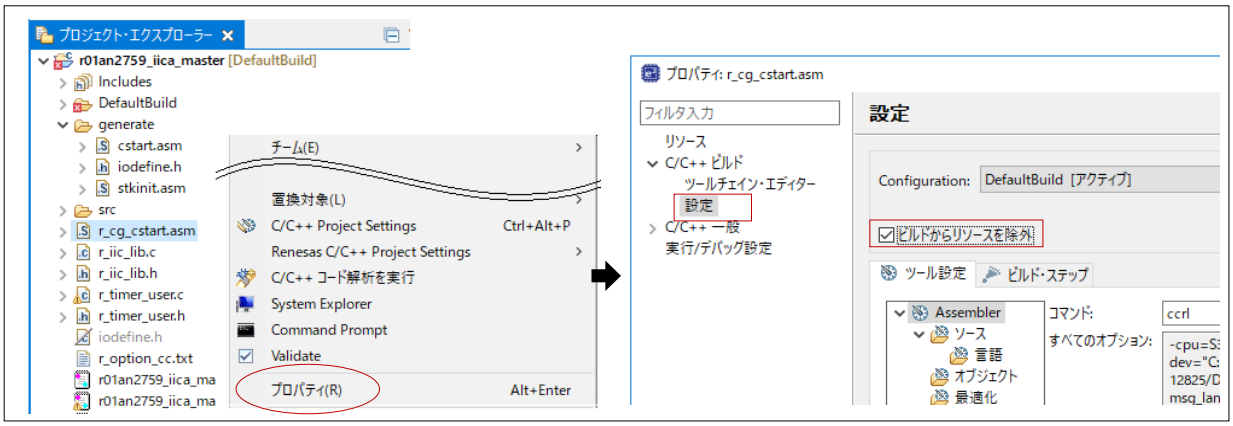


- r_cg_cstart.asm と cstart.asm の比較

  r_cg_cstart.asm と cstart.asm をエディタで開き、ファイルの中身を確認します。確認の結果、両ファイルはスタートアップファイルであるため、デバイス変更後に生成された cstart.asm を残し、r_cg_cstart.asm をビルド対象外にします。

  r_cg_cstart.asm をビルド対象から除外後、「プロジェクト」→「プロジェクトのビルド」メニューを選択し、再度ビルドします。警告は残りますが、すべてのエラーが解決します。

図 3-51　r_cg_cstart.asm をビルドから除外（移植例 2）



4. サンプルコード

サンプルコードは、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。

5. 参考ドキュメント

RL78/G13 ユーザーズマニュアル ハードウェア編 （R01UH0146J）

RL78 ファミリユーザーズマニュアルソフトウェア編 (R01US0015J)

(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

(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。)

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2022.04.04	—	初版発行

。

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。