# 統合開発環境 e² studio

## CMake プロジェクトを作成してビルドする

### はじめに

e² studio では、CMake を使用するプロジェクトを作成して開発を行うことができます。

本ドキュメントでは、e² studio で CMake を使用するプロジェクトを作成し、ビルドを実行する方法について説明します。

### 目次

1. 概要 ..... 2
1.1 目的 ..... 2
1.2 動作確認環境 ..... 2

2. CMake サポート状況 ..... 3
2.1 FSP プロジェクト ..... 3
2.1.1 プロジェクト作成方法 ..... 3
2.1.2 ビルド方法 ..... 4
2.2 FSP プロジェクト以外 ..... 4

3. CMake プロジェクトを作成する ..... 5

4. ビルド関連メニューに CMake コマンドを登録する ..... 6
4.1 「CDT Core Builder」を無効化する ..... 6
4.2 「CMake Configure」コマンドを登録する ..... 7
4.3 「CMake Build」コマンドを登録する ..... 9
4.4 「CMake Clean」コマンドを登録する ..... 10

5. ビルドを実行する ..... 12

6. クリーンを実行する ..... 12

7. スマート・コンフィグレータを使用する場合 ..... 13

8. デバッグ ..... 15

9. 参考情報 ..... 16
9.1 Web サイト ..... 16

改訂記録 ..... 17

# 1. 概要

## 1.1 目的

CMake は、ソフトウェアのビルド設定を記述した CMake 設定ファイルから、Ninja や Make などのビルドファイルを生成するためのオープンソースのビルドシステムです。CMake 設定ファイルは、コンパイラや開発環境の違いを意識せずに記述できるため、開発環境に依存しないビルド環境を構築できます。

e2 studio は、オープンソースの「Eclipse」をベースとしたルネサスマイコン用の統合開発環境です。e2 studio では、e2 studio 2020-04 から CMake を使用するプロジェクトを作成する機能をサポートしました。CMake を使用するプロジェクトを作成してビルドを実行するには、プロジェクトに特別な設定を追加する必要があります。

本ドキュメントでは、CMake を使用するプロジェクトの作成、およびそのプロジェクトでビルド（CMake と Ninja）を実行する方法について説明します。

＜対象読者＞

本ドキュメントは、CMake を使用して組込みソフトウェアを開発するユーザーを対象としています。以下の内容については説明しません。

- CMake の基本的な使用方法
- CMake 設定ファイルの作成方法
- コンパイラ、アセンブラおよびリンカなどのオプション設定方法

これらは、使用するプロジェクトおよびツールチェーンに応じて事前に準備してください。

## 1.2 動作確認環境

このドキュメントで説明する操作手順は、弊社にて以下の環境で確認しました。

［OS］

- Windows 11（日本語版）

［IDE］

- e2 studio 2026-07

［ツール］

- CMake
- Ninja

CMake と Ninja は、e2 studio をインストールすると以下のフォルダにインストールされます。

- CMake
  ＜e2 studio インストールフォルダ＞
  \eclipse\plugins\com.renesas.ide.exttools.cmake.win32.x86_64_X.X.X.vXXXXXXXX-XXXX\cmake\bin
- Ninja
  ＜e2 studio インストールフォルダ＞
  \eclipse\plugins\com.renesas.ide.exttools.ninja.win32.x86_64_X.XX.X.vXXXXXXXX-XXXX\n

## 2.　CMake サポート状況

e² studio における CMake プロジェクトのサポート形態は、プロジェクト種別によって異なります。そのため、プロジェクト種別に合わせて、プロジェクト作成、ビルドなどの操作を行う必要があります。

まず、作成するプロジェクトが FSP プロジェクトかどうか確認してください。

- FSP プロジェクトの場合=>『2.1 FSP プロジェクト』へ
- FSP プロジェクト以外の場合=>『2.2 FSP プロジェクト以外』へ

## 2.1　FSP プロジェクト

以下のプロジェクトは FSP プロジェクトです。

- Renesas RA C/C++ Project、Renesas RA FSP Solution
- Renesas RA for Wireless C/C++ FSP Project
- Renesas RA for Wireless FSP Solution
- Renesas RH850 C/C++ FSP Project
- Renesas RISC-V MCU C/C++ FSP Project
- Renesas RRQ C/C++ FSP Project
- Renesas RX C/C++ FSP Project
- Renesas RX FSP Solution
- Renesas RZ C/C++ FSP Project

FSP プロジェクトでは、プロジェクト作成時に CMake プロジェクトを選択できます。

### 2.1.1　プロジェクト作成方法

CMake プロジェクトを作成するときは、下図に示すウィザードページの［IDE Project Type］ドロップダウンリストで「e² studio CMake」を選択してください。

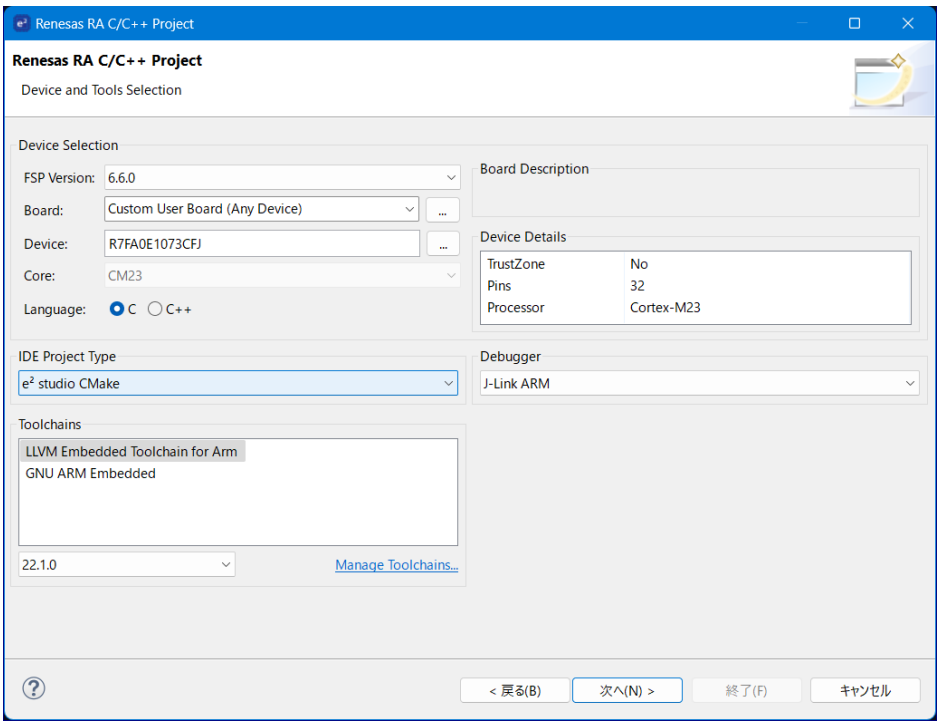



**図 1**

ソース・ファイルとともに CMake 設定ファイルが生成されます。

［注意事項］

- サポートするビルド構成は 1 つのみです。
- Release と Debug の同時管理はサポートしていません。
- ビルド・オプションの GUI 設定はサポートしていません。

### 2.1.2　ビルド方法

［プロジェクト・エクスプローラ］ビューでプロジェクト選択後、以下のメニューから実行してください。

- クリーン
  - 右クリックメニュー［プロジェクトをクリーンにする］
- ビルド
  - 右クリックメニュー［プロジェクトのビルド(B)］
  - メニュー［プロジェクト(P)］->［プロジェクトのビルド(B)］

FSP プロジェクトでは、これ以降の章で説明する手順および設定は必要ありません。

## 2.2　FSP プロジェクト以外

以下のようなプロジェクトでは、e2 studio の標準のビルド機能と CMake を手動で関連付ける必要があります。

- GCC for Renesas RX C/C++ Executable Project
- GNU for RZ C/C++ Executable Project
- LLVM for Renesas RL78 C/C++ Executable Project
- Renesas CC-RH C Executable Project
- Renesas CC-RL C/C++ Executable Project
- Renesas CC-RX C/C++ Executable Project

これらのプロジェクトでは、以降の章の手順に従って、プロジェクト作成およびビルド設定を行ってください。

## 3. CMake プロジェクトを作成する

本章では、CMake Project テンプレートを使用して空の CMake プロジェクトを作成します。作成後に、ソース・ファイルおよび CMake 設定ファイルをプロジェクトに追加してください。

e2 studio を起動し、以下の手順を実施してください。

1) メニュー［ファイル(F)］>［新規(N)］>［C/C++ Project］を選択してください。
2) ［新規 C/C++ Project－新規 C/C++プロジェクトのテンプレート］ダイアログボックスが表示されます。左側リストボックスで「CMake」を選択し、右側リストボックスで「Empty or Existing CMake Project」を選択してください。次に、［次へ(N) >］ボタンをクリックしてください。

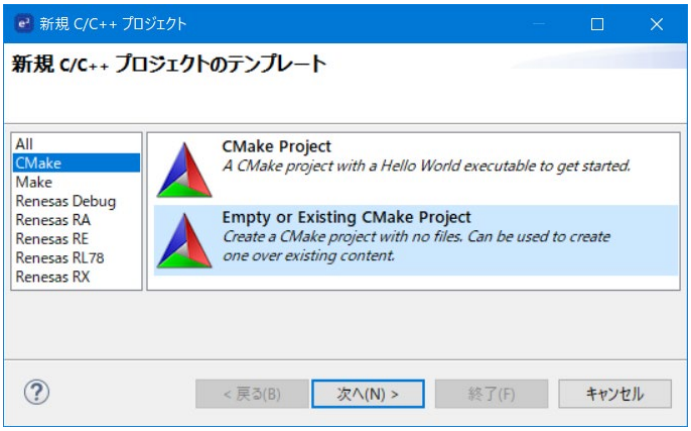



**図 2**

3) ［New CMake Project - Specify properties of new CMake project］ダイアログボックスが表示されます。［プロジェクト名(P):］テキストボックスに任意のプロジェクト名を入力してください。次に、［終了(F) >］ボタンをクリックしてください。
4) 空のプロジェクトが作成され、［プロジェクト・エクスプローラー］ビューに表示されます。
5) 作成されたプロジェクトのフォルダに以下を配置します。
   - ソース・ファイル
   - CMake 設定ファイル（ツールチェーン・ファイル、構成ファイルなど）

次章に進んでください。

## 4.　ビルド関連メニューに CMake コマンドを登録する

作成直後の CMake プロジェクトでは、ビルド関連メニューにビルドを実行するコマンドが関連付けられていません。そのため、以下の設定を行い、ビルド関連メニューに CMake を使用してビルドを実行するコマンドを設定する必要があります。

① 「CDT Core Builder」を無効化する。
② 「CMake Configure」コマンドを登録する。
③ 「CMake Build」コマンドを登録する。
④ 「CMake Clean」コマンドを登録する。

設定後は、通常の

- プロジェクトのビルド
- プロジェクトをクリーンにする

からビルドを実行することができます。

### 4.1　「CDT Core Builder」を無効化する

ビルド実行時に、e² studio 標準の「CDT Core Builder」と CMake によるビルドが同時に実行することを防ぐため、「CDT Core Builder」を無効化します。

以下の手順を実施してください。

1) ［プロジェクト・エクスプローラー］ビューでプロジェクトを選択後、メニュー［ファイル(F)］>［プロパティ(R)］または右クリックメニュー［プロパティ(R)］のいずれかを選択してください。
2) ［プロパティ］ダイアログボックスが表示されます。左側ツリーボックスで「ビルダー」を選択してください。
3) 右側に［ビルダー］パネルが表示されます。［次のプロジェクトのビルダーを構成する(P) : ］チェックボックス付きリストボックスで「CDT Core Builder」チェックボックスのチェックを外してください。
4) メッセージが表示されますので［OK］ボタンをクリックしてください。

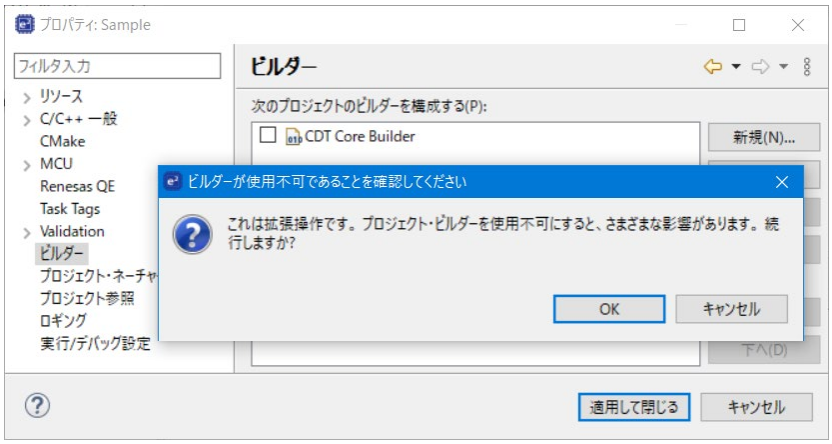



**図 3**

引き続き、［ビルダー］パネルで操作を行います。次の章に進んでください。

## 4.2　「CMake Configure」コマンドを登録する

CMake を実行して CMake 設定ファイルから Ninja 用ビルドファイルを生成するコマンドを作成し、ビルドメニューに関連付けます。

以下の手順に従って、コマンドを登録してください。

（［ビルダー］パネルを開いていない場合は、『4.1 「CDT Core Builder」を無効化する』の手順に従い［ビルダー］パネルを開いてください。）

1) ［新規(N)...］ボタンをクリックしてください。
2) ［構成タイプの選択］ダイアログボックスが表示されます。「プログラム」を選択し、［OK］ボタンをクリックしてください。
3) ［構成の編集］ダイアログボックスが表示されます。［名前(N):］テキストボックスに任意の名前を入力してください。（この例では「CMake Configure」を入力します。）
4) ［メイン］タブをクリックして、以下の項目を設定してください。
   - ［ロケーション(L):］テキストボックスに CMake 実行プログラム（cmake.exe）をパス付きで入力してください。
   - ［作業ディレクトリー(D):］テキストボックスに「${workspace_loc:/${project_name}}」を入力してください。
   - ［引数(A):］テキストボックスに「-S . -B build -G Ninja -DCMAKE_BUILD_TYPE:STRING=Debug -DCMAKE_TOOLCHAIN_FILE:STRING=cmake/cross.cmake -DCMAKE_EXPORT_COMPILE_COMMANDS:BOOL=TRUE」を入力してください。

     パラメータおよび変数の詳細は、以下の Web サイトをご参照ください。
     ccmake(1) — CMake 4.5.0-rc1 Documentation
     cmake-variables(7) — CMake 4.5.0-rc1 Documentation

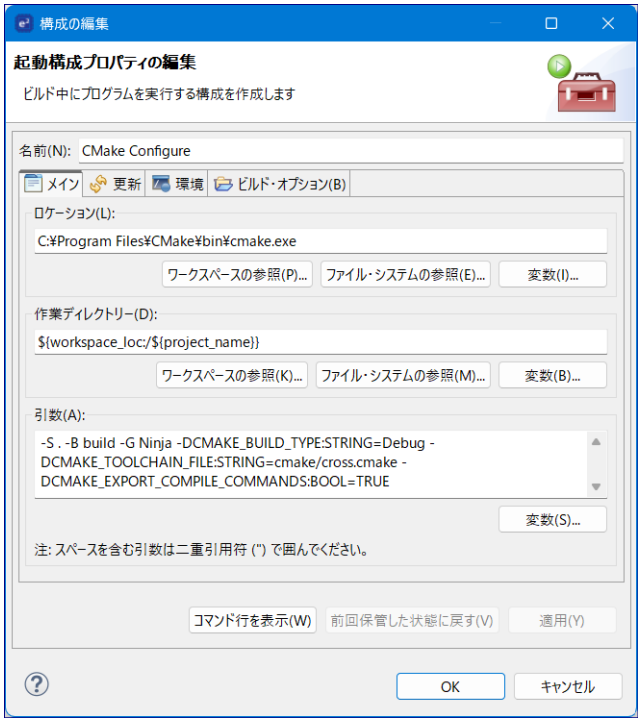



**図 4**

5) ［更新］タブをクリックして、以下の項目を設定してください。

- ［完了時にリソースを更新(U)］チェックボックスをチェックしてください。
- ［選択されたリソースを含むプロジェクト(P)］ラジオボタンをチェックしてください。

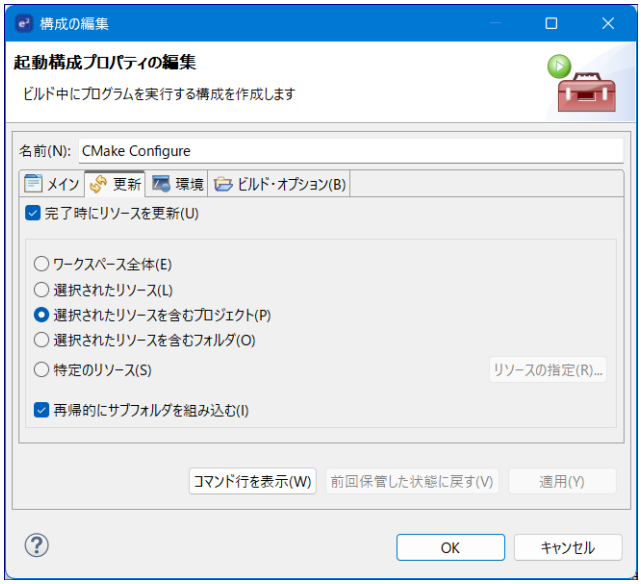



**図 5**

6)　［環境］タブをクリックして、以下の環境変数を設定してください。
［追加］ボタンをクリックすると、［新規環境変数］ダイアログボックスが表示されますので、以下の内容を入力してください。

- PATH=<Ninja のインストールフォルダ>;%PATH%
- CMake、ツールチェーンなどの実行に必要な環境変数

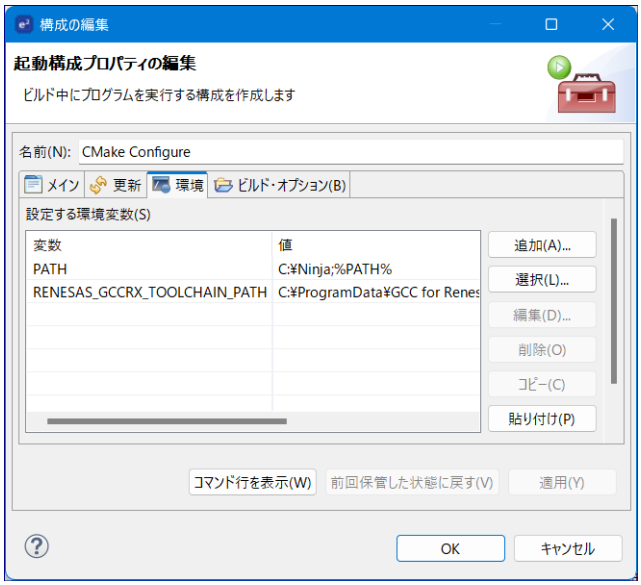



**図 6**

7)　［ビルド・オプション(B)］タブをクリックして、以下の項目を設定してください。

- ［標準入出力］エリアで、［コンソールに割り当て(C)(入力に必要)］チェックボックスをチェックし、それ以外のチェックボックスはチェックを外してください。
- ［バックグラウンドでの起動(K)］チェックボックスをチェックしてください。

- ［ビルダーの実行:］エリアで、［手動ビルド中(D)］チェックボックスをチェックし、それ以外のチェックボックスはチェックを外してください。

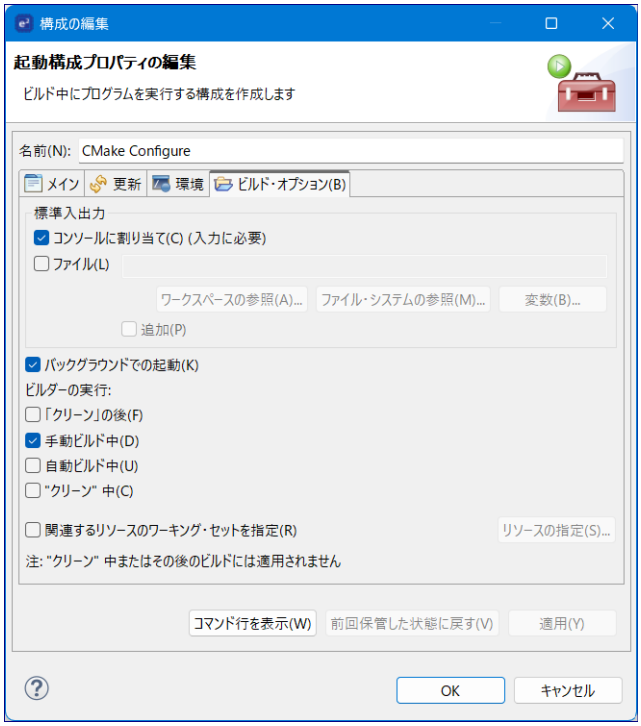



**図 7**

8)　［OK］ボタンをクリックしてください。

引き続き、［ビルダー］パネルで操作を行います。次の章に進んでください。

## 4.3　「CMake Build」コマンドを登録する

生成済みの Ninja 用ビルドファイルを使用してビルドを実行するコマンドを作成し、ビルドメニューに関連付けます。

以下の手順に従って、コマンドを登録してください。

（［ビルダー］パネルを開いていない場合は、『4.1 「CDT Core Builder」を無効化する』の手順に従い［ビルダー］パネルを開いてください。）

1)　［新規(N)...］ボタンをクリックしてください。
2)　［構成タイプの選択］ダイアログボックスが表示されます。「プログラム」を選択して、［OK］ボタンをクリックしてください。
3)　［構成の編集］ダイアログボックスが表示されます。［名前(N):］テキストボックスに任意の名前を入力してください。（この例では「CMake Build」を入力します。）
4)　［メイン］タブをクリックして、以下の項目を設定してください。
    - ［ロケーション(L):］テキストボックスに CMake 実行プログラム（cmake.exe）をパス付きで入力してください。
    - ［作業ディレクトリー(D):］テキストボックスに「${workspace_loc:/${project_name}}」を入力してください。
    - ［引数(A):］テキストボックスに「--build build --config Debug --target all」を入力してください。
5)　［更新］タブをクリックして、以下の項目を設定してください。

- ［完了時にリソースを更新(U)］チェックボックスをチェックしてください。
- ［選択されたリソースを含むプロジェクト(P)］ラジオボタンをチェックしてください。

6) ［環境］タブをクリックして、以下の環境変数を設定してください。
［追加］ボタンをクリックすると、［新規環境変数］ダイアログボックスが表示されますので、以下の内容を入力してください。
- PATH=<Ninja のインストールフォルダ>;%PATH%
- CMake、ツールチェーンなどの実行に必要な環境変数

7) ［ビルド・オプション(B)］タブをクリックして、以下の項目を設定してください。
- ［標準入出力］エリアで、［コンソールに割り当て(C)(入力に必要)］チェックボックスをチェックし、それ以外のチェックボックスはチェックを外してください。
- ［バックグラウンドでの起動(K)］チェックボックスをチェックしてください。
- ［ビルダーの実行:］エリアで、［手動ビルド中(D)］チェックボックスをチェックし、それ以外のチェックボックスはチェックを外してください。

8) ［OK］ボタンをクリックしてください。

引き続き、［ビルダー］パネルで操作を行います。次の章に進んでください。

## 4.4 「CMake Clean」コマンドを登録する

生成済みオブジェクト・ファイルおよび実行ファイルを削除するコマンドを作成し、クリーンメニューに関連付けます。

以下の手順に従って、コマンドを登録してください。

（［ビルダー］パネルを開いていない場合は、『4.1 「CDT Core Builder」を無効化する』の手順に従い［ビルダー］パネルを開いてください。）

1) ［新規(N)...］ボタンをクリックしてください。
2) ［構成タイプの選択］ダイアログボックスが表示されます。「プログラム」を選択して、［OK］ボタンをクリックしてください。
3) ［構成の編集］ダイアログボックスが表示されます。［名前(N):］テキストボックスに任意の名前を入力してください。（この例では「CMake Clean」を入力します。）
4) ［メイン］タブをクリックして、以下の項目を設定してください。
- ［ロケーション(L):］テキストボックスに CMake 実行プログラム（cmake.exe）をパス付きで入力してください。
- ［作業ディレクトリー(D):］テキストボックスに「${workspace_loc:/${project_name}}」を入力してください。
- ［引数(A):］テキストボックスに「--build build --config Debug --target clean」を入力してください。

5) ［更新］タブをクリックして、以下の項目を設定してください。
- ［完了時にリソースを更新(U)］チェックボックスをチェックしてください。
- ［選択されたリソースを含むプロジェクト(P)］ラジオボタンをチェックしてください。

6) ［環境］タブをクリックして、以下の環境変数を設定してください。
［追加］ボタンをクリックすると、［新規環境変数］ダイアログボックスが表示されますので、以下の内容を入力してください。
- PATH=<Ninja のインストールフォルダ>;%PATH%
- CMake、コンパイラなどの実行に必要な環境変数

7)　［ビルド・オプション(B)］タブをクリックして、以下の項目を設定してください。

- ［標準入出力］エリアで、［コンソールに割り当て(C)(入力に必要)］チェックボックスをチェックし、それ以外のチェックボックスはチェックを外してください。
- ［バックグラウンドでの起動(K)］チェックボックスをチェックしてください。
- ［ビルダーの実行:］エリアで、［"クリーン"中(D)］チェックボックスをチェックし、それ以外のチェックボックスはチェックを外してください。




**図 8**

8)　［OK］ボタンをクリックしてください。

9)　［ビルダー］パネルに戻りましたら、［適用して閉じる］ボタンをクリックしてください。

次の章に進んでください。

## 5. ビルドを実行する

以下の手順を実施してください。

1) ［プロジェクト・エクスプローラー］ビューでプロジェクトを選択してください。次に、以下のいずれかを実施してください。

- 右クリックメニュー［プロジェクトのビルド(B)］を選択する。
- メニュー［プロジェクト(P)］->［プロジェクトのビルド(B)］を選択する。

ビルドを実行すると、以下の順序でコマンドが実行されます。

① 「CMake Configure」コマンド
Ninja 用ビルドファイルを生成します。
② 「CMake Build」コマンド
コンパイラ、アセンブラ、リンカを実行します。

## 6. クリーンを実行する

以下の手順を実施してください。

1) ［プロジェクト・エクスプローラー］ビューでプロジェクトを選択してください。次に、右クリックメニュー［プロジェクトをクリーンにする］を選択してください。

クリーンを実行すると、以下のコマンドが実行されます。

① 「CMake Clean」コマンド
生成済みオブジェクト・ファイルおよび実行ファイルを削除します。

## 7. スマート・コンフィグレータを使用する場合

プロジェクトがスマート・コンフィグレータを使用するプロジェクトでは、外部ツール機能を使用すると、メニューからスマート・コンフィグレータを起動して、コード生成を実行できます。

以下の手順に従って、コマンドを登録してください。

1) メニュー［実行(R)］>［外部ツール(E)］>［外部ツールの構成(E)...］を選択してください。
2) ［外部ツール構成］ダイアログボックスが表示されます。左側ツリーボックスで「プログラム」を選択し、「新規の起動構成」ボタンをクリックしてください。
3) 左側ツリーに「新規構成 [local]」が追加されます。
4) 右側パネルで［名前(N):］テキストボックスに任意の名前を入力してください。
5) 右側パネルで［メイン］タブをクリックして、以下の項目を設定してください。
   - ［ロケーション(L):］テキストボックスにスマート・コンフィグレータ実行プログラム（SmartConfigurator.exe）をパス付きで入力してください。
   - ［作業ディレクトリー(D):］テキストボックスに「${workspace_loc:/${project_name}}」を入力してください。
   - ［引数(A):］テキストボックスに、使用するスマート・コンフィグレータに合わせて以下の内容を入力してください。
     - RX 用スマート・コンフィグレータ：
       "${workspace_loc:/＜プロジェクトフォルダ名＞}\＜ファイル名＞.scfg" -enable=cmake,rxcbankmode -disable=iar
     - RL78 用スマート・コンフィグレータ：
       "${workspace_loc:/＜プロジェクトフォルダ名＞}\＜ファイル名＞.scfg" -enable=cmake -disable=iar
     - RH850 用スマート・コンフィグレータ：
       "${workspace_loc:/＜プロジェクトフォルダ名＞}\＜ファイル名＞.scfg" -enable=cmake -disable=iar

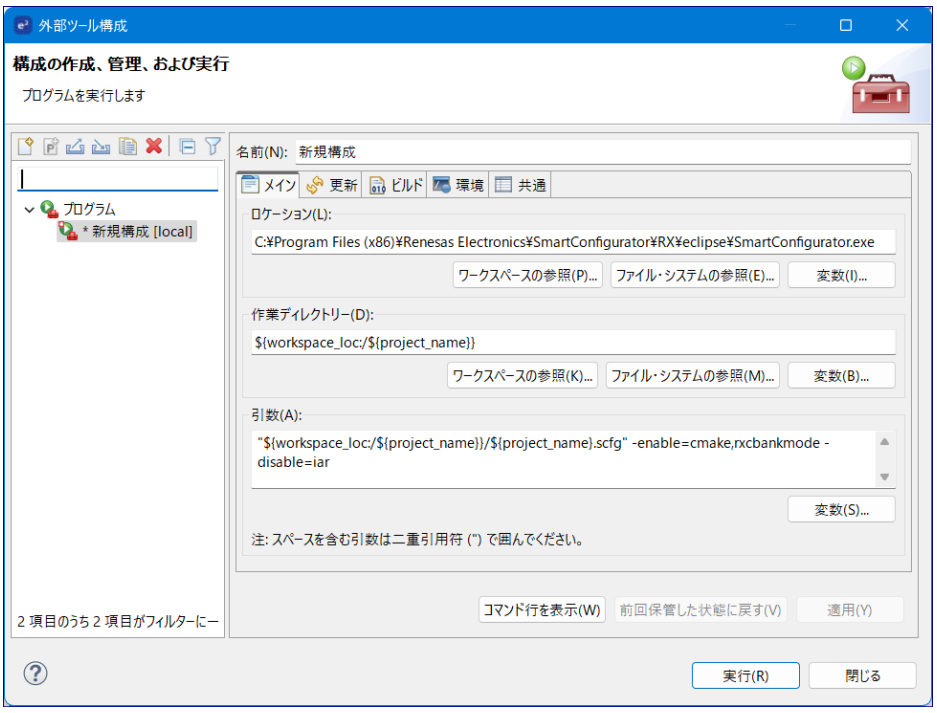



**図 9**

6)　右側パネルで［更新］タブをクリックして、以下の項目を設定してください。
- ［完了時にリソースを更新(U)］チェックボックスをチェックしてください。
- ［選択されたリソースを含むプロジェクト(P)］ラジオボタンをチェックしてください。
- ［再帰的にサブフォルダを組み込む(I)］チェックボックスをチェックしてください。

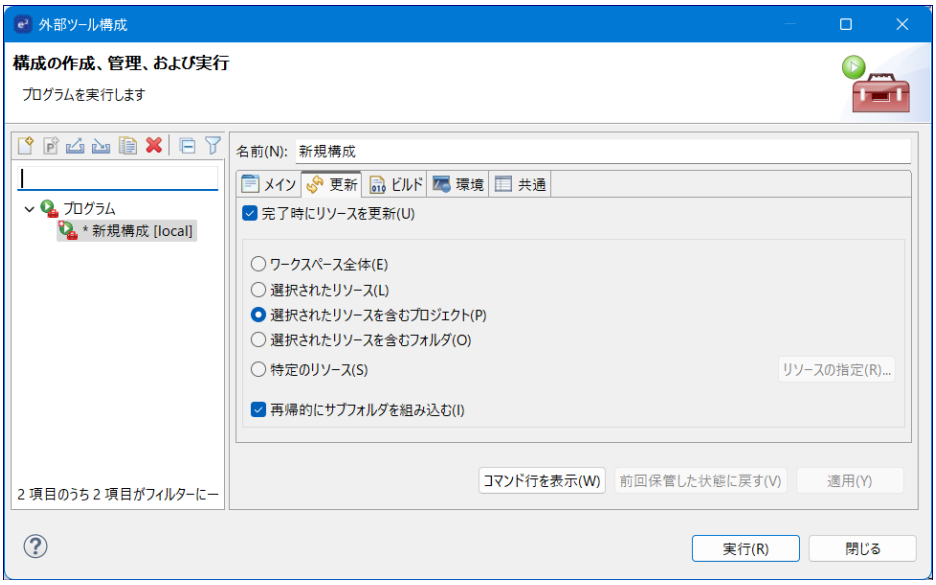



**図 10**

7)　右側パネルで［ビルド］タブをクリックして、以下の項目を設定してください。
- ［起動前にビルド(B)］チェックボックスのチェックを外してください。

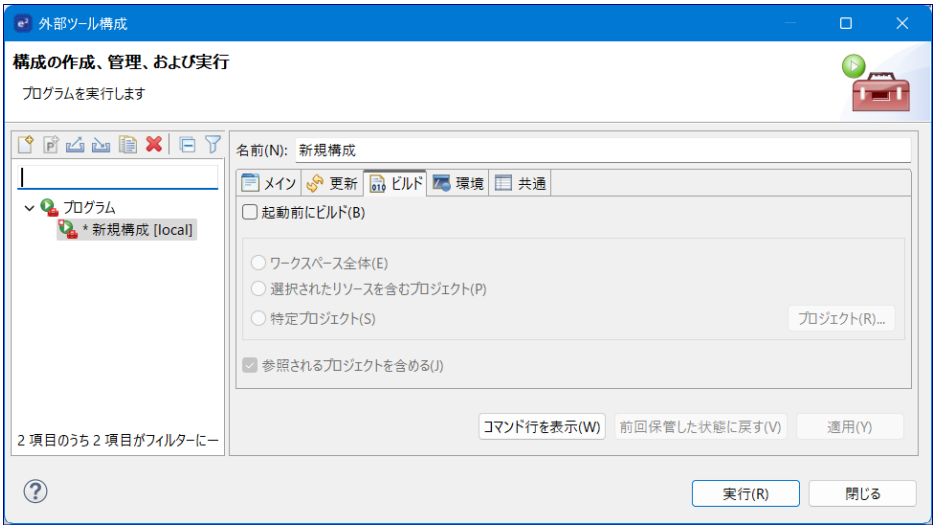



**図 11**

8)　右側パネルで［共通］タブをクリックして、以下の項目を設定してください。
- ［お気に入りメニューに表示(R)］チェックボックス付きリストボックスで［外部ツール］をチェックしてください。
- ［エンコード］エリアで［Use system encoding (windows-31j)］ラジオボタンをチェックしてください。
- ［標準出力およびエラー出力をマージ（エラー出力の色付けを無効にします）(M)］チェックボックスをチェックしてください。
- その他は、デフォルトのままで構いません。

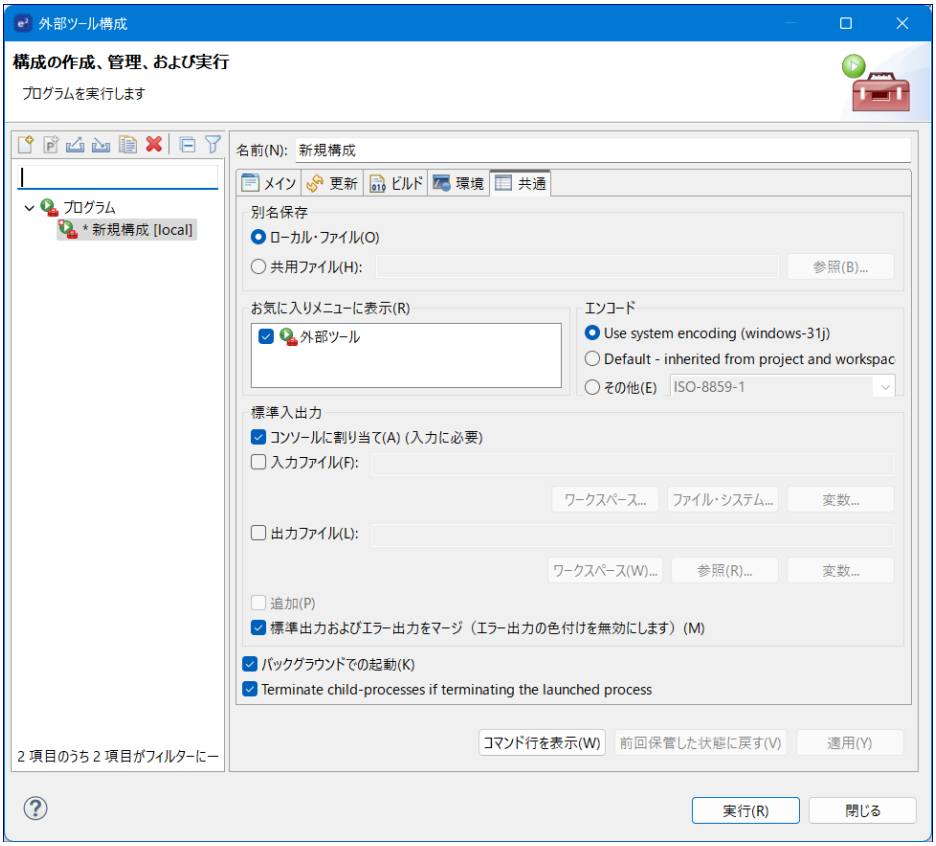



**図 12**

9)　［適用(Y)］ボタンをクリックして、次に［閉じる］ボタンをクリックしてください。

スマート・コンフィグレータを実行するときは、［プロジェクト・エクスプローラー］ビューでプロジェクトを選択後、メニュー［実行(R)］>［外部ツール(E)］>［<任意の名前>］を選択してください。

## 8.　デバッグ

CMake プロジェクトでは、デバッグ構成は自動的に作成されません。デバッガを実行するには、既存プロジェクト（CMake プロジェクト以外）のデバッグ構成を参考にして、ユーザー自身でデバッグ構成を作成してください。

デバッグ構成は、メニュー［実行(R)］>［デバッグの構成(B)...］を選択すると表示される［デバッグ構成］ダイアログボックスで作成してください。

# 9. 参考情報

## 9.1 Web サイト

- e2 studio
  https://www.renesas.com/e2studio
- CMake
  https://cmake.org/
- Ninja
  https://ninja-build.org/

## 改訂記録

| Rev. | 発行日 | 改訂内容 | |
|---|---|---|---|
| | | ページ | ポイント |
| 1.00 | Dec.25.20 | - | 新規作成 |
| 2.00 | Jun.30.21 | すべて | 全面見直しを実施<br>・不要な情報を削除してより汎用的な設定内容に改善 |
| 2.01 | Jun.28.22 | 6 ページ | 「「XXXX」は、ユーザー毎に異なります。」を「「XXXX」は、e2 studio のバージョン毎に異なります。」に変更 |
| 3.00 | Oct.13.10 | すべて | ビルド方法を Make ビルドから Ninja ビルドに変更<br>それに伴い、章構成から見直しを実施 |
| | | | |

# 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。

© 2026 Renesas Electronics Corporation. All rights reserved.