User's Manual

統合開発環境
e^2 studio 2021-04 以上

ユーザーズマニュアル クイックスタートガイド

ルネサスマイクロコントローラ
RA ファミリ

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

ルネサス エレクトロニクス
www.renesas.com

Rev.1.00 2021.09

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

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。

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/

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。

目次

1. 概説 1
1.1 システム構成 3
1.2 システム要件 3
1.2.1 PCハードウェア環境 3
1.2.2 動作環境 3
1.3 サポートするツールチェーン 3
1.4 サポートするエミュレータ 3
1.5 RAプロジェクト開発の概要 4

2. インストール 5
2.1 プラットフォームインストーラを用いたインストール 5
2.2 e[2] studioとFSPの個別インストール 11
2.2.1 e[2] studioのインストール 11
2.2.2 GNU Arm Embedded Toolchainの設定 16
2.2.3 ルネサスRA Flexible Software Package（FSP）のインストール 16
2.3 e[2] studioの更新 18
2.4 FSPの更新 18
2.5 e[2] studioのアンインストール 18
2.6 Keil MDKおよびIAR EWARM用RAスマート・コンフィグレータのインストール 19

3. プロジェクトの作成 22
3.1 TrustZoneに対応していないデバイス用の新規RAプロジェクトの作成 22
3.2 TrustZone対応デバイス用の新規RAプロジェクトの作成 27
3.2.1 フラットプロジェクト（TrustZoneに対応していないプロジェクト） 27
3.3 既存RAプロジェクトのインポート 32
3.4 RAスタティックライブラリの作成と使用 34
3.4.1 スタティックライブラリプロジェクトの作成 34
3.4.2 スタティックライブラリを実行プロジェクトで使用する 39
3.5 RAプロジェクトコンフィグレーションエディタ 44
3.5.1 概要（Summary）ページ 45
3.5.2 BSPページ 46
3.5.3 クロック構成（Clocks Configuration）ページ 47
3.5.4 端子構成（Pin Configuration）ページ 49
3.5.5 スタック構成（Stacks Configuration）ページ 53
3.5.6 コンポーネント構成（Components Configuration）ページ 60
3.5.7 割り込み構成（Interrupts Configuration）ページ 61
3.5.8 イベントリンク構成（Event Links Configuration）ページ 64
3.6 エディタ画面の吹き出し機能 67

4. ビルド 68
4.1 ビルドオプションの設定 68
4.2 サンプルプロジェクトのビルド 70
4.3 プロジェクトレポート機能（ビルドオプション一覧の出力） 71

5. デバッグ 72
5.1 既存デバッグ構成の変更 73
5.2 新規デバッグ構成の作成 76
5.3 基本的なデバッグ機能 77
5.3.1 デバッグビュー 79
5.3.2 ブレークポイントビュー 80
5.3.3 式ビュー 81

5.3.4 レジスタービュー 82
5.3.5 メモリービュー 83
5.3.6 メモリー使用量ビュー 85
5.3.7 逆アセンブル ビュー 87
5.3.8 変数ビュー 88
5.3.9 IOレジスタ（IO Registers）ビュー 89
5.3.10 イベントポイントビュー 90
5.3.11 トレースビュー 93
5.3.12 Fault Statusビュー 95
5.3.13 Run Break Timer 96

6. FreeRTOSアプリケーションの設定 98
6.1 FreeRTOSでの汎用PWMタイマ使用例 98
6.2 サンプルプロジェクトの作成 99

7. Azure RTOSアプリケーションの設定 105
7.1 Azure RTOSでの汎用PWMタイマ使用例 105
7.2 サンプルプロジェクトの作成 106

8. ヘルプ 111

1. 概説

e2 studio は、ルネサス製マイクロコントローラをサポートする統合開発環境です。e2 studio は、オープンソース Eclipse IDE と CDT（C/C++ 開発ツール）をベースに作られており、ビルド（エディタ、コンパイラ、リンカ）から、デバッグまでをカバーします。デバッグは GDB（GNU Debugger）の拡張インタフェースにより実現されます。

e2 studio は Flexible Software Package（FSP）をサポートしています。FSP は、組み込みシステムを開発するためのソフトウェアパッケージで、使いやすく、拡張性があり、高い品質を備えています。組み込みシステムの一般的な用途にマッチした、コンパクトで効率のよいドライバを提供することが FSP の目標です。

e2 studio は GUI によるさまざまなウィザードを備えており、コードの自動生成、既存ドライバの組み込みと設定、ビルドやデバッグのオプション設定、作成したアプリケーションの実行などに使用できます。ドライバの情報はツールチップの形で提供され、コードエディタビュー上で参照できます。

FSP は e2 studio のリリース 2021-04（64 ビット）および V7.6（32 ビット）以上でサポートしています。ルネサス製 RA ファミリ・マイクロコントローラとオープンソース Arm Embedded Toolchain 用の各種ビューやエディタが用意されています。

また、e2 studio は Arm® TrustZone® テクノロジをサポートしています。Arm® TrustZone® テクノロジでは、システムとアプリケーションをセキュアな環境と非セキュアな環境に分離します。Arm® TrustZone® に対応したルネサス製デバイス向けに、TrustZone を有効にした新規プロジェクト作成を支援し、セキュアアプリケーションと非セキュアアプリケーション両方のデバッグ機能を提供します。RA Arm® TrustZone® ツールの詳細は、以下のリンクを参照してください。
https://www.renesas.com/jp/ja/document/apn/ra-arm-trustzone-tooling-primer

本マニュアルでは、TrustZone に対応していないデバイス、および TrustZone 対応デバイスでのフラットプロジェクト（TrustZone を使用しないプロジェクト）を対象として説明します。

サードパーティ製の IDE やツールチェーンを使用する場合は、ルネサス RA スマート・コンフィグレータを用いて RA ファミリ・マイクロコントローラ用のソフトウェア（BSP、ドライバ、RTOS、ミドルウェア）を生成できます。

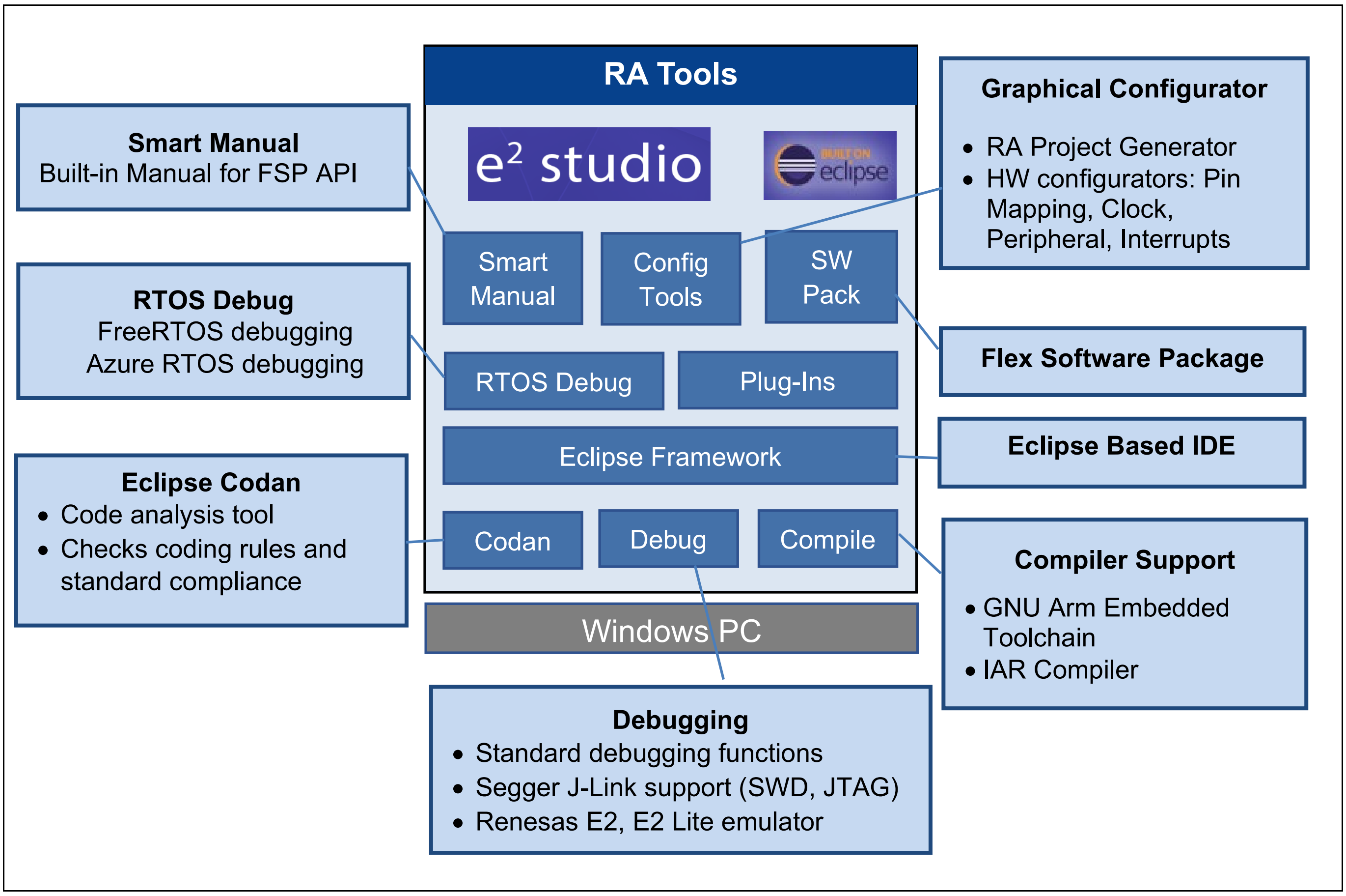


図 1　ルネサス RA ファミリ向け e² studio

1.1　システム構成

一般的なシステム構成では、下図のようにホストコンピュータとターゲットボードを使用します。

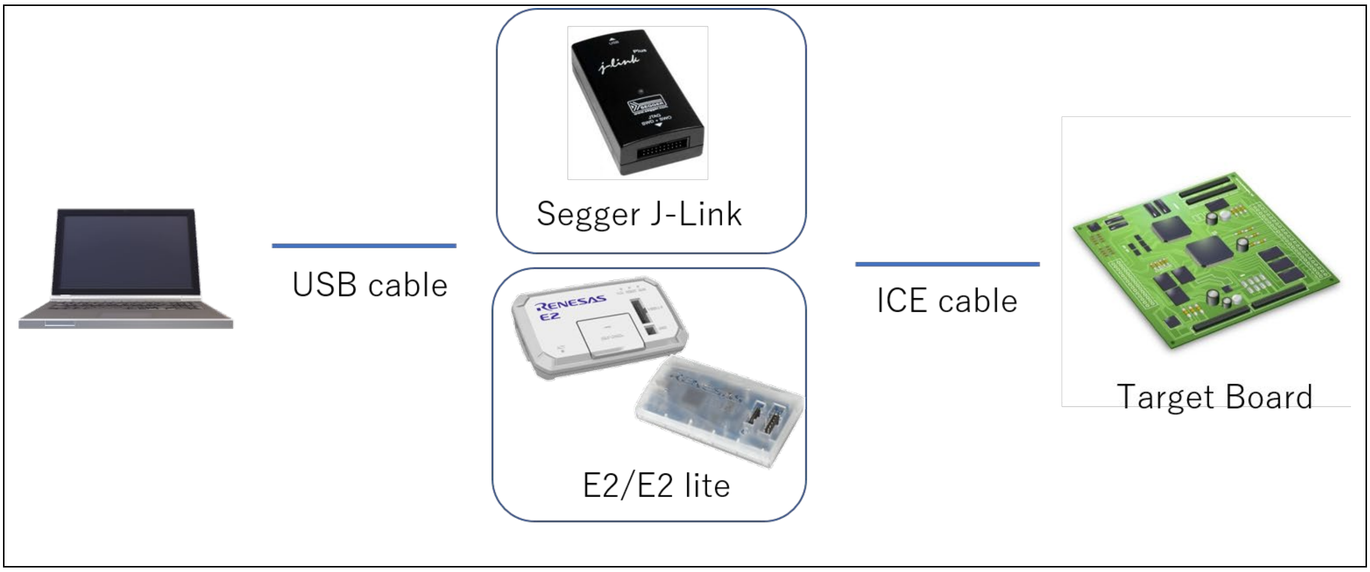


図 2　システム構成

1.2　システム要件

1.2.1　PC ハードウェア環境

プロセッサ：	1GHz 以上（ハイパースレッディング及びマルチコア CPU）
メインメモリ：	2GB 以上の空きエリア
ハードディスク：	2GB 以上の空きエリア
ディスプレイ：	解像度 1,024 x 768 ピクセル以上; 65,536 色以上
インタフェース：	USB 2.0（ハイスピードまたはフルスピード。ハイスピードが望ましい。）

1.2.2　動作環境

アーキテクチャ	Windows	e^2 studio
64 ビットバージョン	Windows 10	2021-04

注：e^2 studio 2021-04 以上では、64 ビットバージョンが必須。

1.3　サポートするツールチェーン

GNU Arm Embedded Toolchain（バージョン：9-2020-q2-update）
IAR Compiler 8.50.9 以上
Arm Compiler（バージョン： 6.16 以上）

1.4　サポートするエミュレータ

Segger J-Link、E2、E2 emulator Lite

1.5　RA プロジェクト開発の概要

本ドキュメントでは、ルネサス製 RA ファミリ・マイクロコントローラを用いた開発方法の入門について詳しく説明します。手順の概要を以下に示します。この手順を把握することで、3 章や 4 章の説明が理解しやすくなります。

(1) RA プロジェクトを作成する

(2) ハードウェア仕様（クロック、ICU、端子機能など）に適合した RA プロジェクトを構成する

(3) FreeRTOS を構成する

(4) Azure RTOS を構成する

(5) BSP を構成する（HAL ドライバモデルを選択）

(6) ユーザコードを追加する

(7) プロジェクトをビルドする

(8) デバッガを設定し、デバッグを開始する

2. インストール

プラットフォームインストーラ（FSP with e2 studio Installer）または標準の e2 studio インストーラのいずれかを用いてインストールできます。

2.1 プラットフォームインストーラを用いたインストール

プラットフォームインストーラは、e2 studio ツール、FSP パッケージ、GCC ツールチェーン、その他のインストールに必要なツールを含んでいます。以下の手順で、プラットフォームインストーラをダウンロードしてインストールしてください。

(1) GitHub のルネサス製 RA ファミリ・マイクロコントローラ用 Flexible Software Package（FSP）のページにアクセスしてください。
https://github.com/renesas/fsp/releases

(2) プラットフォームインストーラ（“setup_fsp<version>_e2s_<version >.exe”）を選択し、ダウンロードの直接リンクをクリックしてください。

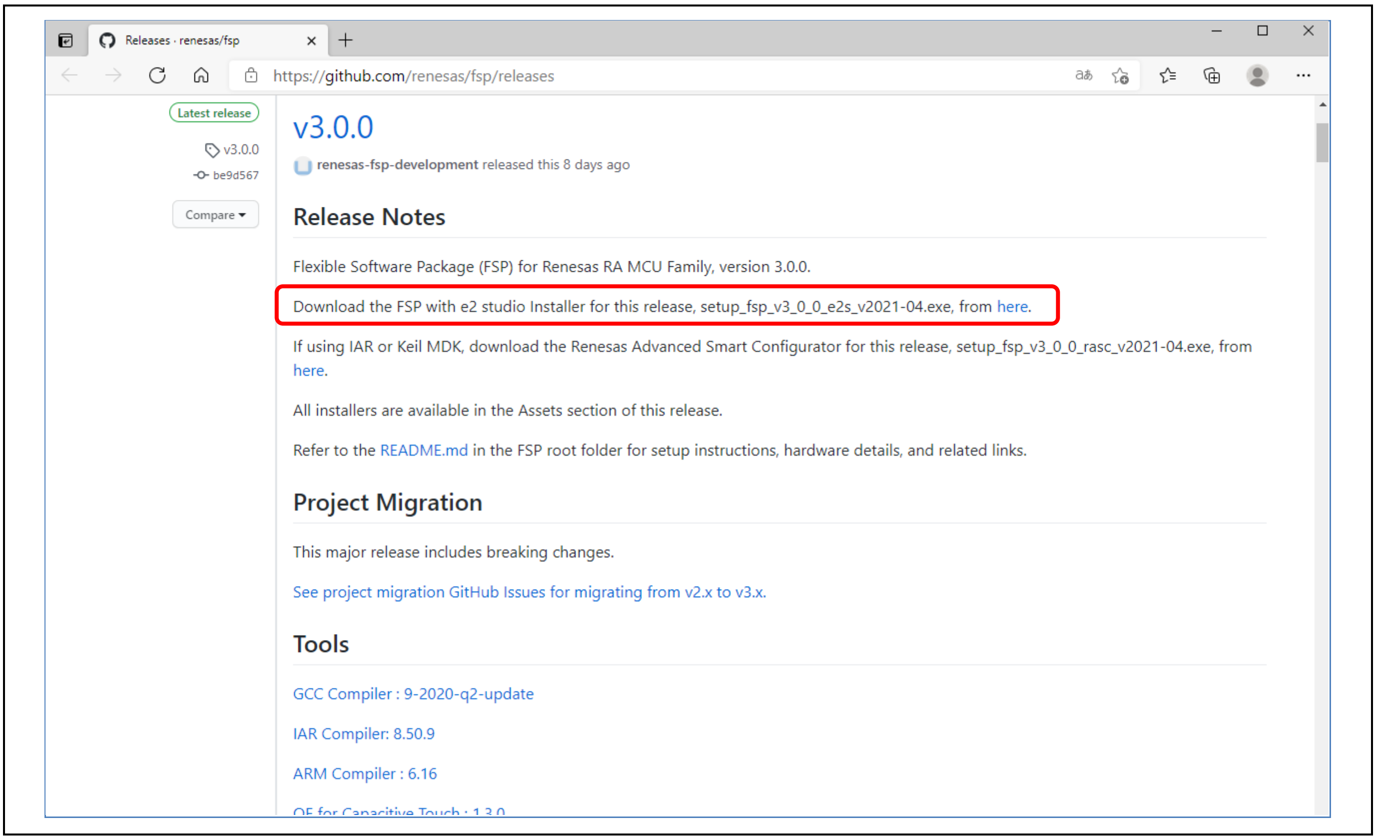


図 3　インストール ― FSP パッケージのダウンロード

(3) インストールファイルを実行してください。

(4) インストールするコンポーネントをカスタマイズしたい場合は、[インストール・タイプ] ページで “Custom Install” を選択し、[Next] ボタンを押してください。

初めて e^2 studio を使うユーザは、“Quick Install” オプションで簡単な手順を選択することをお勧めします。“Quick Install” オプションでは、デフォルトで e^2 studio、FSP、GCC ARM Embedded Toolchain をインストールします。“Quick Install” を選択した場合は、手順(6)から(8)の画面は表示されません。

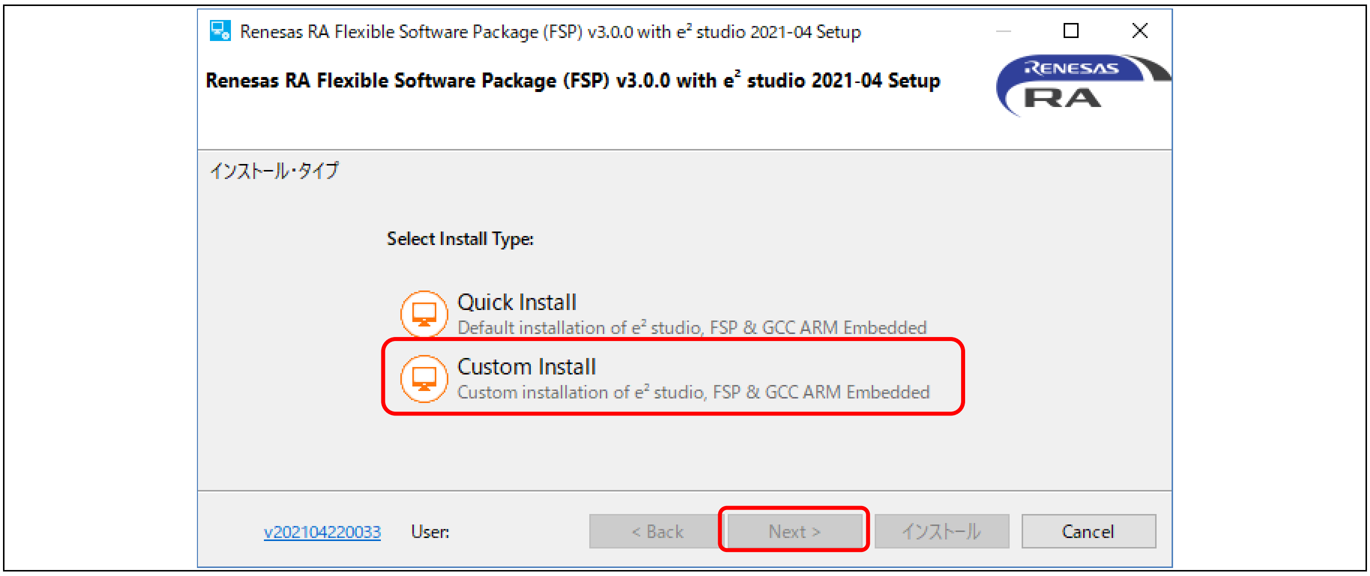


図 4　インストール ― インストール・タイプの選択

(5) [ようこそ] ページ
デフォルトのフォルダを使用するか、あるいは [変更…] をクリックしてフォルダを変更できます。[Next] で次に進みます。

図 5　インストール ― [ようこそ] ページ

(6) [拡張機能] ページ
必要な機能を選択し、[Next] ボタンを押してください。
“Quick Install” を選択した場合は、このページは表示されません。

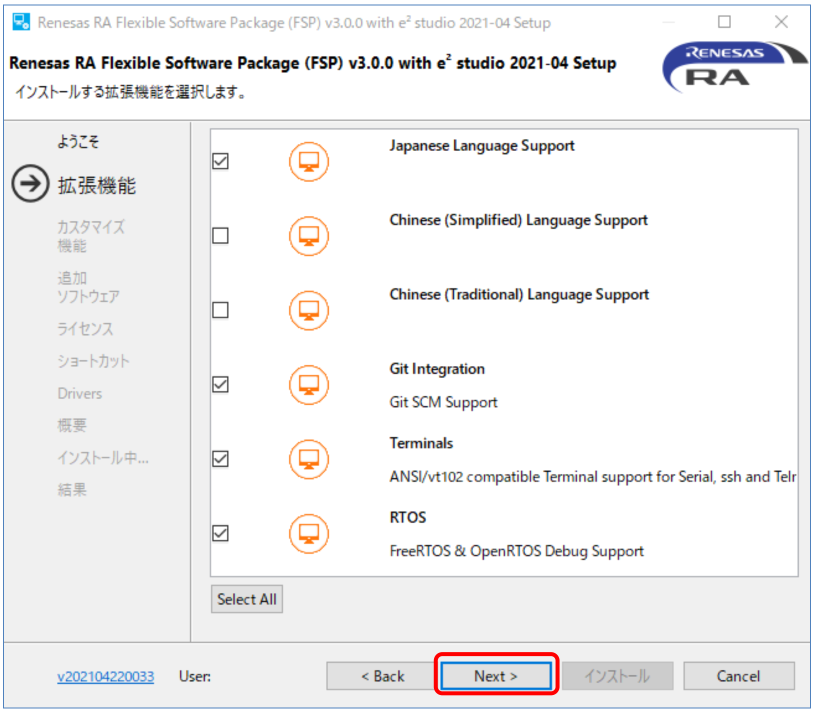


図 6 インストール ー 拡張機能

(7) [カスタマイズ機能] ページ
インストールするコンポーネントを選択し、[Next] ボタンで次に進みます。
“Quick Install” を選択した場合は、このページは表示されません。

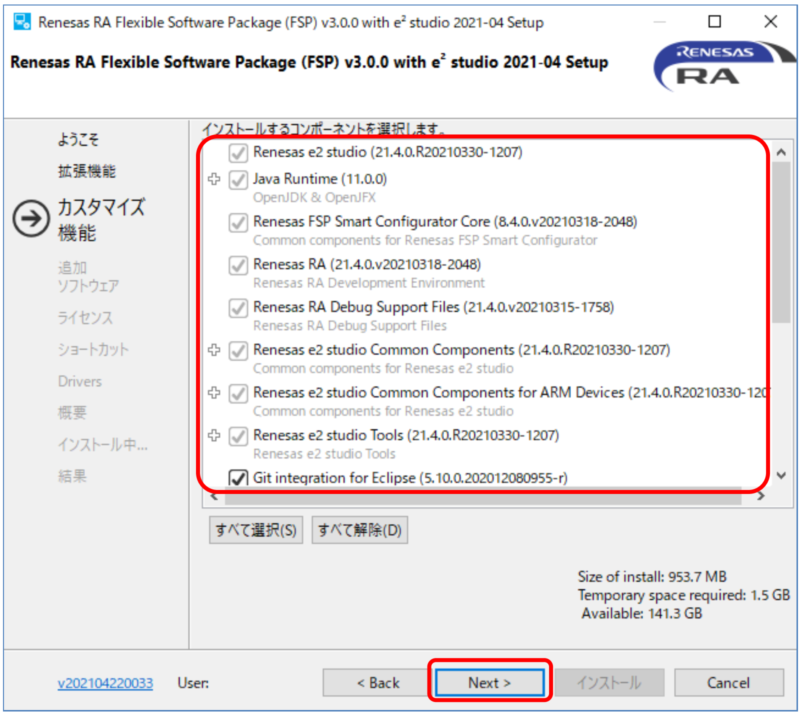


図 7 インストール ー カスタマイズ機能

(8) [追加ソフトウェア] ページ
インストールする “GNU ARM Embedded 9.3.1 2020q2” を選択し、[Next] ボタンを押してください。
“Quick Install” を選択した場合は、このページは表示されません。

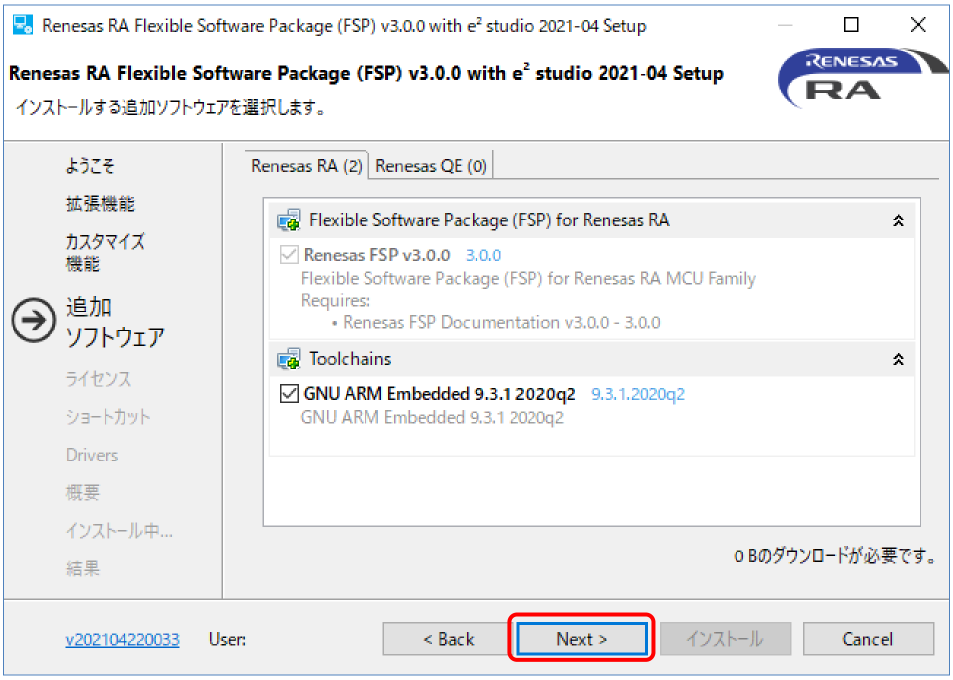


図 8 インストール ― 追加ソフトウェアの選択

(9) ソフトウェア契約に同意いただけましたらチェックを入れてください。[Next] で次に進みます。

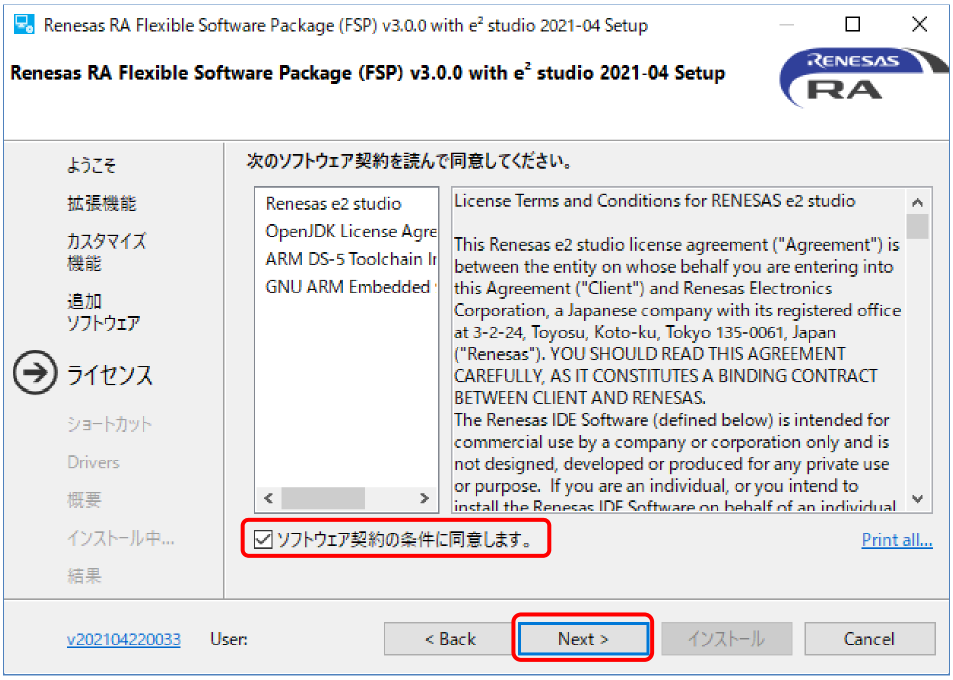


図 9 インストール ― ソフトウェア契約

(10) [ショートカット] ページ
　　スタートメニューに登録するショートカット名を入力します。[Next] で次に進みます。

注： すでに別のバージョンの e² studio がインストールされている場合は、それと区別が付くような名前に書き換えることをお勧めします。

図 10　インストール ー ショートカット

(11) [概要] ページの内容を確認してください。[インストール] ボタンを押すとインストールが始まります。

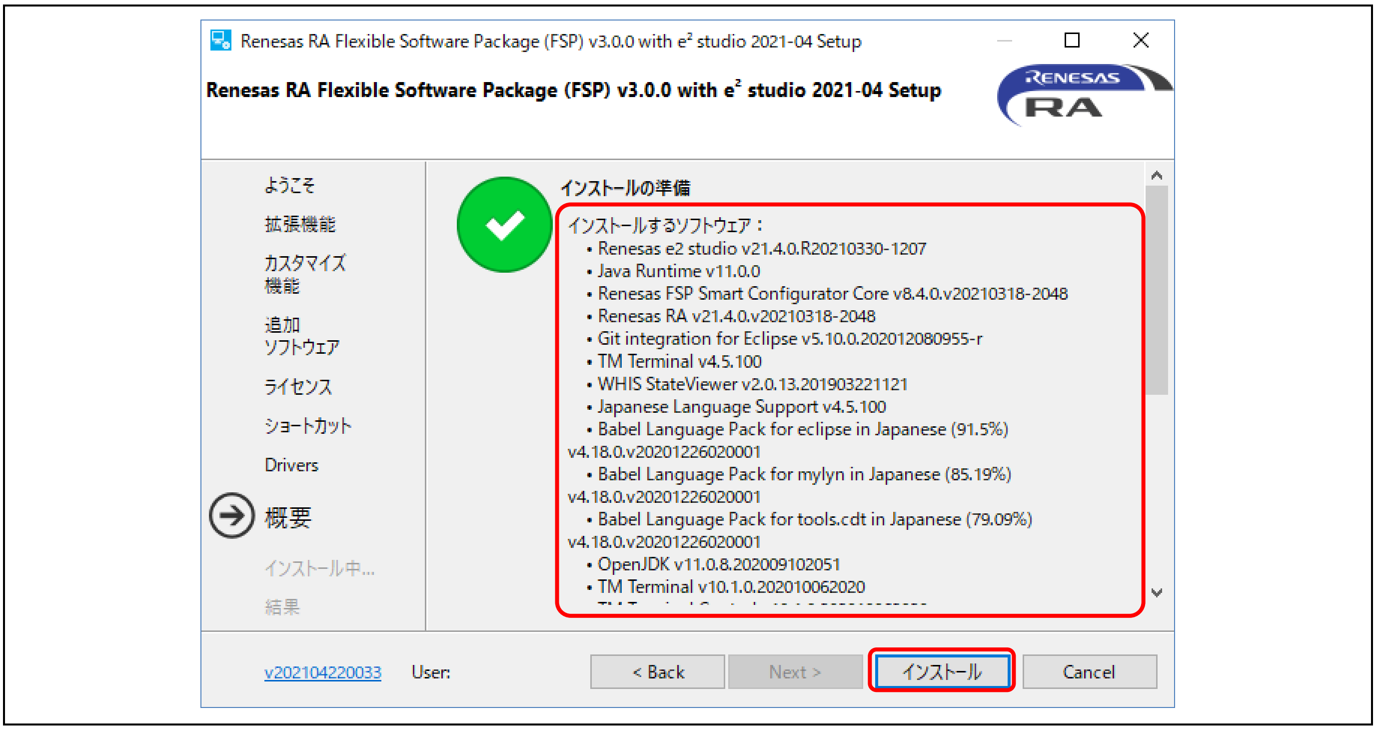


図 11　インストール ー 概要

(12) [OK] ボタンを押してインストールを終了します。

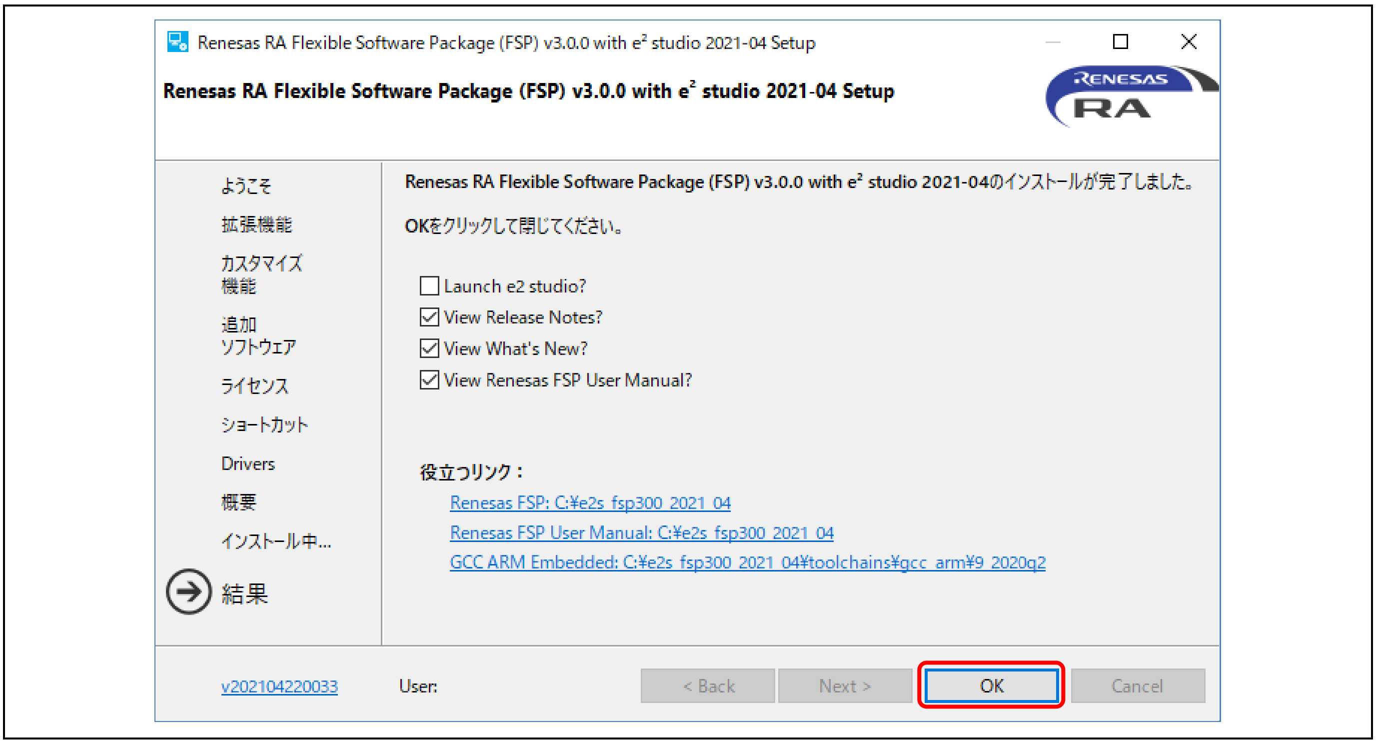


図 12　インストール ― インストールの完了

2.2 e² studio と FSP の個別インストール

この章では、以下のコンポーネントを個別にインストールする場合の手順を説明します。

- e2 studio
- GCC ARM Embedded Compiler
- ルネサス Flexible Software Package（FSP）

2.2.1 e² studio のインストール

以下の手順で、RA ファミリ用 e2 studio をインストールしてください。

(1) e2 studio 2021-04（64 ビットバージョン）オフラインインストーラを以下の場所からダウンロードしてください。
https://www.renesas.com/jp/ja/software-tool/e-studio の「Upgrades」

(2) ダウンロードしたファイルを解凍し、e2 studio インストーラを起動すると e2 studio インストールウィザードページが開きます。

(3) すでに e2 studio がインストールされている場合は、[変更]（インストール済 e2 studio の変更）、[削除]（アンインストール）、[インストール]（別の場所にインストール）の選択肢が表示されます。複数のバージョンを別々の場所にインストールすることも可能です。[Next] ボタンで次に進みます。

図 13　e² studio 複数バージョンのインストール

(4) [ようこそ] ページ
デフォルトのインストール先である “C:\Renesas\e2_studio” が設定されています。
[変更...] をクリックすると、インストール先を変更できます。[Next] ボタンで次に進みます。

図 14　インストール － [ようこそ] ページ

(5) [デバイス・ファミリ] ページ
“RA” を選択してください。必要に応じて他のデバイスファミリも選択可能です。
[Next] ボタンで次に進みます。

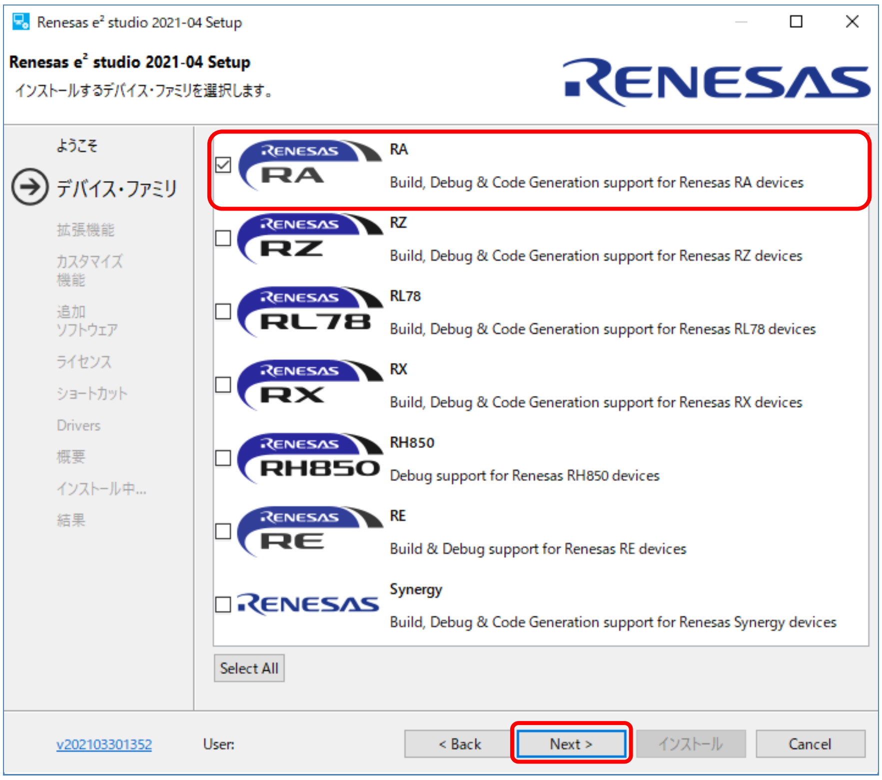


図 15　インストール － [デバイス・ファミリ] ページ

(6) [拡張機能] ページ

インストールする拡張機能（言語パック、Git、RTOS のサポートプラグイン）を選択してください。英語以外の言語を利用する場合はここで言語パックを選択しておく必要があります。

[Next] ボタンで次に進みます。

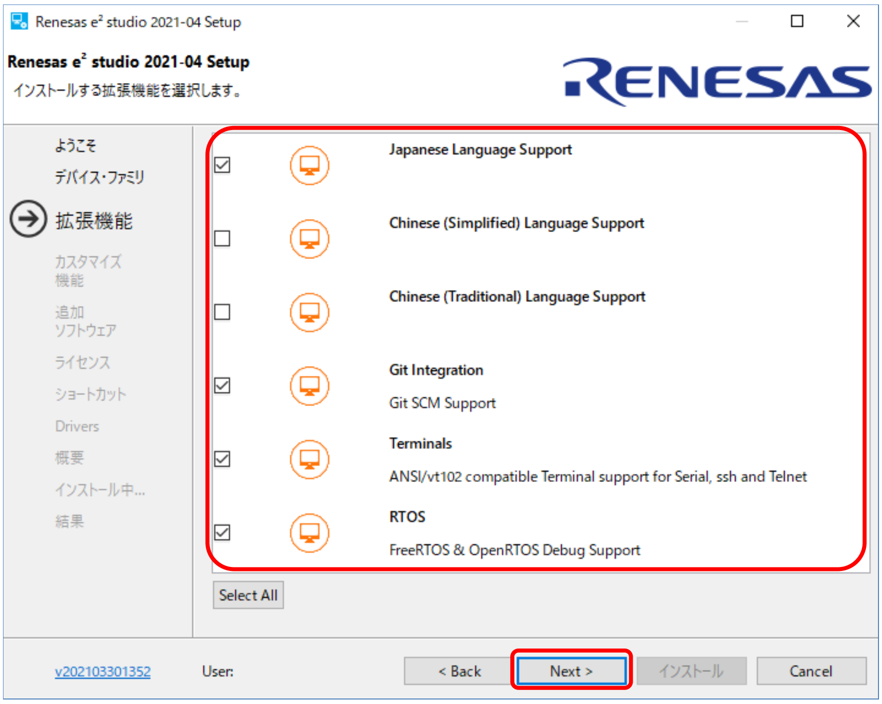


図 16　インストール ― [拡張機能] ページ

(7) [カスタマイズ機能] ページ

必ず “Renesas RA Family Support” を選択してください。[Next] ボタンで次に進みます。

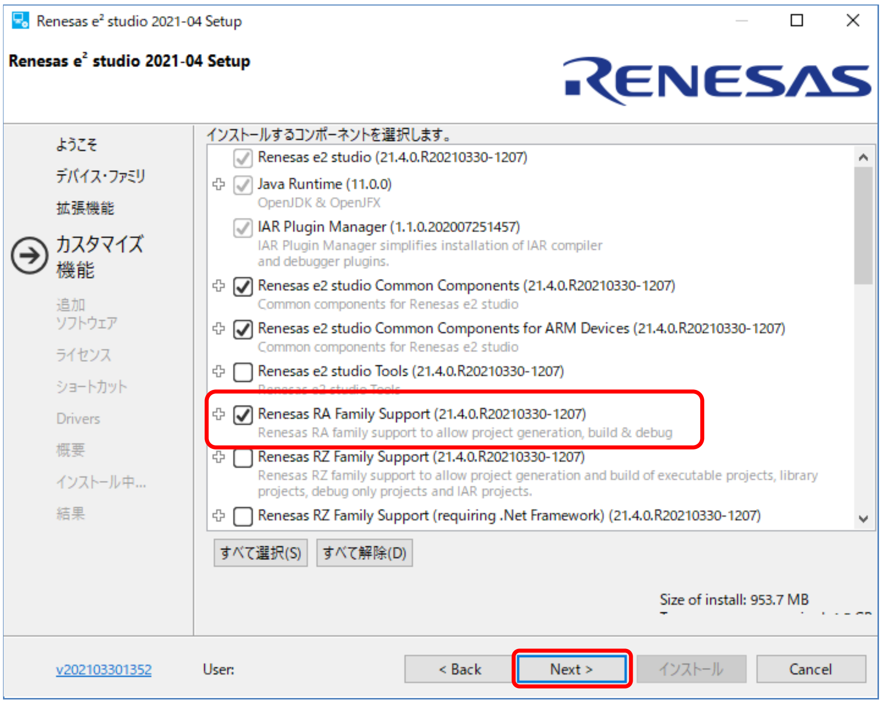


図 17　インストール ― [カスタマイズ機能] ページ

(8) [追加ソフトウェア] ページ

“GCC Toolchains & Utilities” タブを開き、“GNU Arm Embedded 9.3.1 2020q2” を選択して GNU Arm Embedded Toolchain をインストールします。

また、“Renesas FSP” タブを開き、ルネサス RA 用 FSP の最新バージョンを選択してインストールしてください。

[Next] ボタンで次に進みます。

注：インターネット接続のない環境ではソフトウェアカタログがダウンロードできないとの警告が表示され追加ソフトウェアはインストールできませんが、e² studio のインストールは続けることができます。

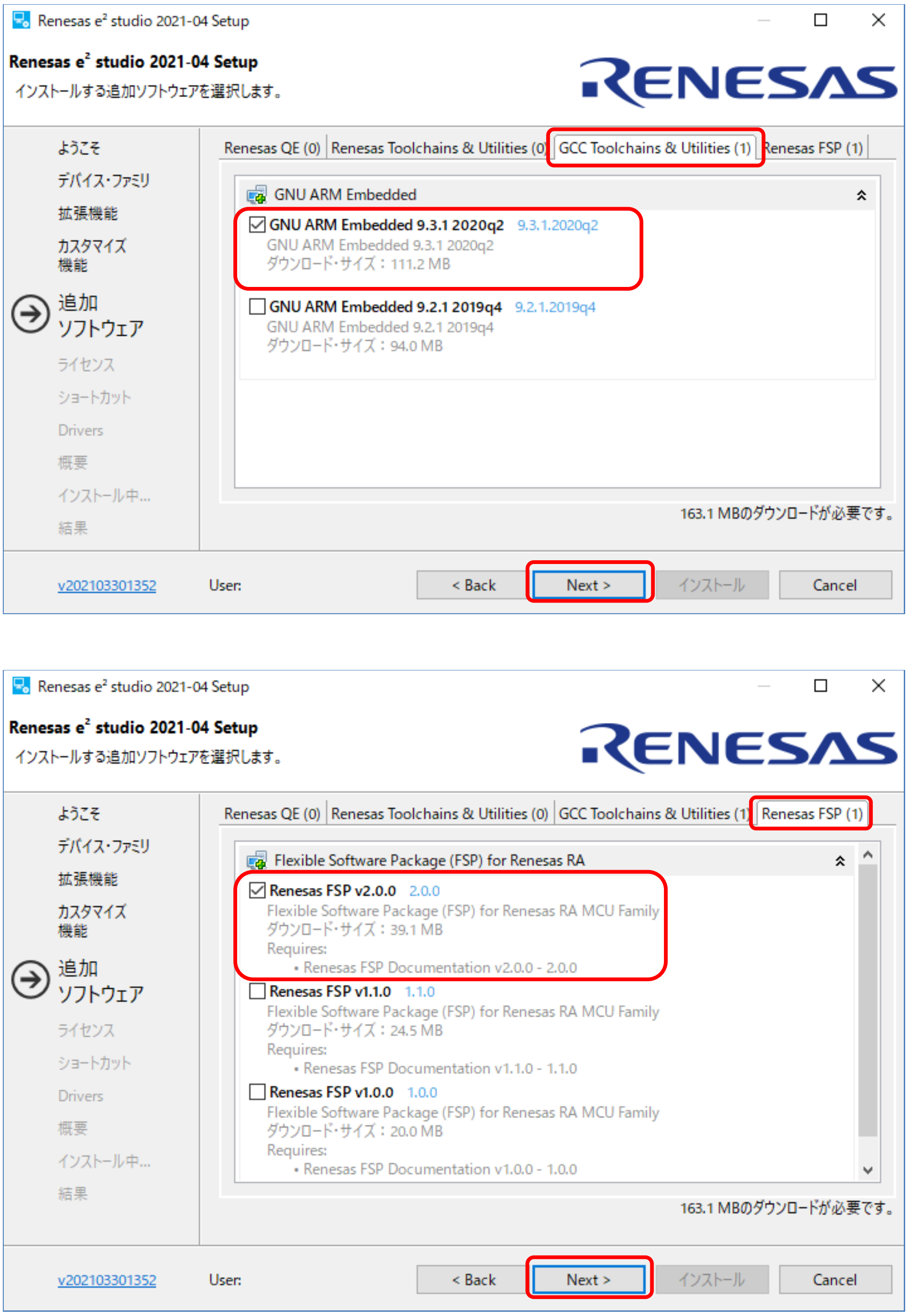


図 18 インストール ー [追加ソフトウェア] ページ

(9) ライセンス

ソフトウェア契約の条項を確認したのち、同意いただけましたらチェックを入れて [Next] ボタンを押してください。同意いただけない場合はインストールを継続できません。

(10)ショートカット

スタートメニューに登録するショートカット名を入力します。[Next] ボタンで次に進みます。

(11) 概要

[インストール] ボタンを押して e² studio をインストールしてください。

(12) インストール中...

インストールが始まります。選択した追加ソフトウェアによっては、そのインストーラが途中で起動しますので画面に従って操作を行ってください。

2.2.2　GNU Arm Embedded Toolchain の設定

GNU Arm Embedded Toolchain は e² studio のインストールと同時にインストールできます。また、同時にインストールせずに、e² studio のインストール後に別途インストールすることもできます。

GNU Arm Embedded Toolchain を別途インストールするには、以下の手順を実行してください。

(1) ルネサス RA ファミリがサポートする GNU Arm Embedded Toolchain の 9-2020-q2-update バージョン（gcc-arm-none-eabi-9-2020-q2-update-win32.exe）を次の場所からダウンロードします。
https://developer.arm.com/tools-and-software/open-source-software/developer-tools/gnu-toolchain/gnu-rm/downloads

(2) ホスト PC で GNU Arm Embedded Toolchain のインストーラを実行します。

(3) インストールする言語を選択します。インストール確認の画面で [Yes] を選択してください。

(4) インストールウィザードでは、すべてデフォルト設定のままにします。

(5) [Install wizard Complete] の画面が表示されたら、“Add path to environment variable” を選択し、[Finish] ボタンを押してインストールを完了します。

2.2.3　ルネサス RA Flexible Software Package（FSP）のインストール

ルネサス RA Flexible Software Package（FSP）は e² studio のインストールと同時にインストールできます。また、同時にインストールせずに、e² studio のインストール後に別途インストールすることもできます。

FSP を別途インストールするには、以下の手順を実行してください。

(1) GitHub のルネサス RA ファミリ・マイクロコントローラ用 Flexible Software Package（FSP）のページにアクセスしてください。
https://github.com/renesas/fsp/releases

(2) 最新の FSP パッケージインストーラ（“FSP_Packs_<version>.exe”）をダウンロードしてください。FSP パッケージインストーラには、ドライバライブラリ、HTML 版のユーザーズマニュアル、readme ファイルが含まれています。

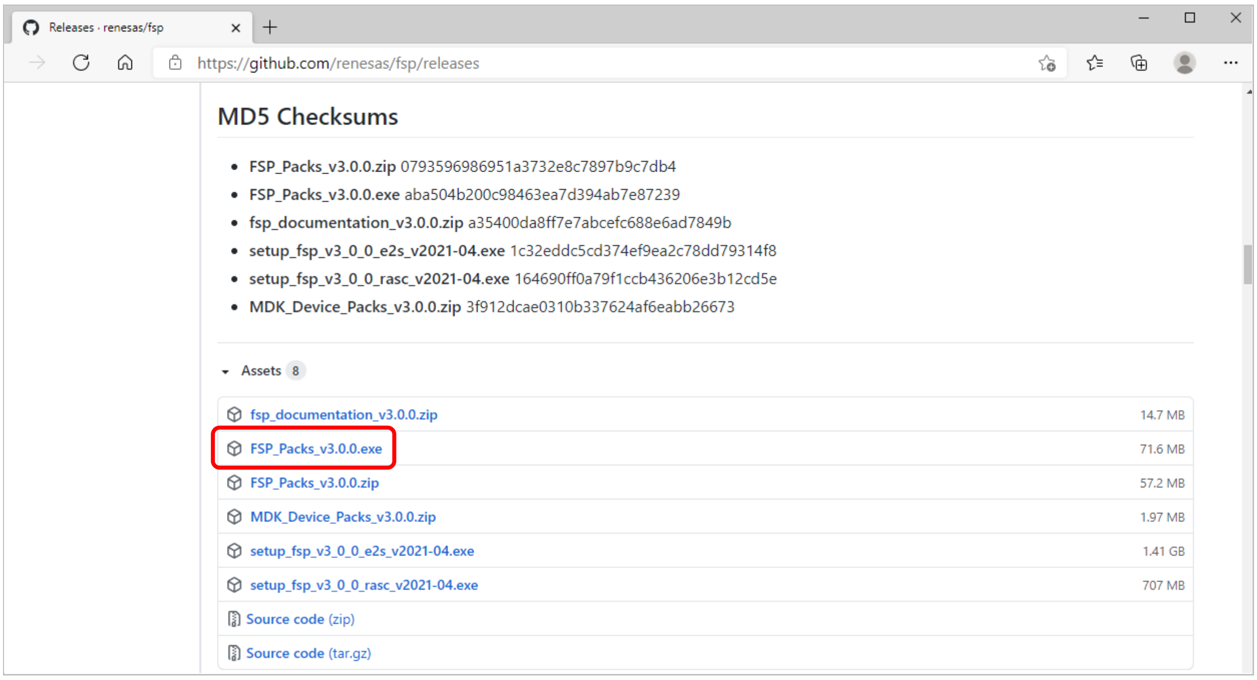


図 19　インストール ー FSP パッケージのダウンロード

(3) 互換性のある e^2 studio がインストール済みであること、e^2 studio が実行中でないことを確認してください。また、インストール中は e^2 studio を実行しないでください。

(4) FSP パッケージインストーラを実行してください。[Next] で次に進みます。

(5) [I Agree] をクリックして、契約条項に同意してください。

(6) e^2 studio をインストールしたフォルダ（例：C:\Renesas\e2_studio）を選択し、[Install] ボタンを押してください。

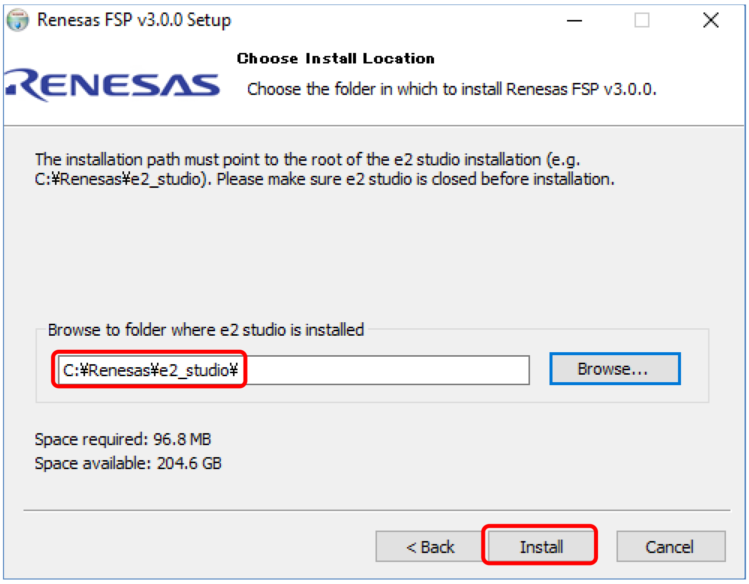


図 20　インストール先の選択

(7) [Finish] ボタンを押してインストールを終了します。

2.3　e² studio の更新

e^2 studio を更新するには、新しいバージョンの e^2 studio インストーラ（プラットフォームインストーラまたは標準の e^2 studio インストーラ）を実行します。インストーラのダウンロード手順は「2 章　インストール」を参照してください。

インストール済みのバージョンを上書きしないように注意してください。e^2 studio の更新前に、旧バージョンをアンインストールします。旧バージョンと新バージョンを両方使用したい場合は、新しいフォルダを作成してそのフォルダを新バージョンのインストール先にします。

2.4　FSP の更新

FSP を更新するには、新しいバージョンの FSP インストーラを実行します。インストーラのダウンロード手順は、「2.2.3　ルネサス RA Flexible Software Package（FSP）のインストール」を参照してください。

2.5　e² studioのアンインストール

e^2 studio のアンインストールは、Windows OS での通常のプログラムアンインストール手順で行えます。

(1) [スタート] → [コントロールパネル] → [プログラムと機能] を選択します。
(2) インストール済みプログラムのリストから、“e^2 studio” を選択し、[アンインストール(U)] ボタンを押してください。
(3) [アンインストール] ダイアログボックスの [はい] ボタンを押して削除を実行してください。

アンインストール処理の最後に、e^2 studio はインストール先から削除され、Windows のショートカットメニューも削除されます。

注：コントロールパネルに該当する e^2 studio が表示されないか、アンインストールができない場合は、以下のフォルダにあるアンインストーラを直接実行してください。
{e^2 studio のインストールフォルダ}/uninstall/uninstall.exe

2.6 Keil MDK および IAR EWARM 用 RA スマート・コンフィグレータのインストール

RA スマート・コンフィグレータ（RA SC）は、サードパーティ製の IDE（Keil MDK および IAR EWARM）を使用する際に、ルネサス RA ファミリ・マイクロコントローラ向けプロジェクト用にデバイスハードウェアの設定（クロック設定や端子割り当てなど）や FSP ソフトウェアコンポーネントの初期化を行なうデスクトップアプリケーションです。

以下の手順で RA SC インストーラをダウンロードしてインストールしてください。

(1) GitHub のルネサス RA ファミリ・マイクロコントローラ用 Flexible Software Package（FSP）のページにアクセスしてください。
https://github.com/renesas/fsp/releases

(2) RA SC インストーラ（“setup_fsp<version>_rasc_<version >.exe”）をダウンロードしてください。

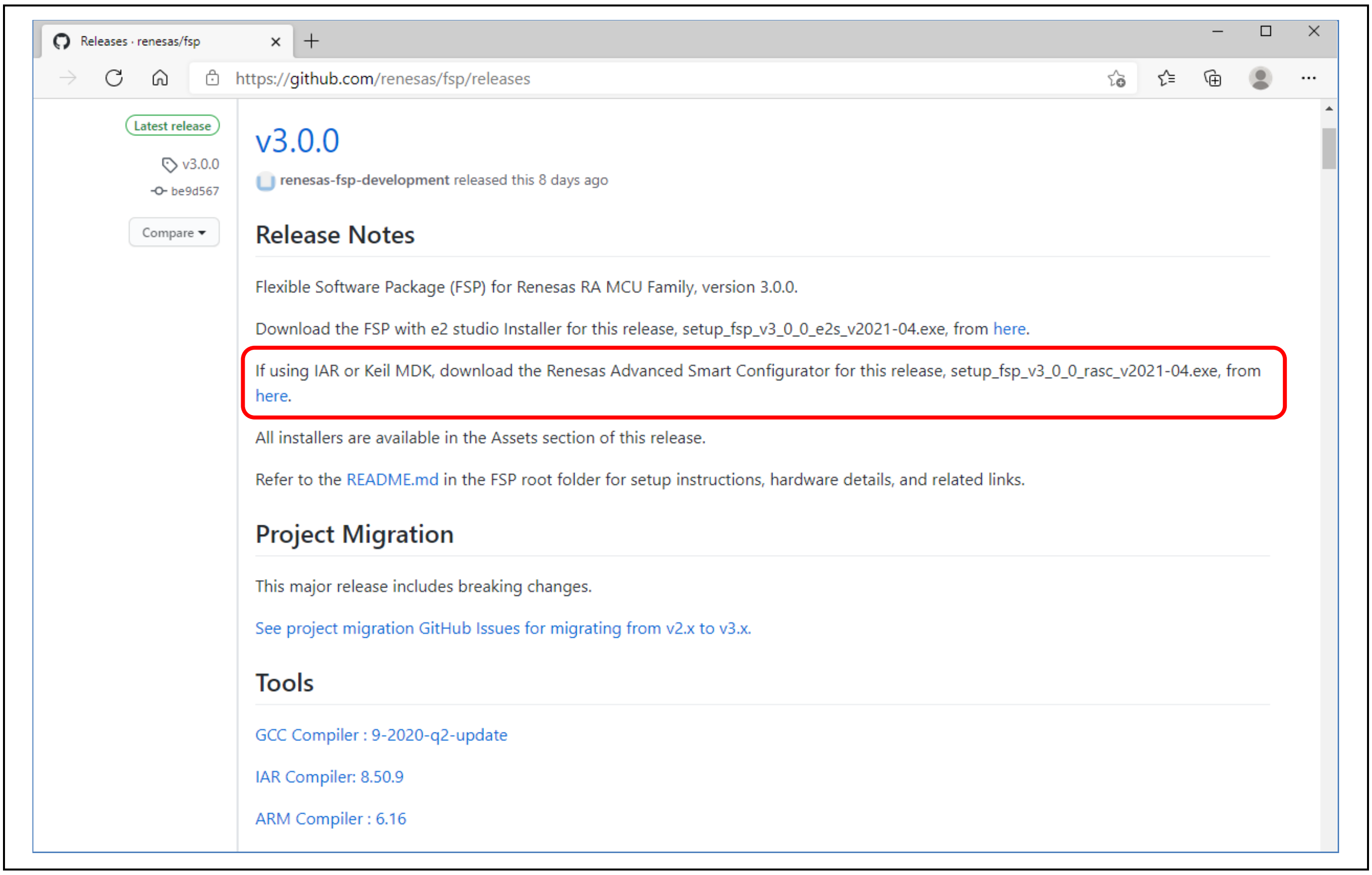


図 21　インストール ー RA SC パッケージのダウンロード

(3) インストールファイルを実行します。

(4) [ようこそ] ページ
デフォルトのフォルダを使用するか、あるいは [変更...] をクリックしてフォルダを変更できます。[Next] で次に進みます。

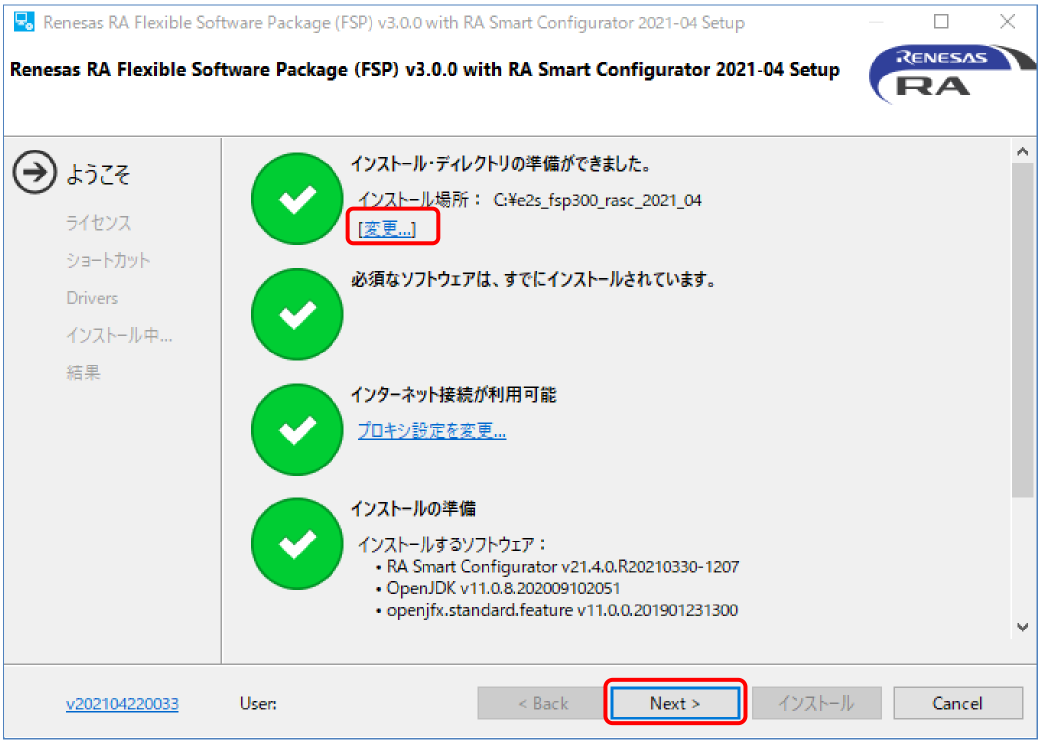


図 22　インストール － [ようこそ] ページ

(5) ソフトウェア契約に同意いただけましたらチェックを入れてください。[Next] で次に進みます。

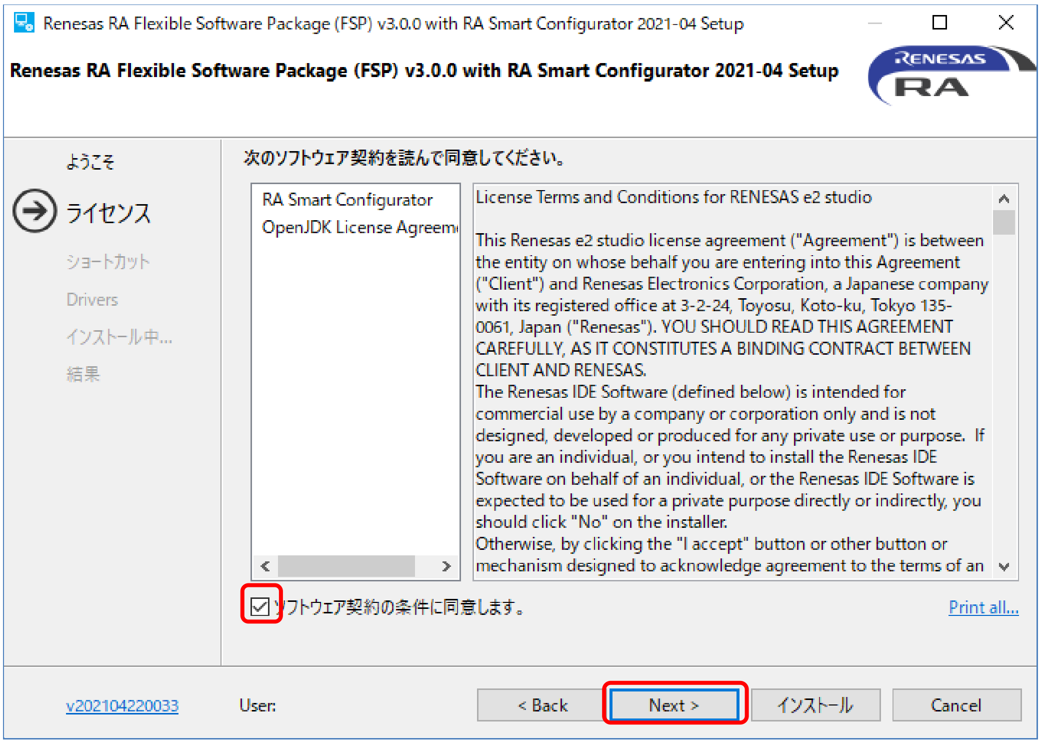


図 23　インストール － ソフトウェア契約

(6) ショートカット名を確認してください。[インストール] ボタンを押すと次に進みます。

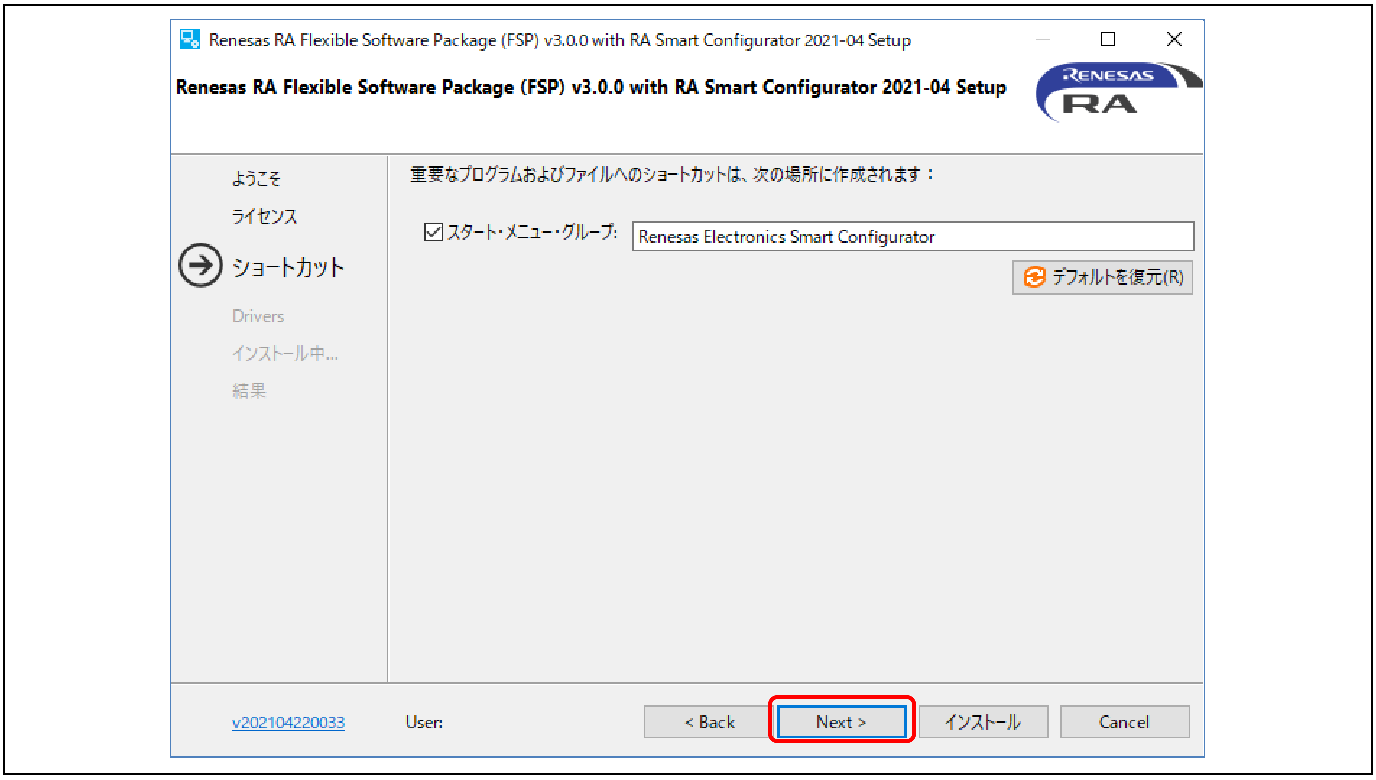


図 24　インストール ― ショートカット

(7) [OK] ボタンを押してインストールを終了します。

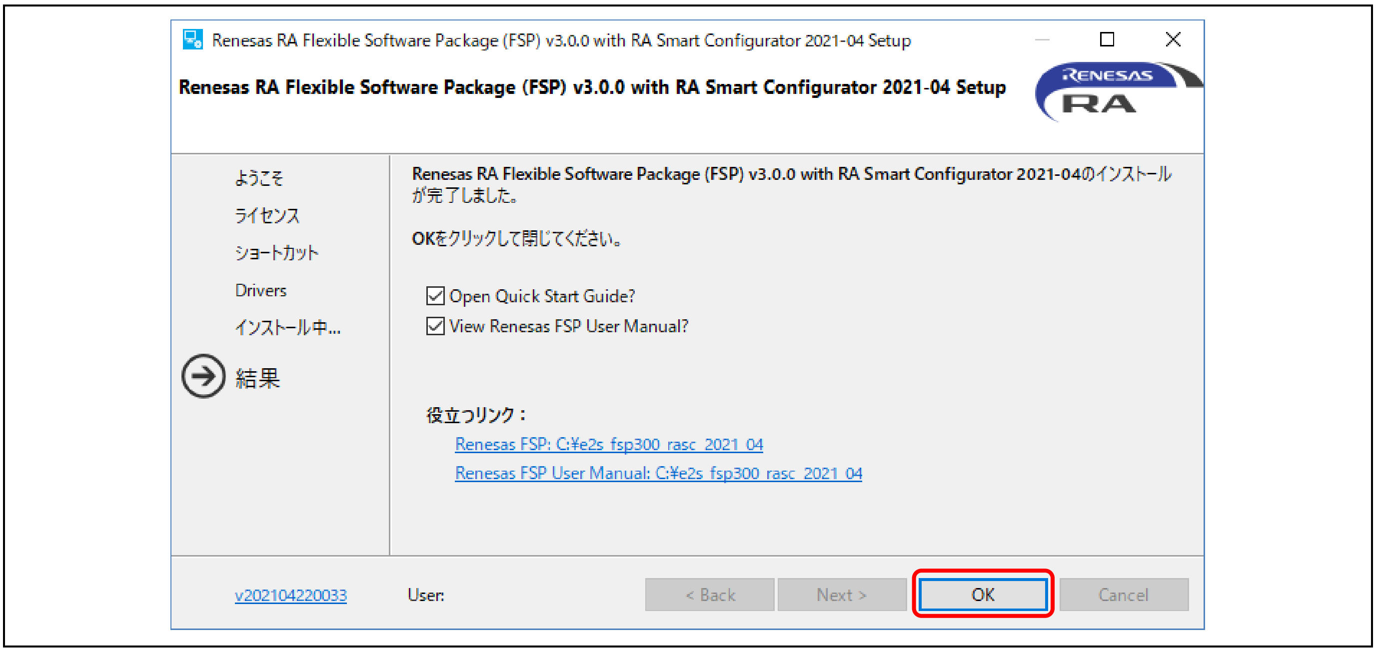


図 25　インストール ― インストールの完了

注：Keil MDK や IAR EWARM と RA SC を使用する方法の詳細については、FSP ドキュメントの「RA SC User Guide for MDK and IAR」を参照してください。
https://renesas.github.io/fsp/_s_t_a_r_t__d_e_v.html#RASC-MDK-IAR-user-guide

3. プロジェクトの作成

この章では、新規 RA プロジェクトの作成について説明します。e2 studio は、新規プロジェクトをすぐに作成できるようなウィザードを用意しています。このウィザードは、使用する RA デバイスとユーザボードに適したプロジェクトを作成できます。

RA 用プロジェクトジェネレータでは、端子構成、割り込み、クロック構成を設定可能で、必要なドライバソフトウェアも設定できます。

プロジェクト作成の前に、「2 章　インストール」の説明にしたがって FSP とツールチェーンがホスト PC にインストールされていることが必要です。

3.1 TrustZone に対応していないデバイス用の新規 RA プロジェクトの作成

この章では、TrustZone に対応していないデバイス用の RA プロジェクトの作成方法を説明します。TrustZone 対応デバイスについては、「3.2　TrustZone 対応デバイス用の新規 RA プロジェクトの作成」を参照してください。

e2 studio ではプロジェクト作成用に簡単なウィザードが用意されています。プロジェクト名、対応するデバイスとボード、ボードレベルドライバを指定して、新規 RA プロジェクトを作成できます。

e2 studio を起動し、ワークスペースフォルダを選択します。 以下の手順で新規 RA プロジェクトを作成してください。

(1) [ファイル] → [新規] → [Renesas C/C++ Project] → [Renesas RA]の順に選択します。

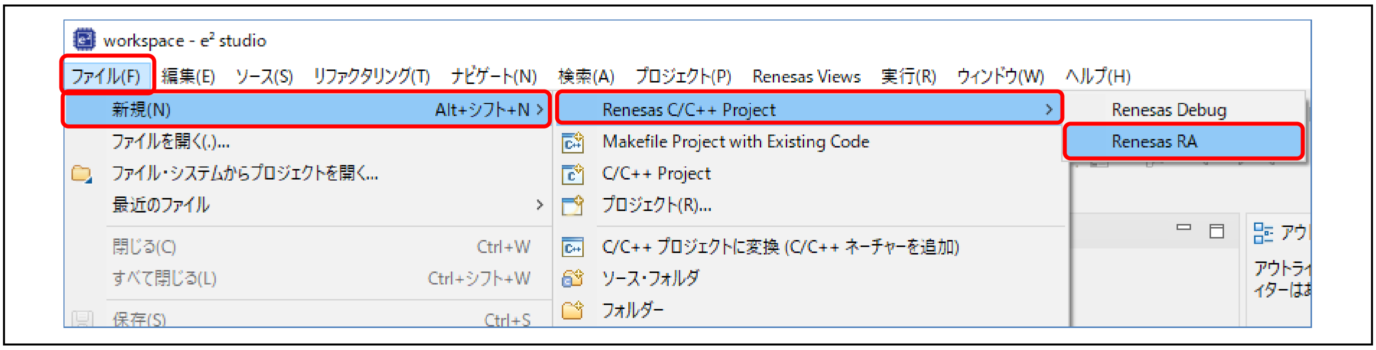


図 26　プロジェクトの作成 ― 新規プロジェクトの作成

(2) “Renesas RA: Renesas RA C/C++ Project” テンプレートを選択してください。[次へ] で次に進みます。

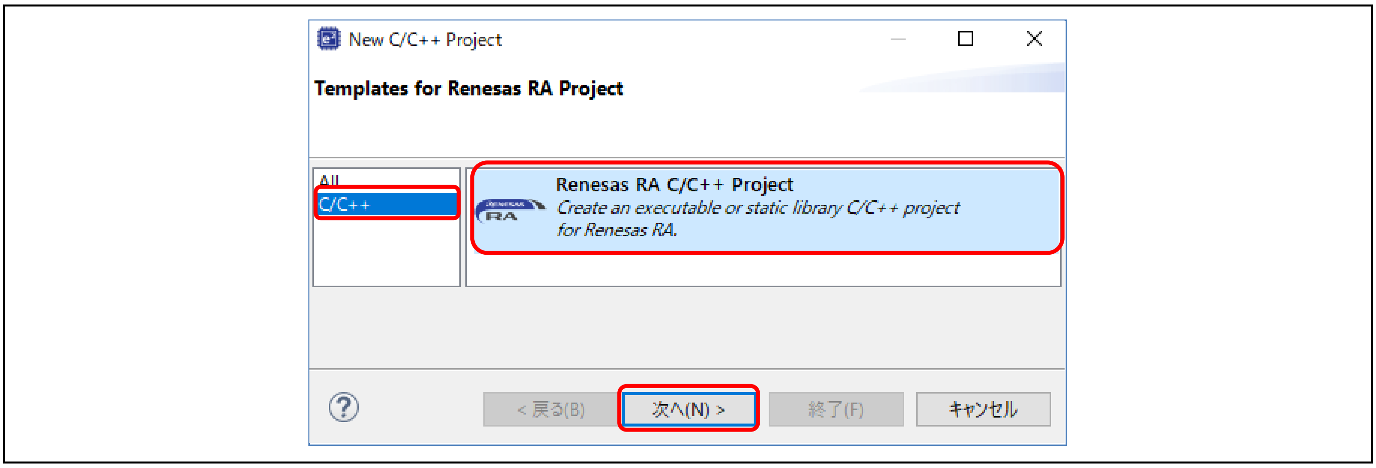


図 27　プロジェクトの作成 ― RA プロジェクトの選択

(3) プロジェクト作成ウィザードで、プロジェクト情報を次のように入力します。

Project name：プロジェクト名を入力します（例："RA_Tutorial"）。

デフォルト・ロケーションの使用：選択します。別の場所にプロジェクトを作成したいときは、このチェックボックスを選択せずに新しい作成先を入力します。

[次へ] で次に進みます。

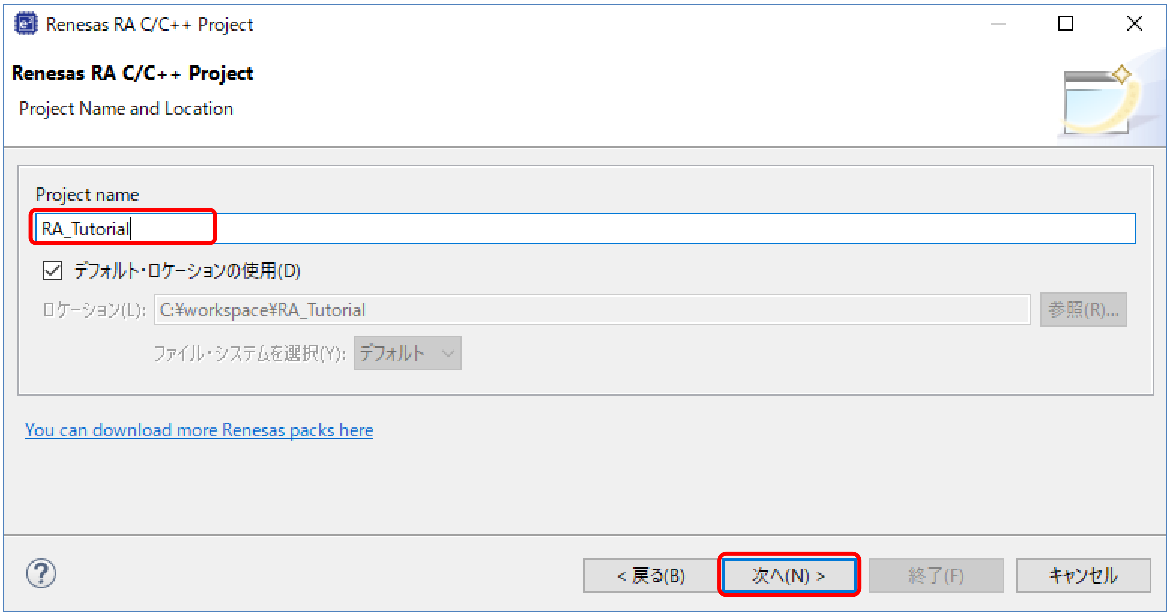


図 28 プロジェクトの作成 ― 新規 RA プロジェクト作成ウィザード

(4) デバイス選択の画面で、デバイスとツールの情報を入力します。

Board：EK-RA6M3

Toolchains：ルネサス RA ファミリ用に認定された最新の GNU Arm Embedded Toolchain を選択します（例：GCC ARM Embedded 9.3.1.20200408）。

Debugger：J-Link ARM

その他はすべてデフォルト設定のままにします。

[次へ] で次に進みます。

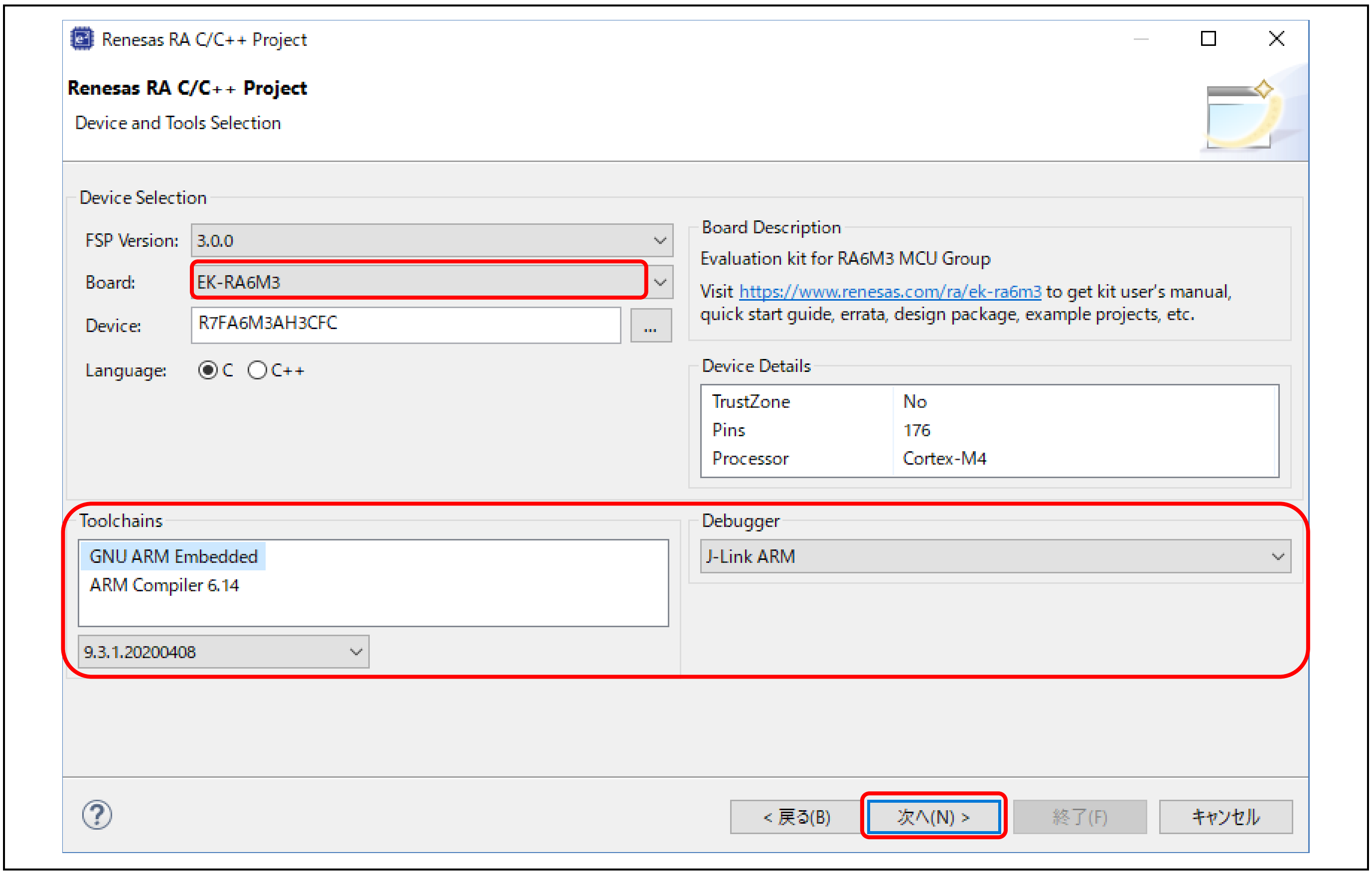


図 29 プロジェクトの作成 ― デバイスの選択

(5) Build Artifact Selection：Executable を選択します。

RTOS Selection：No RTOS を選択します。

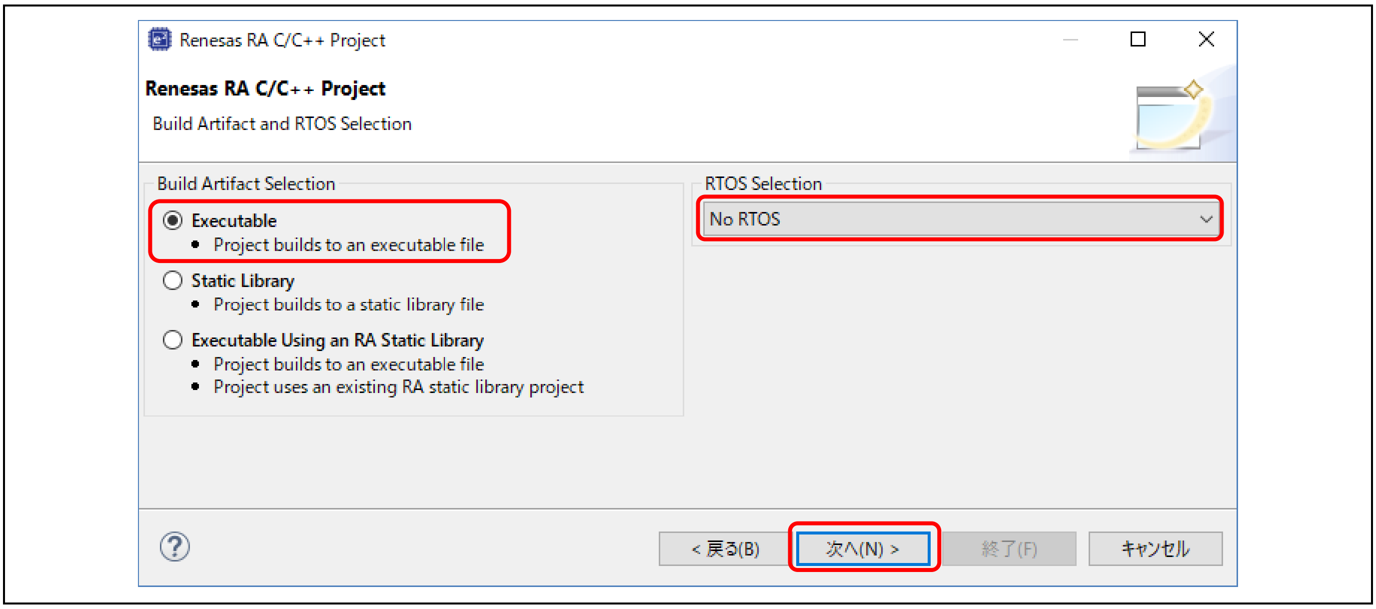


図 30　Artifact と RTOS の選択

(6) プロジェクトテンプレート選択の画面で、Blinky のテンプレートを選択します。

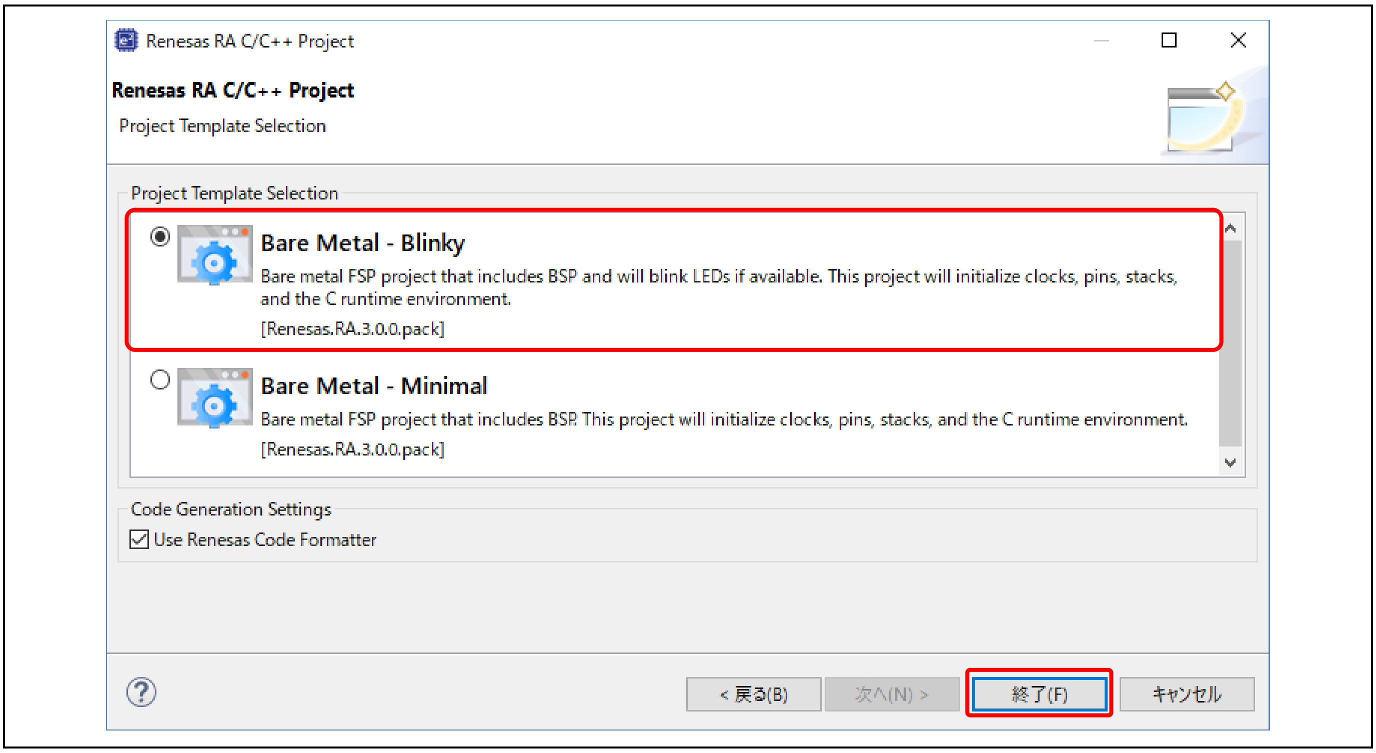


図 31　プロジェクトの作成 ― プロジェクトテンプレート

(7) [終了] ボタンを押すと新規プロジェクトが作成されます。

[FSP Configuration] パースペクティブを開くようメッセージが表示される場合があります。[パースペクティブを開く] を選択して開いてください。

（Eclipse で「パースペクティブ」とは、あらかじめペインとビューを組み合わせてレイアウトを定義したものを指します。）

e2 studio が作成する新規プロジェクトには、[プロジェクトエクスプローラー] ビュー、[RA プロジェクトコンフィグレーションエディタ] ビュー、[Visualization] ビューなど各種ビューが用意されています。

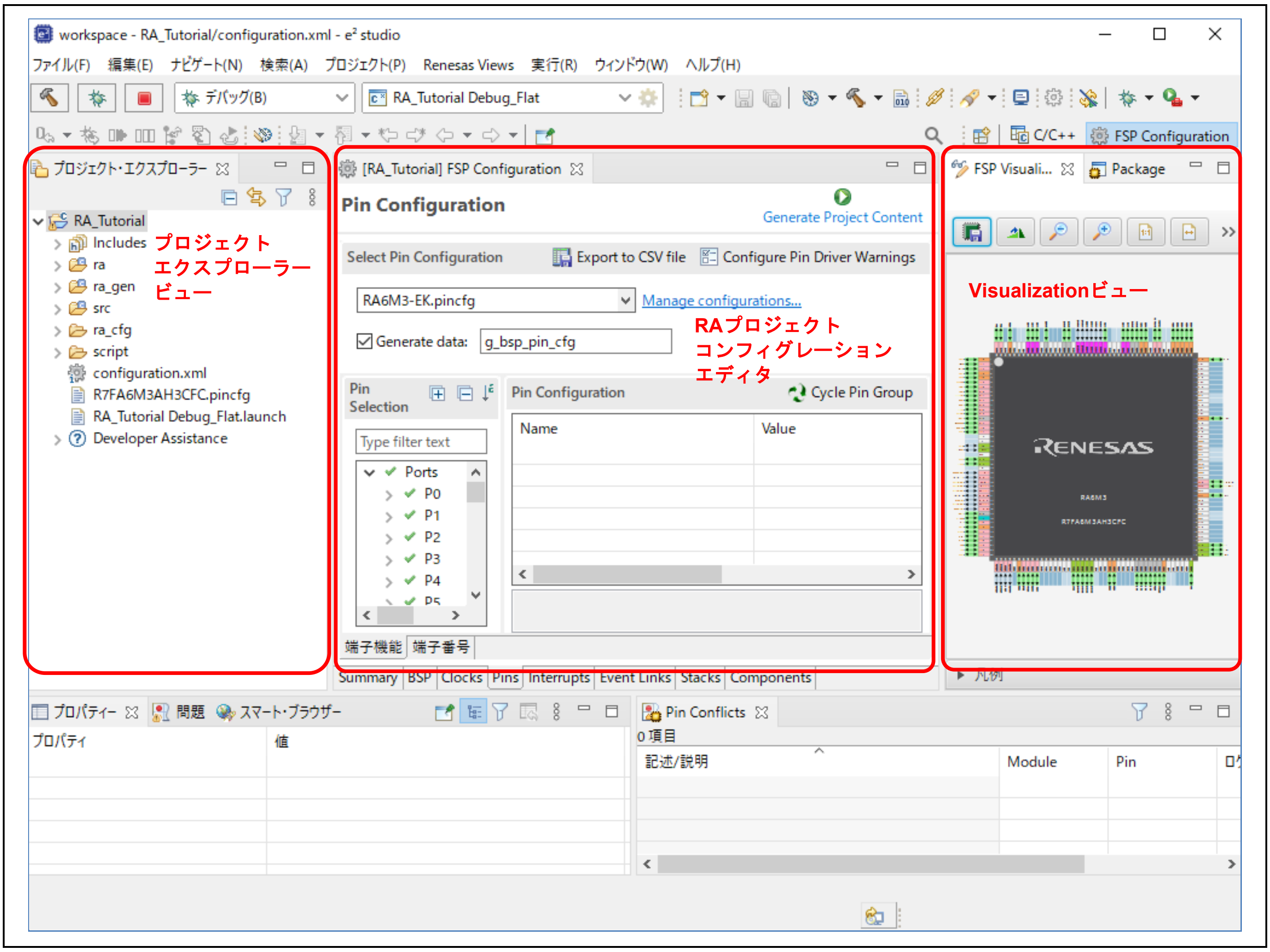


図 32　プロジェクトの作成 ― 新規プロジェクトのビュー

3.2 TrustZone 対応デバイス用の新規 RA プロジェクトの作成

この章では、TrustZone 対応デバイス用の RA プロジェクトの作成方法を説明します。TrustZone に対応していないデバイスについては、「3.1　TrustZone に対応していないデバイス用の新規 RA プロジェクトの作成」を参照してください。

Arm® TrustZone® テクノロジでは、セキュアおよび非セキュア両方のアプリケーション用のプロジェクト作成が可能です。RA Arm® TrustZone® ツールの詳細については、以下のリンクを参照してください。
https://www.renesas.com/jp/ja/document/apn/ra-arm-trustzone-tooling-primer.

フラットプロジェクト（TrustZoneに対応しないプロジェクト）では、セキュリティパーティションのない自己完結したELF実行ファイルを作成するので、ターゲットデバイスで即時実行するのに適しています。

3.2.1 フラットプロジェクト（TrustZone に対応していないプロジェクト）

新規のフラットプロジェクト（TrustZone に対応していないプロジェクト）は以下の手順で作成します。

(1) メニューから、[ファイル] → [新規] → [Renesas C/C++ Project] → [Renesas RA] の順に選択します。

(2) “Renesas RA: Renesas RA C/C++ Project” テンプレートを選択してください。[次へ] で次に進みます。

(3) プロジェクト作成ウィザードで、プロジェクト情報を次のように入力します。

Project name：プロジェクト名を入力します（例：“RA_Flat”）。

デフォルト・ロケーションの使用：選択します。別の場所にプロジェクトを作成したいときは、このチェックボックスを選択せずに新しい作成先を入力します。

[次へ] で次に進みます。

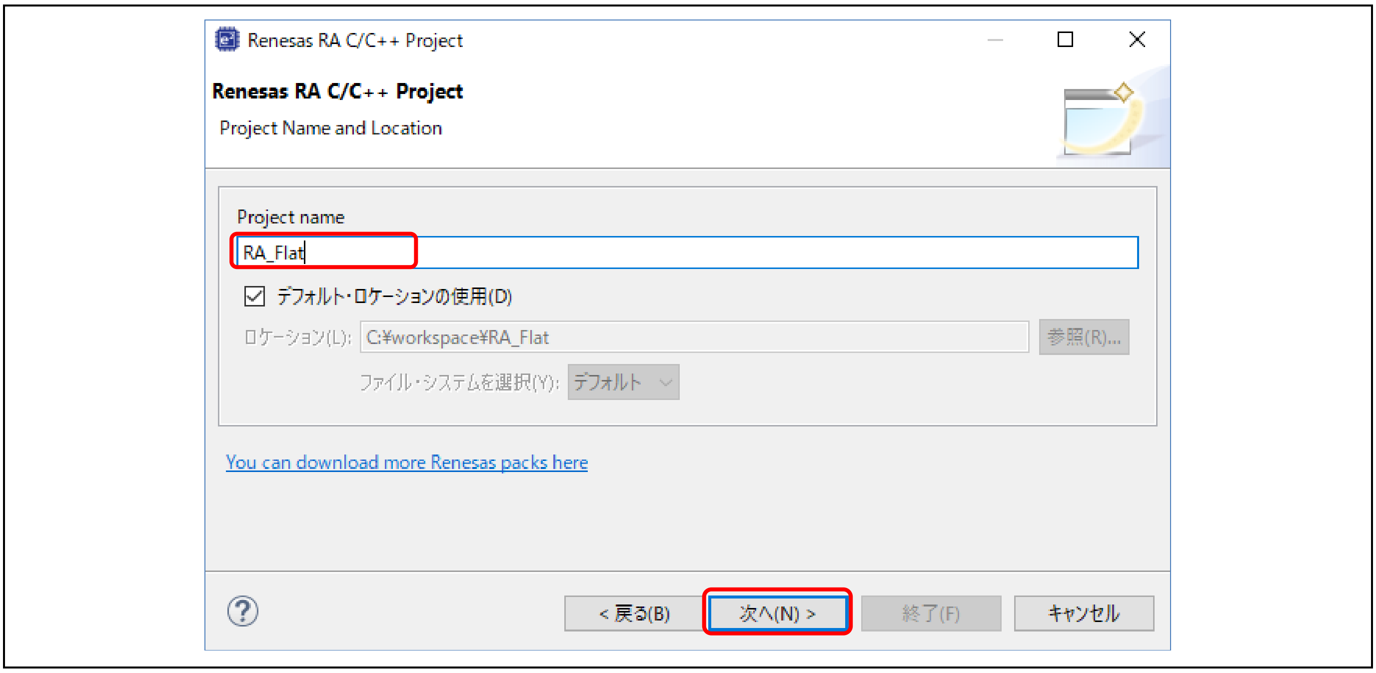


図 33　プロジェクトの作成 ― 新規 RA プロジェクト作成ウィザード

(4) デバイスとツール選択の画面で、デバイスとツールの情報を入力します。

Board：EK-RA6M4

Toolchains：ルネサス RA ファミリ用に認定された最新の GNU Arm Embedded Toolchain を選択します（例：GCC ARM Embedded 9.3.1.20200408）。

Debugger：J-Link ARM

その他はすべてデフォルト設定のままにします。

[次へ] で次に進みます。

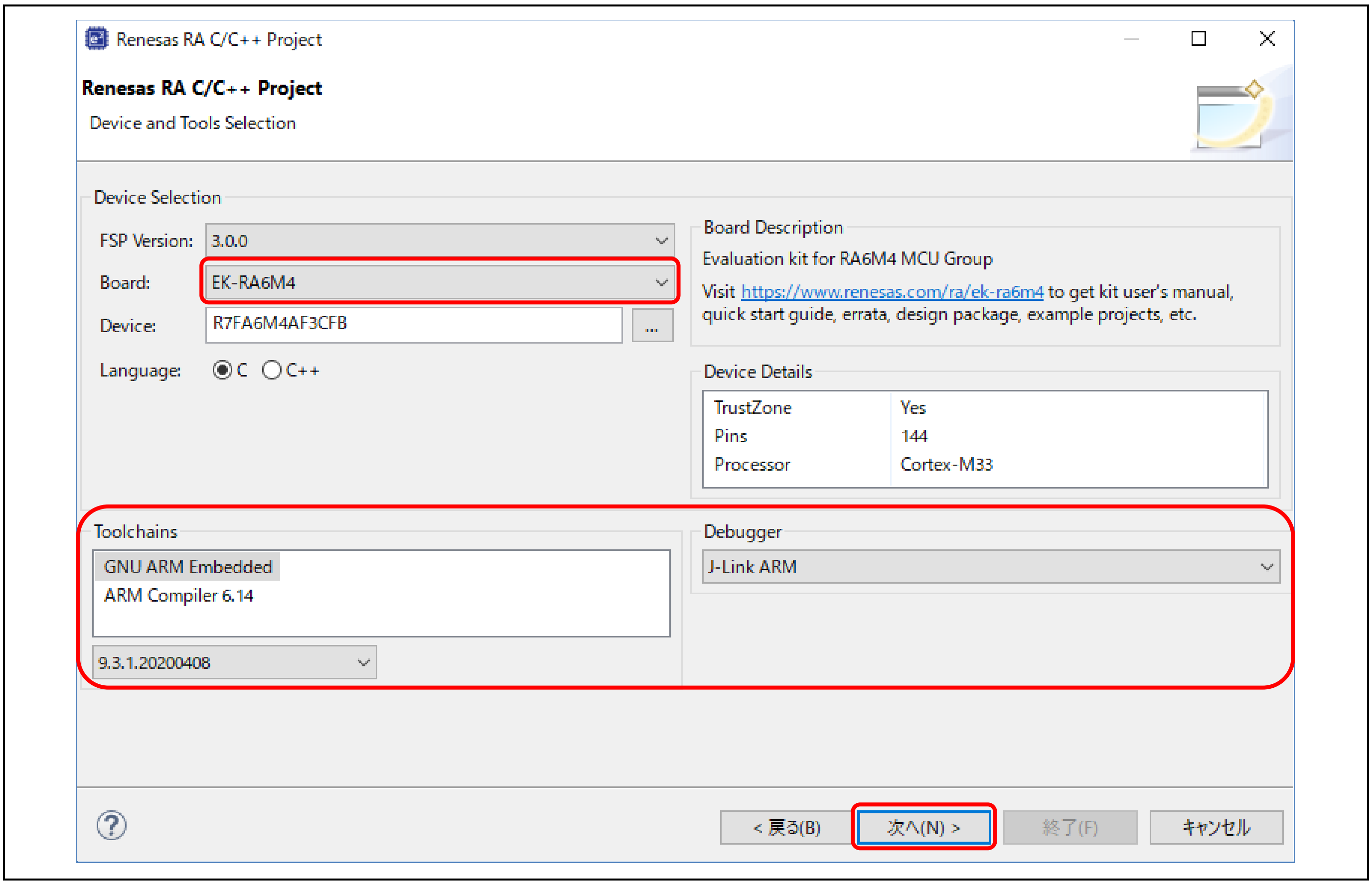


図 34　プロジェクトの作成 ― デバイスの選択

(5) Project Type Selection：Flat (Non-TrustZone) Project を選択します。

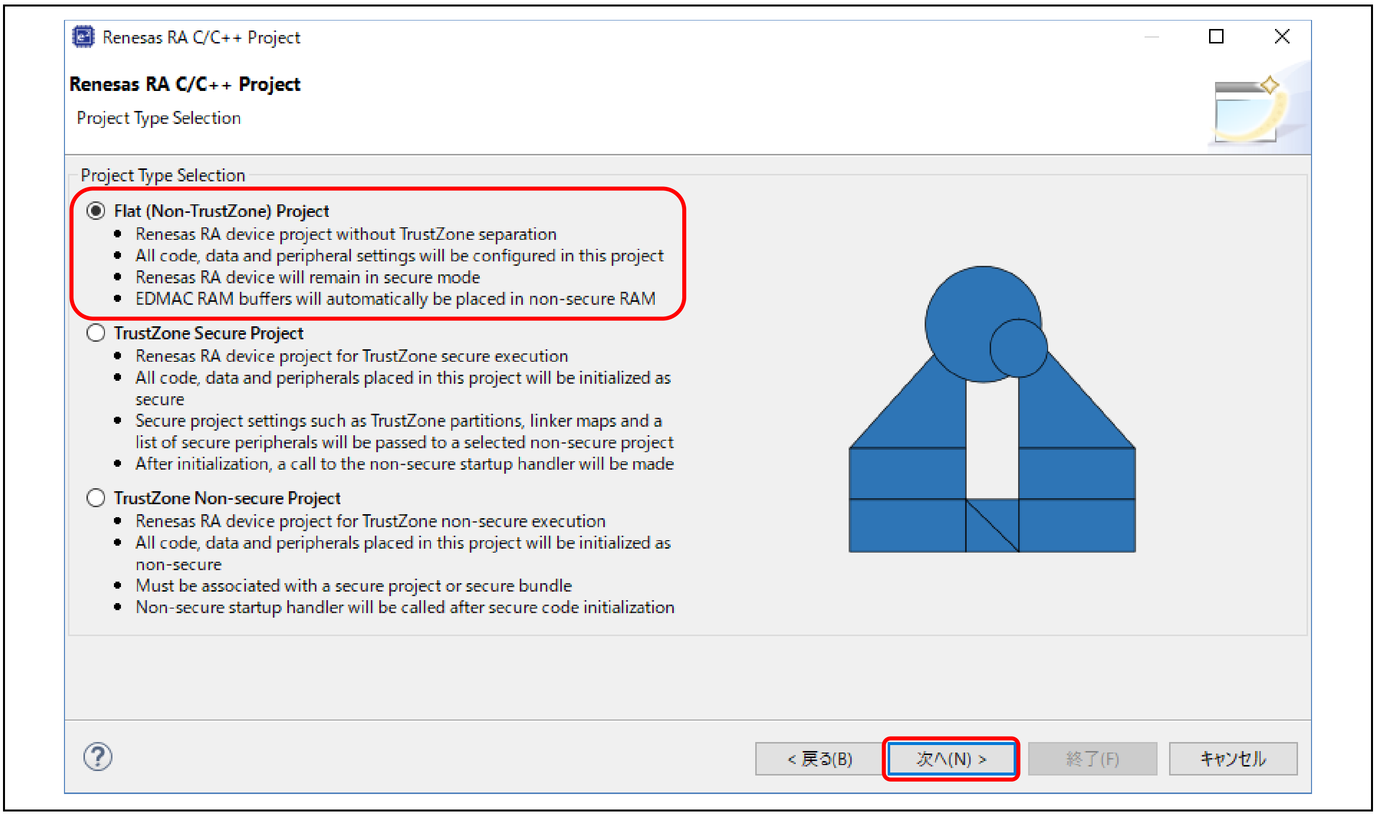


図 35　プロジェクトタイプの選択

(6) Build Artifact Selection：Executable を選択します。

RTOS Selection：No RTOS を選択します。

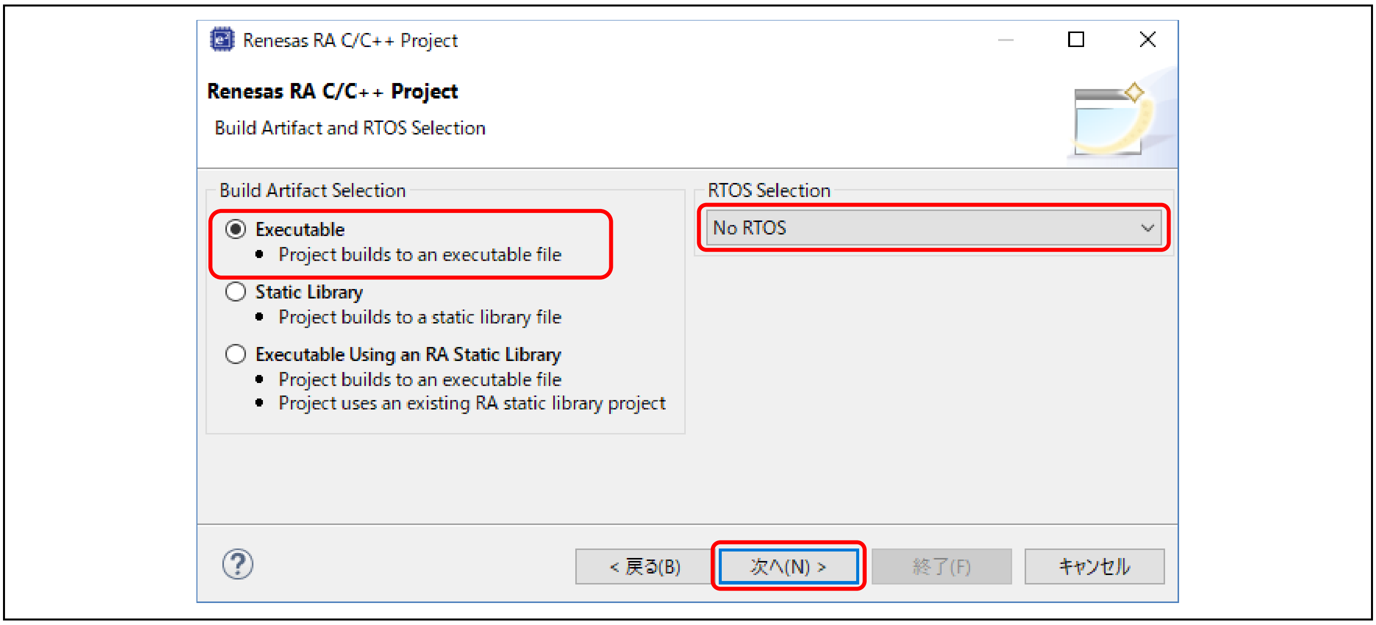


図 36　Artifact と RTOS の選択

(7) プロジェクトテンプレート選択の画面で、テンプレートを選択してください（例：Blinky）。[終了] ボタンを押すとプロジェクトが作成されます。

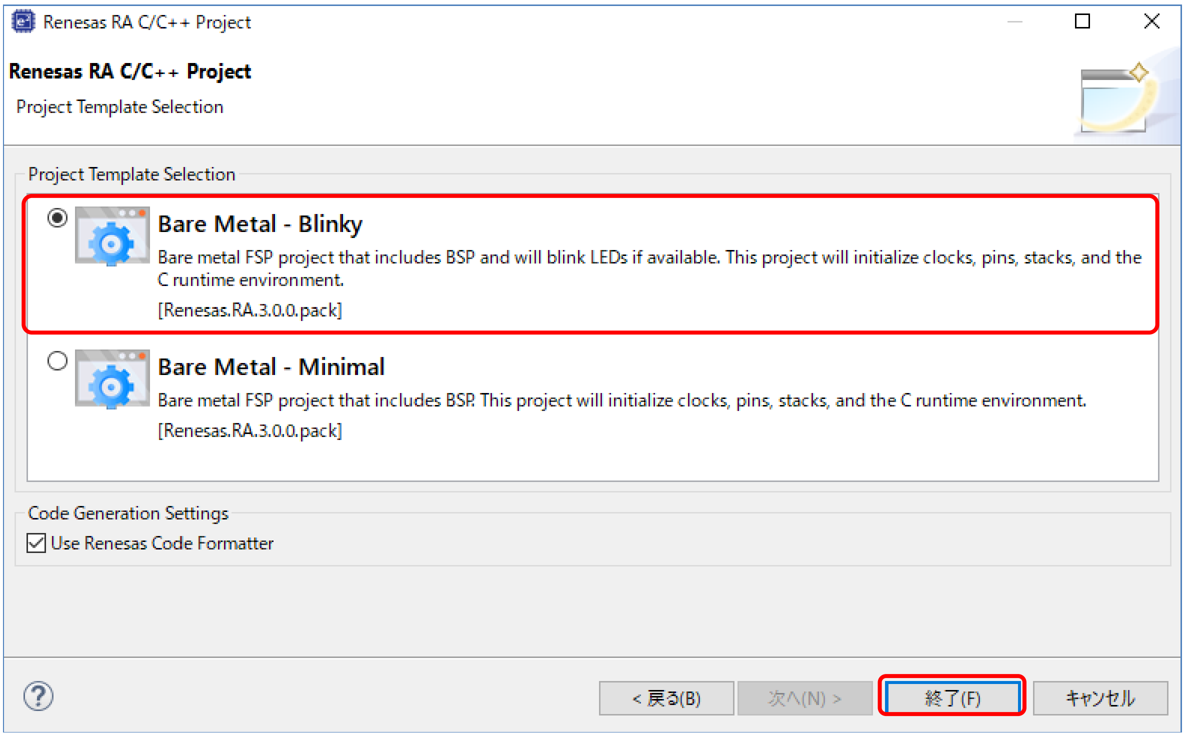


図 37　プロジェクトの作成 ― プロジェクトテンプレート

(8) プロジェクト名を右クリックし、"プロジェクトのビルド" を選択してください。プロジェクトが正しくビルドされます。

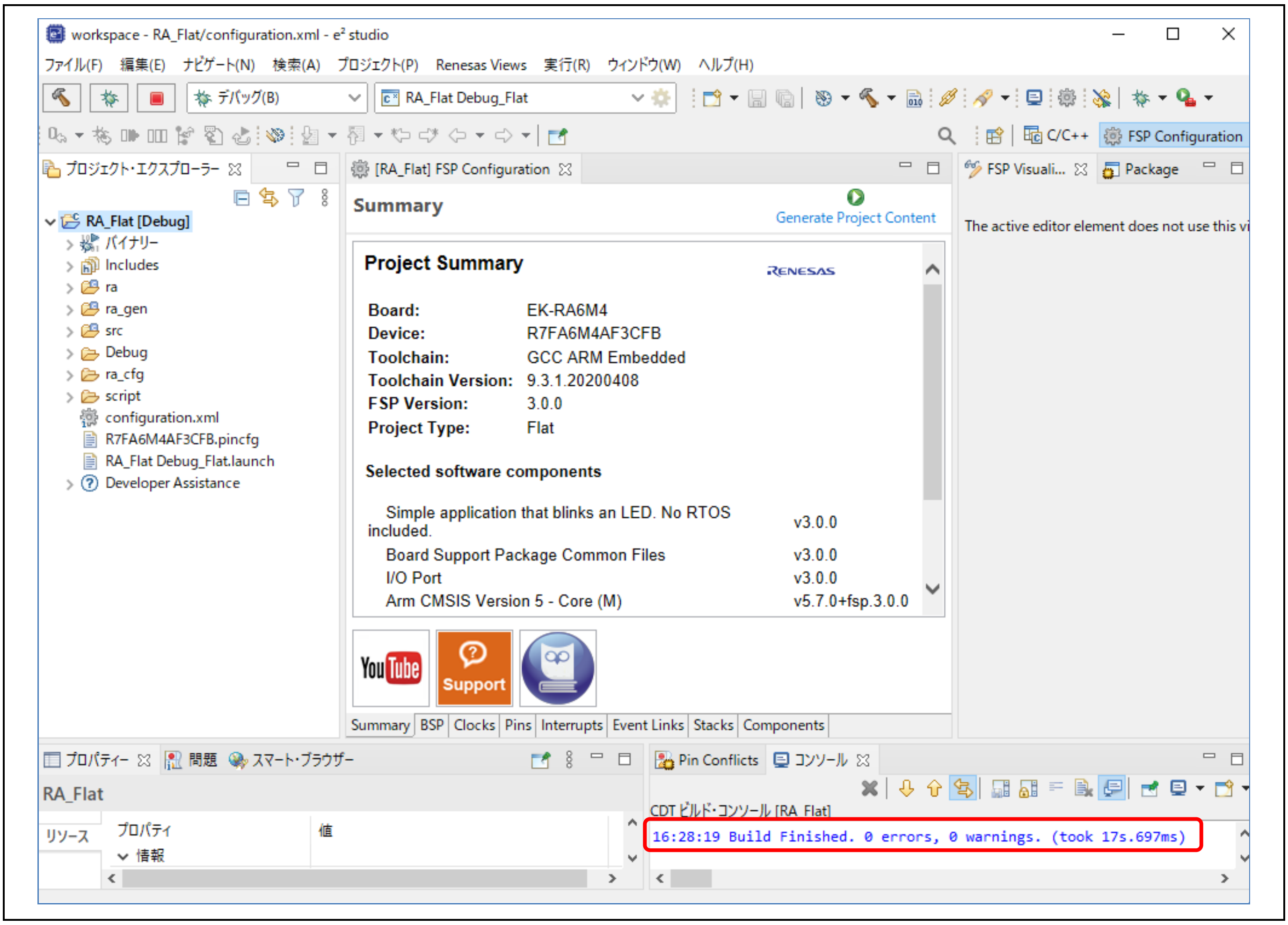


図 38　プロジェクトのビルド成功

3.3 既存 RA プロジェクトのインポート

既存の RA プロジェクトは以下の手順でインポートできます。

(1) [ファイル] → [インポート] の順に選択します。

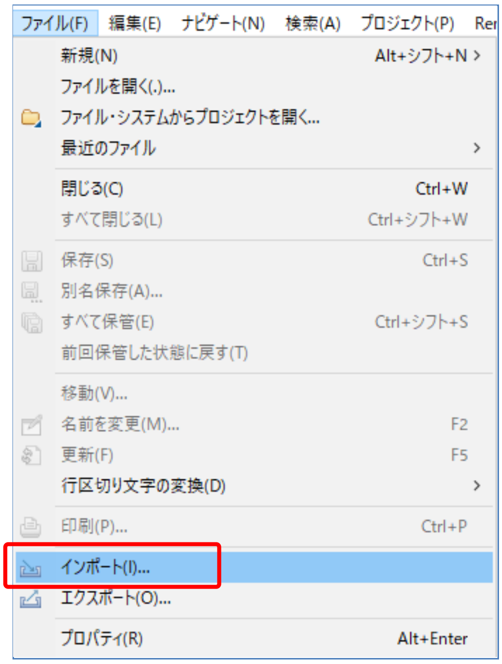


図 39 プロジェクトのインポート

(2) [インポート] 画面の [一般] → [既存プロジェクトをワークスペースへ] を選択してください。[次へ] で次に進みます。

注：インポートするプロジェクトの名称を変更するには、[一般] → [Rename & Import Existing C/C++ Project into Workspace] を選択してください。

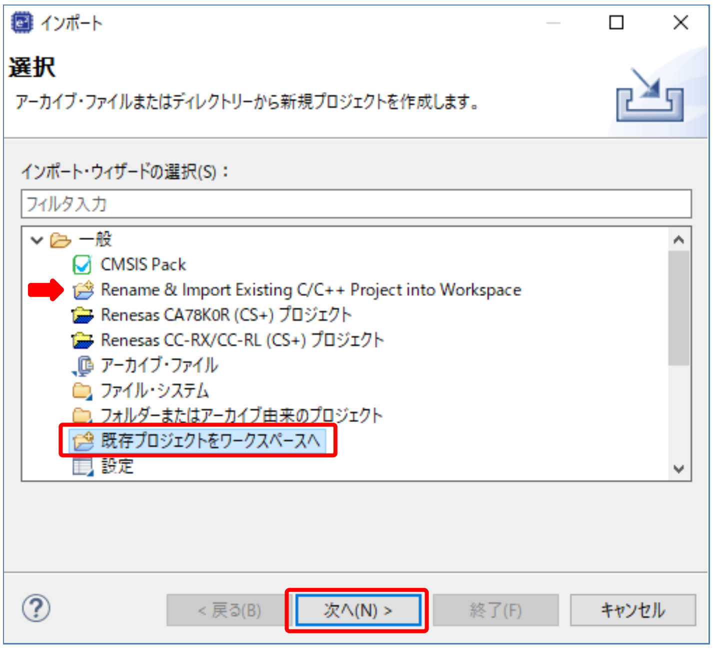


図 40 インポート方法の選択

(3) [プロジェクトをインポート] 画面で “アーカイブ・ファイルの選択” を選び、右の [参照...] ボタンでプロジェクトが入っている圧縮ファイル（.zip）を選択します。
既存プロジェクトがフォルダに保存されている場合は、“ルート・ディレクトリーの選択” を選んでください。

(4) インポートするプロジェクトを選択して [終了] ボタンを押してください。

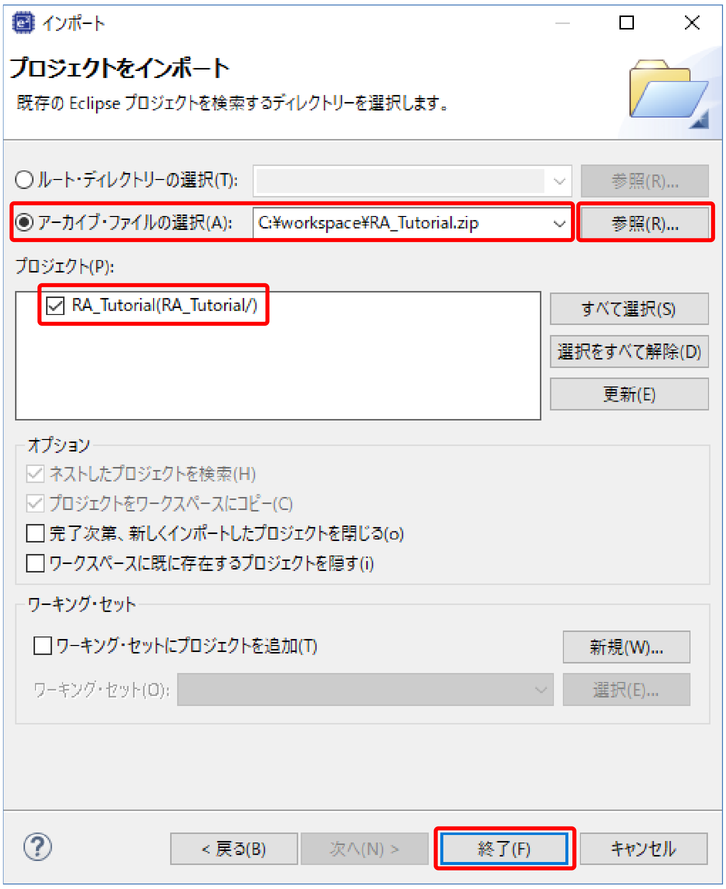


図 41　圧縮ファイルに保存されているプロジェクトの選択

(5) 選択したプロジェクトが e^2 studio にインポートされます。

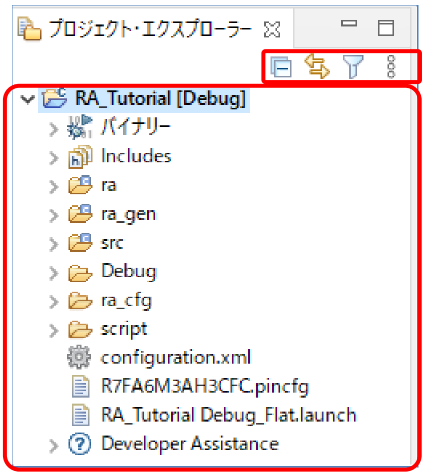


図 42　インポートされたプロジェクト

3.4 RA スタティックライブラリの作成と使用

この章では RA スタティックライブラリのプロジェクトの作成と、作成したライブラリプロジェクトを参照する実行プロジェクトの作成について説明します。

3.4.1 スタティックライブラリプロジェクトの作成

RA スタティックライブラリのプロジェクトを作成する手順例を以下に示します。

(1) [ファイル] → [新規] → [Renesas C/C++ Project] → [Renesas RA] の順に選択します。

(2) “Renesas RA C/C++ Project” テンプレートを選択してください。[次へ] で次に進みます。

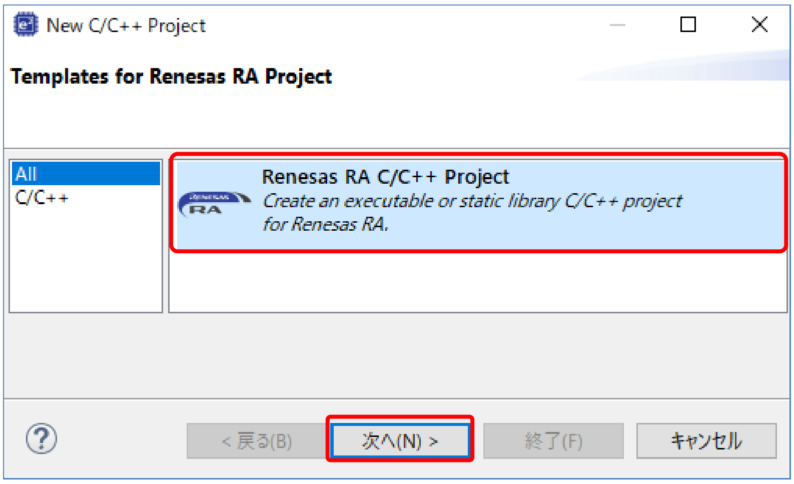


図 43 プロジェクトの作成 ― ライブラリプロジェクトテンプレートの選択

(3) プロジェクト情報の画面で、スタティックライブラリプロジェクトの名称を入力してください（例：RA_Lib）。[次へ] で次に進みます。

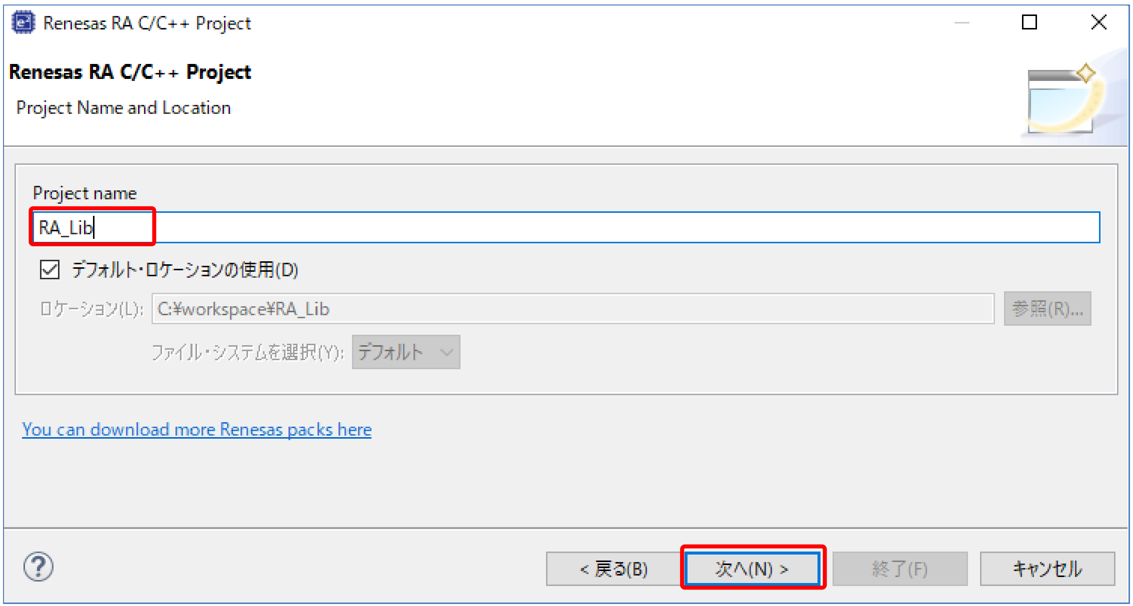


図 44 ライブラリプロジェクトの設定

(4) デバイスとツール選択の画面で、ボードを選択します（ここでは EK-RA6M3）。その他はすべてデフォルト設定のままとし、[次へ] で次に進みます。

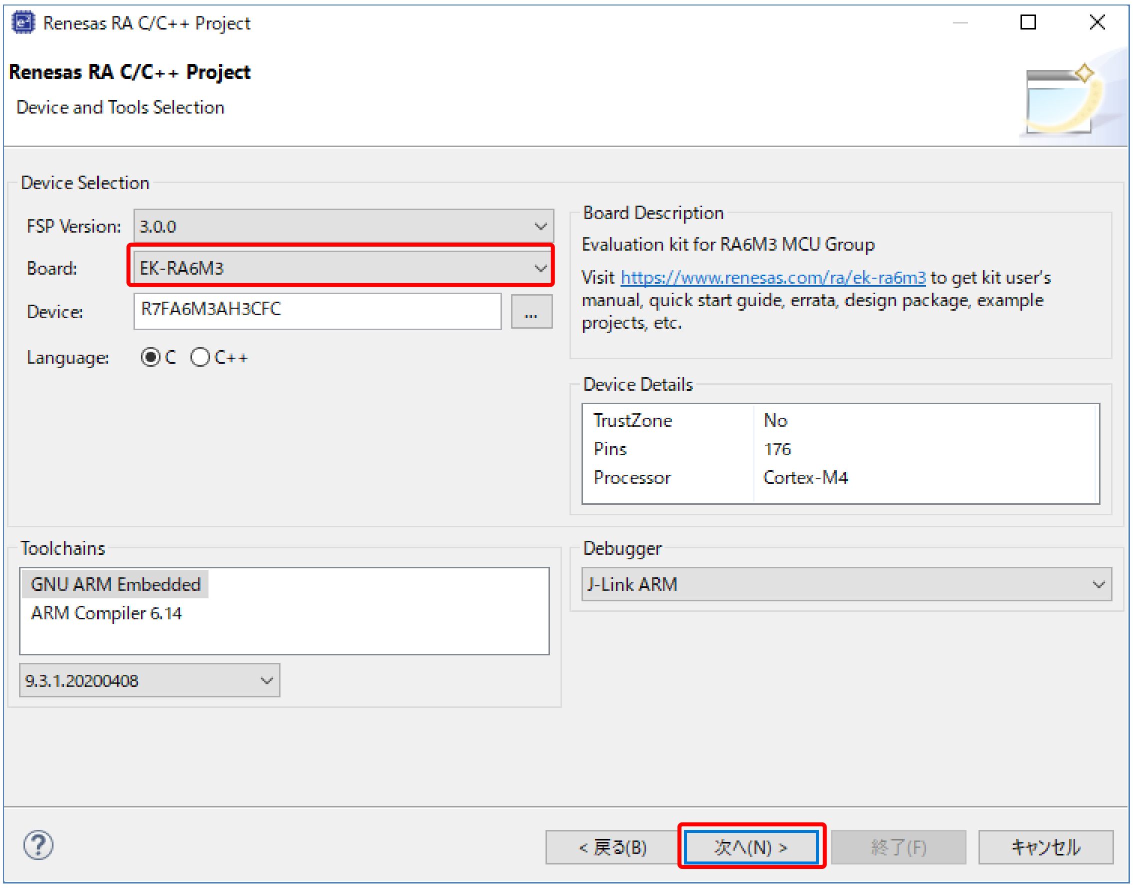


図 45　デバイスとツールの選択

(5) Build Artifact Selection：Static Library を選択します。

RTOS Selection：No RTOS を選択します。

[次へ] で次に進みます。

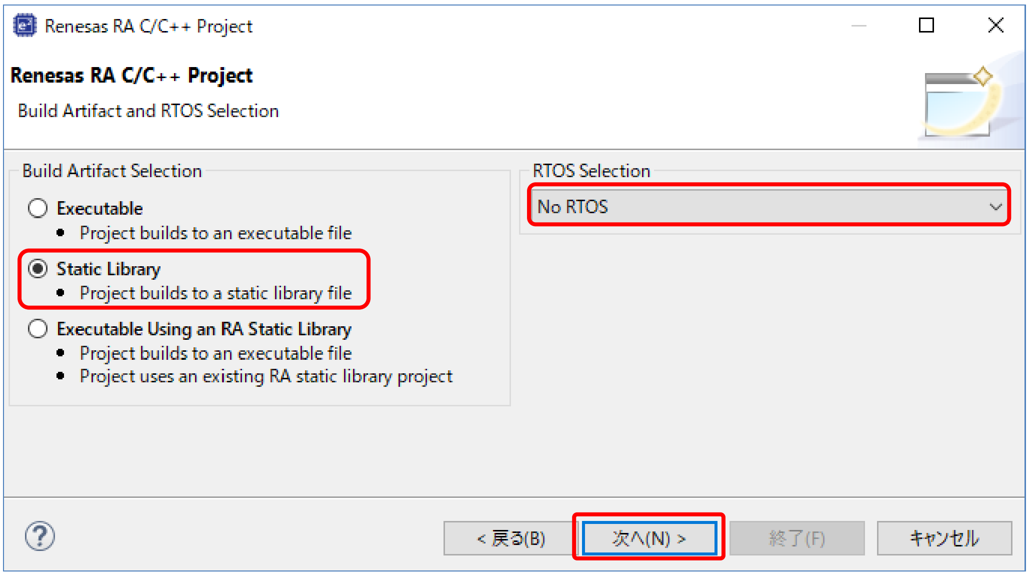


図 46　Artifact と RTOS の選択

(6) プロジェクトテンプレート選択の画面で、“Bare Metal – Blinky”テンプレートを選択してください。
[終了] ボタンを押すとプロジェクトが作成されます。

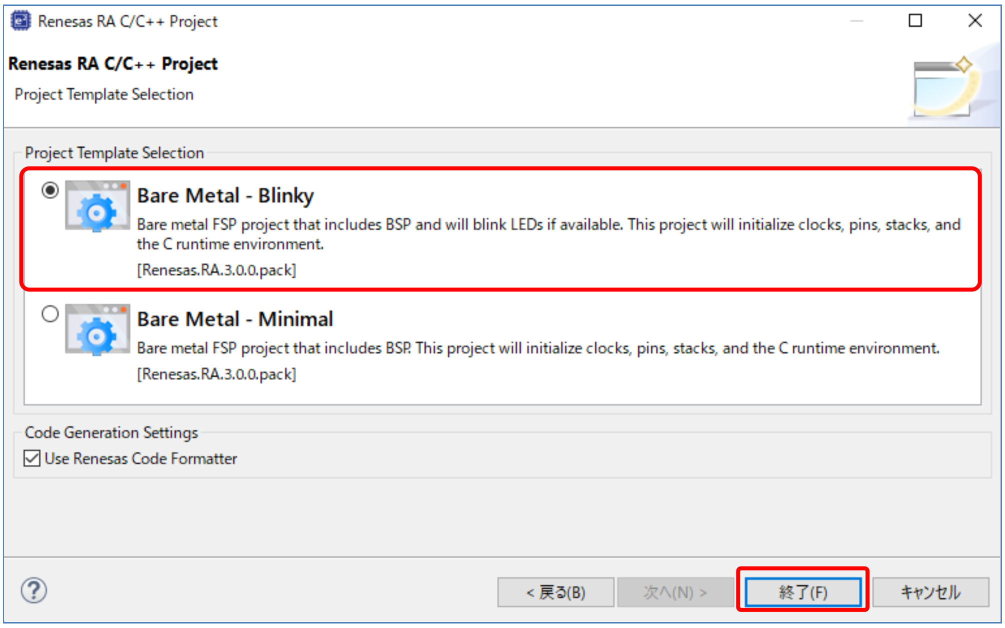


図 47　ライブラリ用プロジェクトテンプレートの選択

(7) [FSP Configuration] パースペクティブを開くよう e² studio のメッセージが表示される場合があります。
[パースペクティブを開く] を選択して開いてください。
(8) [Generate Project Content] をクリックしてください。

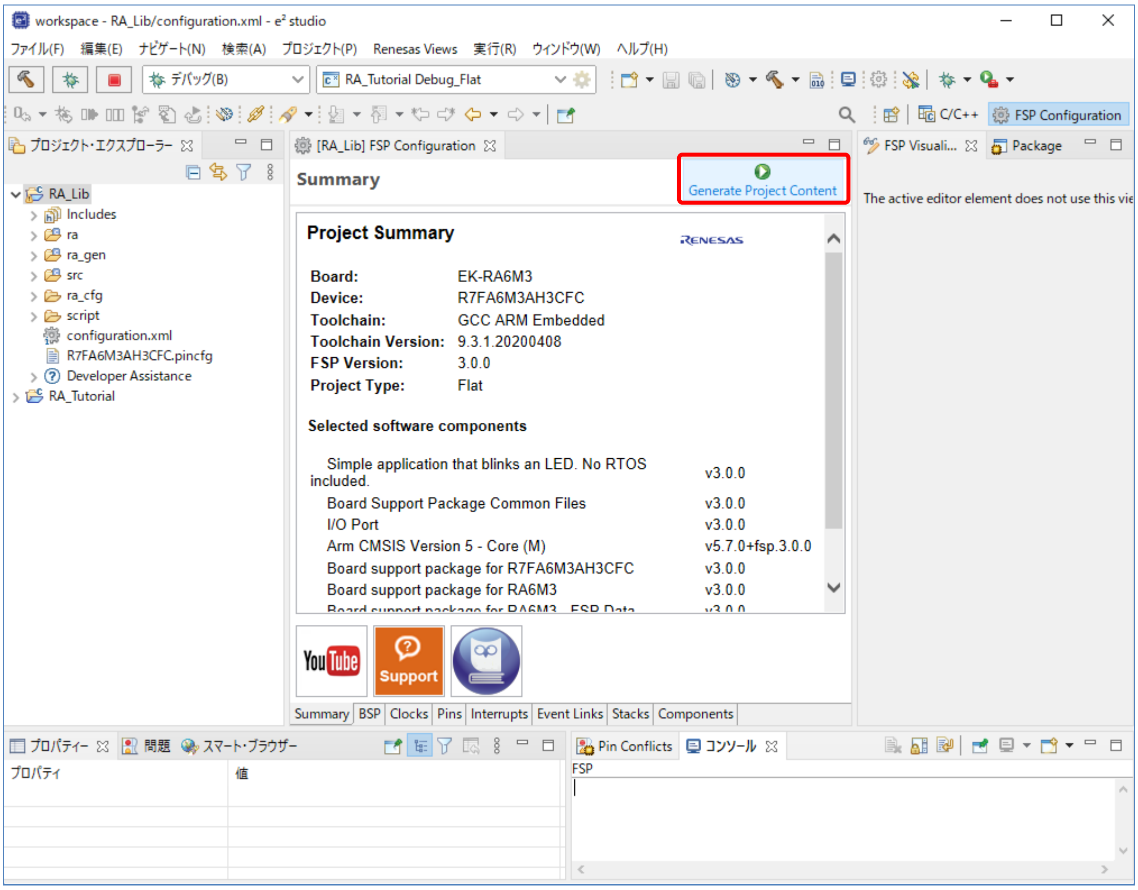


図 48　ライブラリプロジェクトの生成

(9) [プロジェクトエクスプローラー] ビューで、RA_Lib\src\ 下の “hal_entry.c” を開いてください。

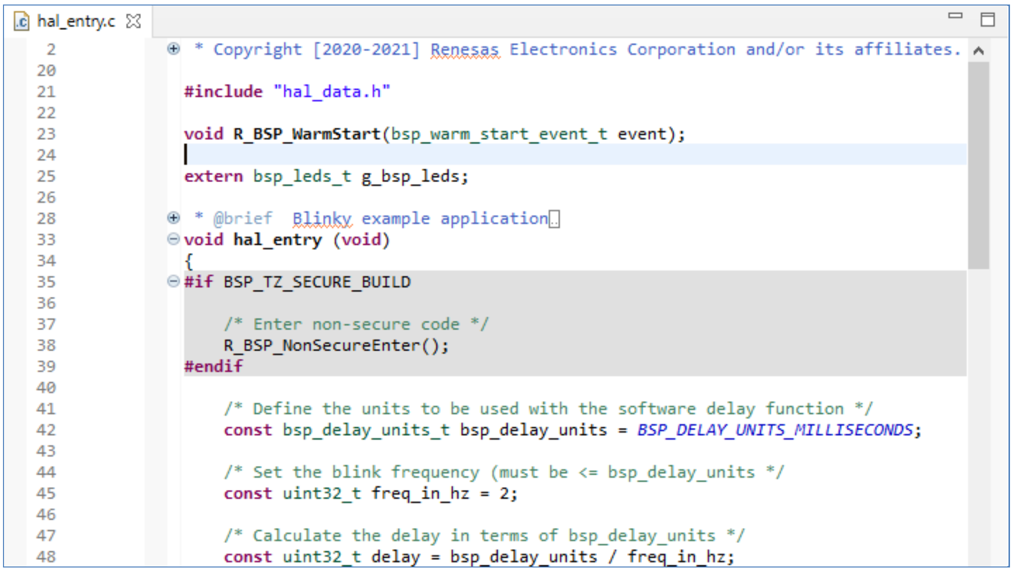


図 49　既存の “hal_entry.c”

hal_entry()関数の名称を hal_entry_lib()に変更し、hal_entry_lib()の宣言を追加します。

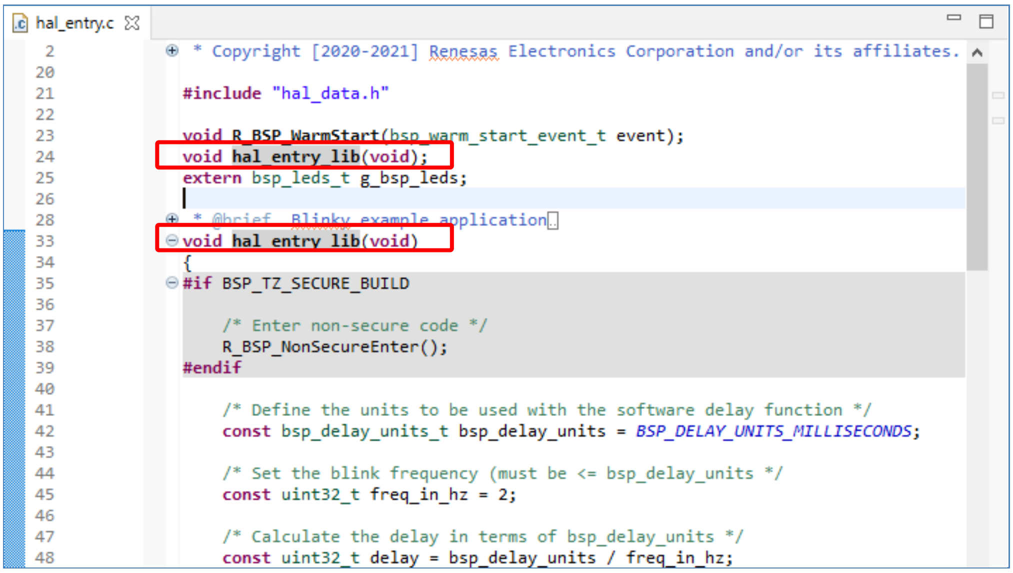


図 50　新しい “hal_entry.c”

(10)作成したライブラリプロジェクトをビルドすると、スタティックライブラリファイル “libRA_Lib.a” が出力されます。

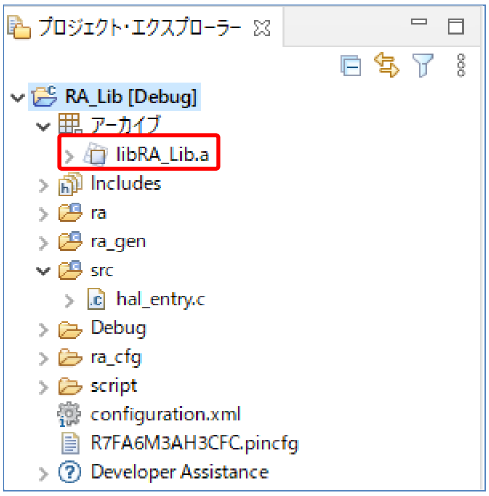


図 51　ビルドされたスタティックライブラリ

3.4.2　スタティックライブラリを実行プロジェクトで使用する

前章（3.4.1　スタティックライブラリプロジェクトの作成）で作成したスタティックライブラリを RA 実行プロジェクトで使用する方法を、以下の手順で説明します。

- RA C 言語実行プロジェクトを作成する
- スタティックライブラリプロジェクトで宣言した hal_entry_lib()関数を呼び出すようにソースコードを変更する
- RA C 言語実行プロジェクトをビルドして実行する

以下の手順を実行します。

(1) [ファイル] → [新規] → [Renesas C/C++ Project] → [Renesas RA] の順に選択します。
(2) “Renesas RA C/C++ Project” テンプレートを選択してください。[次へ] で次に進みます。
(3) プロジェクト名 “RA_App” を入力してください。[次へ] で次に進みます。
(4) デバイスとツール選択の画面で、ボードを選択します（ここでは EK-RA6M3）。その他はすべてデフォルト設定のままとし、[次へ] で次に進みます。
(5) [Build Artifact and RTOS Selection] 画面で、“Executable” を選択します。“RTOS Selection” は、“No RTOS” を選択してください。[次へ] ボタンを押します。

注：　[Build Artifact and RTOS Selection] 画面では “Executable Using an RA Static Library” は選択できません。

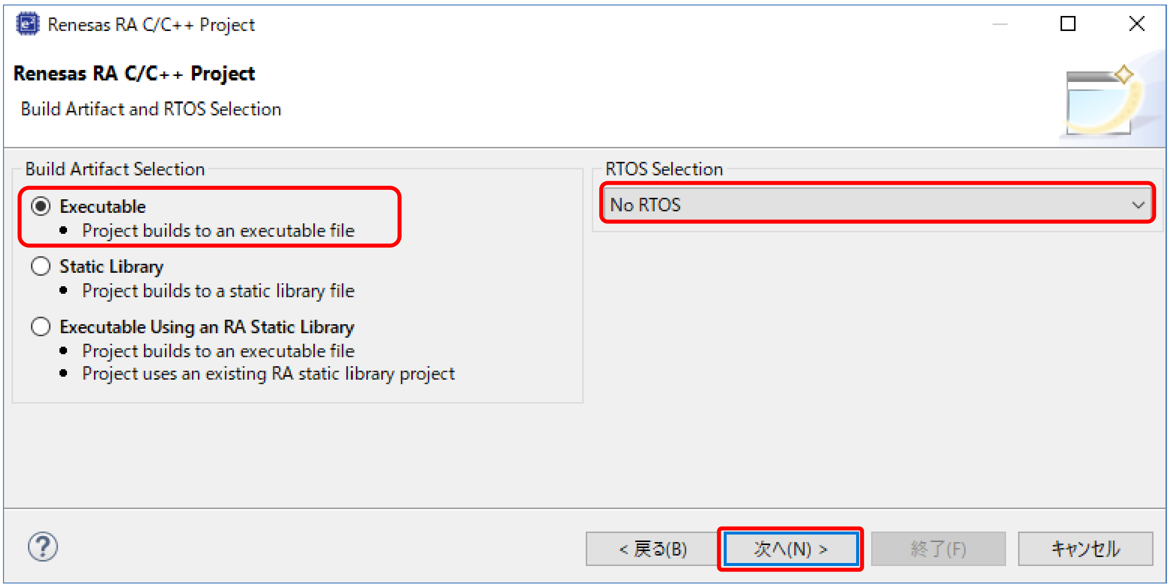


図 52　Artifact と RTOS の選択

(6) プロジェクトテンプレート選択の画面で、“Bare Metal — Minimal” テンプレートを選択してください。
[終了] ボタンを押すとプロジェクトが作成されます。

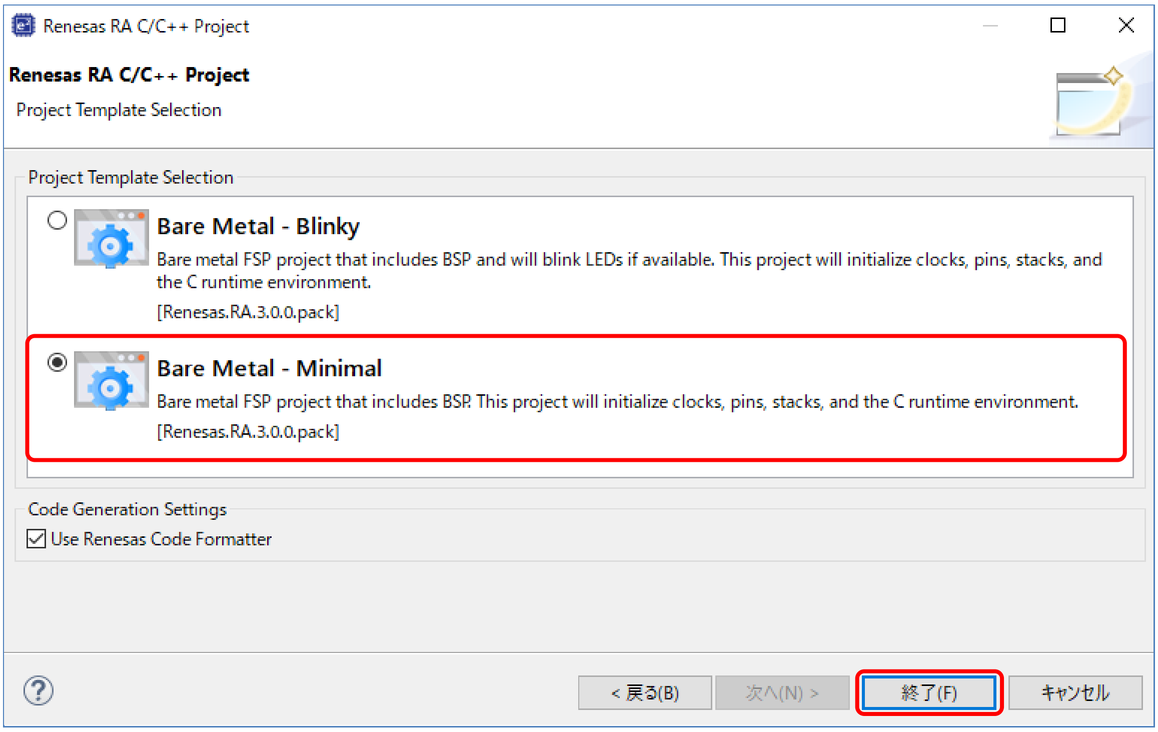


図 53　プロジェクトの作成 — プロジェクトテンプレート

(7) 作成済みの RA ライブラリを追加してください。

プロジェクトプロパティの設定：

— [設定] → [Libraries] → [Libraries (-l)] → "RA_Lib" の順に選択して追加します。
— [設定] → [Libraries] → [Libraries search path (-L)] → "${workspace_loc:/RA_Lib/Debug}"の順に選択して追加します。
— [適用して閉じる] ボタンを押します。

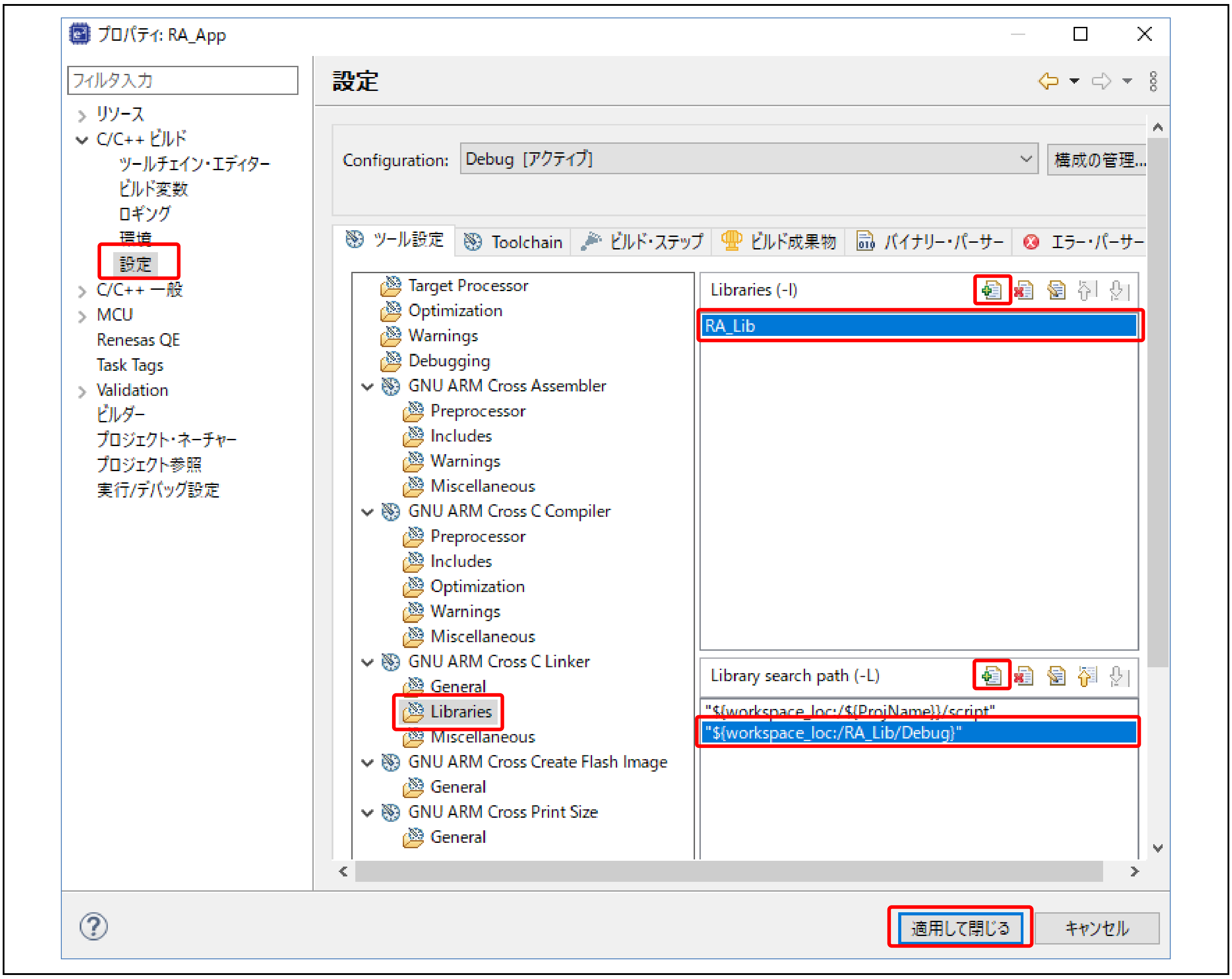


図 54　プロジェクトプロパティの設定

(8) [プロジェクトエクスプローラー] ビューで、RA_App\src\. 下の “hal_entry.c” を開いてください。

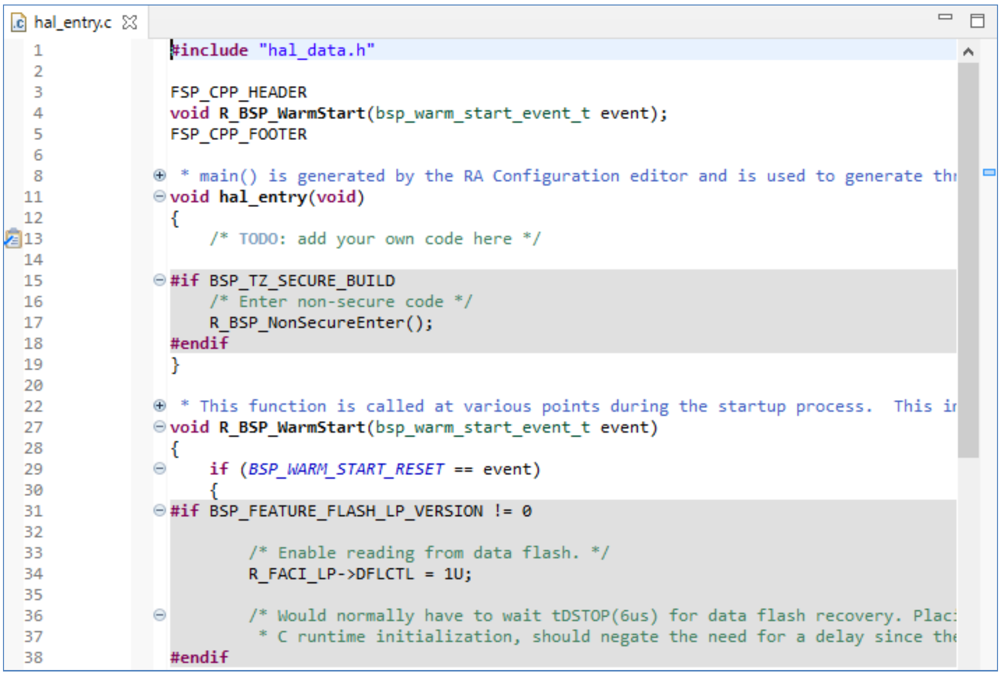


図 55 既存の “hal_entry.c”

R_BSP_WarmStart()関数とその宣言を削除します。

hal_entry()関数内に LED 点滅用ライブラリ関数 “hal_entry_lib()” を呼び出すコードを追加し、ライブラリ関数の宣言を追加します。

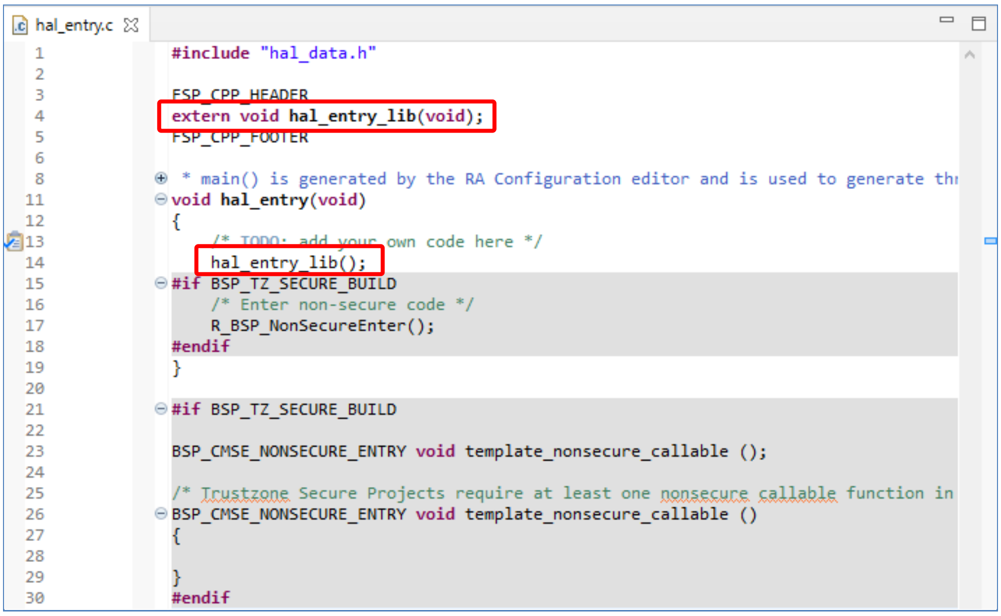


図 56 新しい “hal_entry.c”

(9) アプリケーションプロジェクトをビルドします。

(10)ライブラリ関数 “hal_entry_lib()” の呼び出し位置にブレークポイントを設定して RA_App プロジェクトを実行します。ブレークポイントの設定方法については、「5.3.2 ブレークポイントビュー」を参照してください。

(11)ブレークポイントでプログラムの停止後、実行を再開してください。LED を点滅させるライブラリ関数 “hal_entry_lib()” が実行されることを確認します。

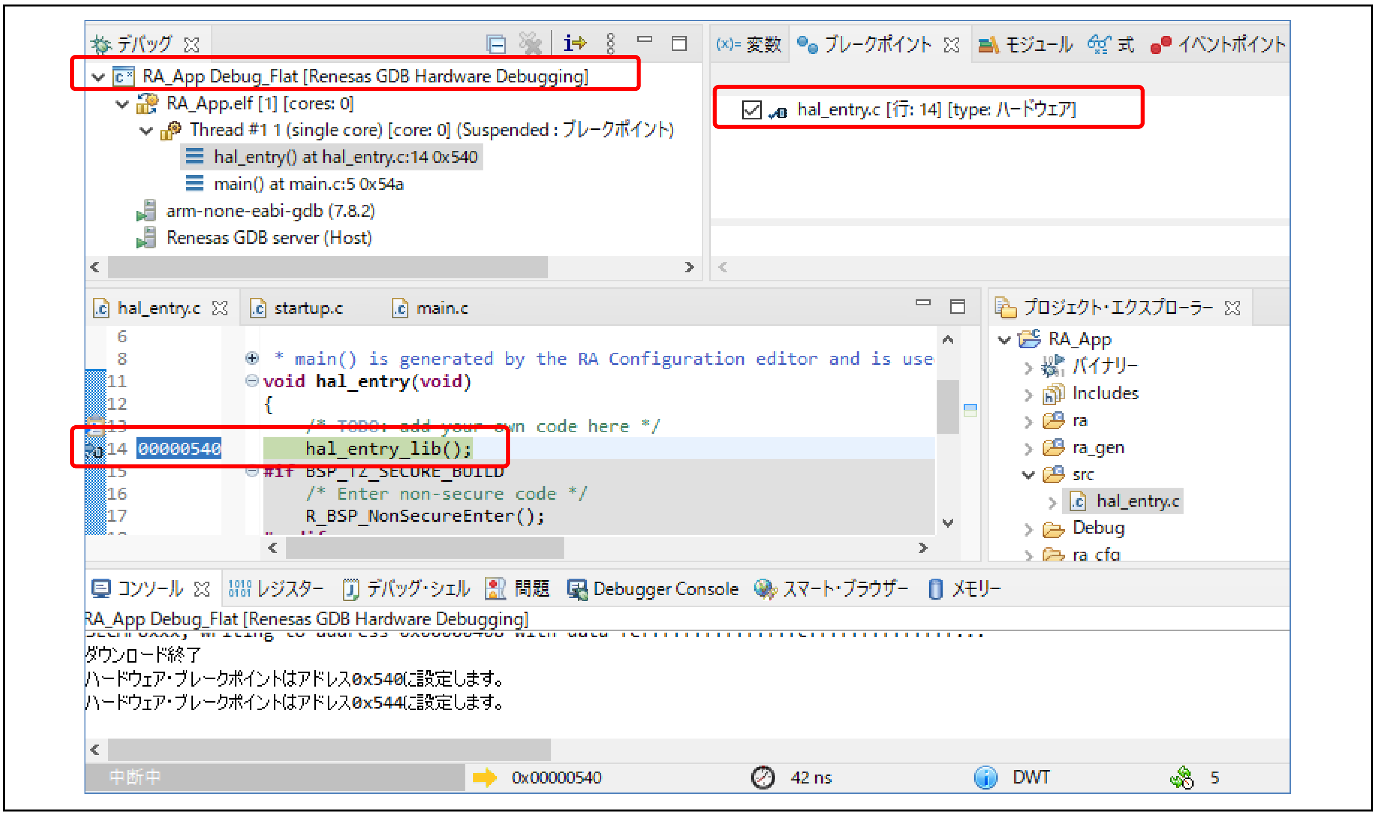


図 57　アプリケーションプロジェクトでのライブラリ関数の実行

3.5 RA プロジェクトコンフィグレーションエディタ

[RA プロジェクトコンフィグレーションエディタ] ビューは、現在のプロジェクト構成（configuration）を表示します。構成は “configuration.xml” ファイルに保存されています。プロジェクト構成の設定はいくつかのページに分類してあり、項目別にプロジェクトの設定を行なうことができます（端子やクロックの設定、使用するドライバなど）。ドライバには、ハードウェアレベルのシンプルなものから、RTOS 対応のアプリケーションまでを含みます。マルチスレッド専用のコンポーネント（ミューテックス、セマフォ、イベントなど）も設定できます。

プロジェクト構成を編集するには、まず、以下を確認してください。

- e² studio ウィンドウの右上の表示で [FSP Configuration] パースペクティブが選択されていることを確認してください。選択されていない場合は、[ウインドウ] メニュー → [パースペクティブ] → [パースペクティブを開く] → [その他] → [FSP Configuration] の順に選択してください。
- “configuration.xml” ファイルが開いていることを確認してください。

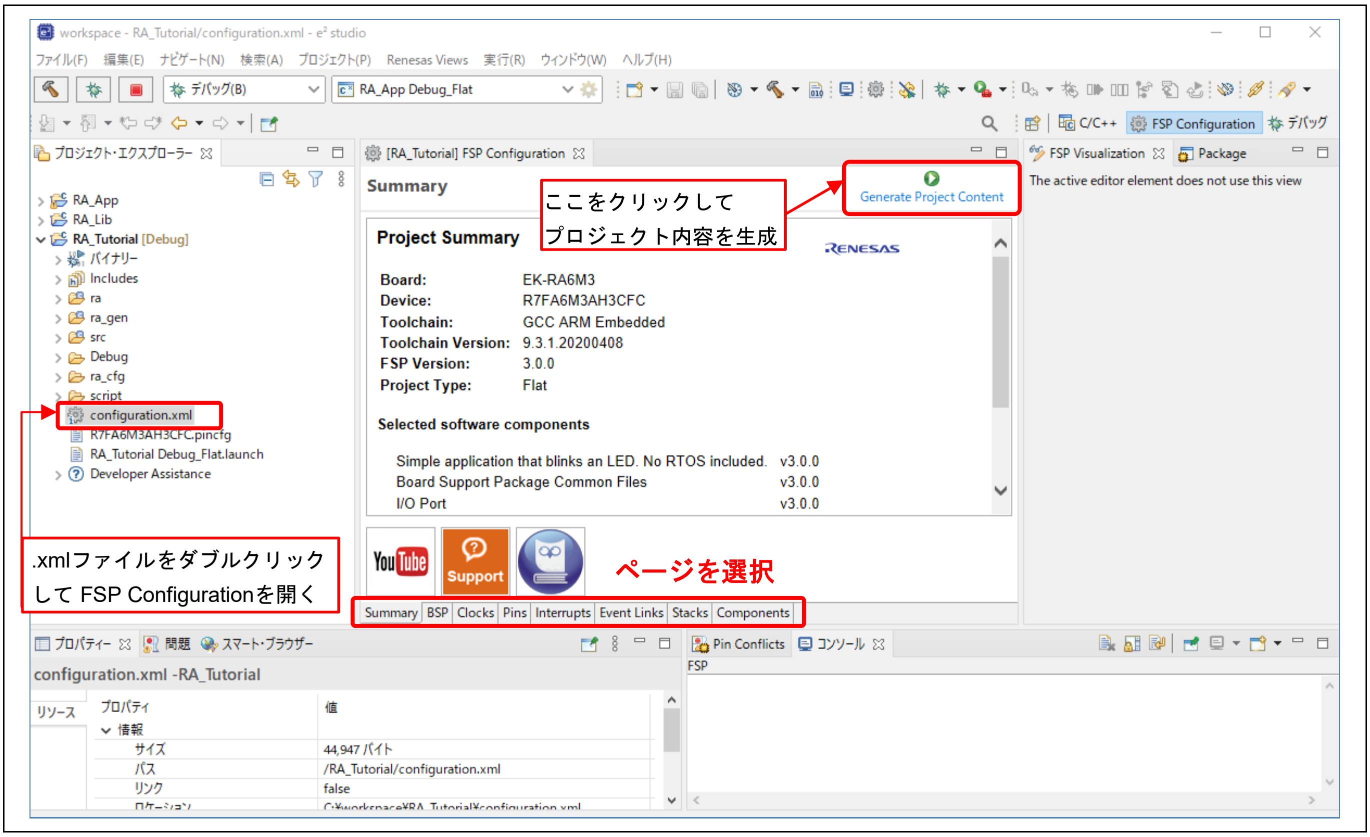


図 58　RA プロジェクト構成 ― [RA プロジェクトコンフィグレーションエディタ] ビュー

[RA プロジェクトコンフィグレーションエディタ] ビューには 8 つのページ（タブ）があります。

概要（Summary）ページは、プロジェクトの概要を表示します。

BSP ページでは、FSP のバージョン、RA 用ボードの種類、デバイスを選択できます。

クロック（Clocks）、端子（Pins）、割り込み（Interrupts）、イベントリンク（Event Links）、スタック（Stacks）、コンポーネント（Components）ページの設定手順やオプションについても後述します。

3.5.1 概要（Summary）ページ

[Summary] ページは、現在選択しているデバイス、ボード、RA 用ソフトウェアコンポーネントなど、プロジェクトの概要を表示します。また、YouTube の 'Renesas Presents' チャンネルや FSP ユーザーズマニュアルなど便利なサイトへのリンクもあります。

スレッドにモジュール／オブジェクトを追加すると、その情報は [Summary] ページにも追加されます。

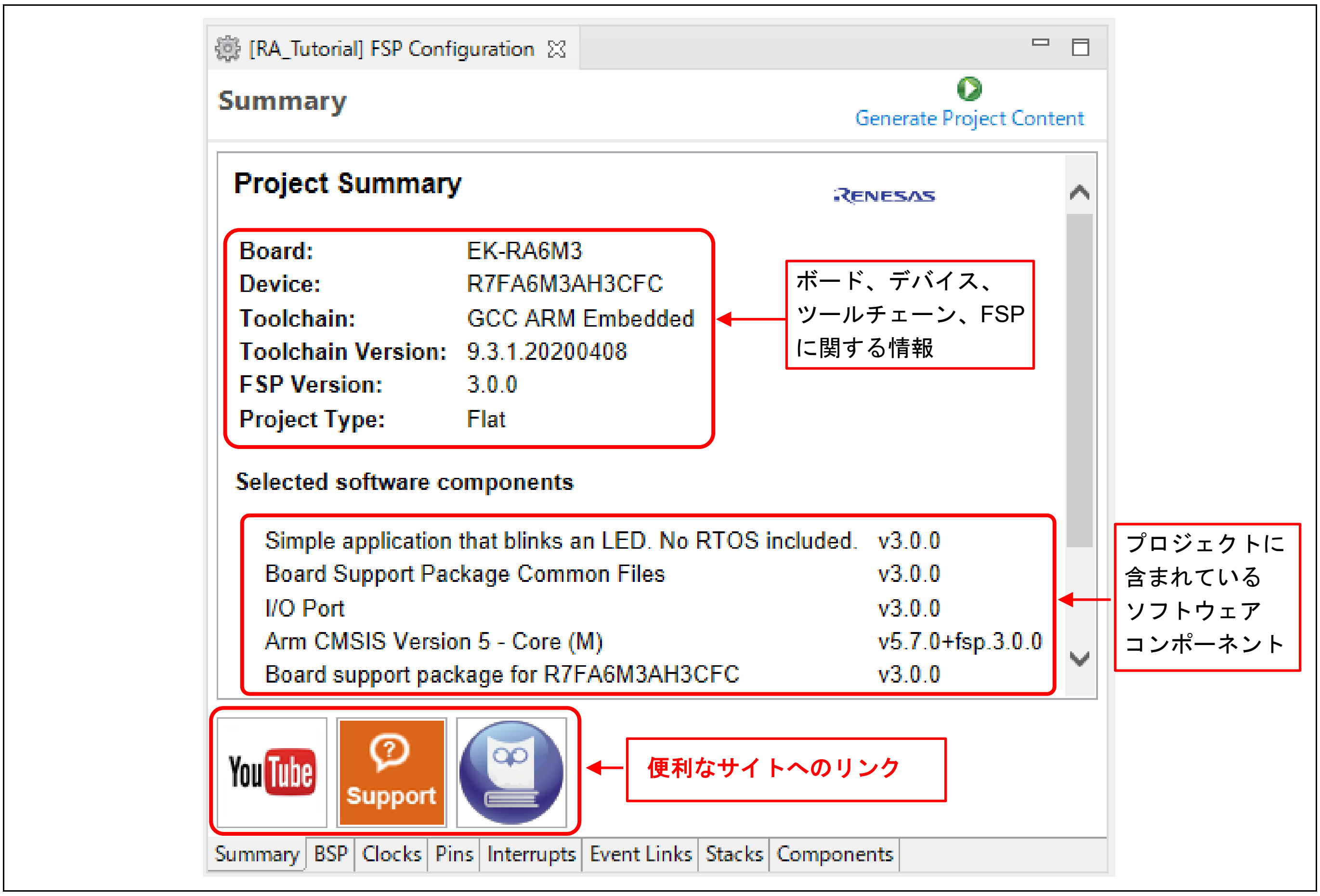


図 59 [Summary] ページ

3.5.2 BSP ページ

[BSP] ページでは、FSP のバージョン、ボード、デバイスを選択できます。CMSIS ボードの情報もこのページからインポートできます。

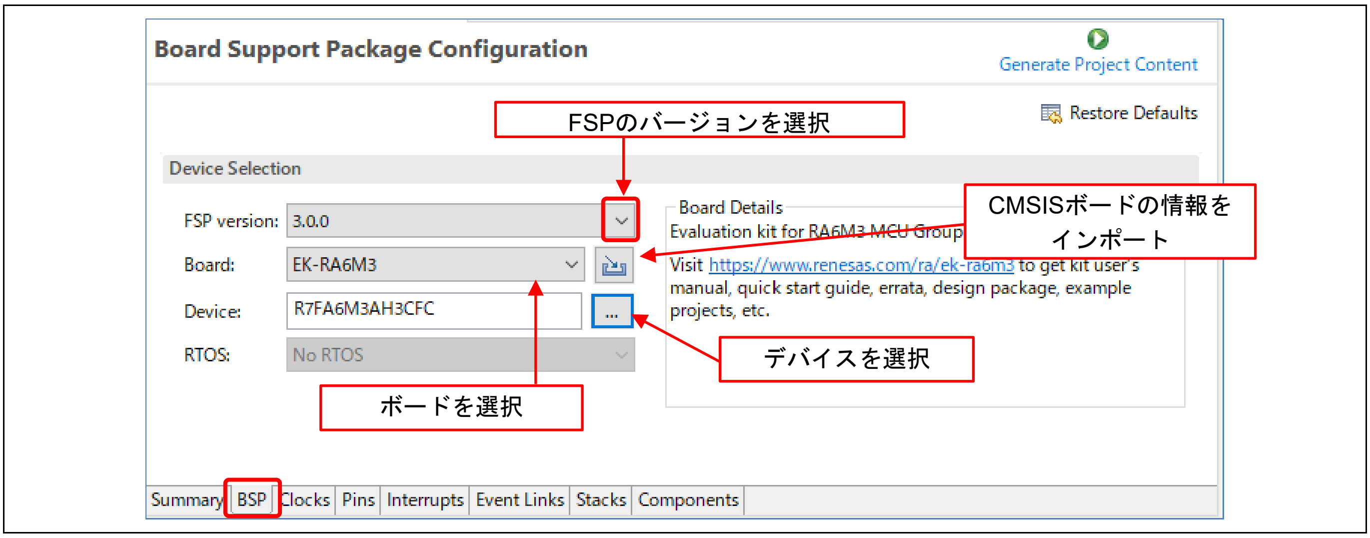


図 60 RA プロジェクト構成 ― [BSP] ページ

[BSP] ページを開くと、選択したボードで設定できるプロパティが e² studio の [プロパティ] ビューに表示されます。プロジェクトの必要に応じて、[プロパティ] ビューでプロパティを変更できます。

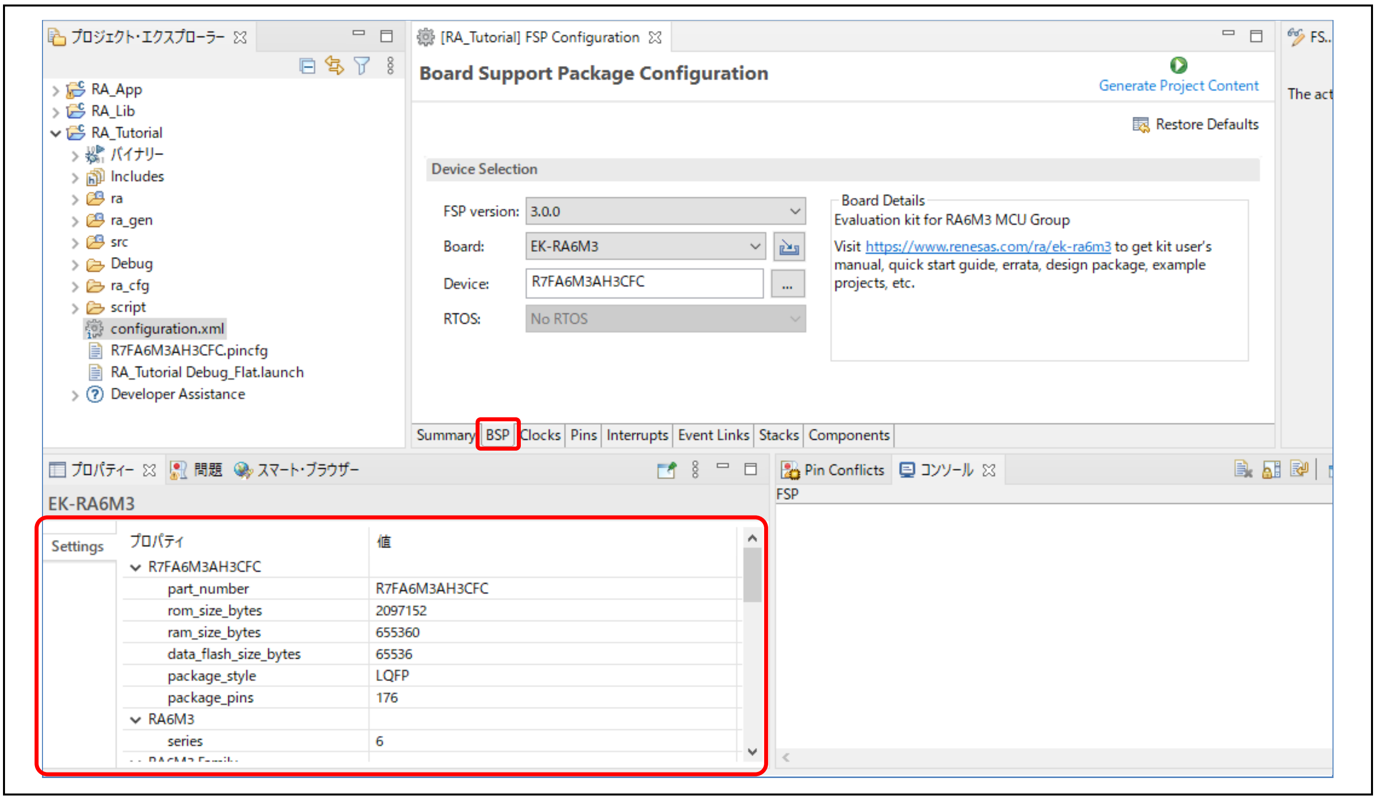


図 61 ボードのプロパティ

3.5.3　クロック構成（Clocks Configuration）ページ

[Clocks Configuration] ページでは、アプリケーションの初期クロックを設定します。出力クロックごとに、クロックソース、PLL 設定、クロック分周設定を選択できます。
クロック生成回路（CGC）の詳細は、RA ファミリの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。
以下の手順でプロジェクトを更新します。

(1) 画面上のクロック設定のドロップダウンリストから、設定したい値を選択します。
注：クロックの値が定められた範囲を越えると、値が赤字になり、“！” マークが表示されます。

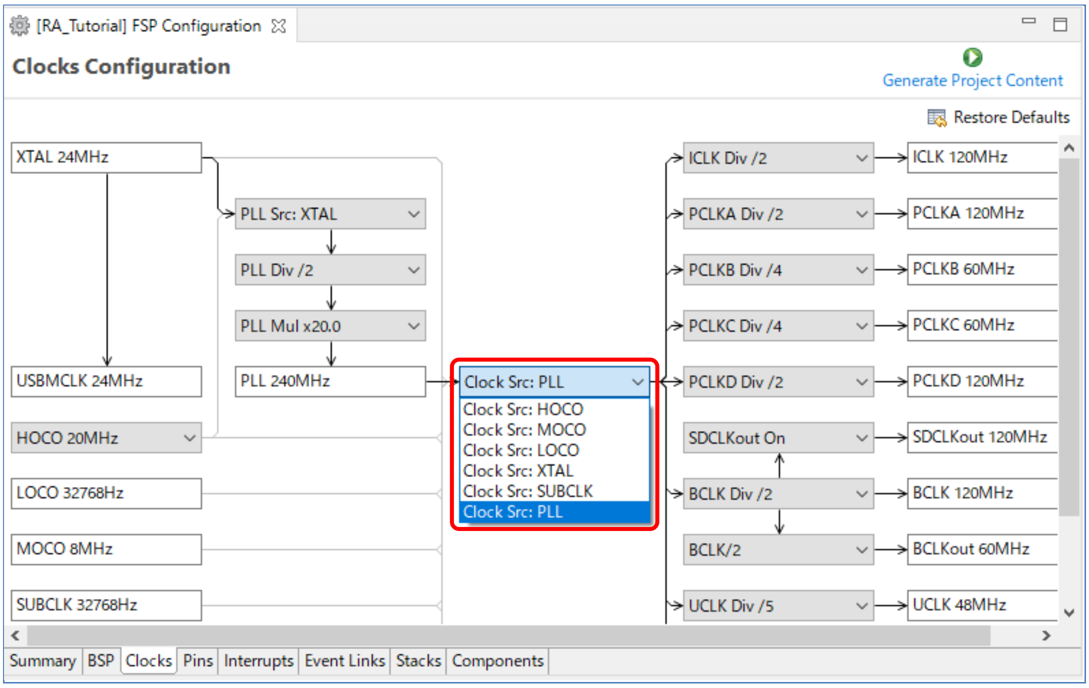


図 62　RA プロジェクト構成 ― [Clocks Configuration] ページ

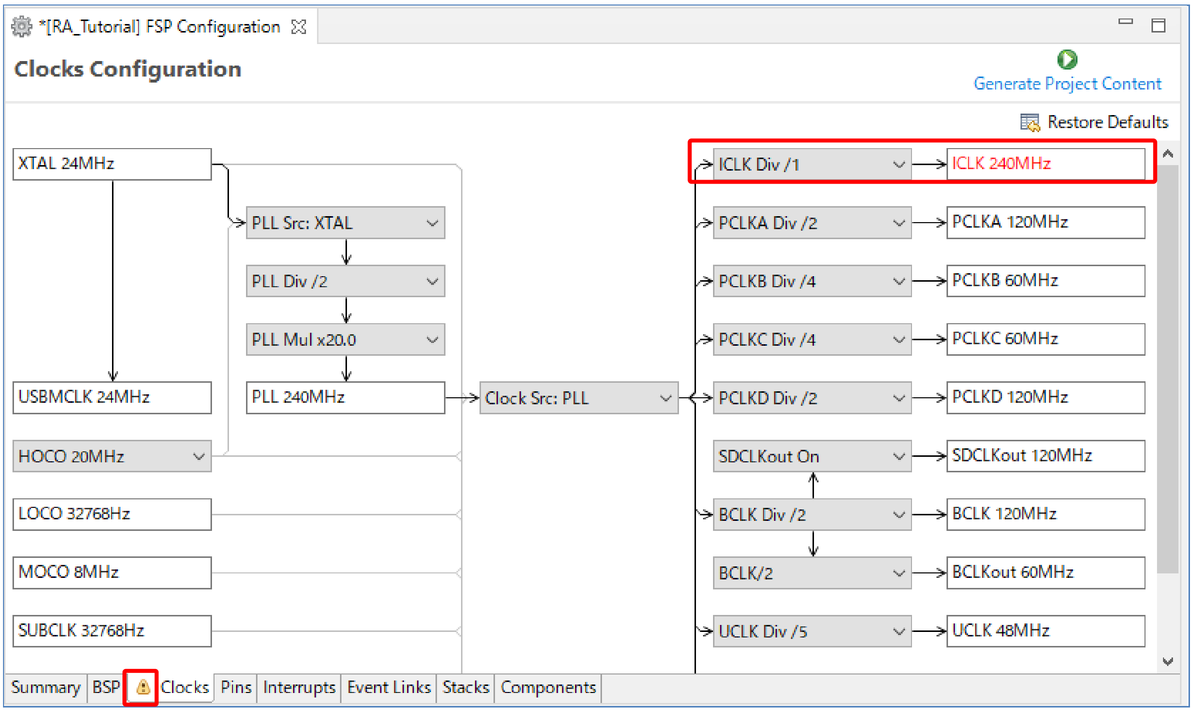


図 63　範囲を超過した値の表示例

(2) [Ctrl+S] キーなどで、プロジェクト構成の設定を保存します。
(3) [Generate Project Content] ボタンを押します。

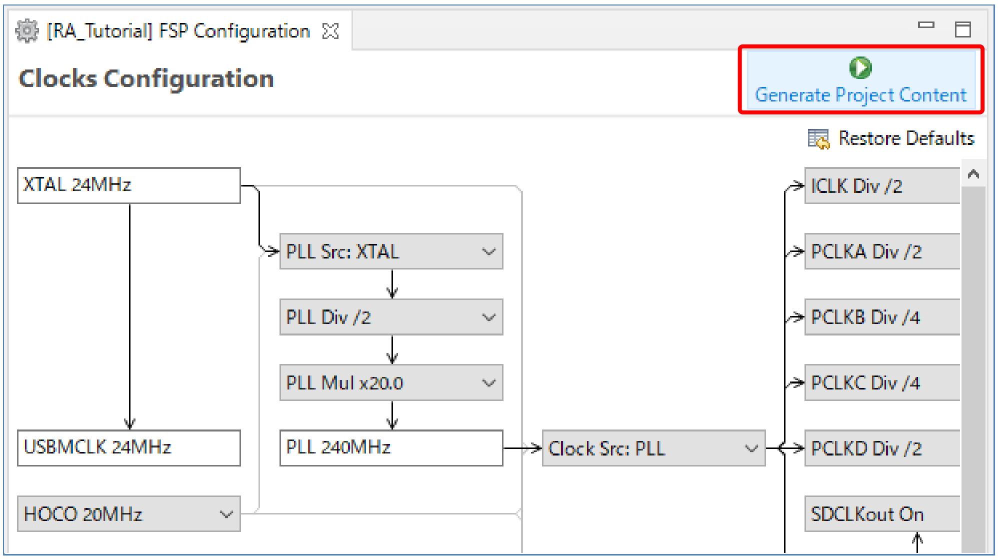


図 64　プロジェクト内容の生成

(4) 選択したクロック設定で、"bsp_clock_cfg.h" ファイルが更新されます。

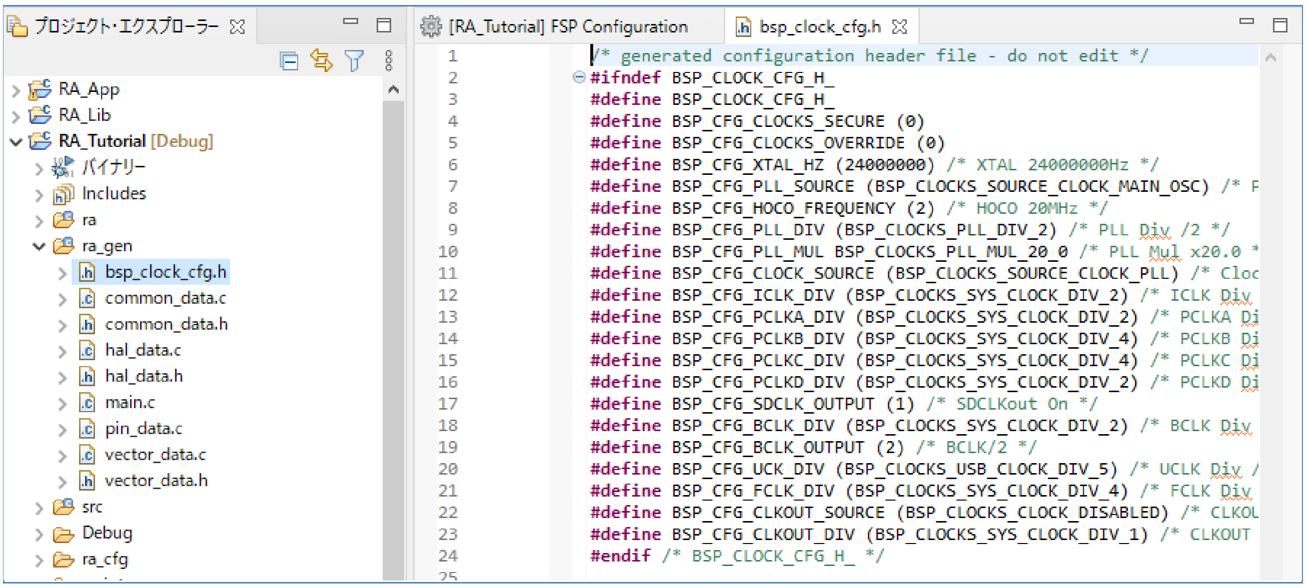


図 65　"bsp_clock_cfg.h" の更新

3.5.4　端子構成（Pin Configuration）ページ

[Pin Configuration] ページでは、プロジェクトの端子構成を生成します。

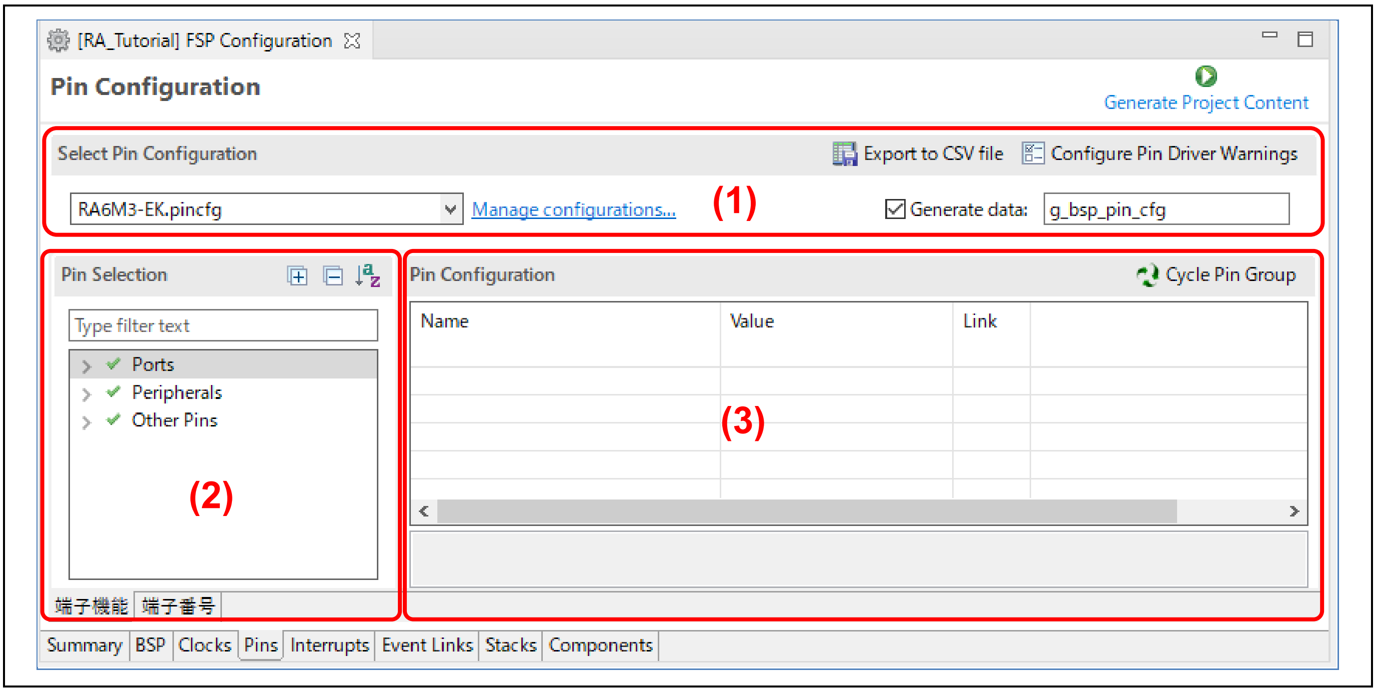


図 66　RA プロジェクト構成 ― [Pin Configuration] ページ

[Pin Configuration] ページは 3 つのペインで構成されています。

(1) Select Pin Configuration：端子構成ファイルを選択し、関連するデータ構造に名前を付けます。以下の手順で、複数の端子構成を設定できます。
   a. 既存の .pincfg ファイルをコピーして、新規の .pincfg ファイル（例：NewName.pincfg）をプロジェクトエクスプローラーに作成します。
   b. 作成した新規 .pincfg ファイル（例：NewName.pincfg）を、“Select Pin Configuration” のドロップダウンリストで選択します。
   c. [Generate data] チェックボックスを選択し、横にあるテキストボックスで、新規端子構成のデータ構造に固有の名称を付けます。
   d. 複数の異なるデータ構造を用いて、複数の端子構成を作成できます。

(2) Pin Selection：設定する端子や周辺機能を選択します。

(3) Pin Configuration：選択した端子や周辺機能のプロパティや機能を設定します。

端子構成を設定するには、プロジェクトで使用する周辺機能ごとに設定するのが最もわかりやすい方法です。以下の手順で行ないます。

(1) [Pin Selection] ペインで周辺機能を選択します（例：[Connectivity:SCI] → [SCI4]）。選択した周辺機能の構成が [Pin Configuration] ペインに表示されます。

(2) 選択した周辺機能の動作モードを “Operation Mode” で選択します（例：“Simple SPI”）。

(3) 選択したモードで周辺機能の入出力に使用したい端子を “Input/Output” で選択します。
注 1. "Lock" カラムで端子割り当ての変更を禁止できますが、ロックの有無は出力するコード内容には影響しません。
注 2. 端子設定の右側にある矢印をクリックすると、ポート設定に飛びます。

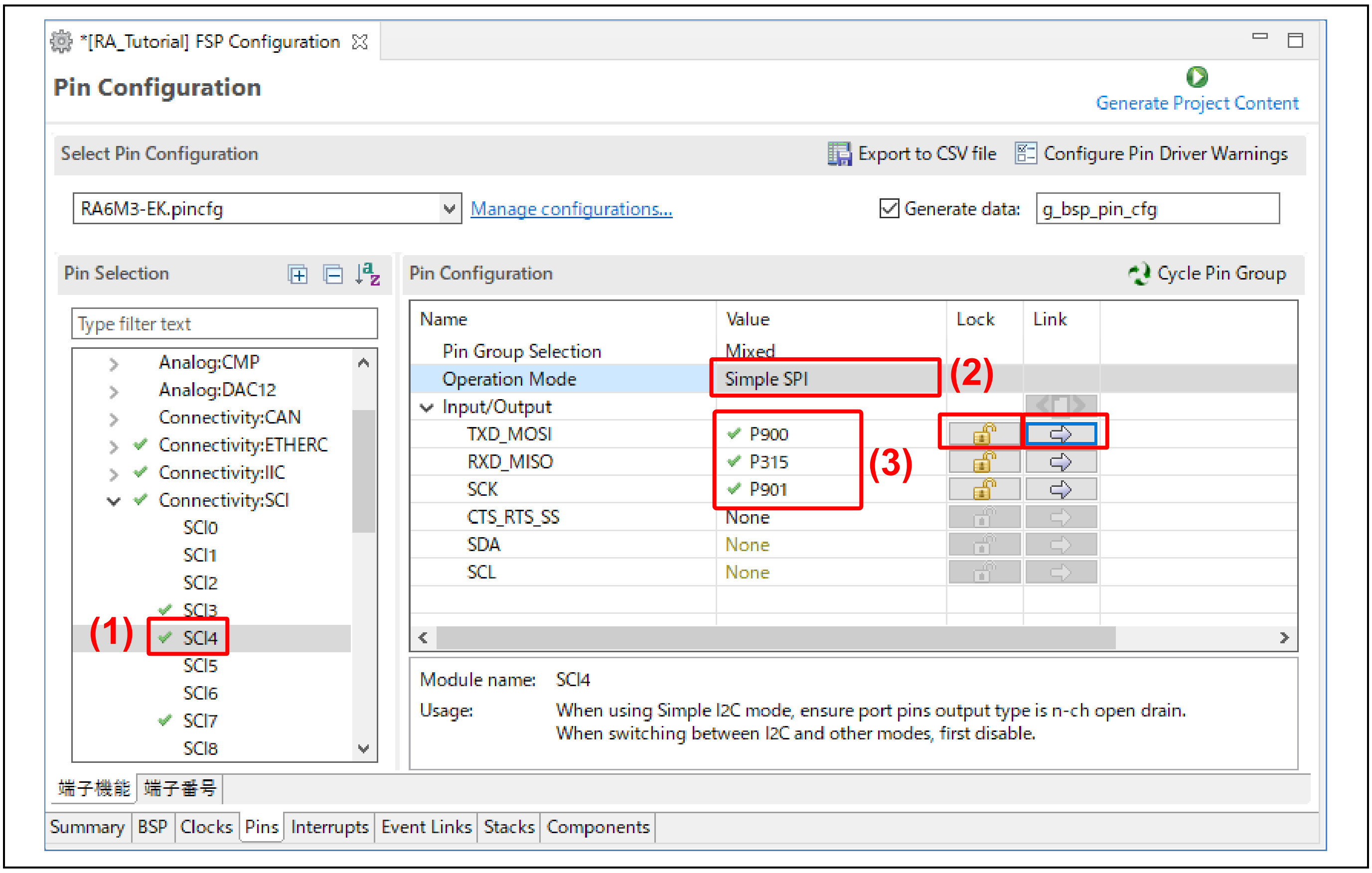


図 67　RA プロジェクト構成 ー 端子構成の設定（周辺機能ごとに設定）

端子ごとに設定することもできます。以下の手順で行ないます。

(1) [Pin Selection] ペインで 1 つの端子を選択します（例：[Ports] → [P0] → [P003]）。選択した端子の設定が [Pin Configuration] ペインに表示されます。

(2) 下図の例のように、端子のプロパティを入力します。

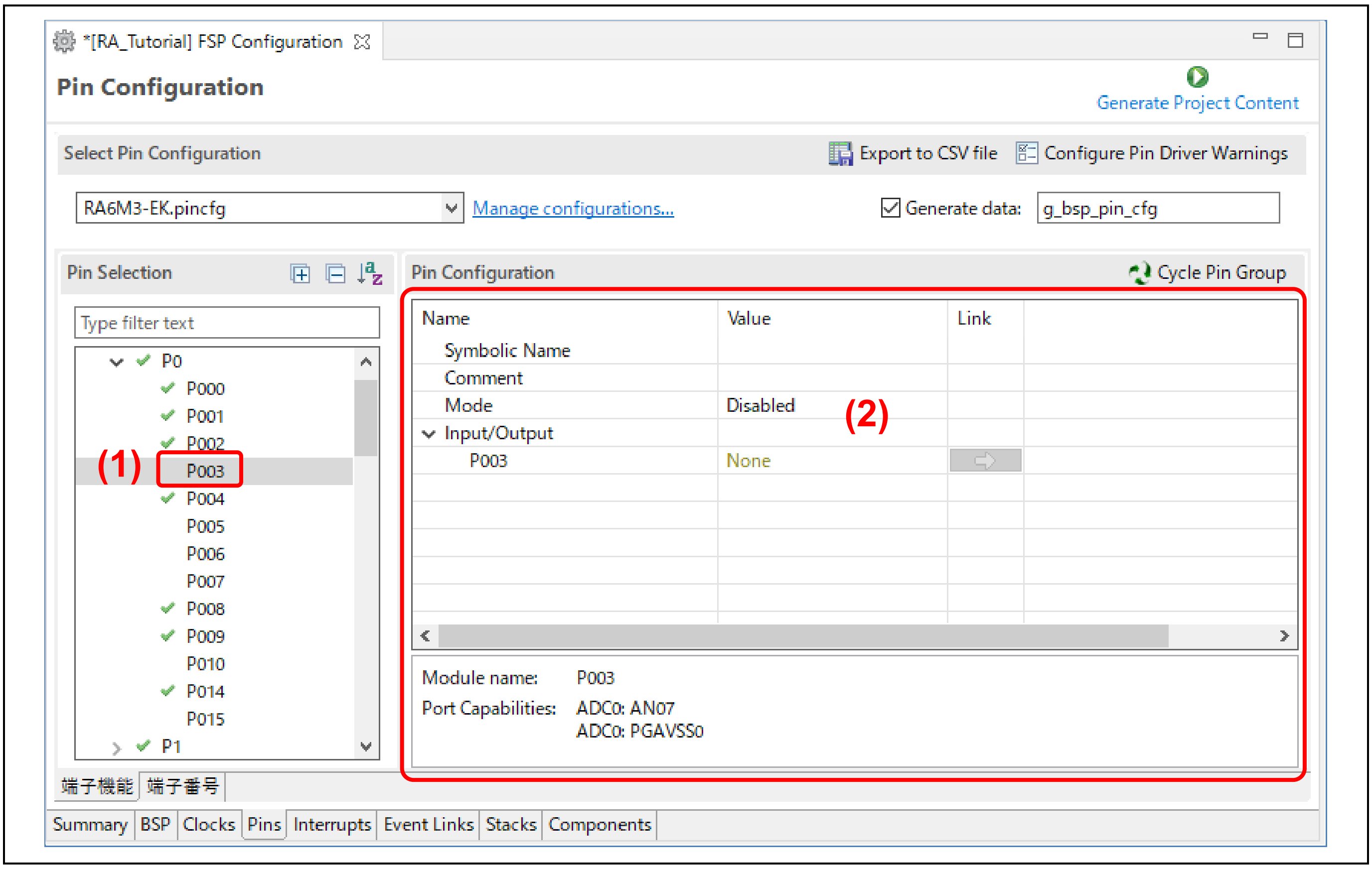


図 68　RA プロジェクト構成 ― 端子構成の設定（端子ごとに設定）

(3) 変更した端子設定は [Visualization] ビューに表示されます。

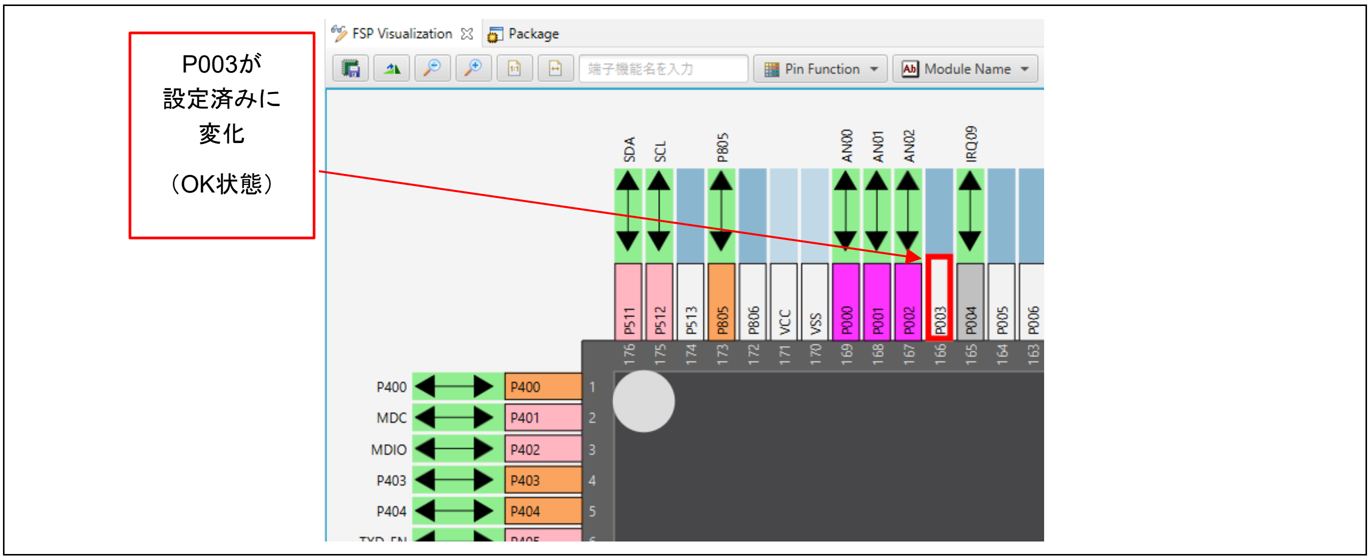


図 69　RA プロジェクト構成 ― [Visualization] ビュー（接続状態）

[Pin Configuration] ページでは、端子構成を別のデバイスから移行することもできます。移行にはツールバー上の [Import a pin configuration] ボタンを使います。この機能は、ユーザが設定した既存の端子構成を新たなデバイスに移行できます。

既存の端子構成を現在のプロジェクトにインポートするには、[Manage configurations…] → [Import] の順にボタンを押し、インポートする構成ファイルを選択します。

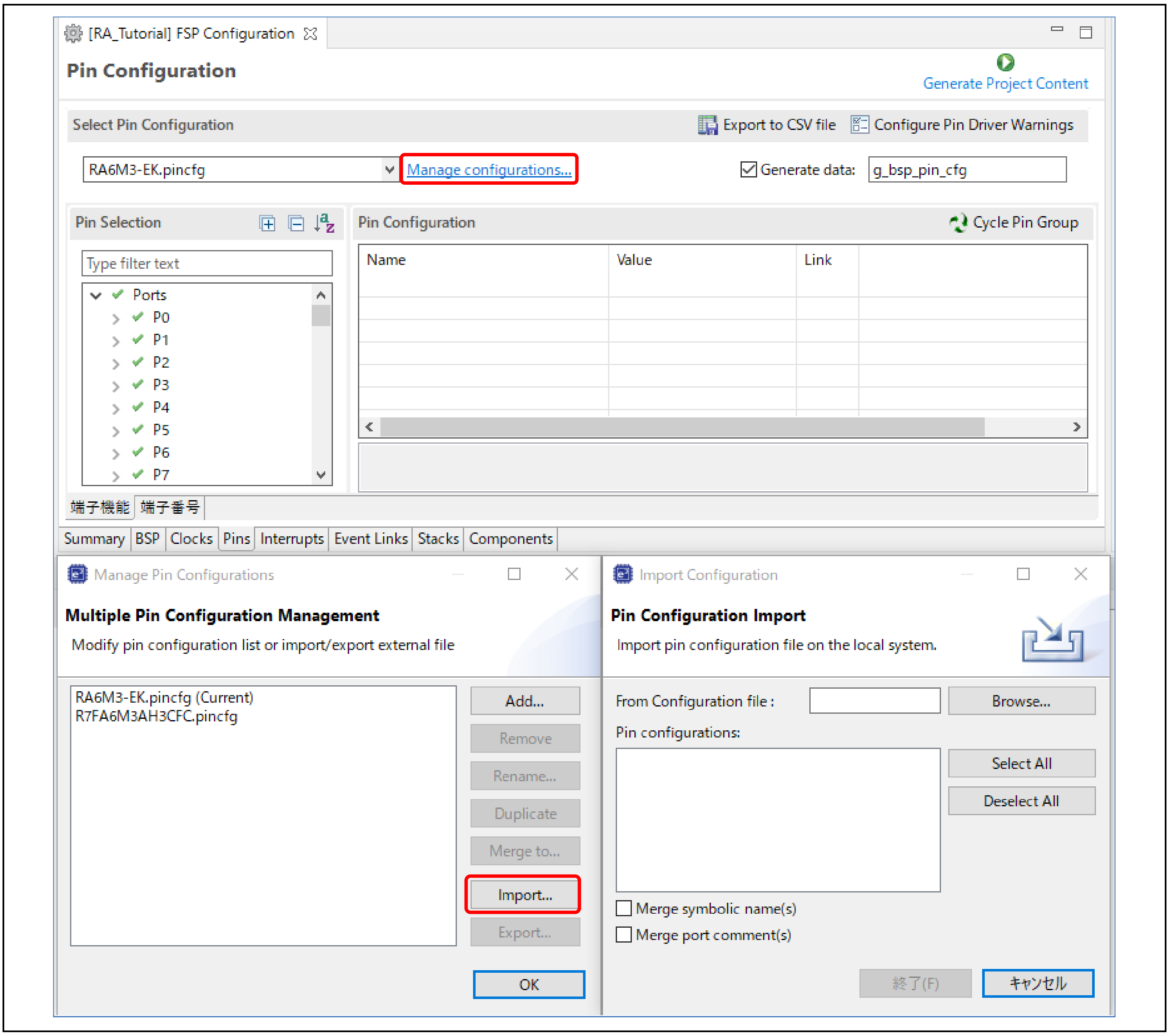


図 70　既存の端子構成を現在のプロジェクトにインポート

競合が発生する場合はメッセージが表示されますので、次のいずれかを選択します。

1. インポート動作をキャンセルする
2. 競合する設定をそのままインポートする
3. 競合する設定をインポートせずに、インポート操作を継続する

3.5.5　スタック構成（Stacks Configuration）ページ

[Stacks Configuration] ページでは次の操作が可能です。

- RA プロジェクト内でスレッドを設定する
- スレッドに RA のソフトウェアモジュールやオブジェクトを追加する
- [プロパティ] ビューでモジュールやオブジェクトのプロパティを変更する

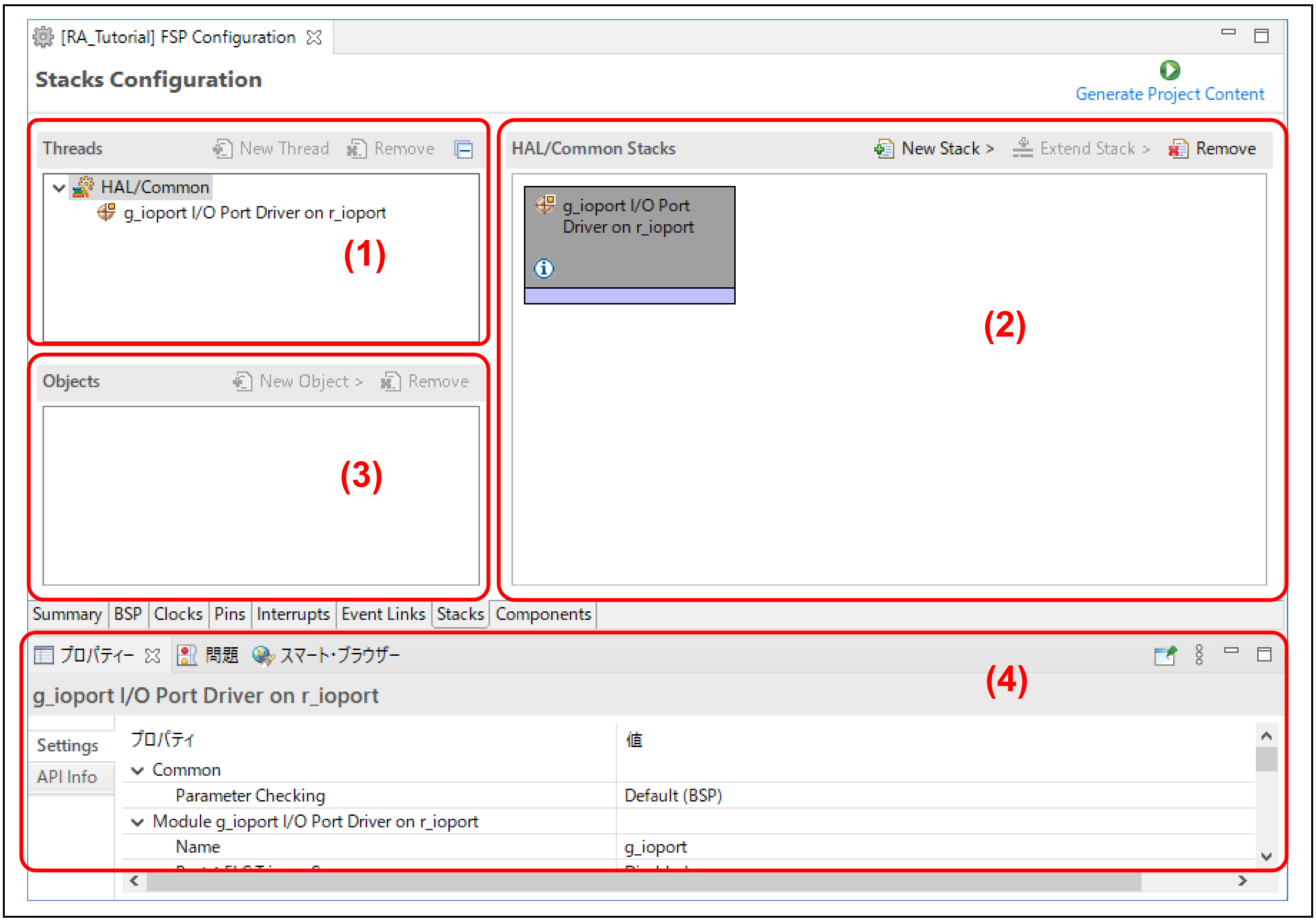


図 71　RA プロジェクト構成 ― [Stacks Configuration] ページ

[Stacks Configuration] ページは 3 つのペインで構成されています。

1. [Threads] ペイン：スレッドを追加または削除します。詳細は「6 章　FreeRTOS アプリケーションの設定」で説明します。
2. [Stacks] ペイン：FSP モジュールインスタンス（I/O ポート、SCI、UART など）を追加または削除します。
3. [Objects] ペイン：カーネルオブジェクトを追加または削除します。詳細は「6 章　FreeRTOS アプリケーションの設定」で説明します。

また、[プロパティ] ビューではスレッドの設定機能をサポートしており、モジュールやオブジェクトのプロパティを変更できます。

以下の手順でモジュールを既存のプロジェクトに追加できます。

(1) スレッド（HAL/Common）を選択します。選択したスレッド内のモジュールとオブジェクトが表示されます。

(2) [Stacks] ペインの [New Stack] ボタンを押してスレッドにモジュールを追加します（例：[New Stack] → [Driver] → [Monitoring] → [Clock Accuracy Circuit Driver on r_cac] の順に選択）。

(3) [Generate Project Content Generate Project Content] ボタンを押してソースコードを生成します。

(4) 選択したモジュールのプロパティが [プロパティ] ビューに表示されます。必要に応じてプロパティを変更できます。

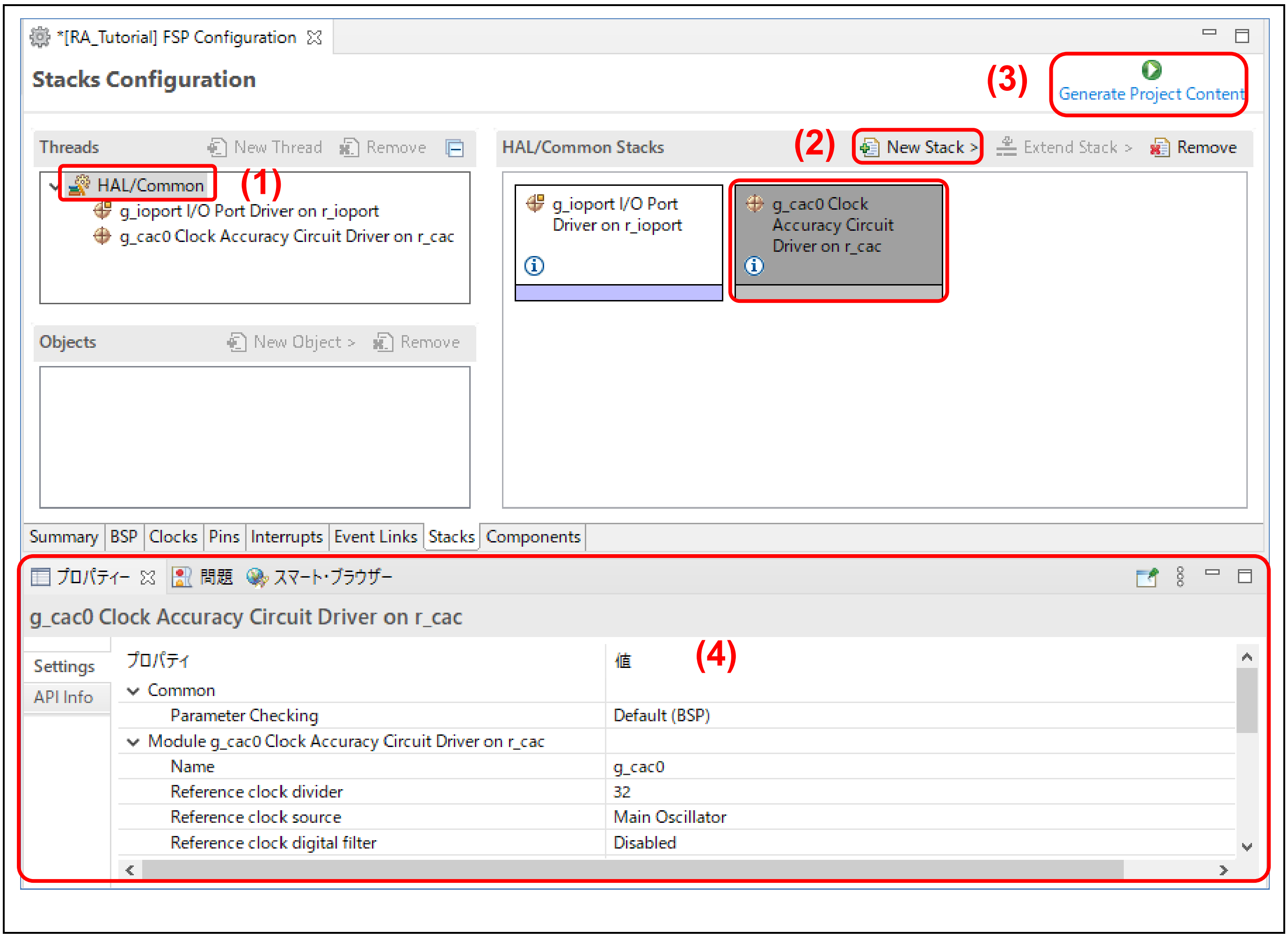


図 72　RA プロジェクト構成 ー スレッドにモジュールを追加

注：別の操作例として、「6.1　FreeRTOS での汎用 PWM タイマ使用例」では汎用 PWM タイマ（GPT）モジュールを “Blinky スレッド” に追加する手順を説明しています。

モジュールを追加すると、依存関係がある別のモジュールの追加や、構成の設定が必要になる場合があります。依存関係があるモジュールのうち、必要なものは自動で追加されます。任意で追加したいものはユーザが手動で追加します。追加可能なモジュールは別に表示されますので、追加したいものをクリックして選択し、プロパティを設定してください。（例：“New Stack” → [Middleware] → [USB] → [USB PCDC driver on r_usb_pcdc] の順に選択）。

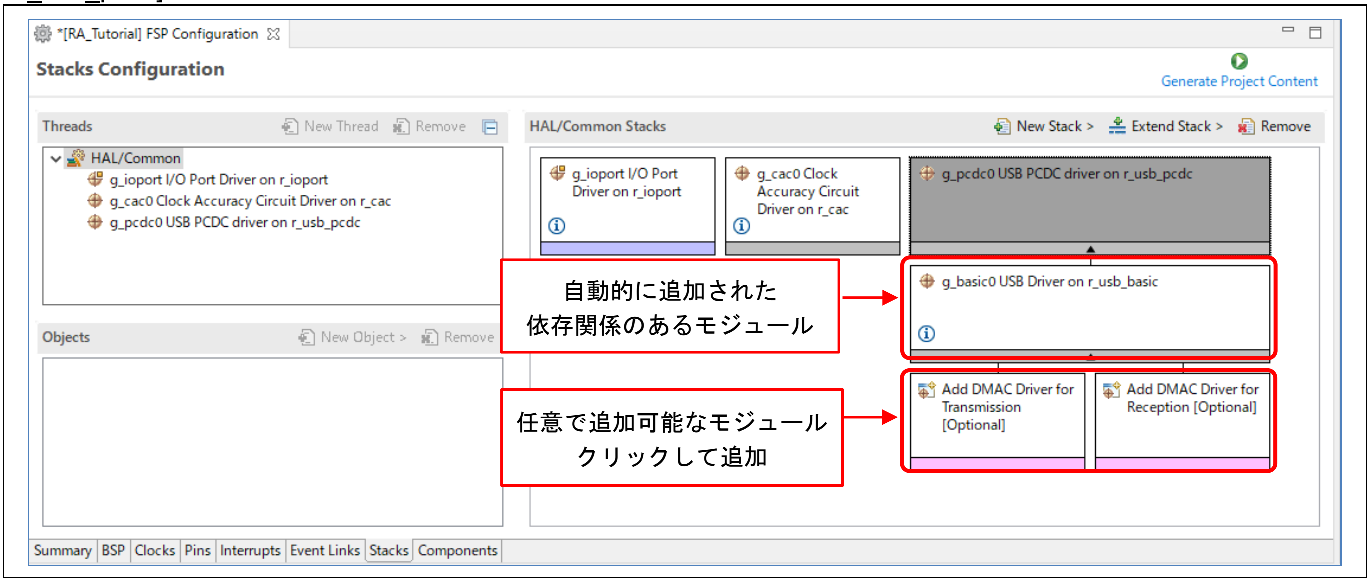


図 73　RA プロジェクト構成 － 依存関係のあるモジュール

[Stacks Configuration] ページでは、コピー＆ペーストでモジュールやモジュールスタックを追加することもできます。モジュール上で右クリックし、[コピー] を選択してコピーしてください。次に、同じプロジェクト内の同じスレッドまたは異なるスレッドの [Stacks] ペインで右クリックし、[貼り付け] を選択してください。カット＆ペーストも可能です。

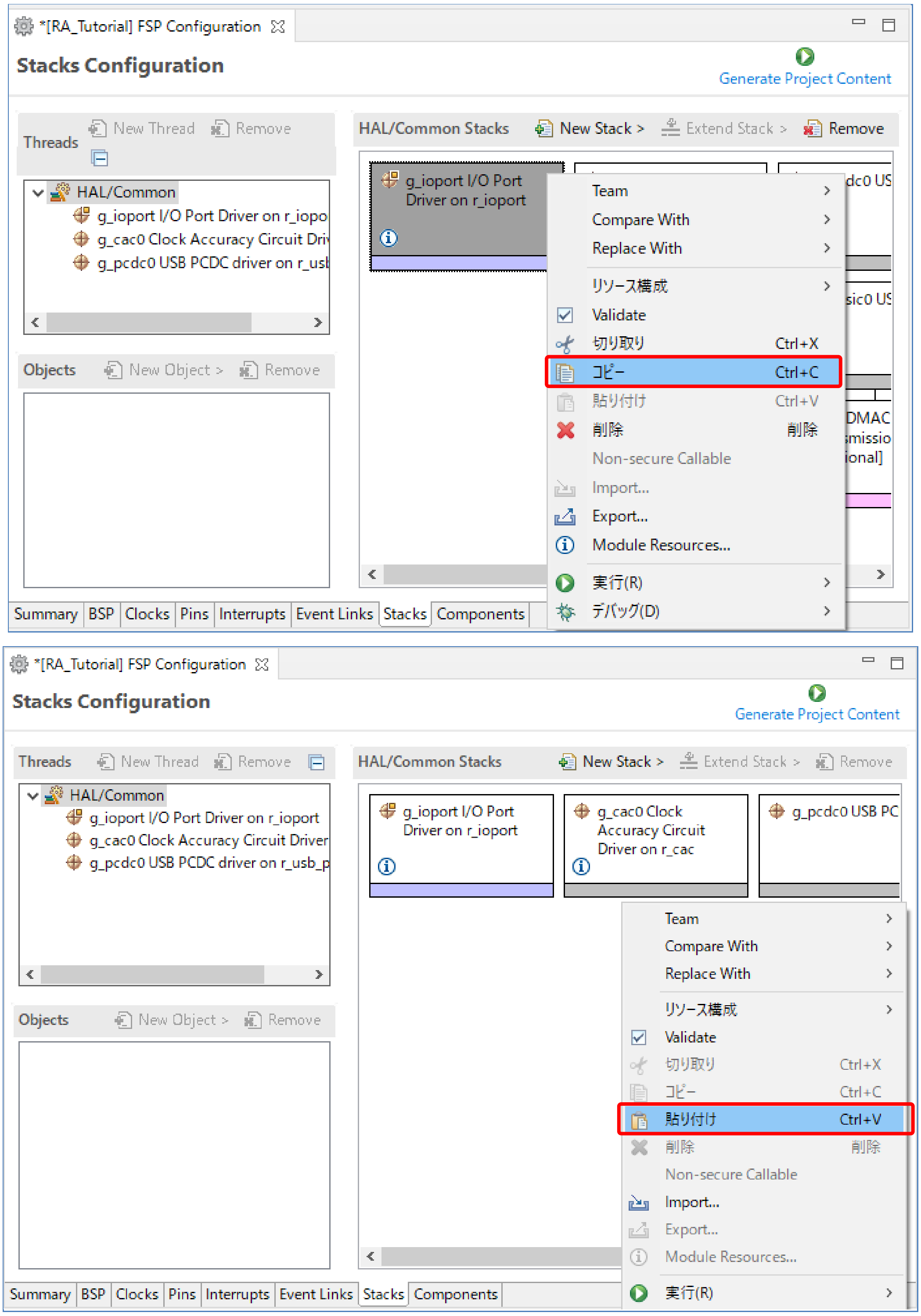


図 74　コピー＆ペーストの操作

モジュールをコピーすると、既存のモジュールとの間でモジュール名が競合します。一方のモジュールインスタンス名を変更して競合を解消してください。
ただし、g_ioport をコピーした場合は競合が発生しません。

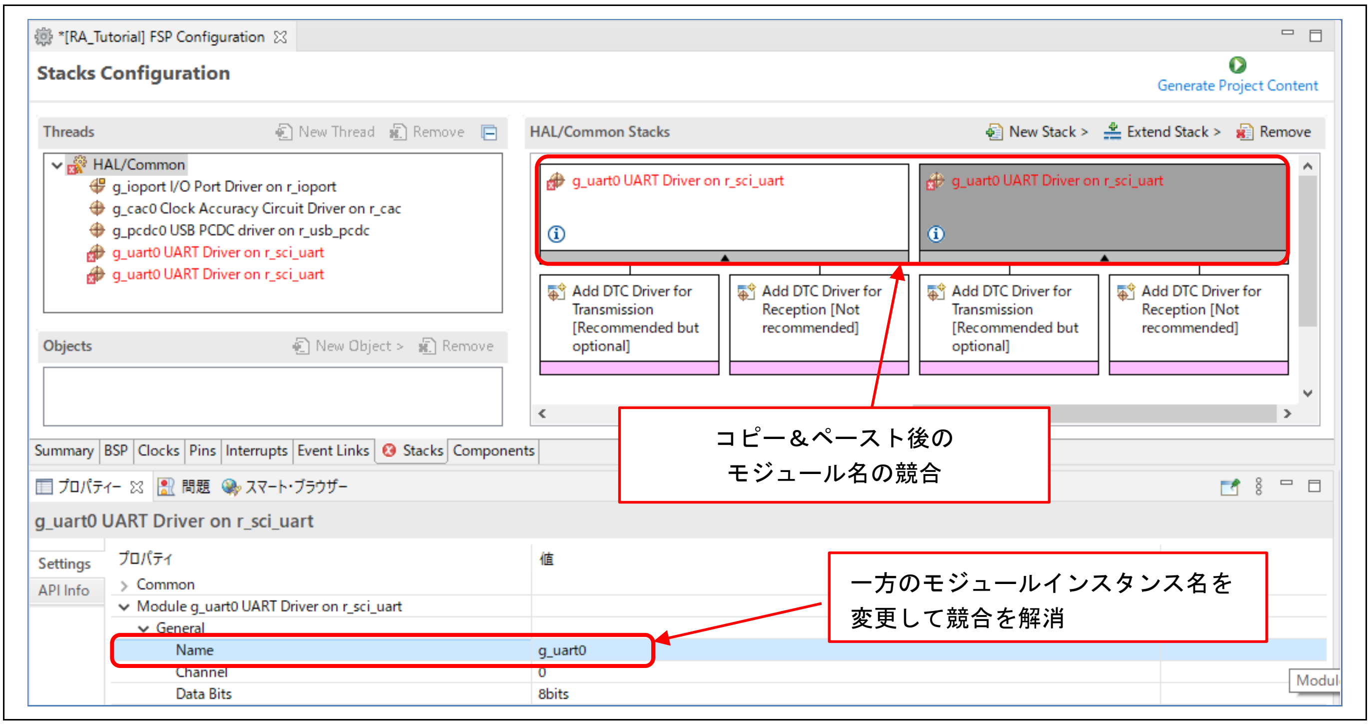


図 75　モジュールインスタンス名の競合

[Stacks Configuration] ページでは、エクスポート&インポート操作でモジュールやモジュールスタックを追加することもできます。モジュール上で右クリックし、[Export…] を選択してモジュールの構成を XML ファイルにエクスポートしてください。次に、同じプロジェクト内の同じスレッドまたは異なるスレッドの [Stacks] ペインで右クリックし、[Import...] を選択して、エクスポート済みの XML ファイルから構成をインポートしてください。モジュール名の競合は、一方のモジュールインスタンス名を変更して解消します。

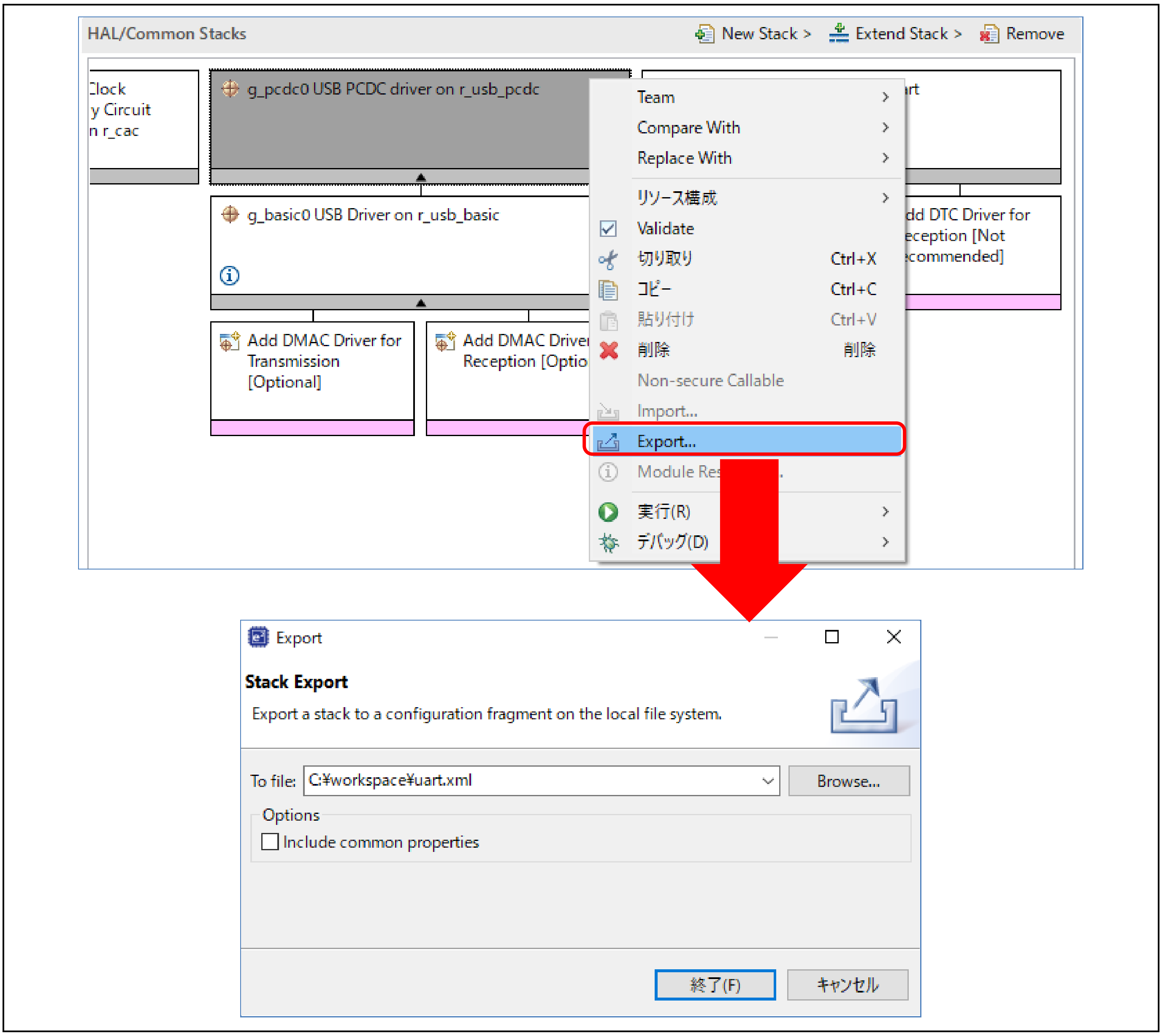


図 76　RA スタックのエクスポート

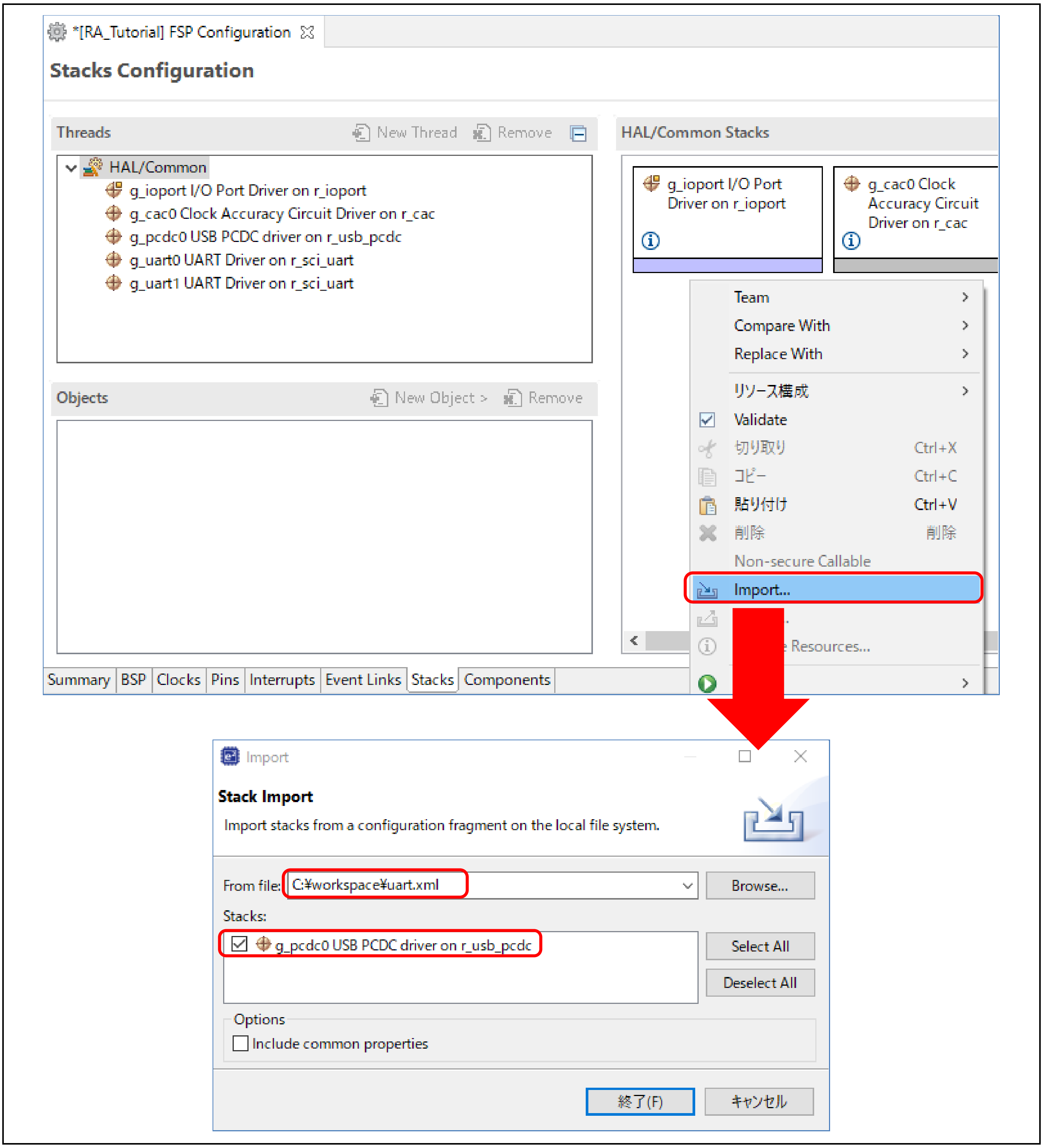


図 77 RA スタックのインポート

3.5.6　コンポーネント構成（Components Configuration）ページ

[Components Configuration] ページでは、アプリケーションで必要な個々のモジュールを取り込んだり、削除したりできます。

すべての RA プロジェクトに共通して必要なモジュールはあらかじめ選択されています（例：HAL Drivers → all → r_cac）。

[Stacks] ページで選択したドライバに必要なモジュールは、すべて自動で取り込まれます。その他のモジュールについては、必要なコンポーネントのチェックボックスをクリックすることで、取り込んだり削除したりできます。

注：アプリケーションにモジュールを追加するには、通常は [Stacks Configuration] ページを用います。[Components Configuration] ページは、主に、インストール済み FSP で使用可能なコンポーネント一覧を参照するために用います

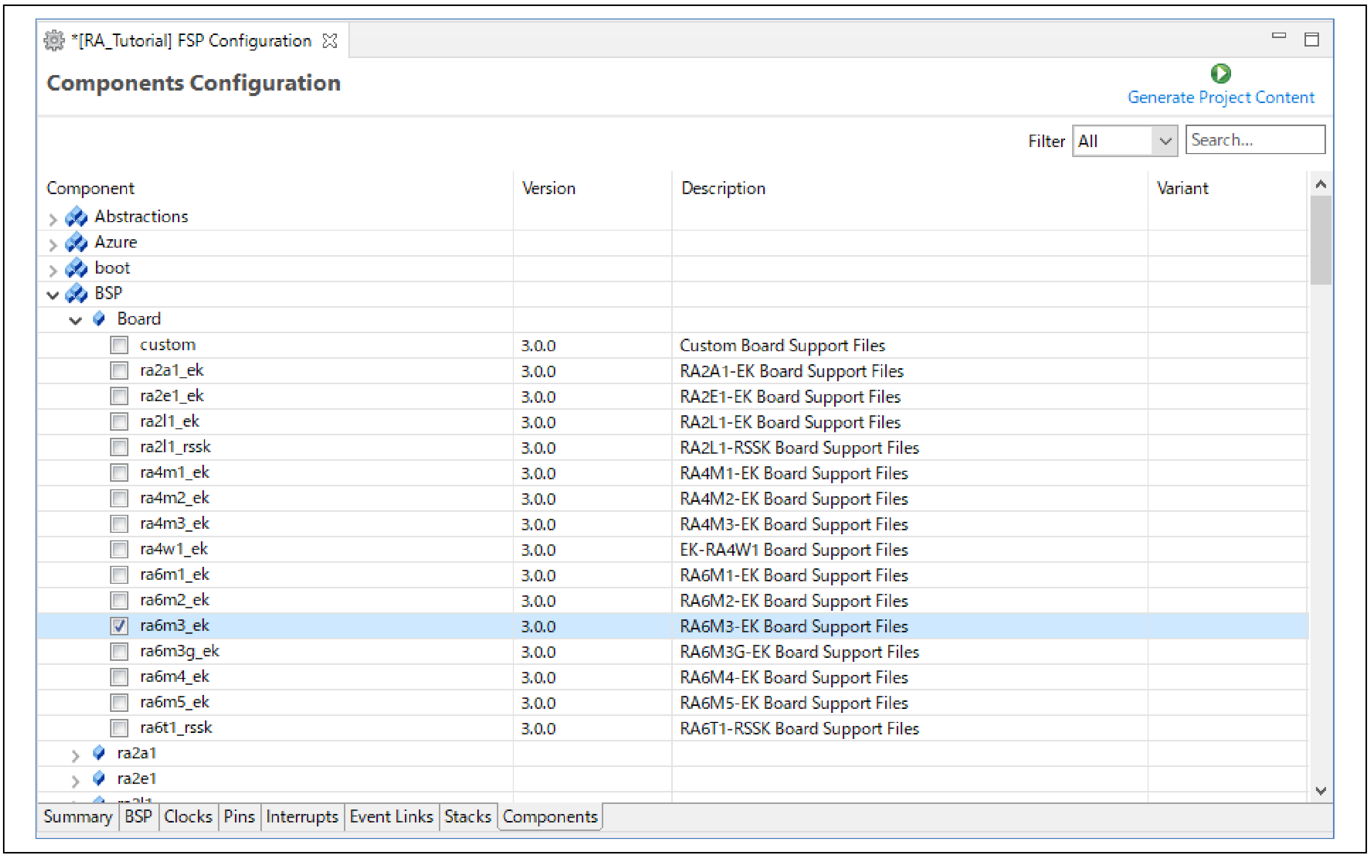


図 78　RA プロジェクト構成 ー [Components Configuration] ページ

3.5.7　割り込み構成（Interrupts Configuration）ページ

[Interrupts Configuration] ページでは、RA 割り込みフレームワークで使用するイベント（割り込み）や ISR（割り込みサービスルーチン）の管理ができます。

[Interrupts Configuration] ページは 2 つのペインで構成されています。

(1) [User Events] ペイン：ユーザが独自に作成したイベントの一覧を表示します。

(2) [Allocations] ペイン：「3.5.5　スタック構成（Stacks Configuration）ページ」で追加した RA モジュールのイベント一覧を表示します。

両ペインで、“Event” カラムはイベント名を表示します。“ISR” カラムは、対応する “Event” カラムのイベントの割り込みハンドラ名を表示します。

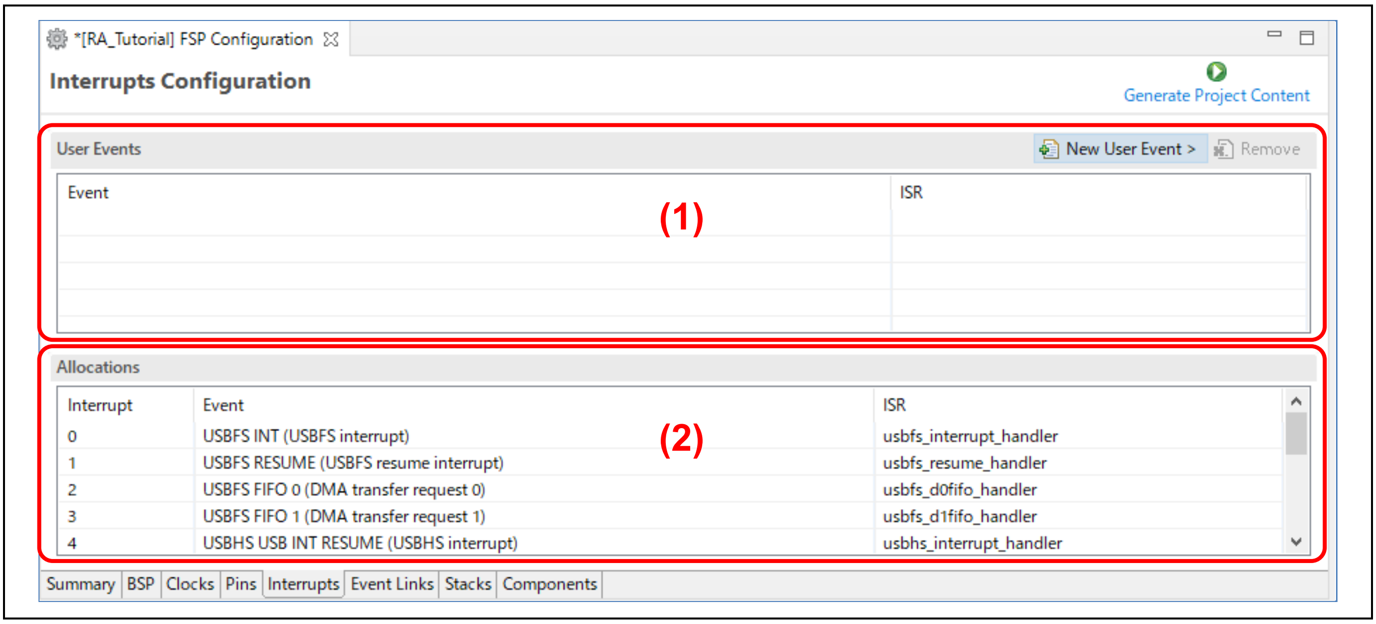


図 79　RA プロジェクト構成 ー [Interrupts Configuration] ページ

ユーザイベントと、それに対応する ISR を作成するには、[New User Event] ボタンをクリックし、作成するイベントを選択してください。

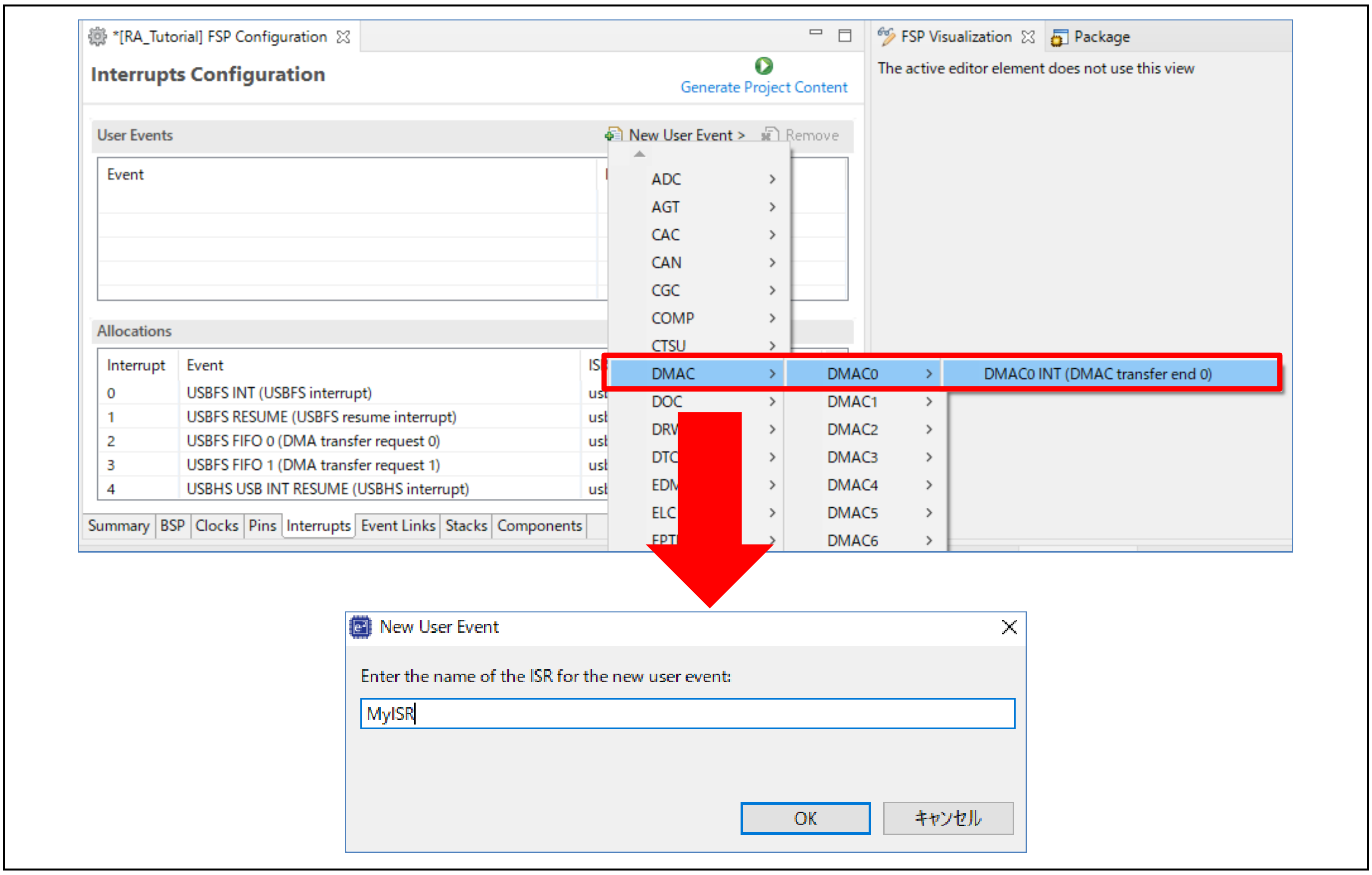


図 80 [Interrupts Configuration] ページ ― 新規ユーザイベントの追加

作成された新規イベントは “User Events” ペインに表示されます。

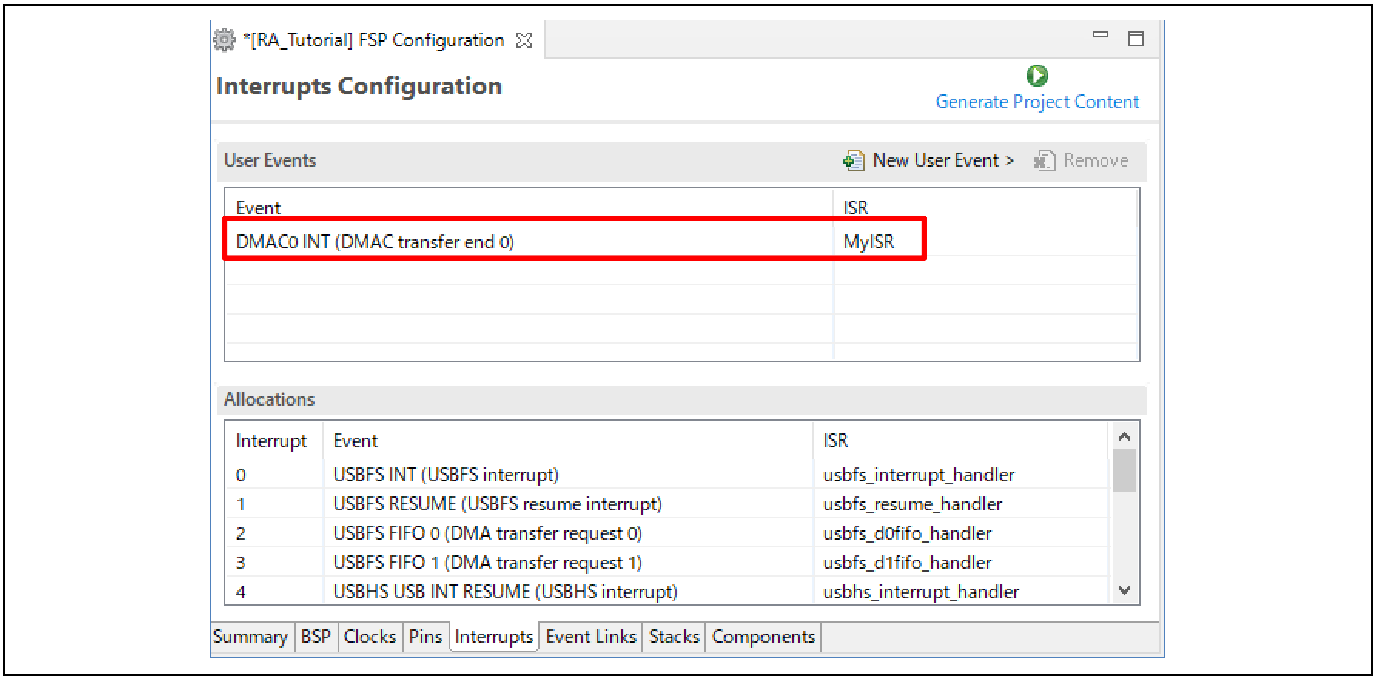


図 81 [Interrupts Configuration] ページ ― ソースコードの生成

ユーザイベントを削除するには、“User Events” ペインでイベントを選択し、 ボタンを押してください（RA のモジュール追加によって “Allocations” ペインに表示されたイベントは削除できません）。

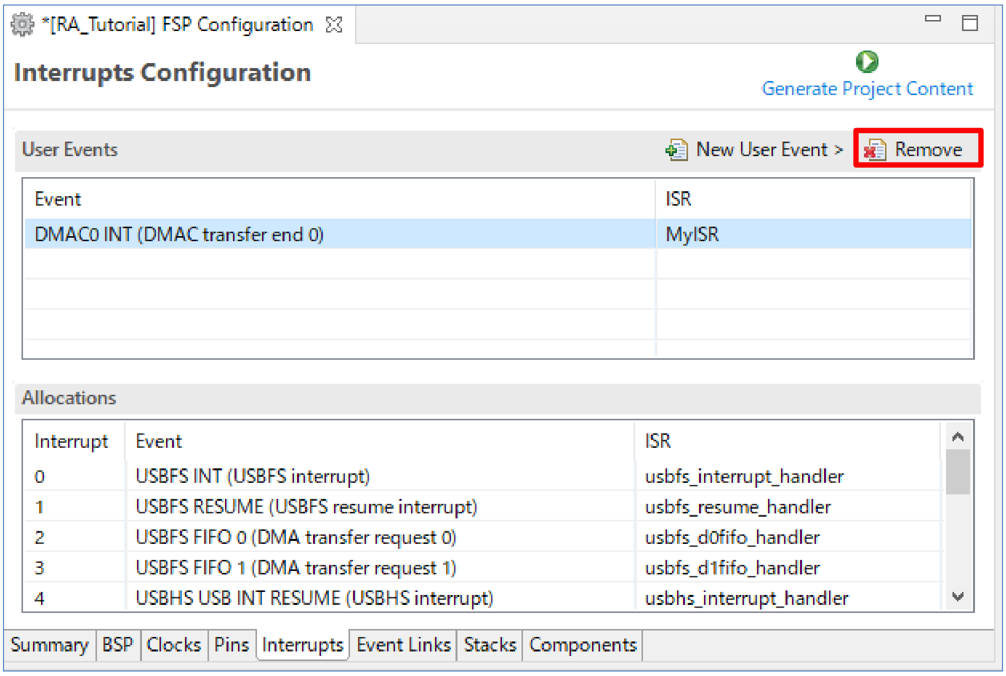


図 82　ユーザイベントの削除

3.5.8　イベントリンク構成（Event Links Configuration）ページ

[Event Links Configuration] ページでは、RA プロジェクト内の FSP ドライバ以外のドライバによるイベントリンクコントローラ（ELC）の使用を設定できます。ドライバが周辺機能で発生させる ELC イベントと、使用する ELC イベントを設定します。

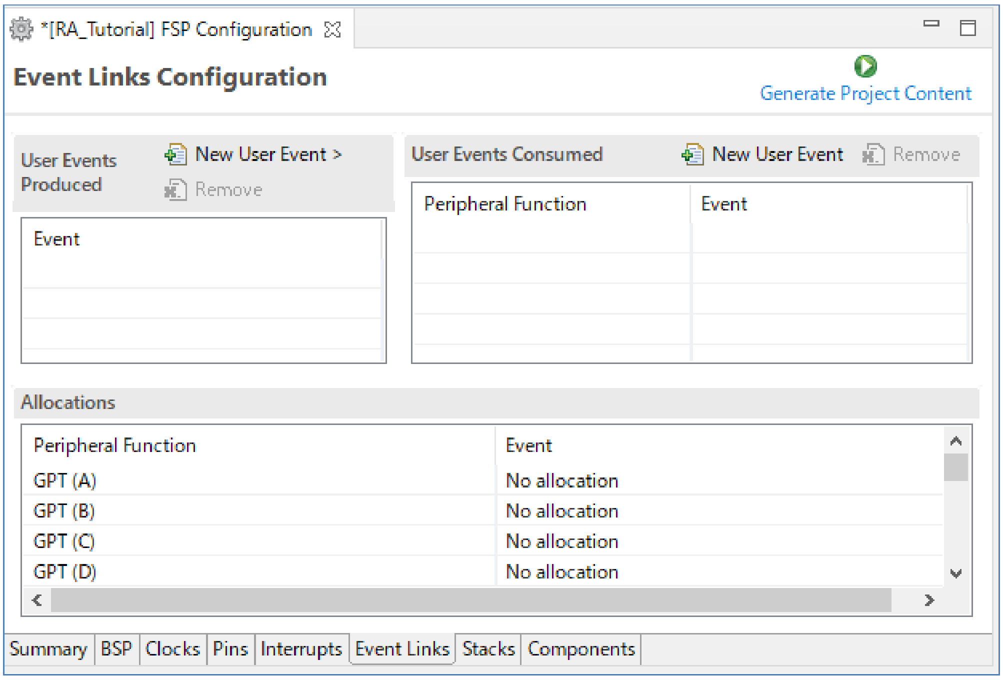


図 83　[Event Links Configuration] ページ

発生させるユーザイベントの設定：

(1) [User Events Produced] リストの [New User Event] ボタンを押します。
(2) カスケードメニューが開き、選択した RA デバイスがサポートする ELC イベントがすべて表示されます。
(3) メニューからイベントを選択すると、[User Events Produced] リストが更新されます。

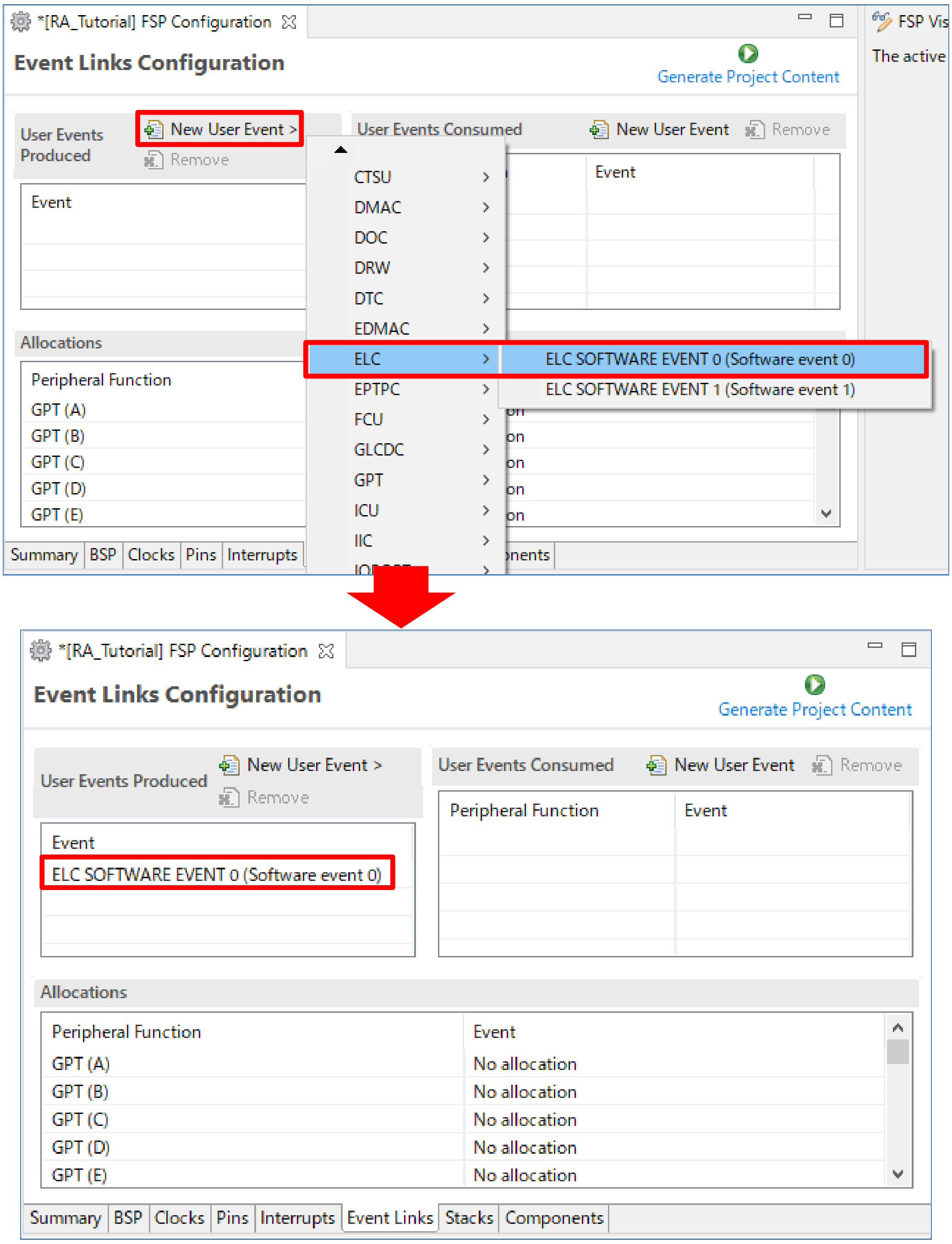


図 84　発生させるユーザイベントの設定

使用するユーザイベントの設定：

(1) [User Events Consumed] リストの [New User Event] ボタンを押します。
(2) [New User Event] ダイアログボックスが開き、使用できる周辺機能と ELC イベントのコンボボックスが表示されます。周辺機能とイベントを選択して [OK] ボタンを押します。
(3) RA configuration で、選択したイベントが選択した周辺機能に割り当てられ、[Event Links Configuration] ページの [User Events Consumed] リストと [Allocations] リストに表示されます。

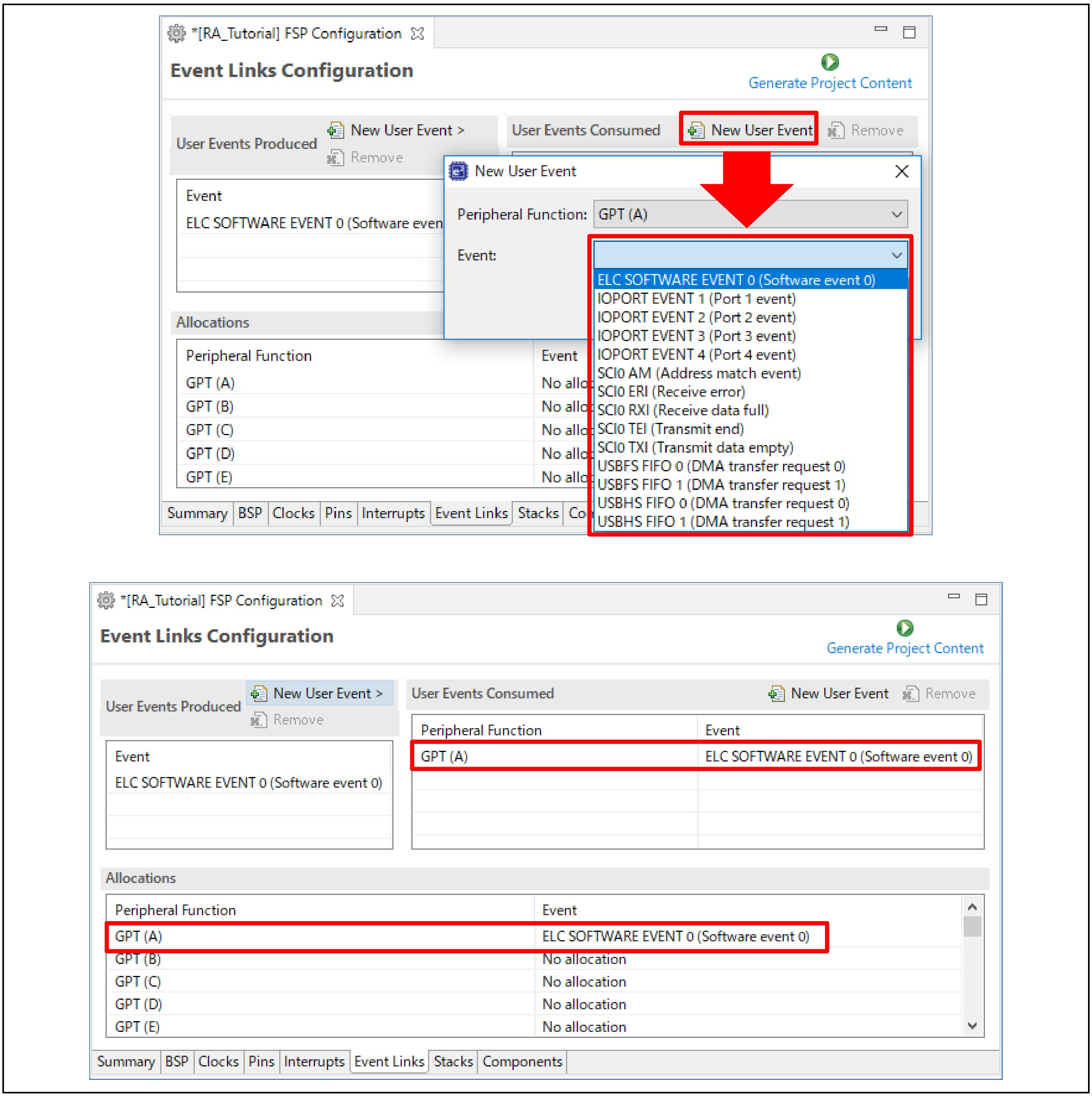


図 85 使用するユーザイベントの設定

3.6 エディタ画面の吹き出し機能

e² studio はテキストエディタ画面での吹き出し機能をサポートしています。[ウインドウ] メニュー → [設定] → [C/C++] → [エディター] → [吹き出し] の順に選択して、吹き出し機能を有効または無効にします。

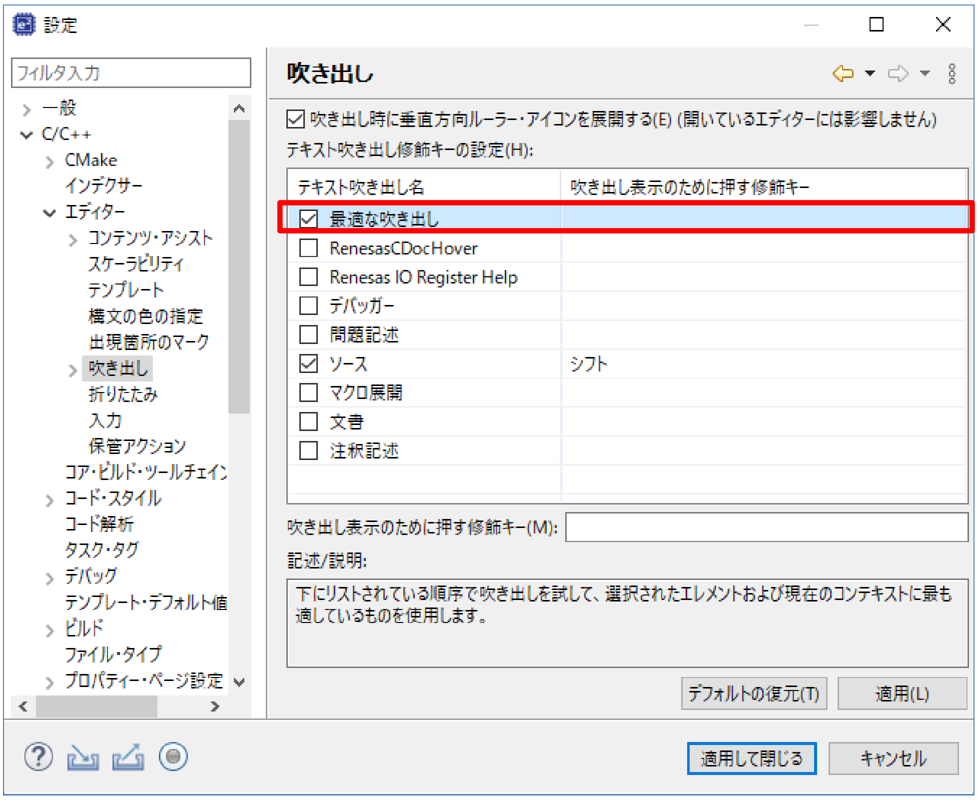


図 86　吹き出しの設定

吹き出しを有効にするには、“最適な吹き出し” を選択します。無効にするには、チェックをはずしてください。吹き出しはデフォルトで有効に設定されています。

吹き出し機能により、ソースコード中の各識別子の詳細情報を見ることができます。識別子の上にマウスポインタを置くとポップアップ表示が出ます。

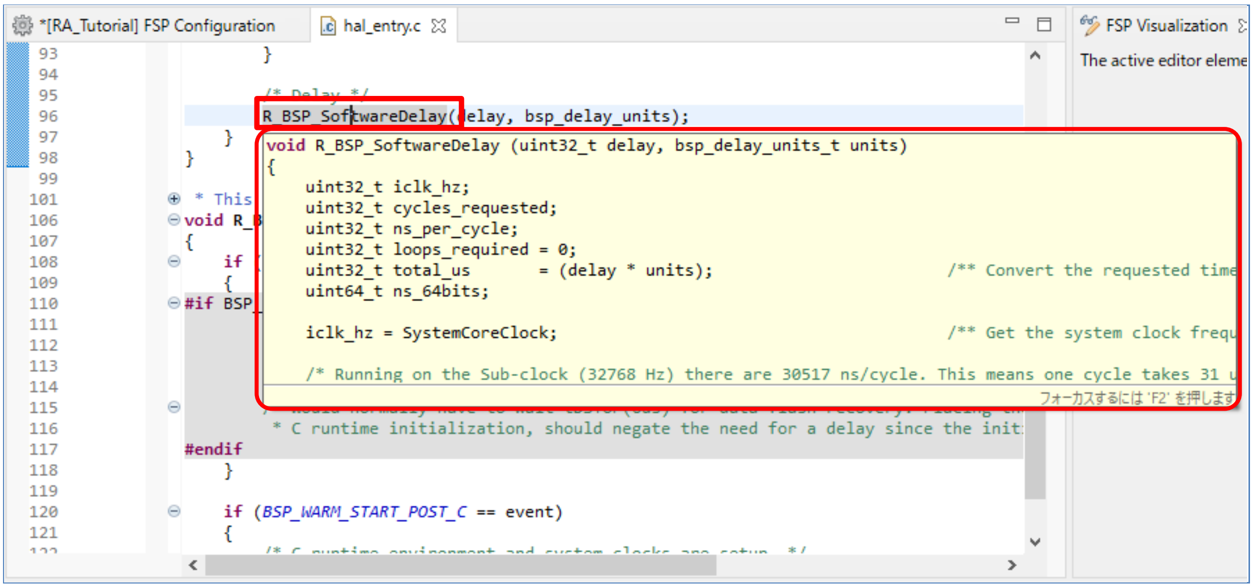


図 87　吹き出し機能による情報の表示

4. ビルド

この章では e² studio のビルドに関する設定や主なビルド機能について解説します。

4.1 ビルドオプションの設定

プロジェクト作成時はデフォルトのビルドオプションが生成され、通常はデフォルト設定でプロジェクトをビルドできます。

ビルドオプション（ツールチェーンのバージョンや、最適化オプションなど）を変更したい場合は、ビルドする前に以下の手順を実行してください。

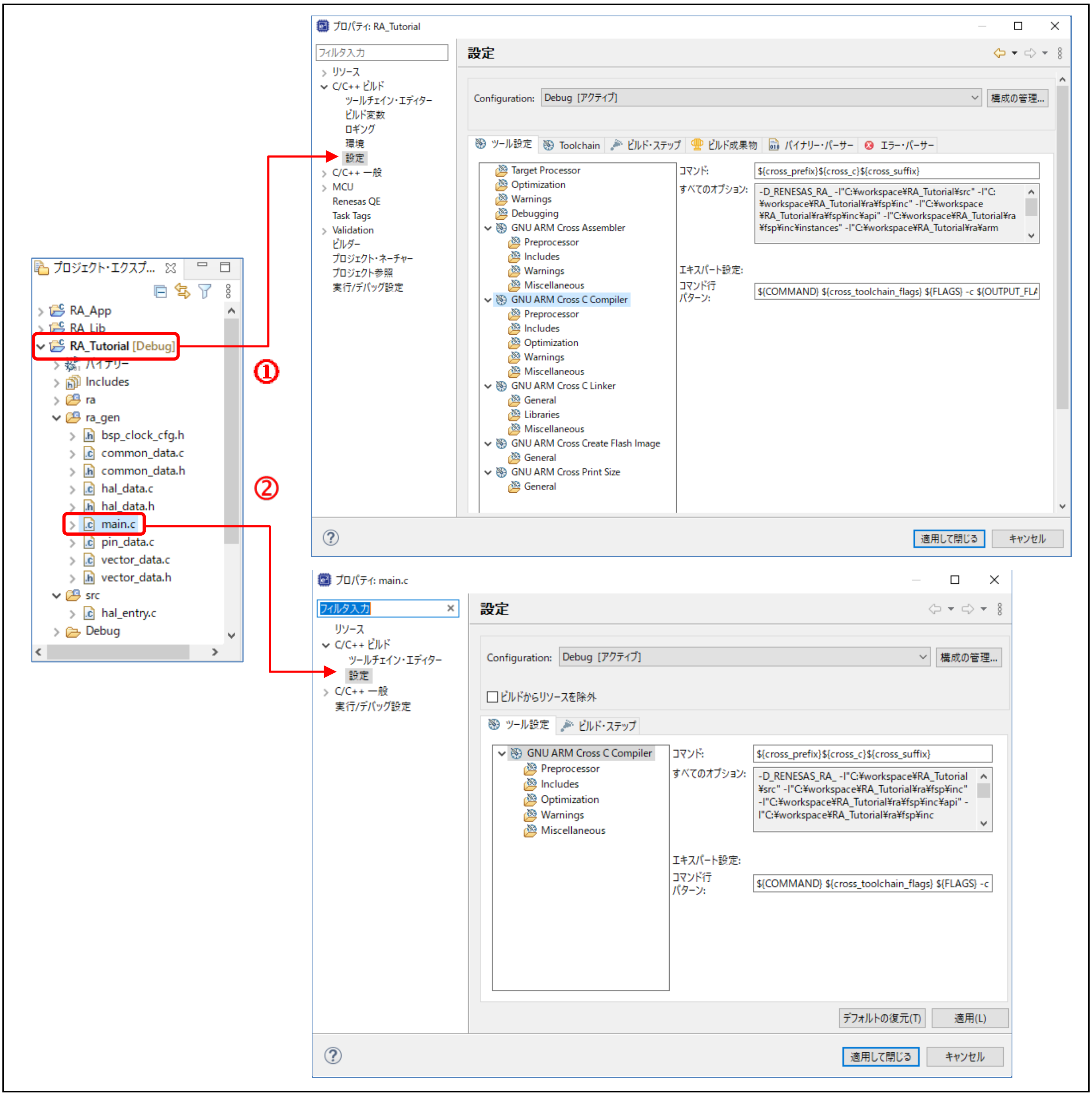


図 88　ビルド － RA プロジェクトのプロパティと “main.c” ソースファイル

ビルドオプションは、プロジェクトまたはソースファイルの [プロパティ] 画面で設定できます。

(1) ① プロジェクト名または ② ソーフファイル名を選択します。
(2) 右クリックして [プロパティ] を選択するか、ショートカットキー[Alt] +[Enter] で [プロパティ] 画面を開きます。
(3) “C/C++ビルド” → “設定” の順に選択して、オプション設定を表示し編集します。

[プロパティ] 画面はプロジェクト単位およびソースファイル単位で設定できます。プロジェクト単位の [プロパティ] 画面ではソースファイル単位より多くのプロパティを設定でき、その設定はプロジェクト内の全てのファイルに共通に適用されます。

4.2 サンプルプロジェクトのビルド

以下の手順でプロジェクトをビルドします。

(1) [プロジェクトエクスプローラー] ビューで RA プロジェクトをクリックして選択します。

(2) [プロジェクト] → [プロジェクトのビルド] の順に選択するか、 アイコンを押してプロジェクトをビルドします。

(3) ビルド終了後、エラーがないことを確認してください。

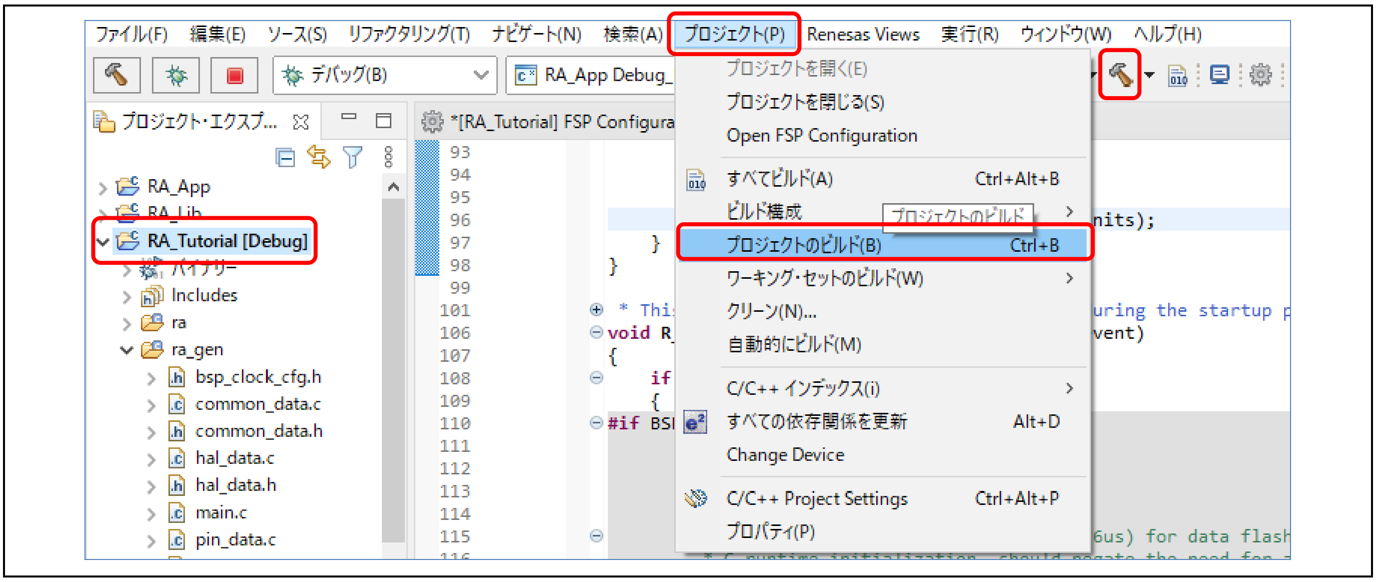


図 89　ビルド ― サンプルプロジェクトのビルド

4.3 プロジェクトレポート機能（ビルドオプション一覧の出力）

プロジェクトレポート機能により、e² studio のプロジェクトビルド設定をファイルに保存できます。

(1) [プロジェクトエクスプローラー] ビューで右クリックし、ポップアップメニューを表示します。

(2) [Save build settings report] を選択してビルド設定のレポートを保存します。

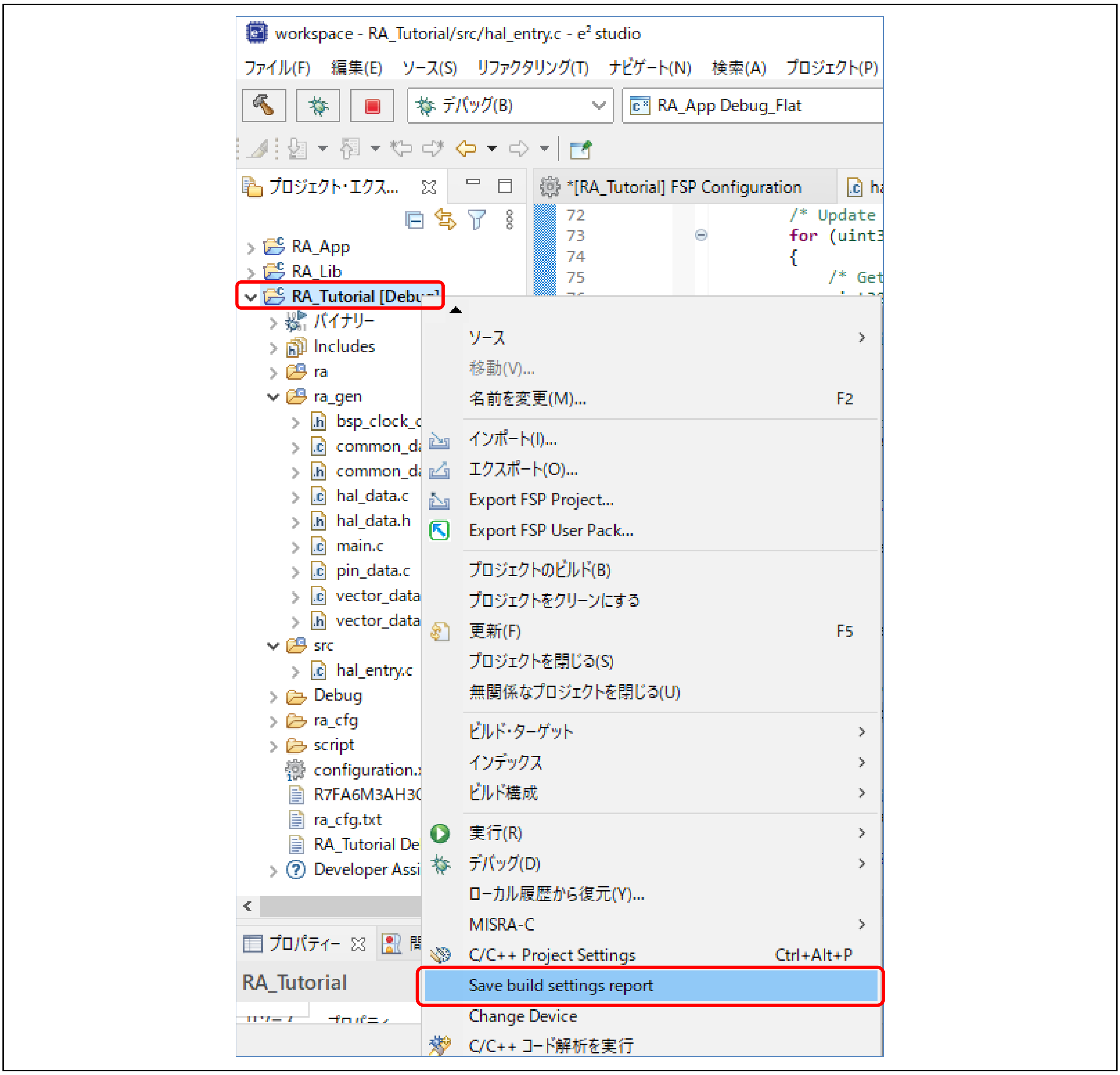


図 90 ビルド ― プロジェクトレポート機能（ビルドオプション一覧の出力）

5. デバッグ

この章では、e² studio のデバッグ構成と主なデバッグ機能の使い方について説明します。J-Link ARM エミュレータと RA6M3 EK ボードを動作環境とする RA プロジェクト（「4.2　サンプルプロジェクトのビルド」参照）を例に説明します。

いずれかのパースペクティブアイコン上で右クリックし、[テキストの表示] を選択してアイコン名を表示します。

図 91　デバッグ － [デバッグ] パースペクティブへの切り替え

e² studio で RA プロジェクトを開き、[デバッグ] をクリックして [デバッグ] パースペクティブに切り替えます。

Eclipse で「パースペクティブ」とは、あらかじめペインとビューを組み合わせてワークベンチウィンドウのレイアウトを定義したものを指します。それぞれのパースペクティブは、特定の用途向けのビュー、メニュー、ツールバーの組み合わせで構成されます。たとえば、以下のようなパースペクティブがあります。

- [デバッグ] パースペクティブ：プログラムのデバッグに必要なビューを表示します。
- [FSP Configuration] パースペクティブ：エディタ画面の “configuration.xml” とあわせて FSP configuration を開き、プロジェクト構成を設定する [Visualization] ビューや [プロパティ] ビューを表示します。
- [C/C++] パースペクティブ：ユーザが C/C++プログラム開発を行うために必要なビューを表示します。

[デバッグ] パースペクティブ以外でユーザがデバッガに接続しようとすると、e² studio は [デバッグ] パースペクティブに切り替えるようユーザに促します。

一つのワークベンチのウィンドウの中に、一つまたは複数のパースペクティブを表示することができます。ユーザはパースペクティグをカスタマイズしたり、新しいパースペクティブを追加したりすることができます。

5.1　既存デバッグ構成の変更

RA プロジェクトを作成すると、プロジェクトごとに一度だけデフォルトのデバッグ構成が自動で作成されます。既存のデバッグ構成は以下のように変更できます。

(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名をクリックします。

(2) [実行] → [デバッグの構成…] あるいは  アイコン（下向き矢印） → [デバッグの構成] の順にクリックし、[デバッグ構成] ウィンドウを開きます。

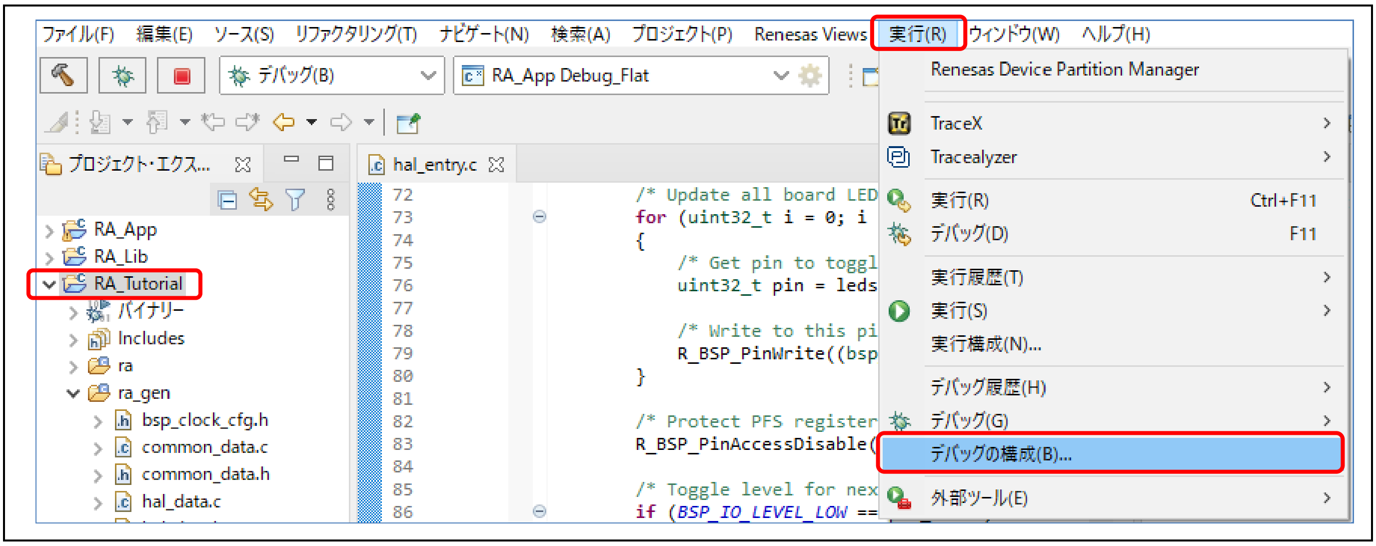


図 92　デバッグ － [デバッグ構成] ウィンドウを開く

(3) [デバッグ構成] ウィンドウで、“Renesas GDB Hardware Debugging” デバッグ構成の表示を展開し、既存のデバッグ構成（例：“RA_Tutorial Debug_Flat”）をクリックしてください。

(4) [メイン] タブを選択し、プロジェクトビルドフォルダにあるロードモジュール（例："RA_Tutorial.elf"）を指定してください。

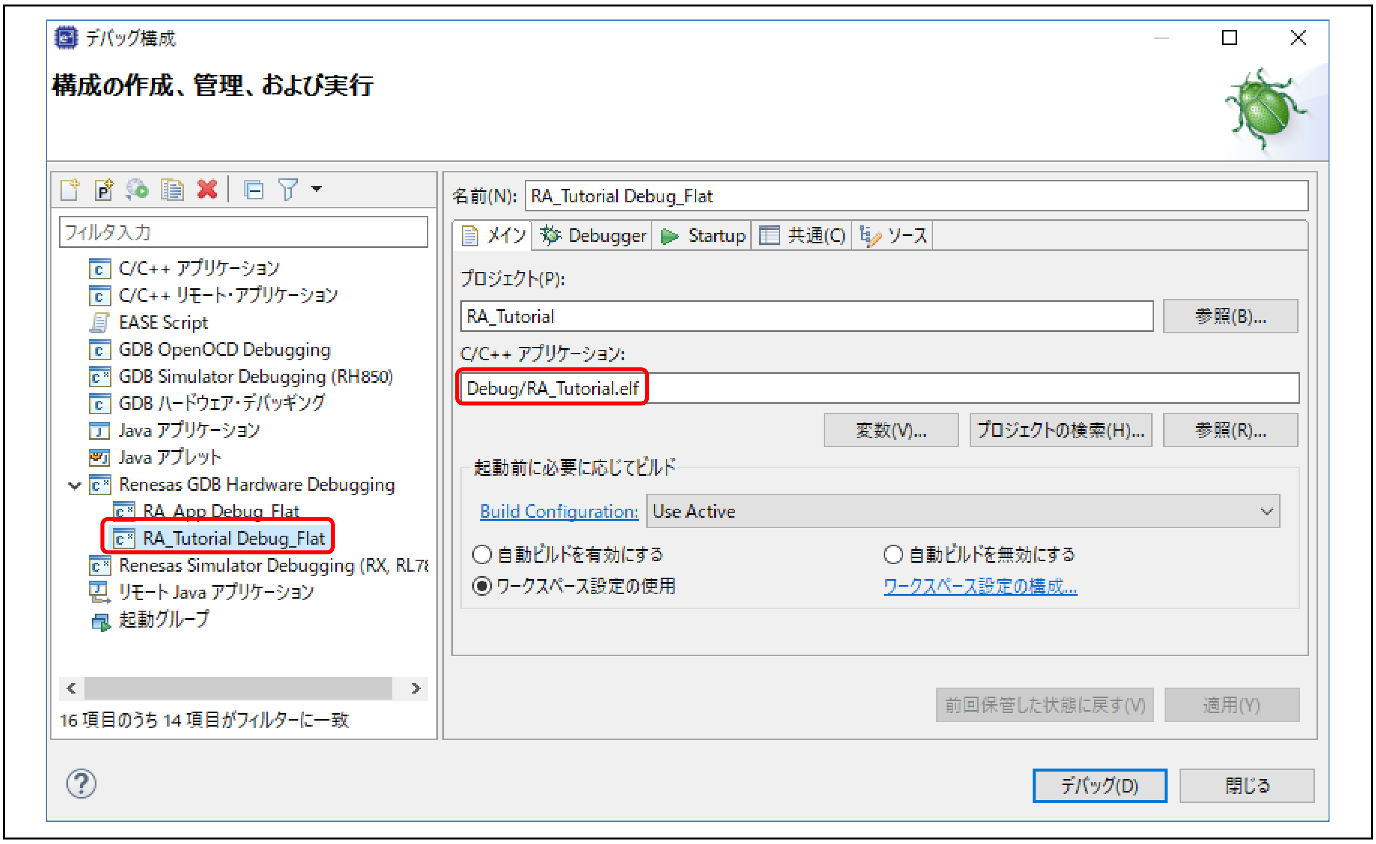


図 93　デバッグ ― ロードモジュールの選択

(5) [Debugger] タブに切り替え、エミュレータの種類とデバイス名を選択してください。

Debug hardware："J-Link ARM"

Target Device："R7FA6M3AH"

(6) [適用] ボタンを押すと設定が保存されます。

(7) [デバッグ] ボタンを押すと、デバッグ起動構成が実行され、J-Link エミュレータと RA ボードへ接続されます。

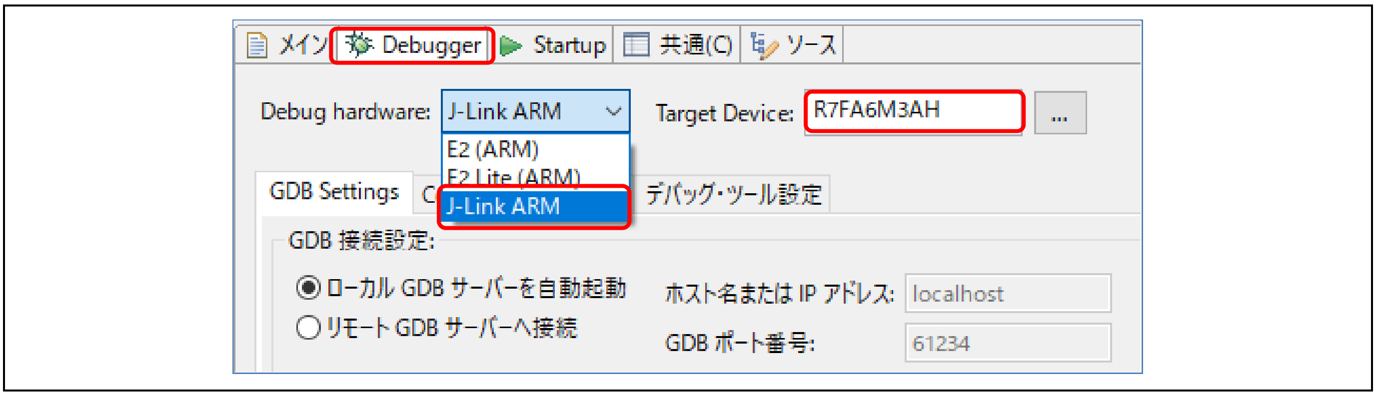


図 94　デバッグ ― 接続設定の変更

(8) 正しく接続できた場合は、図に示すような [デバッグ] ビュー画面が表示されます。接続後はエントリポイント（例："startup.c" の "Reset_Handler()"）でプログラムの実行が一旦中断されます。

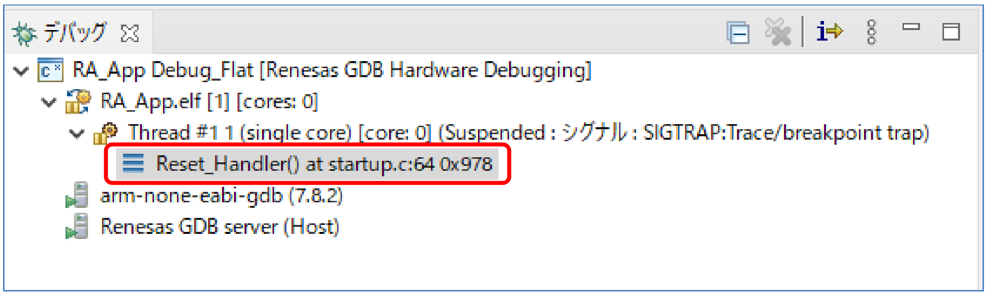


図 95　デバッグ ー ターゲット接続後の [デバッグ] ビュー表示

5.2　新規デバッグ構成の作成

別の種類のエミュレータを使用したいなどで、デバッグ構成を追加する場合に、簡単な方法は既存の構成を複製する方法です。以下の手順で行います。

(1) [デバッグ構成] ウィンドウを開きます（「図 92」参照）。

(2) [デバッグ構成] ウィンドウの左ペインに表示されたデバッグ構成（例：“RA_Tutorial Debug_Flat”）を選択し アイコンをクリック（現在選択している起動構成をコピー）すると、複製されたデバッグ構成（例：“RA_Tutorial Debug_Flat(1)”）が作成されます。

(3) デバッグ構成は「5.1　既存デバッグ構成の変更」で説明した方法で設定できます。

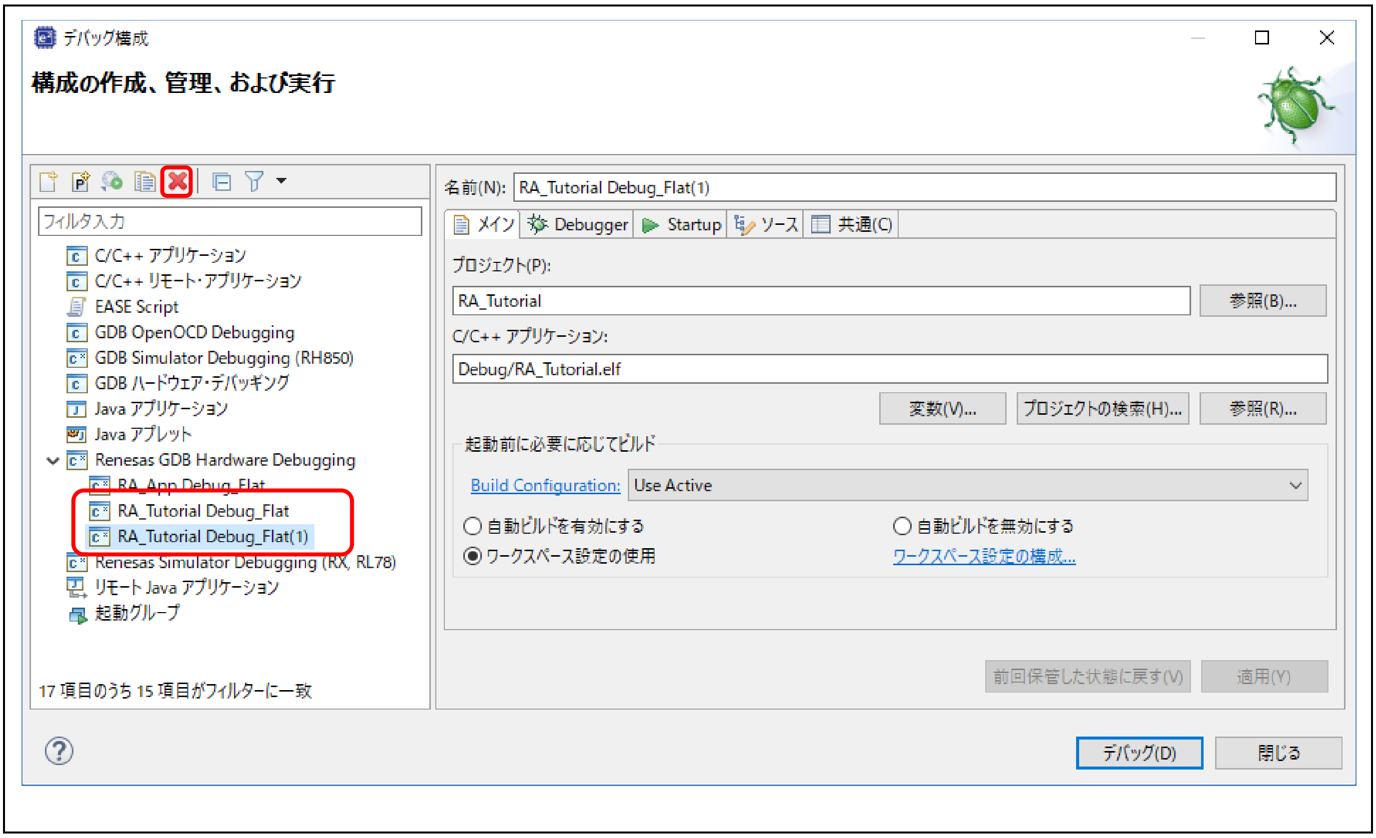


図 96　デバッグ ー 選択したデバッグ構成の複製

5.3　基本的なデバッグ機能

この章では、e² studio がサポートする以下の代表的なデバッグビューを説明します。

- Eclipse フレームワークに標準的な GDB デバッガの機能のビュー（ウィンドウ）：ブレークポイント、式、レジスタ、メモリ、逆アセンブル、変数（MMU ビューは RA 用にはサポートされていません）
- GDB デバッガの Renesas による拡張機能のビュー：IO レジスタ、イベントポイント、トレース、Fault Status

[デバッグ] ビューには、下記のような便利なツールバーがあります。

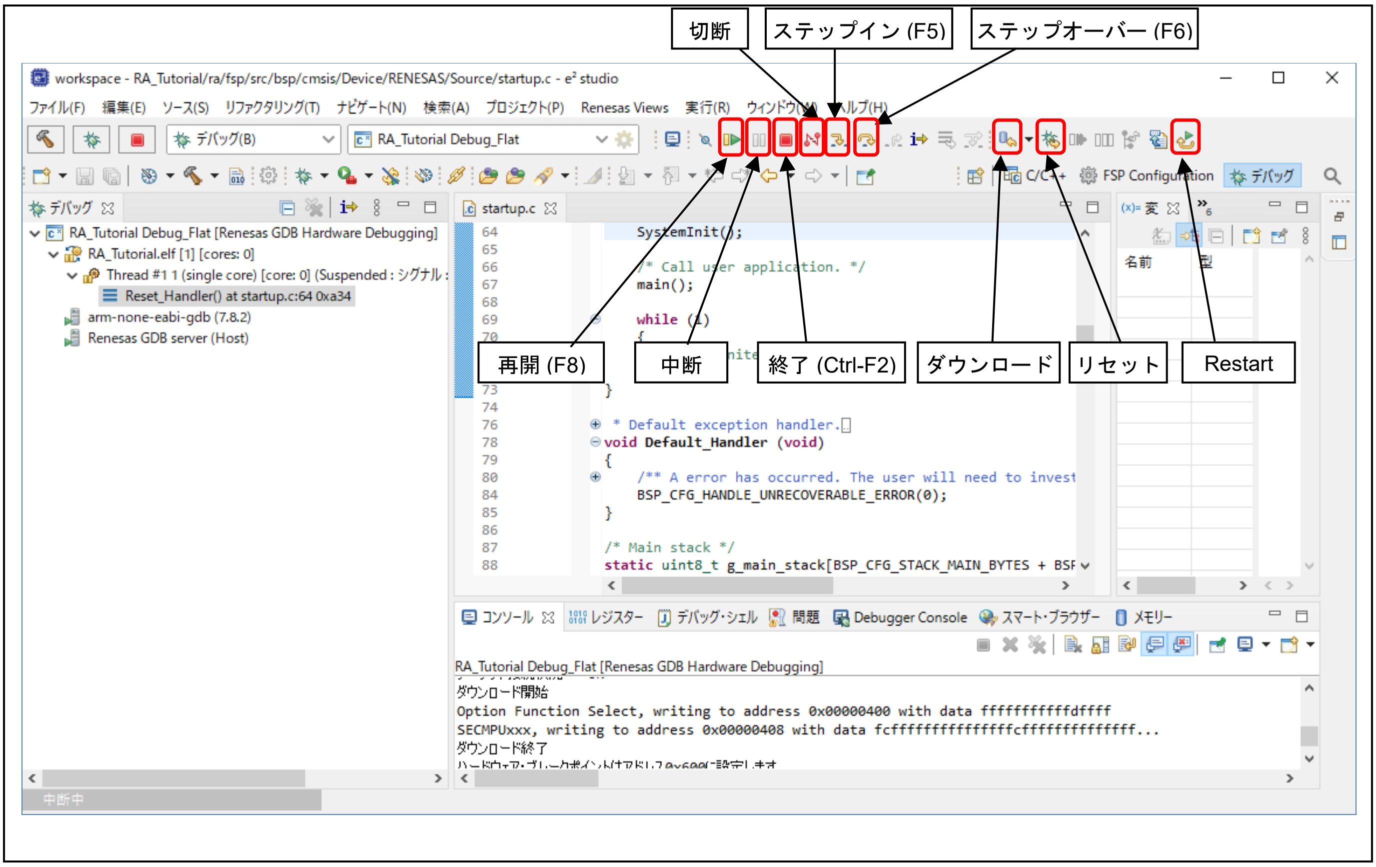


図 97　デバッグ ― [デバッグ] ビューの便利なツールバー

プログラムを実行するには ボタンをクリックするか [F8] キーを入力します。

プログラムは、ブレークポイントで、あるいは ボタンをクリックすることで一時停止します。一時停止中は以下の操作が可能です。

- ボタンまたは [F5] キーは、現在実行中のプログラム行にある次の関数呼び出しへステップイン実行します（関数内に入って１ステップ実行）。
- ボタンまたは [F6] キーは、現在実行中のプログラム行にある次の関数呼び出しをステップオーバー実行します（１行実行するが関数内には入らない）。
- ボタンで、実行を再開します。

デバッグセッションの停止は、選択したデバッグセッション／プロセスを ボタンで停止するか、選択したプロセスとデバッガを ボタンで切断します。

他に以下のような操作が可能です。

-  ボタンは、リセット後にプログラムを実行します。デバッグ構成でブレークポイントが設定されていると、main()関数で実行停止します。
-  ボタンは、プログラムをパワーオンリセットのエントリポイントにリセットします。
-  ボタンは、ターゲットシステムにバイナリファイルを再びダウンロードします。

5.3.1　デバッグビュー

[デバッグ] ビューには、実行した関数がスレッドごとに表示されます。

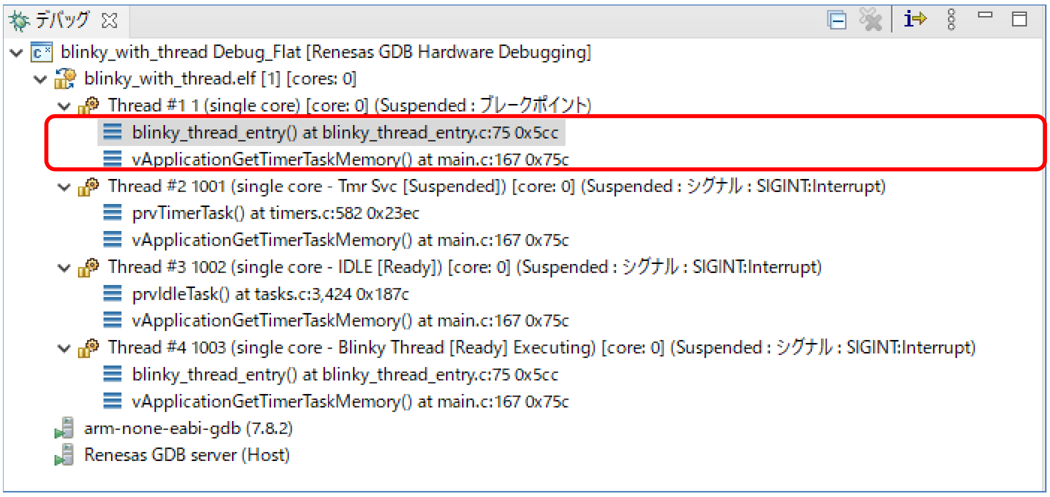


図98　デバッグ － [デバッグ] ビュー

[デバッグ] ビューに実行済み関数を表示する機能は、[デバッグ構成] ウィンドウで設定します。

1. メニューから [実行] → [デバッグの構成…] の順に選択し、[デバッグ構成] ウィンドウを開きます。
2. [Debugger] タブを選択し、次に [デバッグ・ツール設定] タブを選択してください。
3. "デバッグビューにRTOSのスレッド情報を表示する" を "はい" に設定してください。"いいえ" を選択すると、関数を表示する機能は使用できません。デフォルト設定は "はい" です。

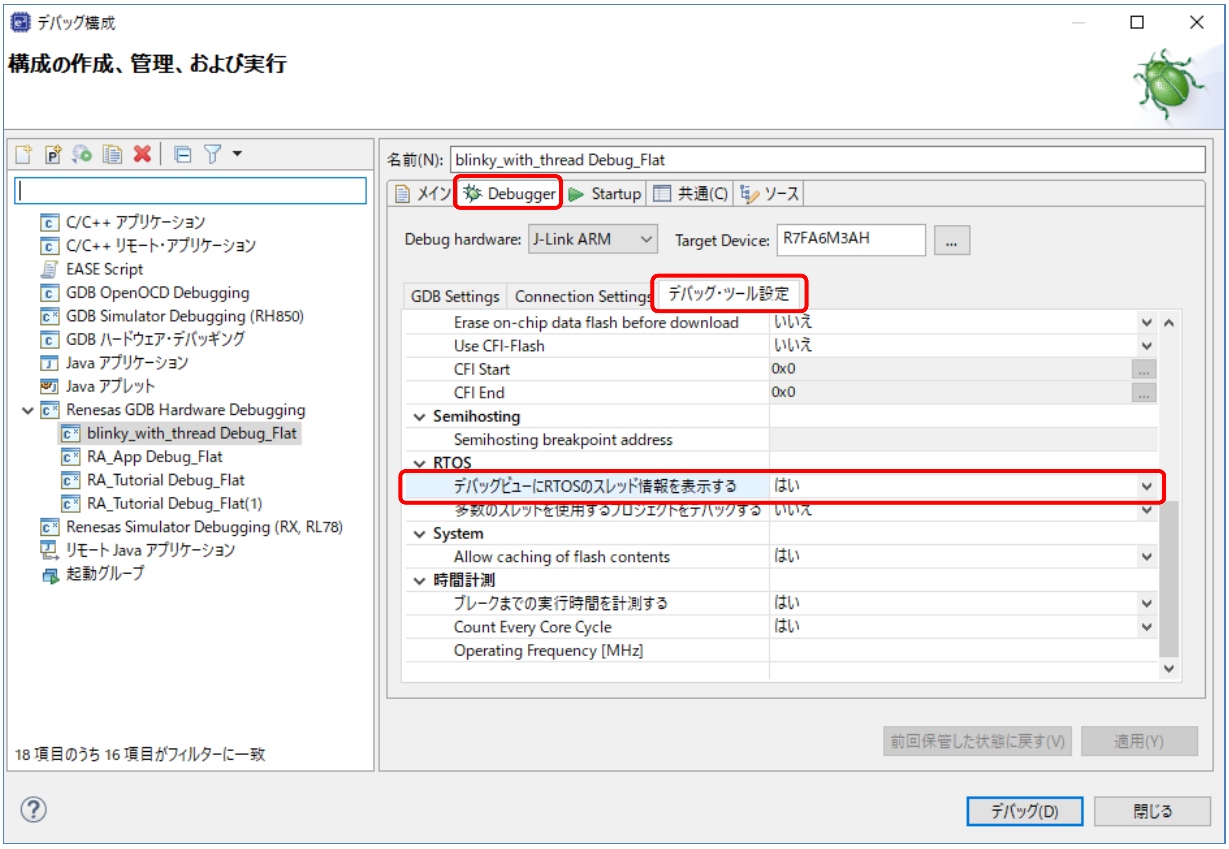


図99　デバッグ – デバッグ・ツール設定

5.3.2 ブレークポイントビュー

[ブレークポイント] ビューには、プログラムの実行可能な行に設定されたブレークポイントが表示されます。デバッガでプログラムを実行中、有効なブレークポイントの行またはアドレスに達すると実行が中断されます。

e2 studio ではソフトウェアブレークポイント（ のマーカー）とハードウェアブレークポイント（ のマーカー）を区別して設定できます。ダブルクリックによりブレークポイントを設定すると、デフォルトのブレークポイント型が適用されます。ハードウェアブレークポイントの場合、ブレークポイント用のハードウェアリソースが残っていないときは設定できません。その場合は、ソフトウェアブレークポイントに置き換えるとのメッセージが表示されます。

ブレークポイントを設定するには、デバッガが起動した状態で以下の操作を行います。

(1)　エディタ画面上でブレークポイントを設定する

ソースファイルエディタの左余白をダブルクリックすると、デフォルトの種別でブレークポイントが設定され、ブレークポイントマーカーが表示されます。マーカーが ならハードウェアブレークポイント、 ならソフトウェアブレークポイントです。

ダブルクリックではなく、ソースコードの左余白を右クリックすると [Toggle Software Breakpoint] または [Toggle Hardware Breakpoint] で直接ブレークポイント種別を選択できます。

(2)　「ブレークポイント」ビューでブレークポイントプロパティを設定する

[ウィンドウ] メニューの [ビューの表示] （または [Alt] +[Shift] +[Q] ショートカットに続けて [B] を押す）で [ブレークポイント] ビュー（ のアイコンのビュー）を開くとブレークポイントの有効・無効、ブレークポイント種別やブレークポイントを設定する行番号やアドレス等の変更が行えます。

以下の方法でブレークポイントを無効（ブレークポイントで止まらない）にすることができます。

- エディタの左余白に表示されたブレークポイントマーカー（ または ）を右クリックし、コンテキストメニューの「ブレークポイントを使用不可にする」を選択すると無効に切り替わります。無効になるとブレークポイントマーカーが白くなります（ または ）。
  再度コンテキストメニューの「ブレークポイントを使用可能にする」で有効に戻せます。あるいは、シフトを押しながらダブルクリックすると無効・有効が切り替わります。
- 全てのブレークポイントを一括して無効に切り替えるには、ツールバーまたはブレークポイントビューの ボタンをクリックします。一括して無効になっている間は、全てのブレークポイントマーカーに斜線（＼）が重ねて表示され、どのブレークポイントでも止まらなくなります。

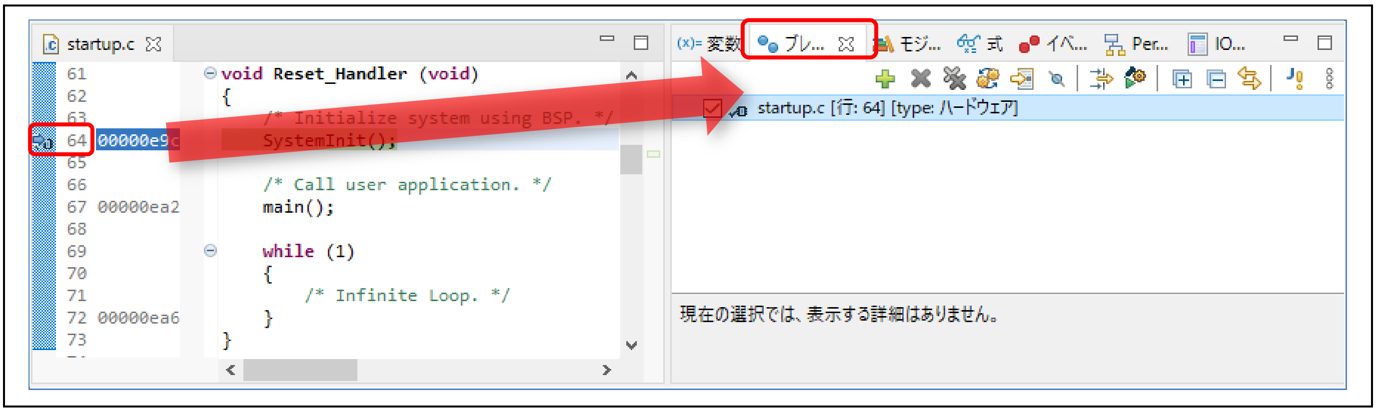


図 100　デバッグ － [ブレークポイント] ビュー

5.3.3 式ビュー

[式] ビューでは、デバッグ中のグローバル変数、静的変数、ローカル変数の値をモニタできます。
変数名で登録する他に、計算式（例：“Aval+Bval*2”）や型キャスト（例：“(struct mystr *)&buf[1] ”）を使って表示させることができます。

変数を見るには以下の手順を行なってください。

(1) [ウィンドウ] メニュー → [ビューの表示] → [式] の順に選択、あるいは アイコンをクリックし、[式] ビューを開きます。
(2) エディタ上で変数名（例：“bsp_common.c” 内の “g_fsp_version”）を選択状態にしてから [式] ビューへドラッグ&ドロップします。（または、変数を右クリックして [監視式を追加(A)...] メニュー項目を選択し、[式] ビューに追加します。）

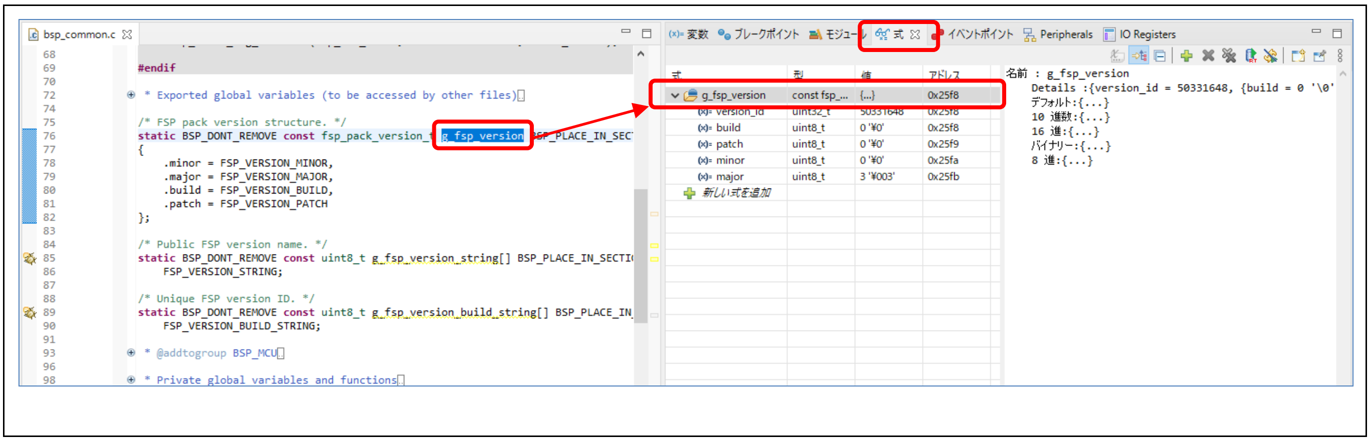


図 101　デバッグ ー [式] ビュー

5.3.4　レジスタービュー

[レジスター] ビューは、RA ファミリ・マイクロコントローラの汎用レジスタについての情報を表示します。

(1) のアイコンが付いた [レジスター]ビューを選択するか、表示されていなければ [ウィンドウ] メニューの [ビューの表示] → [レジスター]を選択して表示させます。
(2) レジスタ名を選択すると詳細欄には各基数のフォーマットで値が表示されます。

前回ブレーク時以降に変化のあった値は強調表示（背景色が黄色）になっています。

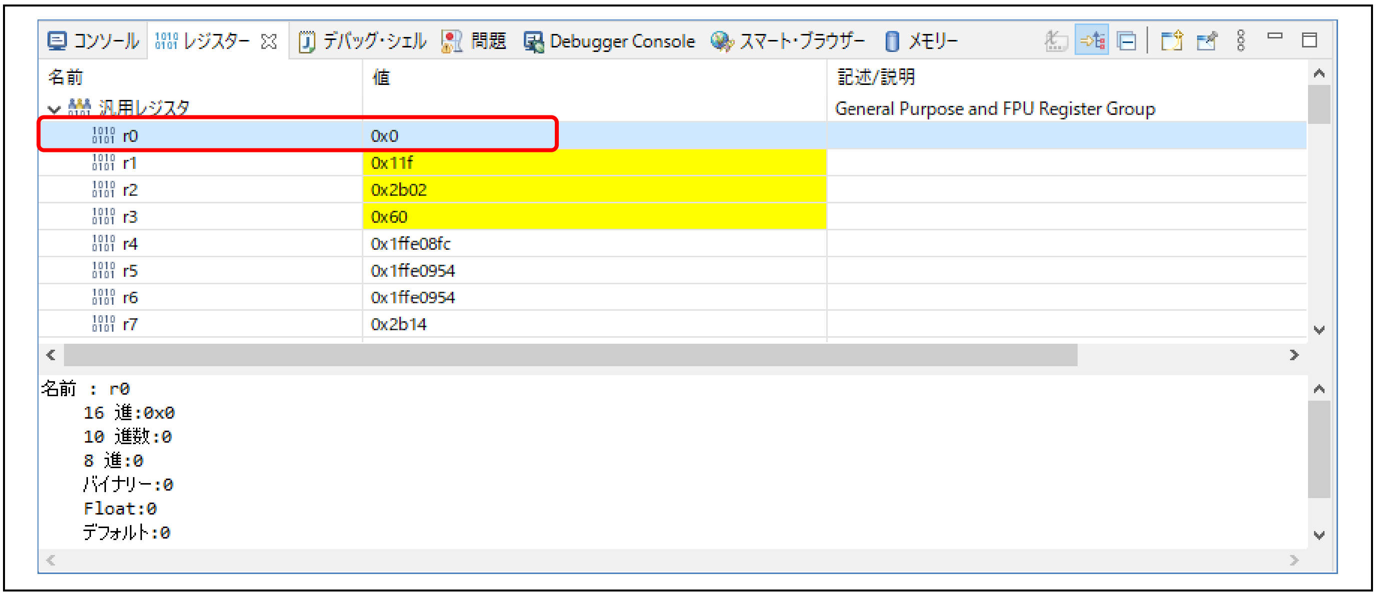


図 102　デバッグ ー [レジスター] ビュー

5.3.5　メモリービュー

[メモリー] ビューでは、ユーザは “メモリーモニター”（表示開始アドレスと表示形式の組み合わせ）を指定してメモリを表示し編集することができます。各モニターは “ベースアドレス” と呼ばれる格納位置によって特定される記憶場所を表します。各メモリーモニターの中のメモリデータは異なる “メモリレンダリング” で表示することができます。メモリレンダリングはあらかじめ設定したデータフォーマット（例えば、16 進数、符号付き整数、符号なし整数、ASCII イメージなど）です。

変数（例：“g_fsp_version_build_string”）をメモリービューで確認するには、

(1) メモリー ビュー（ のアイコン）が表示されていればそれを選択、表示されていなければ [ウィンドウ] メニュー → [ビューの表示] → [メモリー] でビューを開いてください。
(2) ビュー内の アイコンをクリックして [モニター・メモリー] のダイアログを開き、メモリの表示開始アドレスを指定してください。下記の例では変数のアドレス “&g_fsp_version_build_string” を指定しています。

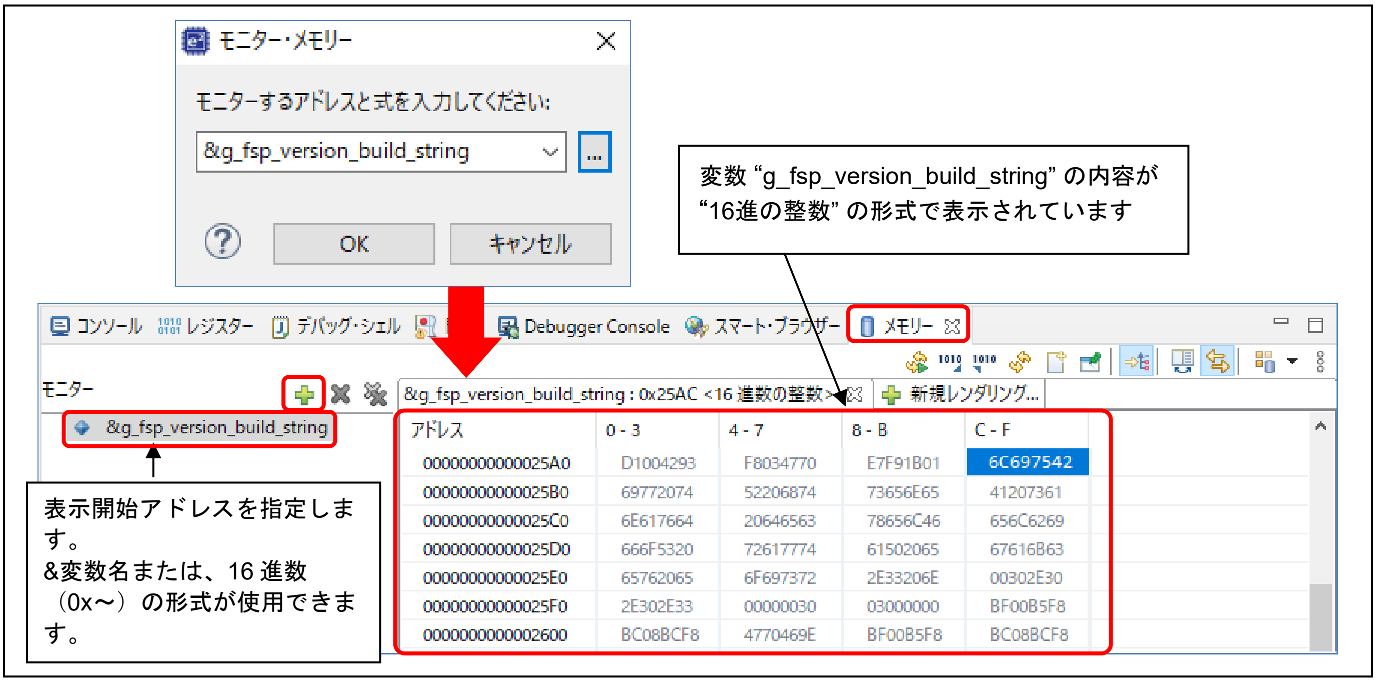


図 103　デバッグ ー [メモリー] ビュー

指定したメモリーモニターに対するレンダリング（表示形式）を追加するには、

(1) 新規レンダリング...のタブをクリックし、レンダリングを選択してから [レンダリングの追加] ボタンをクリックすると指定したレンダリングで表示するタブが追加されます。

下記の例では変数 “g_fsp_version_build_string” を「16 進数の整数」と「ASCII」の二種類で表示を切り替えられるようにしています。

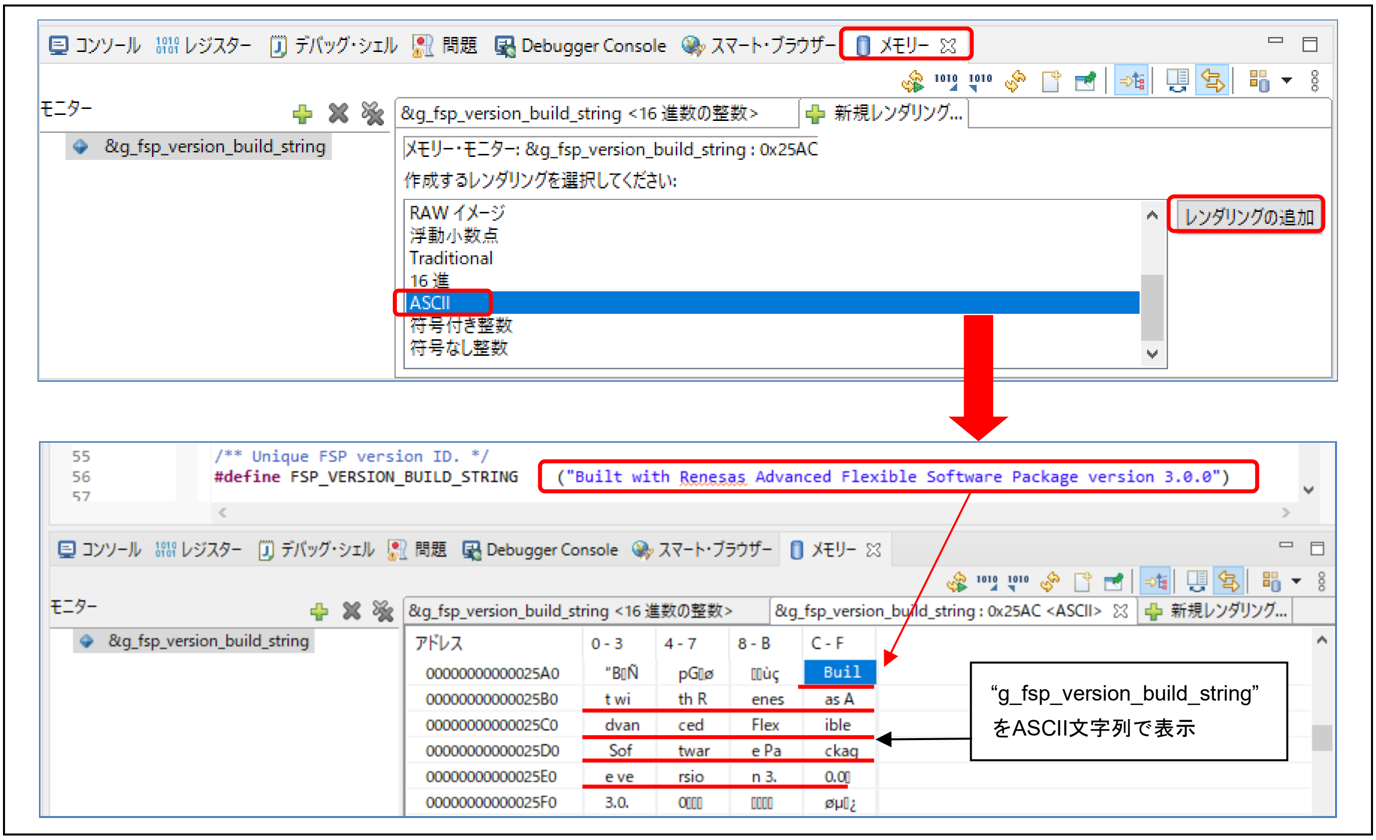


図 104　デバッグ ー [メモリ] ビューの新規レンダリング（表示方式）

5.3.6 メモリー使用量ビュー

[メモリー使用量] ビューを使用して、プロジェクトのマップファイル（*.map）やライブラリリストファイル（*.lbp）の情報を取得します。表示するのは対象プロジェクトの合計のメモリサイズ、ROM/RAM の各使用率、セクション/オブジェクト/シンボル/モジュール単位の情報、ベクタテーブルやクロスリファレンスです。

e2 studio のバージョン 7.3 以上では、ROM/RAM の使用量をグラフ表示できます。

[メモリー使用量] ビューを使うには、[ウィンドウ] メニュー → [ビューの表示] → [その他...] の [C/C++] カテゴリから [メモリー使用量] を選択し、[開く] ボタンでビューを開いてください。

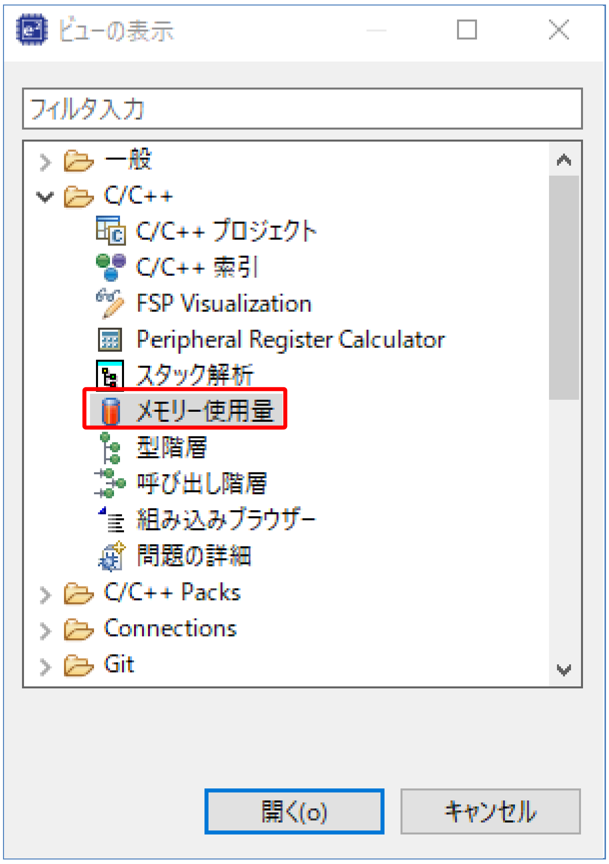


図 105 [メモリー使用量] ビューの表示

[メモリー使用量] ビューは次の 3 つのエリアで構成されています。
(1) サイズ、(2) メモリ領域使用量（アドレス空間使用量は未サポート）、(3) 詳細

注：選択したプロジェクトにリンカスクリプトファイルがない場合、またはリンカスクリプトファイルにメモリ領域が定義されていない場合は、メモリ領域使用量に「リンカ・スクリプト・ファイルは正しくありません」というメッセージが表示されます。

図 106　[メモリー使用量] ビューのエリア

[メモリー使用量] ビューでは以下の操作が可能です。

：メモリー使用量を表示するためのマップファイル（*.map）やライブラリリストファイル（*.lbp）を選択します。

：[メモリー使用量] ビューの情報をすべて更新します。

：詳細エリアのすべてのタブのデータをエクスポートします。

：マップファイルやライブラリリストファイルをエディタで開きます（RA ライブラリプロジェクトにはライブラリリストファイルはありません）。

：選択したプロジェクトのマップファイルを出力するためのページを開きます。

:：選択したプロジェクトの [Sections] ページを開きます。

5.3.7 逆アセンブル ビュー

[逆アセンブル] ビューは、ロードしたプログラムのソースコードとアセンブラ命令を混在して表示します。現在実行中の行は画面上で矢印のマーカで強調表示されます。[逆アセンブル] ビューでは、アセンブラ命令へのブレークポイントの設定、ブレークポイントの有効化／無効化、逆アセンブル命令のステップ実行、プログラムの特定の命令へのジャンプが可能です。

逆アセンブル ビューを使うには、

(1) デバッガを起動し、エディタ左端のアドレスが表示されている場所を右クリックし [逆アセンブリーへジャンプ] を選ぶか、[ウィンドウ] メニュー → [ビューの表示] → [逆アセンブル] を選択します。すでに [逆アセンブル] ビュー（ アイコンのついたタブ）が表示されていればそれをクリックします。

(2) （アクティブなデバッグ・コンテキストにリンク）ボタンが有効になっていれば逆アセンブルビューは PC カウンタのアドレスに対応した行に自動的にスクロールします。

(3) [逆アセンブル] ビューの左端、アドレスカラムを右クリックし [オペコードを表示]、[関数オフセットを表示] でオペコードと関数先頭からのオフセットの表示を切り替えられます。

図 107 デバッグ ― [逆アセンブル] ビュー

5.3.8 変数ビュー

[変数] ビューは、現在実行中のスコープ内で表示可能な全てのローカル変数を表示します。

ローカル変数を見るには、

(1) [変数] ビュー（(x)= のアイコン）を選択するか、[ウィンドウ] メニュー → [ビューの表示] → [変数] でビューを開きます。

(2) ステップ実行で関数内に入ると、変数ビューにローカル変数とその値が表示されます。（下記の例では hal_entry () 内の leds）

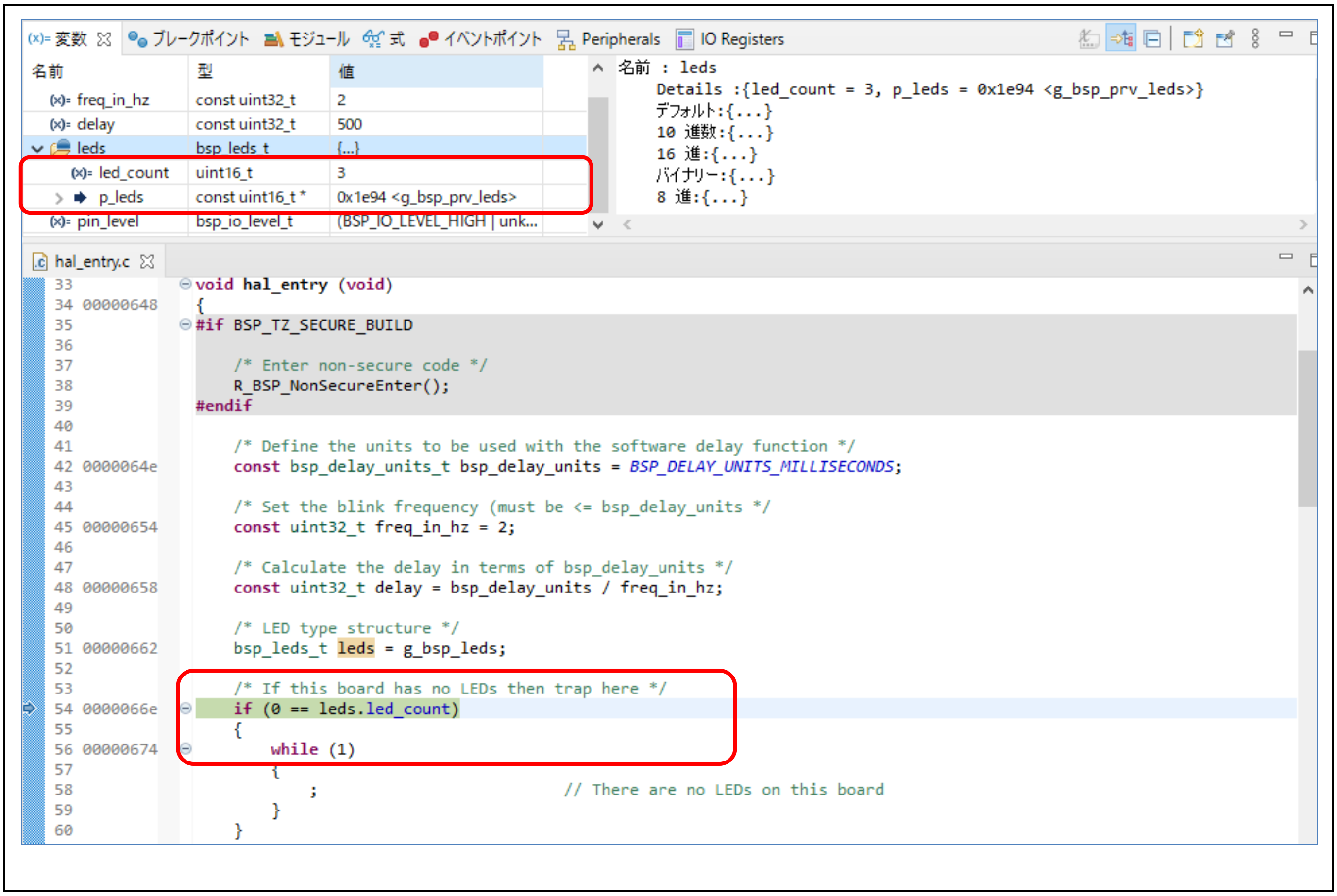


図 108　デバッグ ― [変数] ビュー

5.3.9 IO レジスタ（IO Registers）ビュー

[IO Registers] ビューは、ターゲット専用の IO ファイルで定義された全レジスタを表示します。ユーザは、[選択されたレジスター] に必要な IO レジスタを選択して追加することによって、[IO レジスタ] ビューをカスタマイズすることができます。

選択した IO レジスタの値を見るには、

(1) [IO Registers] ビュー（のアイコン）を選択するか、[ウィンドウ] メニュー → [ビューの表示] → [IO Registers] を選択してビューを開いてください。
(2) [すべてのレジスター] タブで、対象のモジュール（例：CAC）を探し、モジュールの IO レジスタ一覧を展開してください。
(3) レジスタ（例：CAICR と CASTR）を [選択されたレジスター] ペインにドラッグ＆ドロップします。レジスタ名左側の ● は、選択されたレジスタであることを示します。
(4) [選択されたレジスター] タブを選んで、選択したレジスタが表示されることを確認してください。

[すべてのレジスター] タブでの表示には時間がかかるため、複数のレジスタを見るには [選択されたレジスター] を使うことをお勧めします。

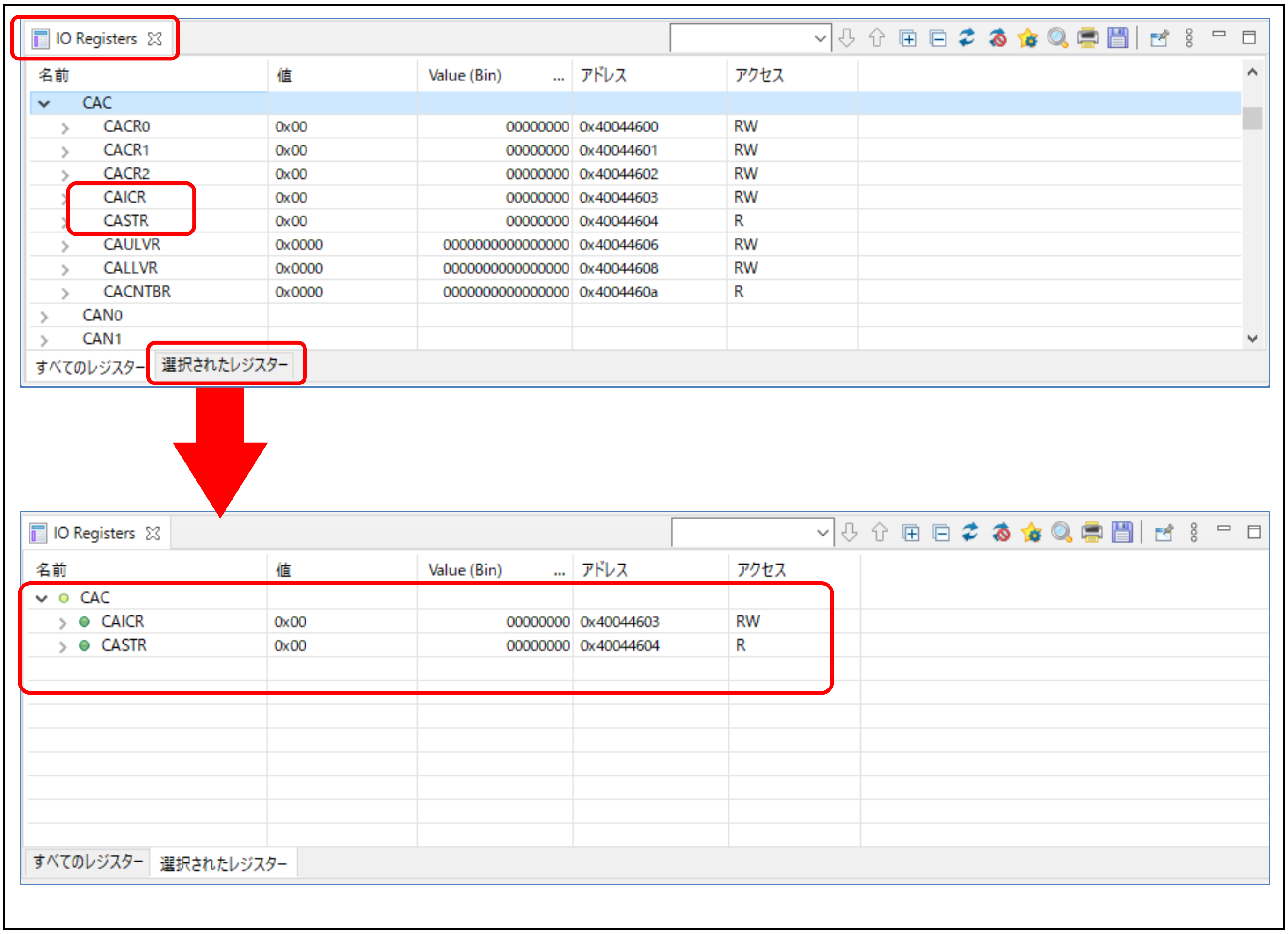


図 109 デバッグ ー [IO Registers] ビュー

5.3.10 イベントポイントビュー

イベントは、プログラム実行中にブレークあるいはトレース機能を実行するために設定された条件の組み合わせです。ユーザは [イベントポイント] ビューで、異なる種類の定義されたイベント、たとえば、トレース開始、トレース終了、イベントブレークなどを設定、表示することができます。

RA プロジェクトでは、データアクセスによるイベントブレークをサポートしています。エミュレータは指定された条件での指定アドレスあるいは指定アドレス範囲へのアクセスを検出します。これにより、アドレスとデータを組み合わせた条件を設定することができます。

イベントの組み合わせ（OR、AND（およびその組み合わせ）、シーケンシャル）は 2 つ以上のイベントに使用できます。

表 1　イベントの組み合わせ条件

組み合わせ条件	説明
OR	指定したイベントのうち 1 つが一致すると条件成立。
AND（組み合わせ）	指定したイベントが、順序は問わず、すべて一致すると条件成立。
シーケンシャル	指定したイベントが、指定した順序で一致すると条件成立。

アドレスまたはデータが一致する条件（例：g_bsp_leds へのアクセス時）で、グローバル変数にイベントブレークを設定するには、

(1) [Renesas Views] → [デバッグ] → [イベントポイント] の順に選択して [イベントポイント] ビューを開きます。

(2) タイプカラムの “イベント・ブレーク” をダブルクリックし、[編集 イベント・ブレーク] のダイアログを開きます。

(3) [追加...] ボタンをクリックして次に進みます。

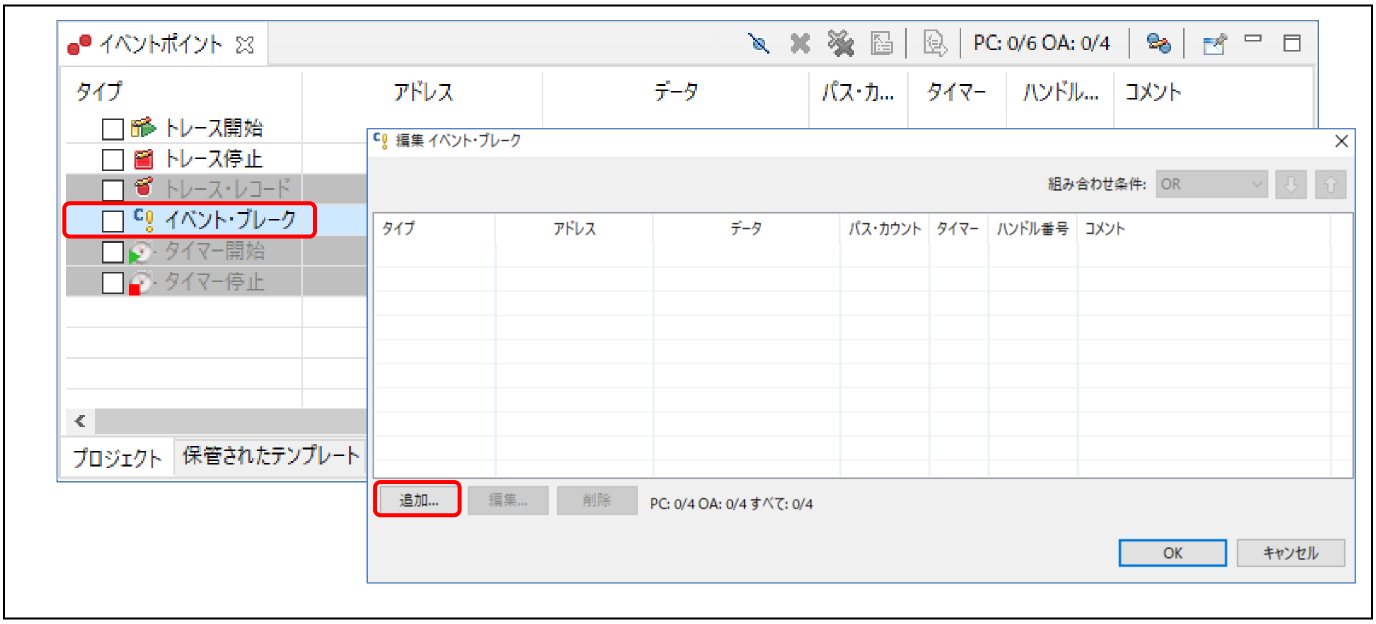


図 110　デバッグ ー [イベントポイント] ビュー(1/2)

(4) [イベントポイント・タイプ] に “Data Access” を選択します。

(5) [アドレス設定] タブに進み、… アイコンをクリックしてシンボル “g_bsp_leds” を検索します。（グローバル変数のアドレスは “&g_bsp_leds” で示されます。）

(6) 次に、[データ・アクセス設定] タブに切り替え、[リード/ライト] に “Read” を選択します。

(7) [OK] ボタンで次に進みます。

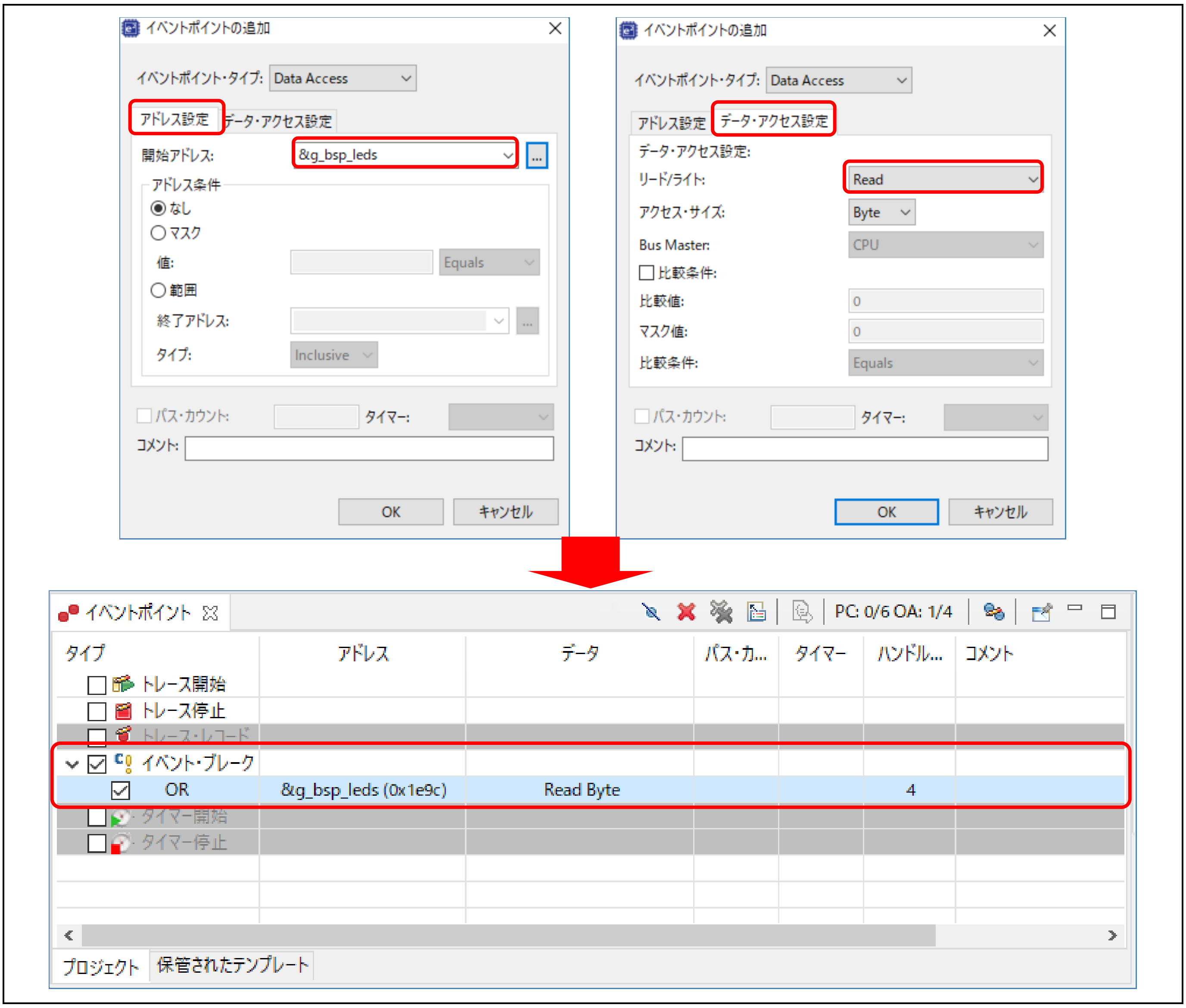
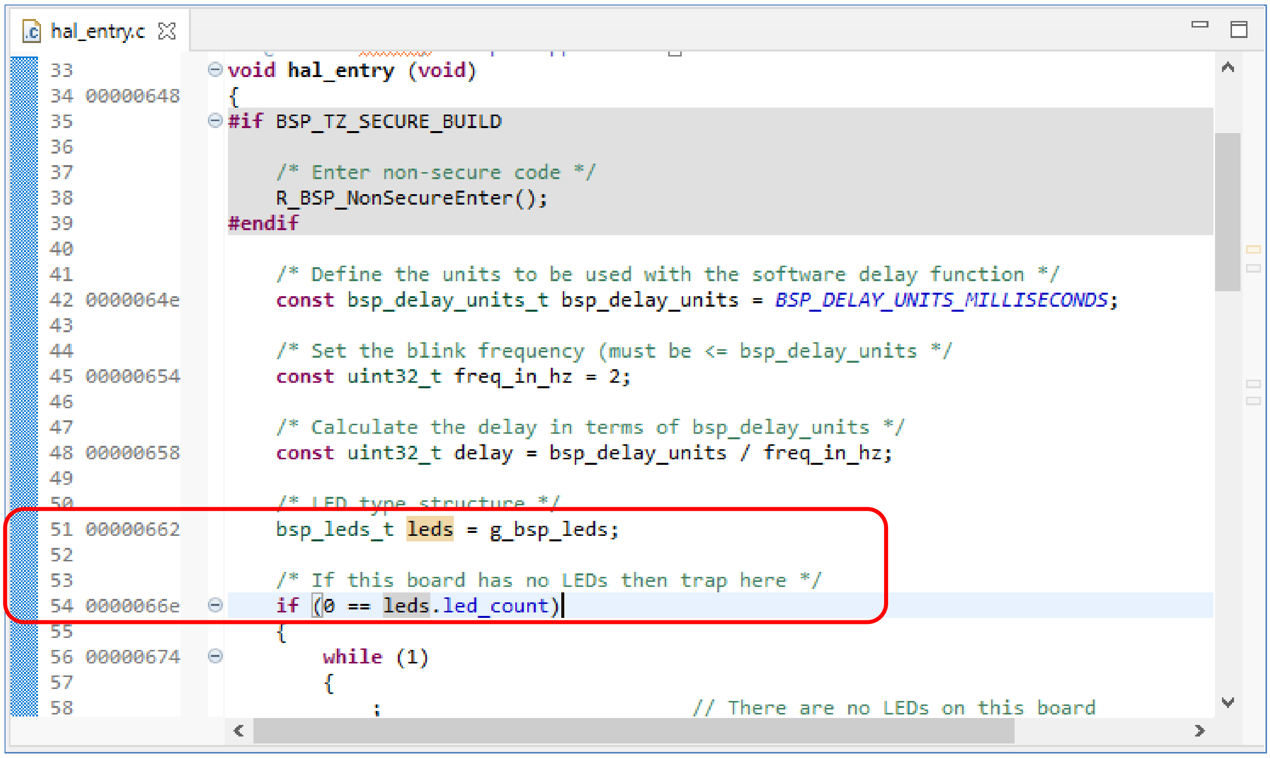


図 111　デバッグ ー [イベントポイント] ビュー(2/2)

(8) プログラムを最初から実行するためにリセットします。

(9) 下図は、条件が成立してイベントブレークが掛かった様子を示します。“g_bsp_leds” のアクセス（リード）でブレークしています。



112　デバッグ ― イベントブレークの実行

5.3.11　トレースビュー

トレースとは、ユーザプログラム実行中、1 サイクルごとのバス情報をトレースメモリから取得することを意味します。取得されたトレース情報は [トレース] ビューに表示されます。それによりユーザはプログラムの実行を追跡し、問題が発生した箇所を探すことができます。

トレースバッファは有限なため、バッファがいっぱいになると、古いトレースデータを新しいデータで上書きします。

プログラム実行が停止するまでのトレース取得を設定するには、

(1) [Renesas Views] → [デバッグ] → [Trace] の順に選択して [Trace] ビューを開きます。

(2) [トレース] ビューの  アイコンをクリックしてトレースの収集を有効にしてください。

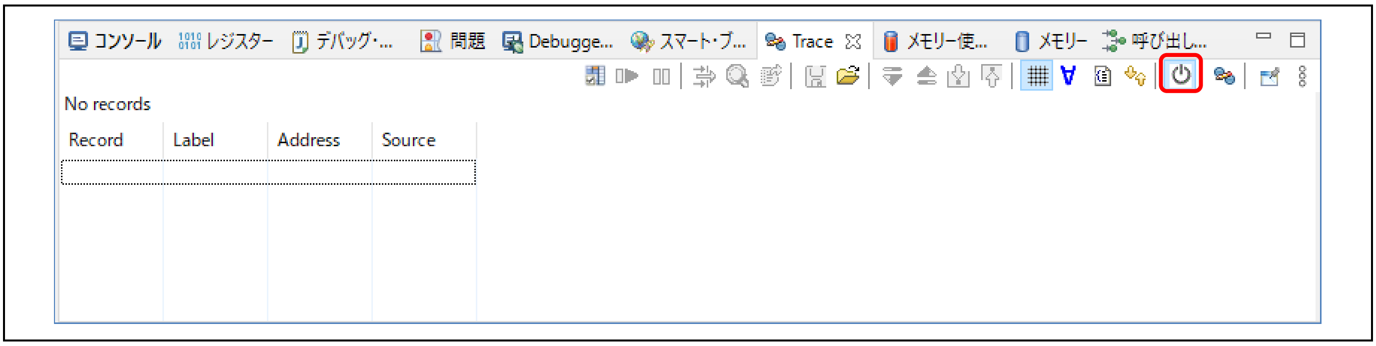


図 113　デバッグ － トレース有効

(3) プログラムを実行し、ブレークポイントで停止させるか、あるいは [デバッグ] ツールバーの [中断] ボタンで停止させます。それまでにトレースメモリに格納された情報がトレース結果として表示されます。

(4) [トレース] ビュー上のボタンで表示モードを選択します。下図は、main() 関数実行前までのトレース結果を示しています。

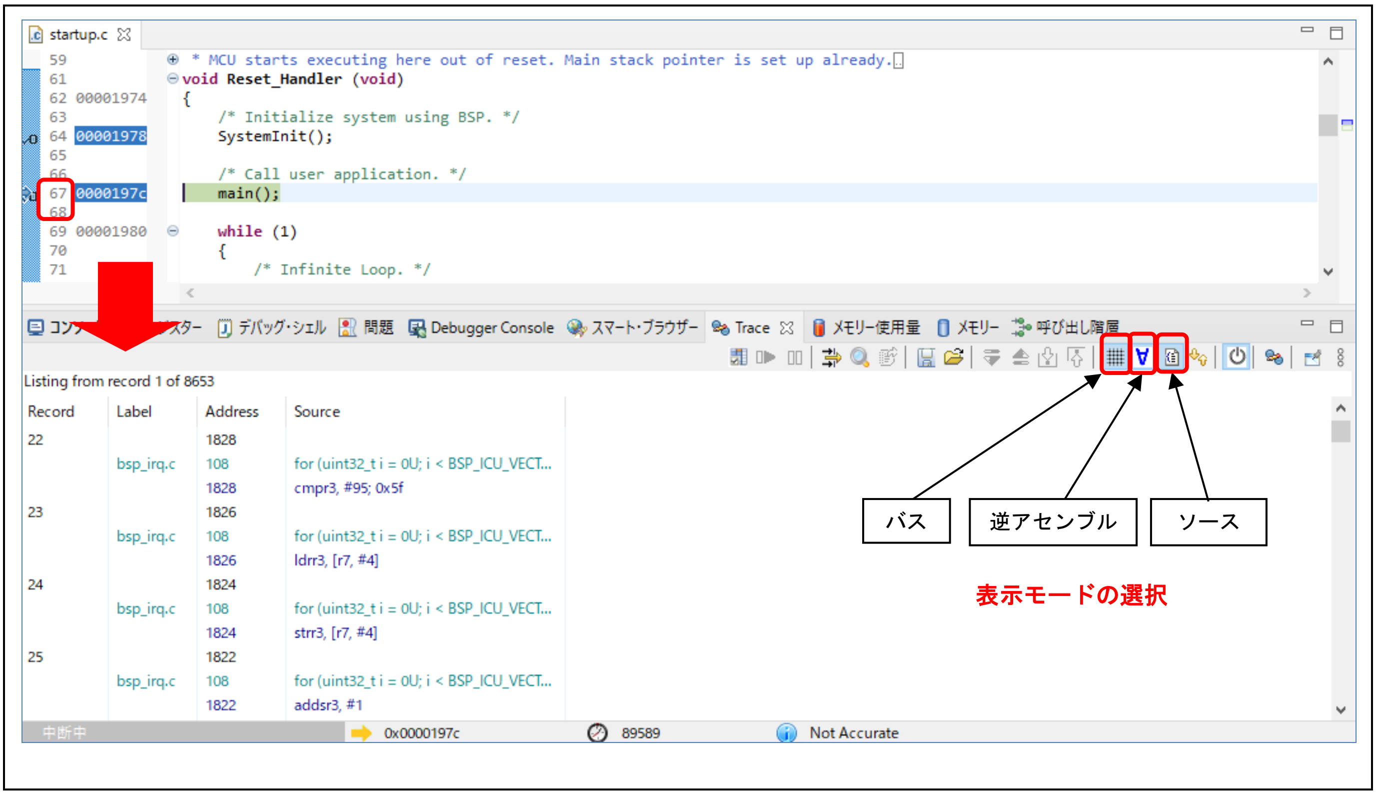


図 114　デバッグ － [トレース] ビューの表示モード選択

(5) トレース結果は、デフォルト設定では古い順から表示されます。表示順は ボタンで変更できます。

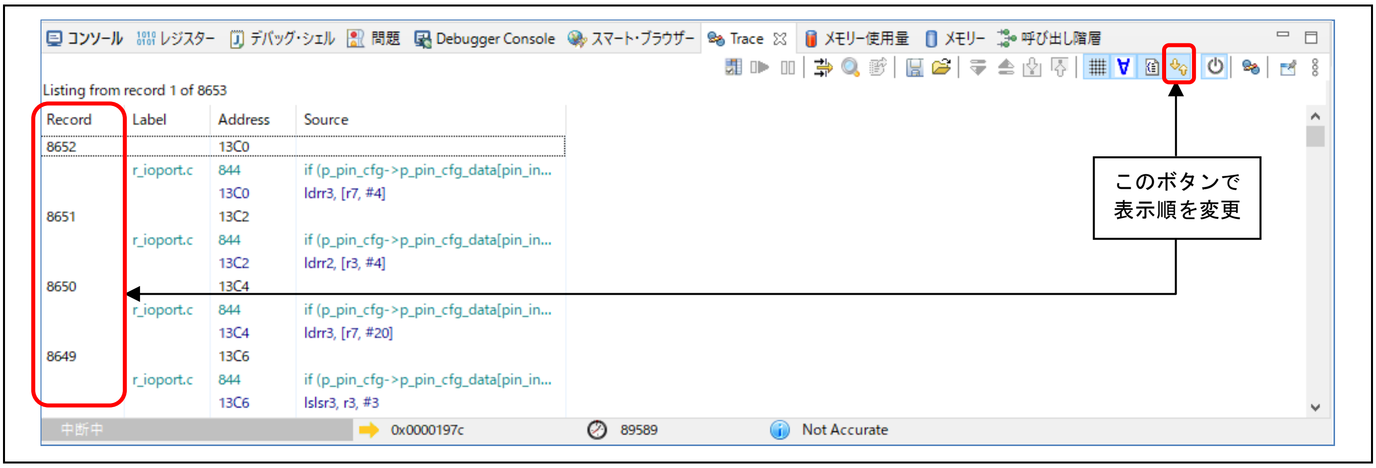


図 115　表示順の変更

(6) トレース結果は ボタンでフィルタすることができます。フィルタ条件として、[Record] と [Address] とが選択できます。

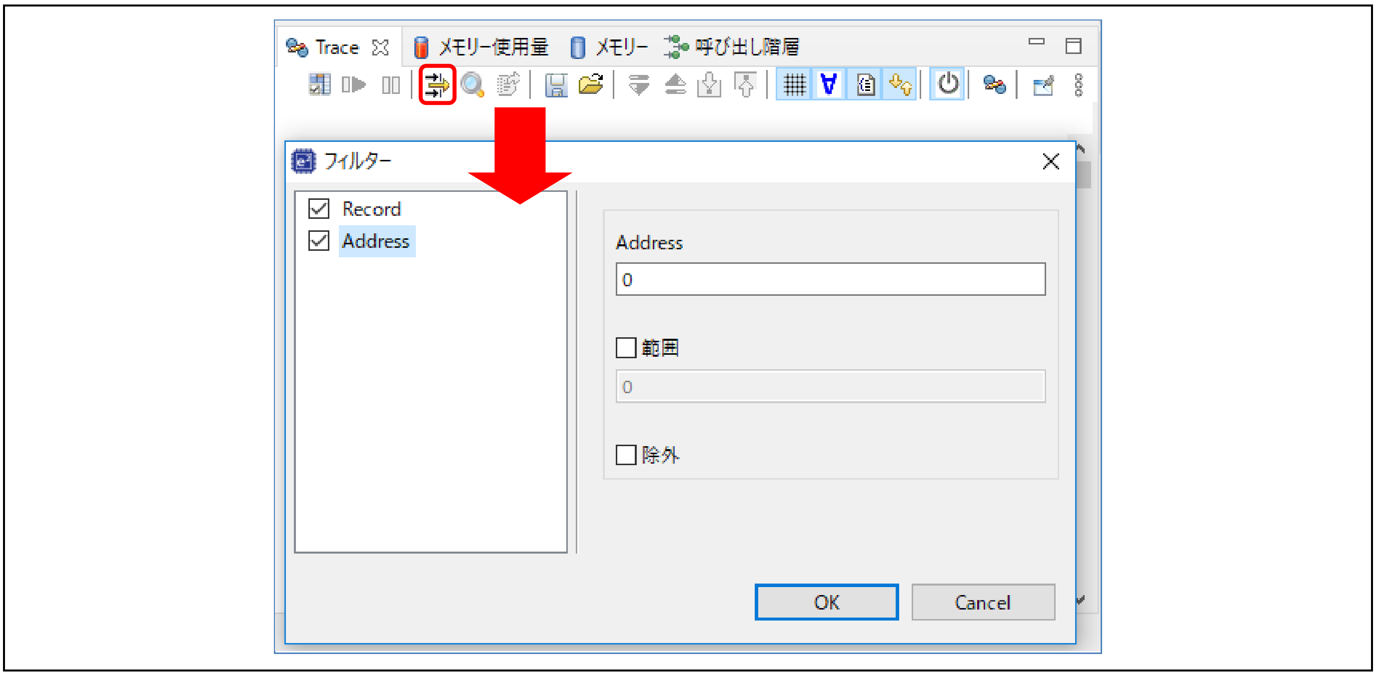


図 116　デバッグ － トレース結果のフィルタ

(7) トレース結果は .csv ファイルに保管することができます（バス、アセンブリ、ソース情報を含む）。また、トレース結果を .csv ファイルから [トレース] ビューにロードすることも可能です。

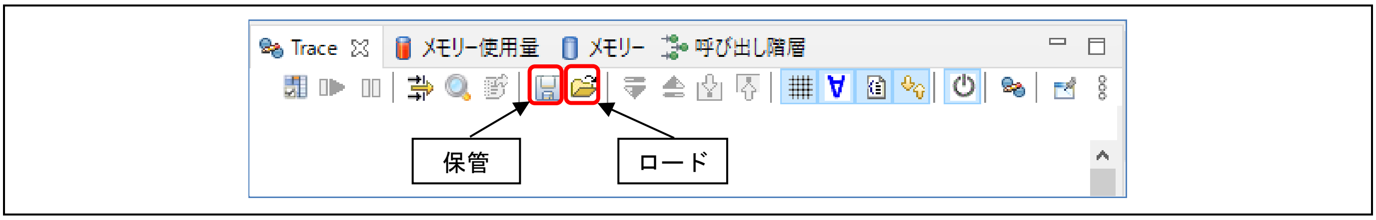


図 117　デバッグ － トレース結果の保管とロード

5.3.12　Fault Status ビュー

このビューは、ハードウェアフォルト発生時に、fault status レジスタのビット値とその他の主なレジスタの値を表示します。ハードウェアフォルトが発生すると、フォルト要因に関するレジスタのビット値をチェックして表示し、また r0、r1、r2、r3、r12、lr、pc、psr レジスタの値を表示します。下図にその画面を示します。この機能は e² studio v5.2 以上でサポートしています。

図 118　[Fault Status] ビュー ― ハードウェアフォルト無し

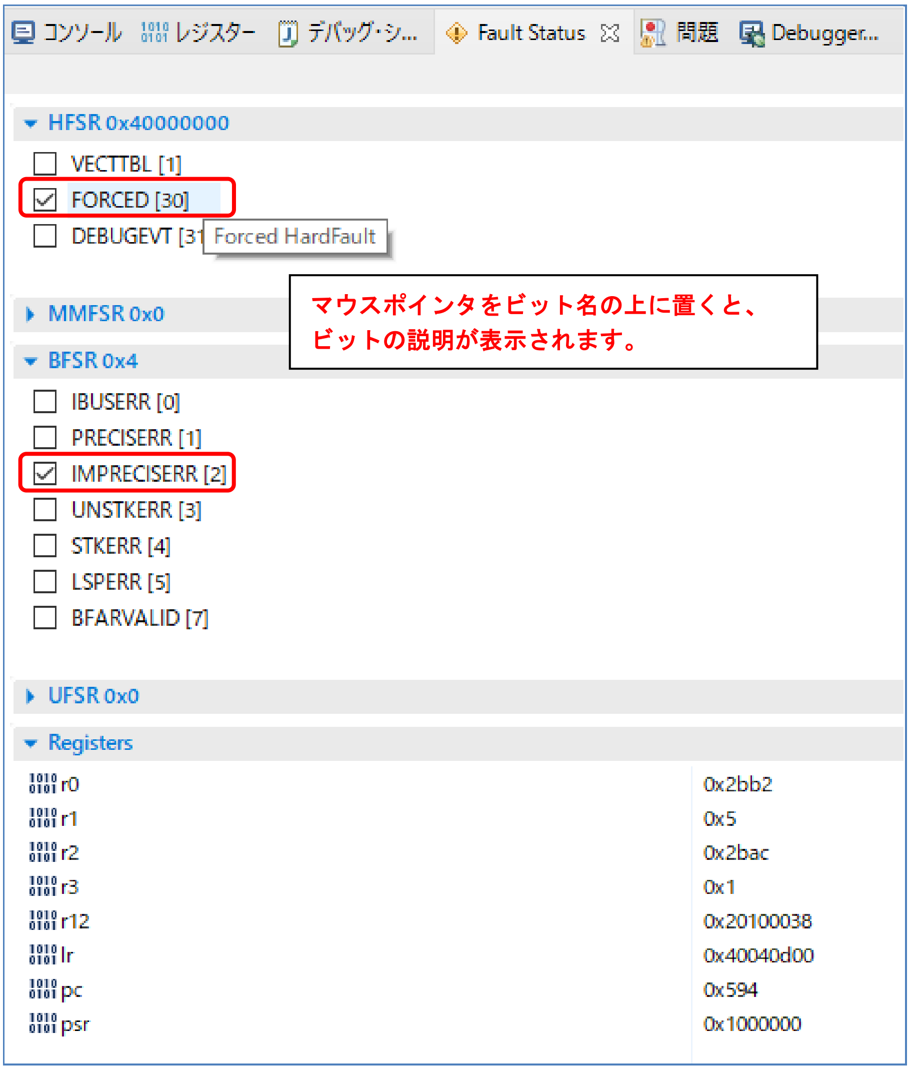


図 119　[Fault Status] ビュー ― ハードウェアフォルト発生

5.3.13 Run Break Timer

Run Break Timer 機能は、直前のプログラム実行のパフォーマンスをステータスバーに表示します。プログラム実行の停止後、現在のプログラムカウンタ（PC）値、直前の実行時間と CPU サイクル、精度または測定の方法を確認できます。

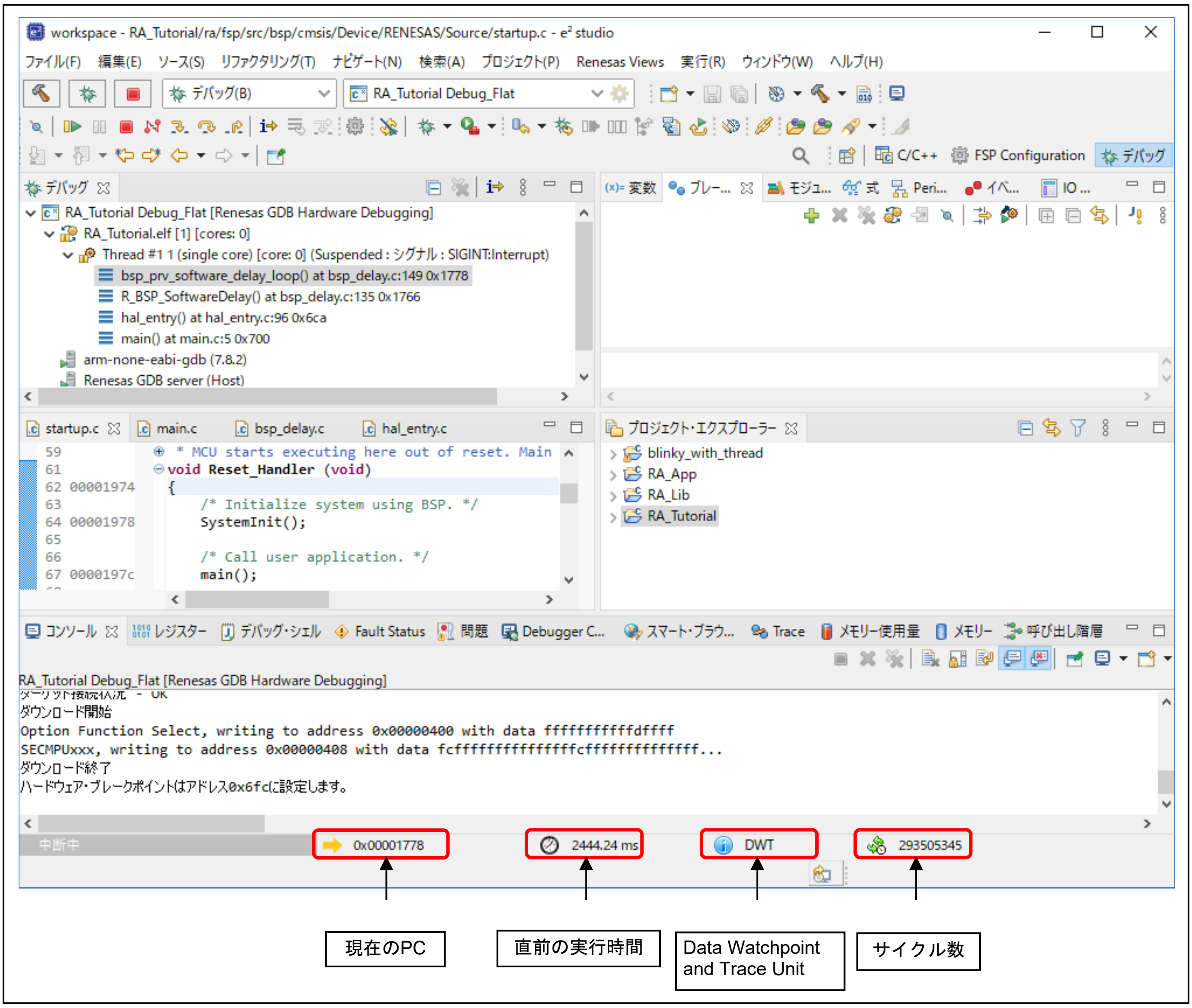


図 120 Run Break Timer による直前のプログラム実行のパフォーマンス表示

RA デバイスごとの Run Break Timer 機能のサポート状況を以下に示します。

表 2 Run Break Timer のサポート状況

デバイス	デバッガ	サポート状況
RA2 シリーズ（Cortex-M23）	J-Link	System Time
RA4、RA6 シリーズ	J-Link	Data Watchpoint and Trace Unit (DWT) — System Time で計算したサイクル数とオーバフロー回数

Run Break Timer は e^2 studio v7.3.0 以上でサポートしています。機能のアップデート情報については、e^2 studio のリリースノート（以下のリンク）を参照してください。
https://www.renesas.com/jp/ja/software-tool/e-studio の「その他資料」

6. FreeRTOS アプリケーションの設定

この章では、FreeRTOS オブジェクトと汎用 PWM タイマ（GPT）モジュールを用いる RA プロジェクトを、プロジェクトテンプレート “FreeRTOS – Blinky – Static Allocation” を使用して作成する例を説明します。

6.1 FreeRTOS での汎用 PWM タイマ使用例

RA プロジェクトの “FreeRTOS – Blinky – Static Allocation” テンプレートでは、LED 状態を切り替える前にタスクの実行にディレイを入れることで LED の点滅を行ないます。

ここで説明する例では、ディレイを入れるかわりに、Blinky スレッドをセマフォ待ち状態にし、GPT のタイマ割り込みを使用し、１秒間隔でセマフォを解放、これによりスレッドの実行を再開します。

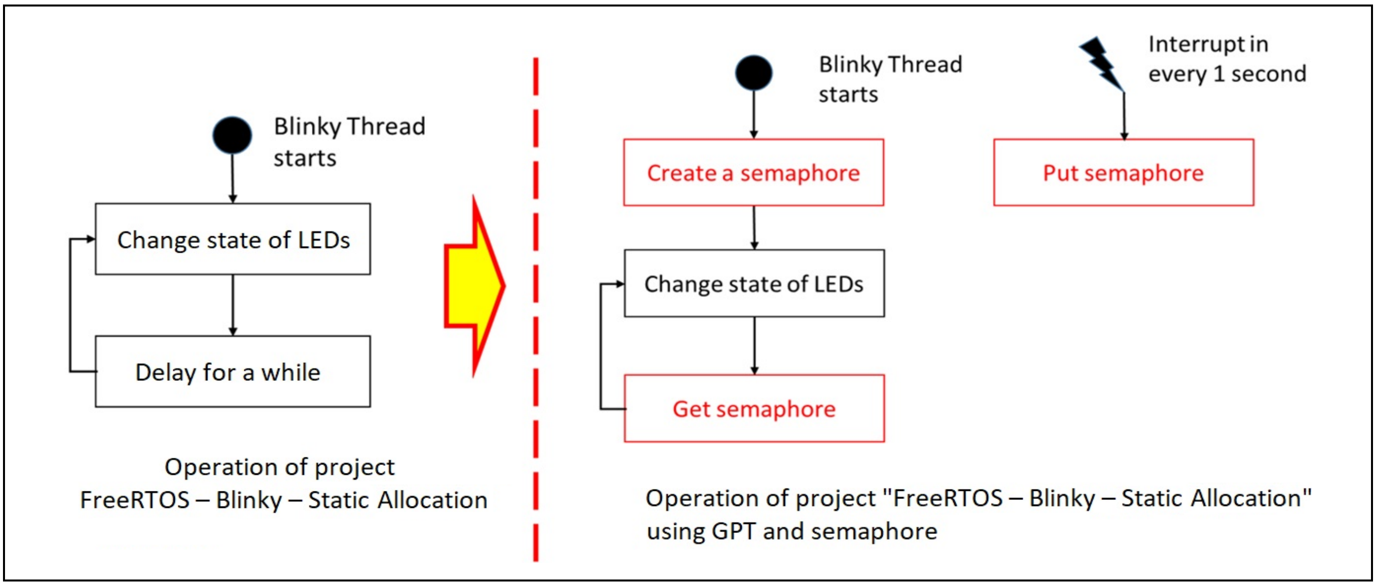


図121 FreeRTOSアプリケーションの設定 ー 概要

6.2 サンプルプロジェクトの作成

GPT とセマフォを使用する FreeRTOS サンプルプロジェクトを作成するには、以下の手順で RA プロジェクトを構成してください。

(1) 新規プロジェクト作成用ウィザードを立ち上げ、「3.1　TrustZone に対応していないデバイス用の新規 RA プロジェクトの作成」の手順に従って新規プロジェクトを作成します。ただし、[Build Artifact and RTOS Selection] 画面では “FreeRTOS” を選択し、[Project Template Selection] 画面では “FreeRTOS – Blinky – Static Allocation” を選択してください。

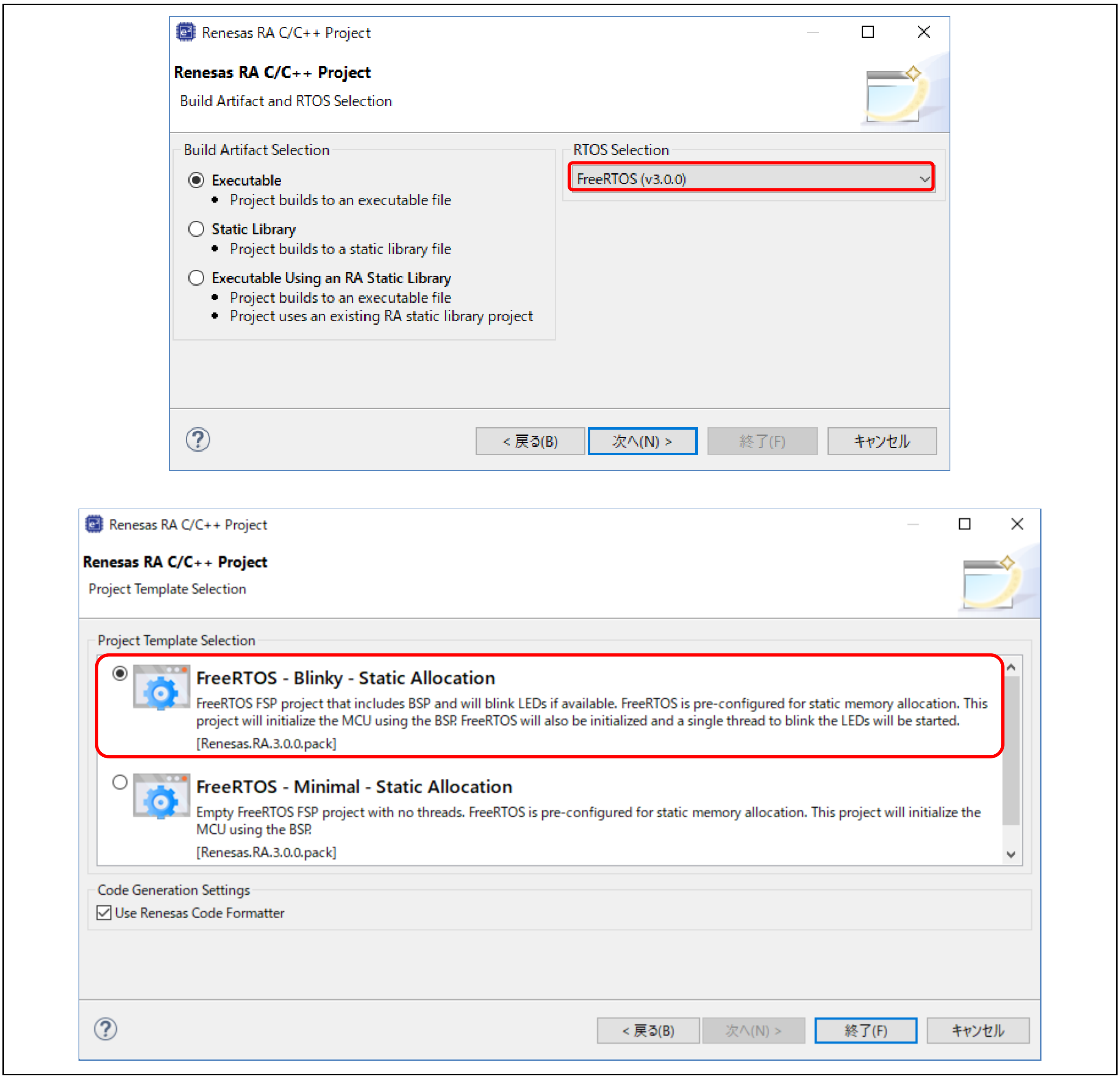


図 122　FreeRTOS アプリケーションの設定 ― 新規プロジェクトの作成

(2) [RA プロジェクトコンフィグレーションエディタ] ビューの [Stacks Configuration] ページを開きます。「3.5.5　スタック構成（Stacks Configuration）ページ」を参照してください。

(3) [Threads] ペインで “Blinky Thread” を選択し、[Stacks] ペインで [New Stack] → [Driver] → [Timers] → [Timer Driver on r_gpt] の順に選択して、Blinky スレッドに GPT モジュールを追加します。

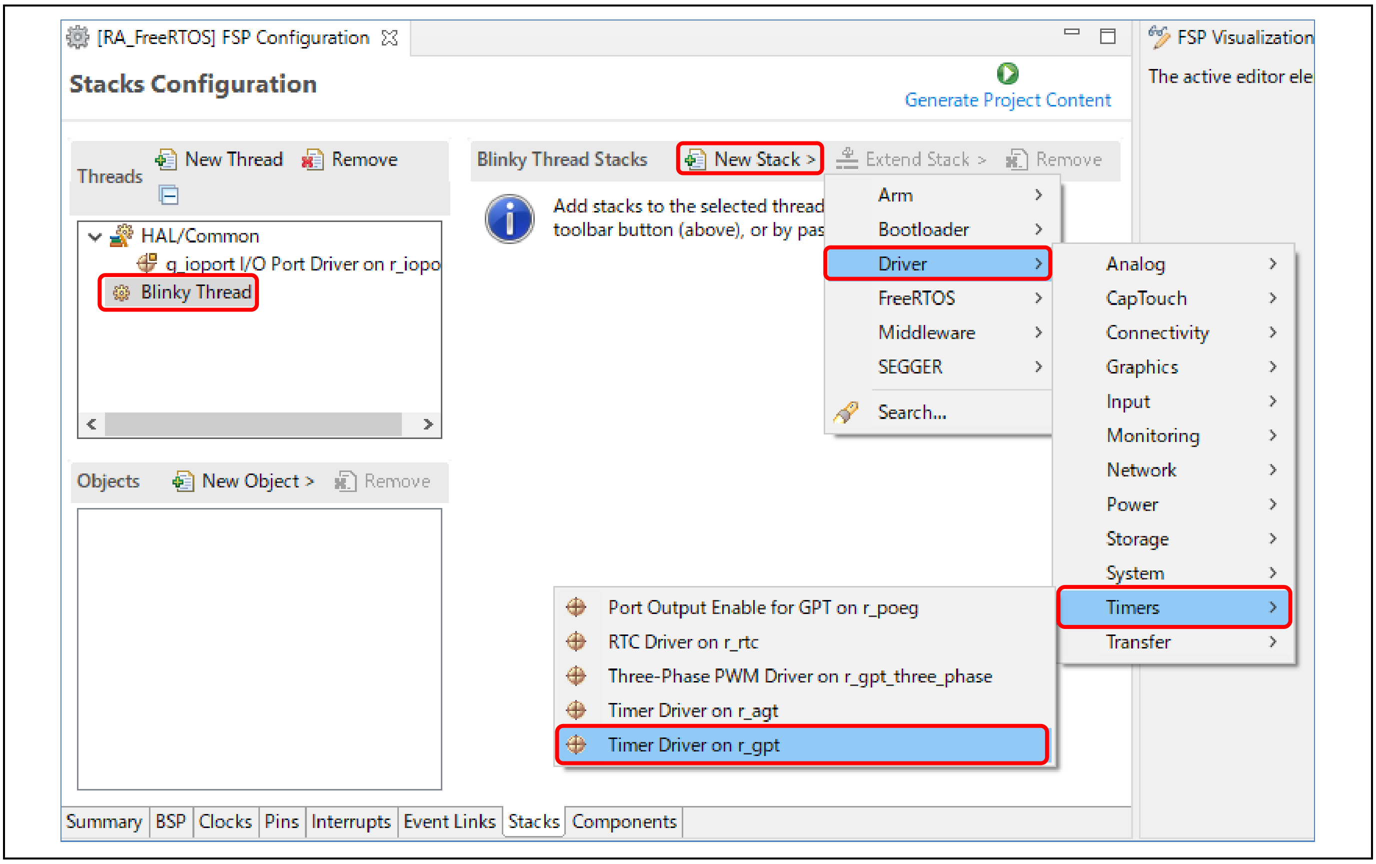


図 123　FreeRTOS アプリケーションの設定 ― GPT モジュールの追加

(4) GPT モジュールを次のように設定します。

- Name：g_timer
- Mode：Periodic
- Period：1
- Period Unit：Seconds
- Callback：gpt_callback
- Overflow/Crest Interrupt priority：Priority 2

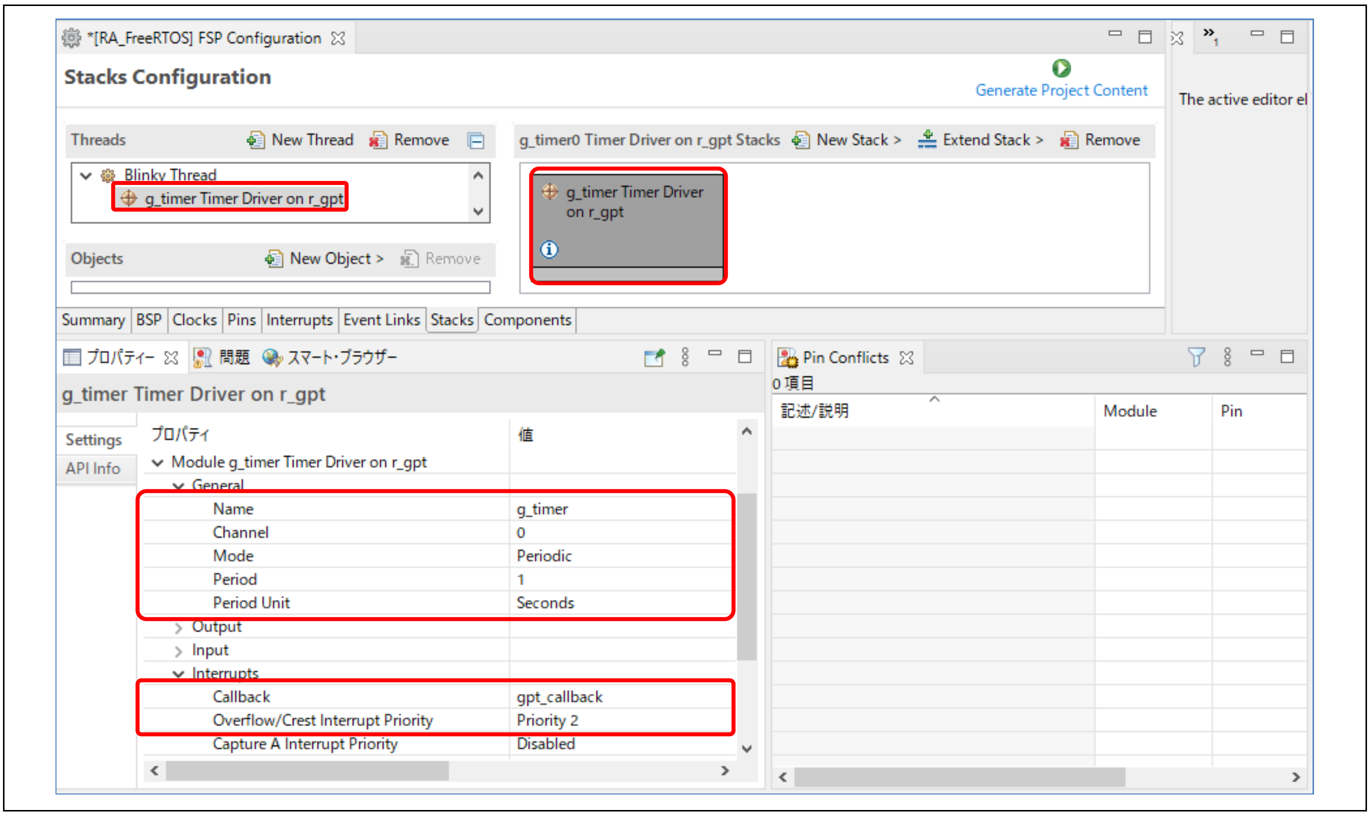


図 124　FreeRTOS アプリケーションの設定 ― GPT モジュールの設定

(5) [Threads] ペインで "Blinky Thread" を選択し、[Objects] ペインで [New Object] → [Binary Semaphore] の順に選択して、Blinky スレッドにセマフォオブジェクトを追加します。

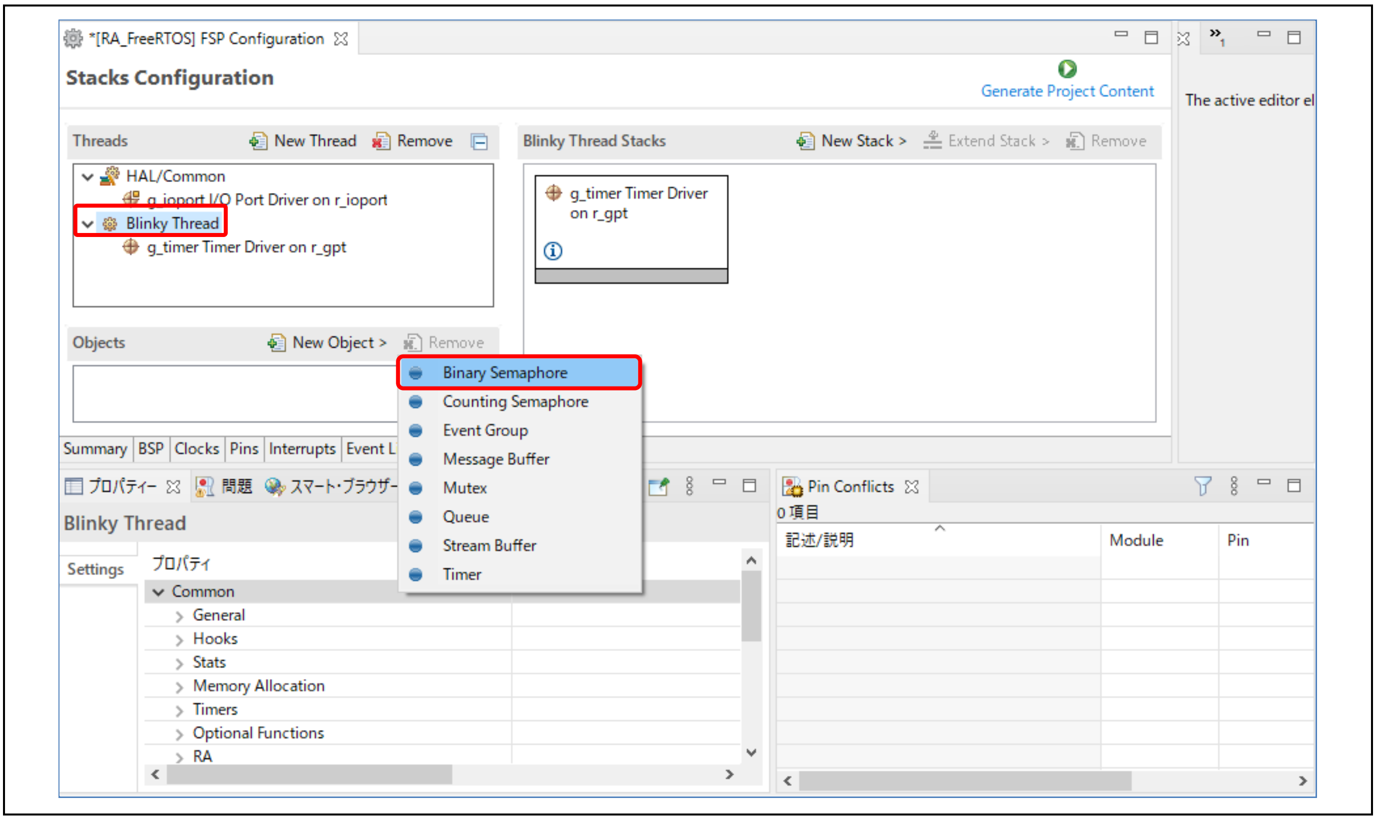


図 125　FreeRTOS アプリケーションの設定 ― セマフォオブジェクトの追加

(6) 作成したセマフォを次のように設定します。

- Name：Blinky Semaphore
- Symbol：g_blinky_semaphore

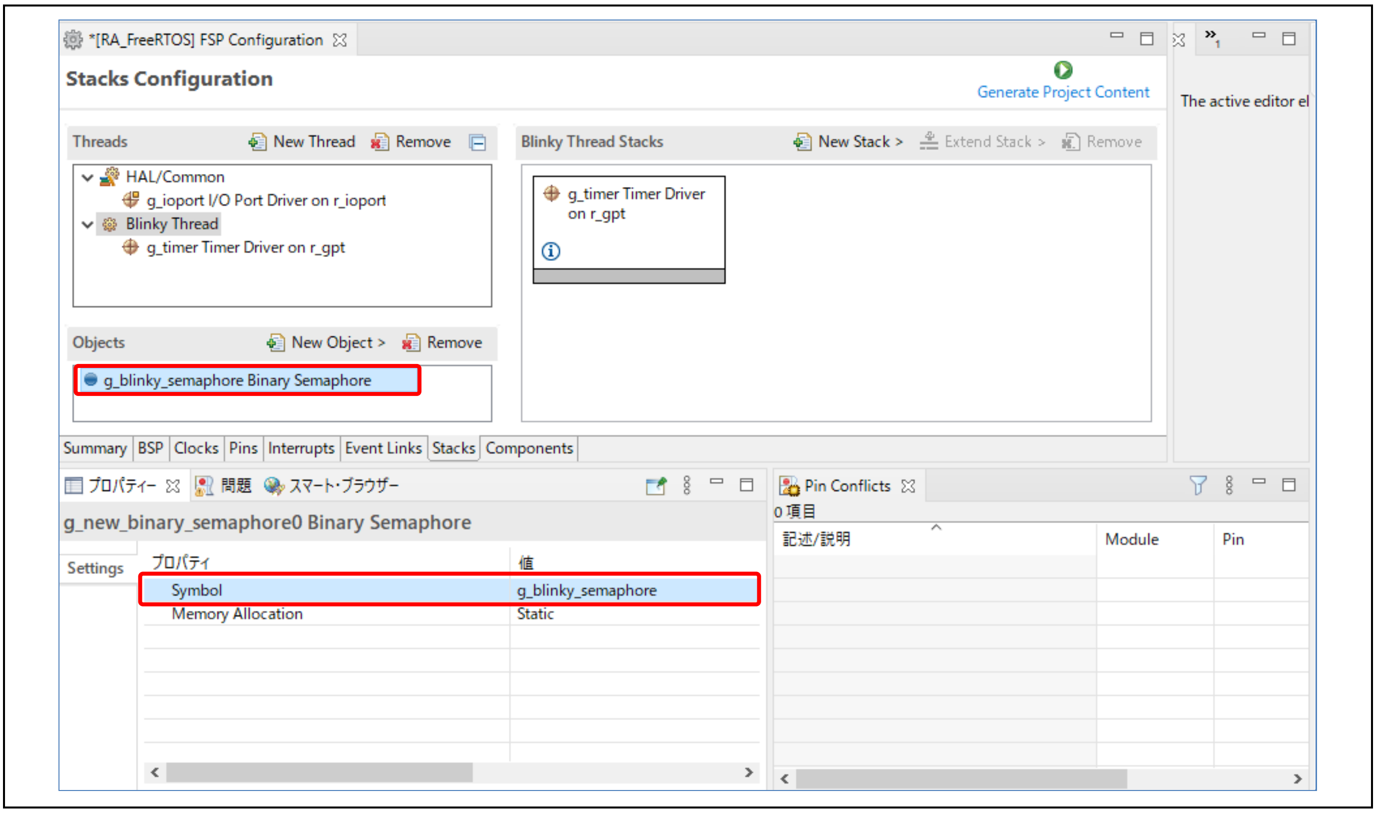


図 126 FreeRTOS アプリケーションの設定 ― セマフォオブジェクトの設定

(7) [Ctrl + S] キーで設定を保存し、Generate Project Content ボタンでソースコードを生成します。
(8) “src\blinky_thread_entry.c” ファイルを開き、次のように変更します。

- blinky_thread_entry() 関数の “while(1)” ループの前に、GPT モジュール初期化のコードを追加します。

```
g_timer.p_api->open(g_timer.p_ctrl, g_timer.p_cfg);
g_timer.p_api->start(g_timer.p_ctrl);
```

- blinky_thread_entry() 関数のタスクディレイ命令を削除し、セマフォを待つコードを追加します。

```
xSemaphoreTake(g_blinky_semaphore, portMAX_DELAY);
```

- Blinky スレッドにセマフォを通知する関数 gpt_callback() を追加します。

```
void gpt_callback(timer_callback_args_t *p_args) {
(void)p_args;
static signed portBASE_TYPE xHigherPriorityTaskWoken;
xSemaphoreGiveFromISR(g_blinky_semaphore, &xHigherPriorityTaskWoken);
}
```

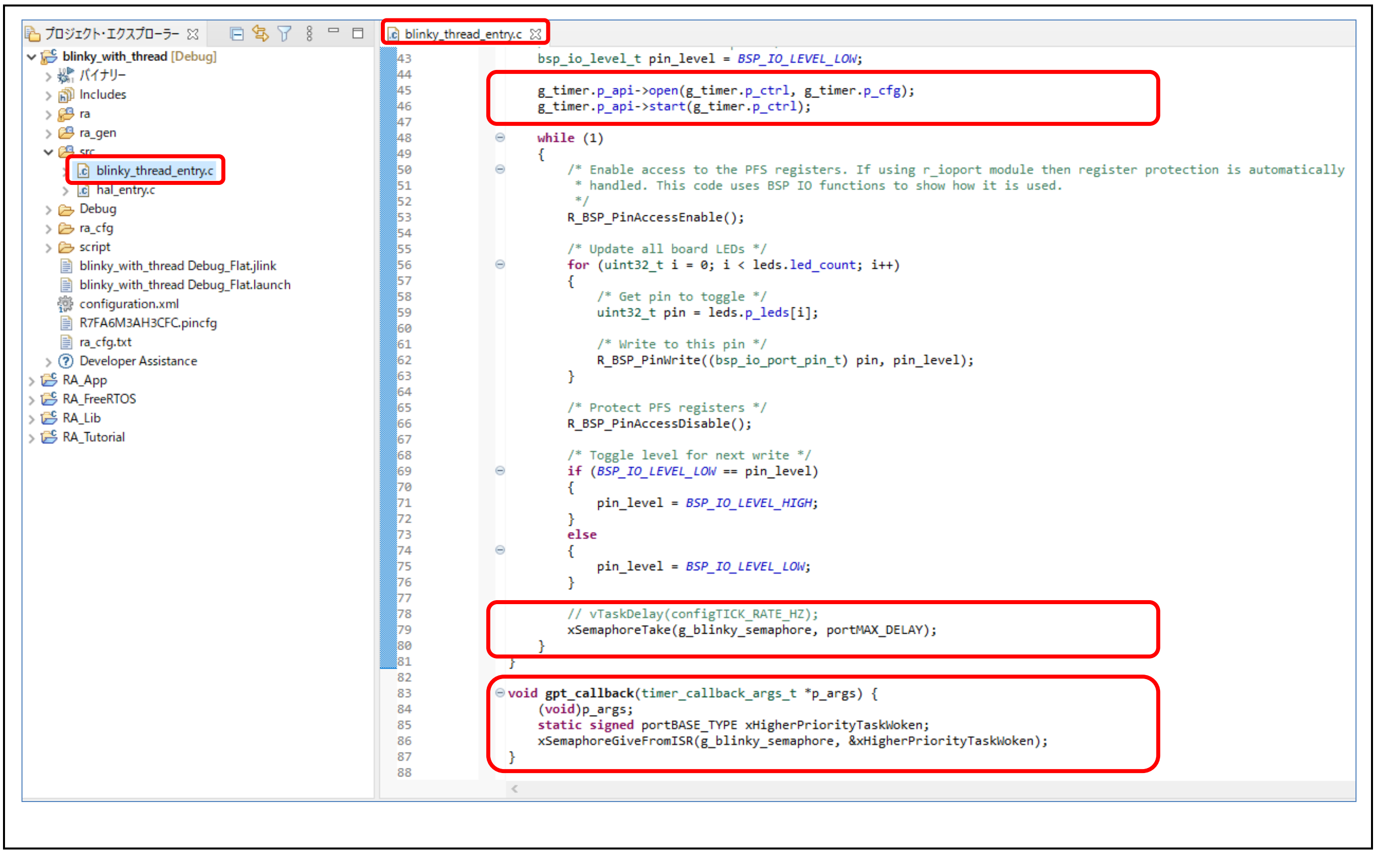


図 127　FreeRTOS アプリケーションの設定 ー ユーザソースコードの追加

(9) プロジェクトをビルドして、EK-RA6M3 ボードで実行します。 1 秒間隔で LED が点滅するのを確認してください。

7. Azure RTOSアプリケーションの設定

この章では、Azure RTOS オブジェクトと汎用 PWM タイマ（GPT）モジュールを用いる RA プロジェクトを、プロジェクトテンプレート “Azure RTOS ThreadX – Blinky” を使用して作成する例を説明します。

7.1 Azure RTOSでの汎用PWMタイマ使用例

RA プロジェクトの “Azure RTOS ThreadX – Blinky” テンプレートでは、LED 状態を切り替える前にタスクの実行にディレイを入れることで LED の点滅を行ないます。

ここで説明する例では、ディレイを入れるかわりに、Blinky スレッドをセマフォ待ち状態にし、GPT のタイマ割り込みで 1 秒間隔でセマフォを解放し、これによりスレッドの実行を再開します。

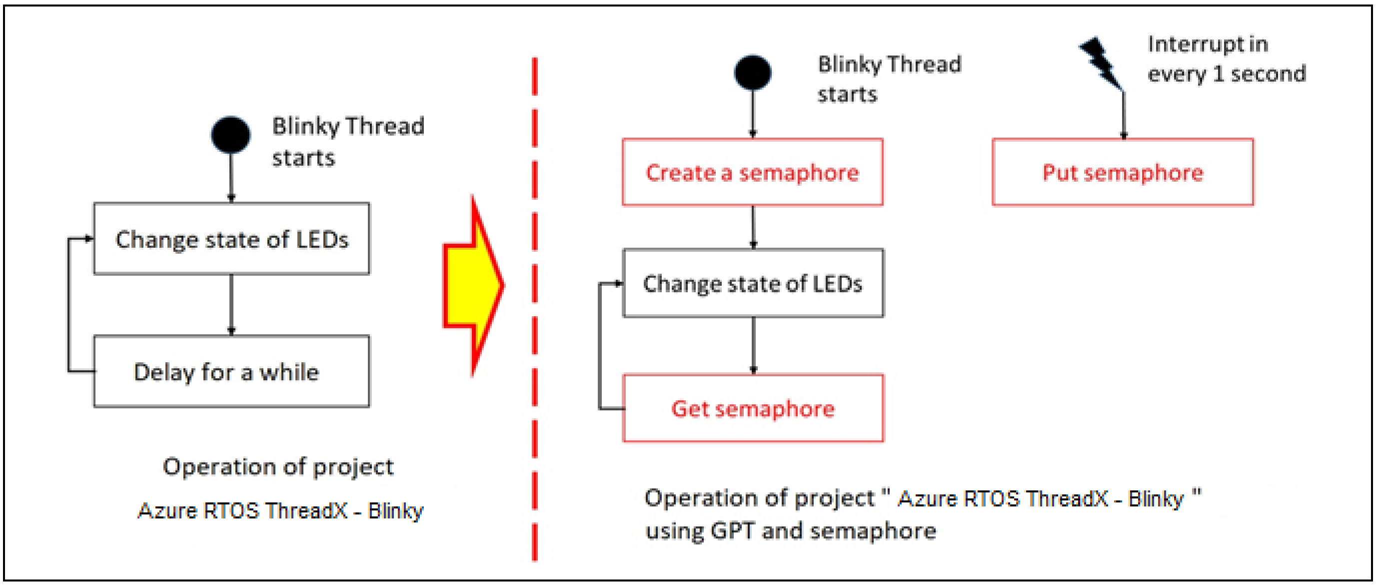


図128 Azure RTOSアプリケーションの設定 ー 概要

7.2 サンプルプロジェクトの作成

GPT とセマフォを使用する Azure RTOS サンプルプロジェクトを作成するには、以下の手順で RA プロジェクトを構成してください。

(1) 新規プロジェクト作成用ウィザードを立ち上げ、「3.1　TrustZone に対応していないデバイス用の新規 RA プロジェクトの作成」の手順に従って新規プロジェクトを作成します。ただし、[Build Artifact and RTOS Selection] 画面では “Azure RTOS ThreadX” を選択し、[Project Template Selection] 画面では “Azure RTOS ThreadX – Blinky” を選択してください。

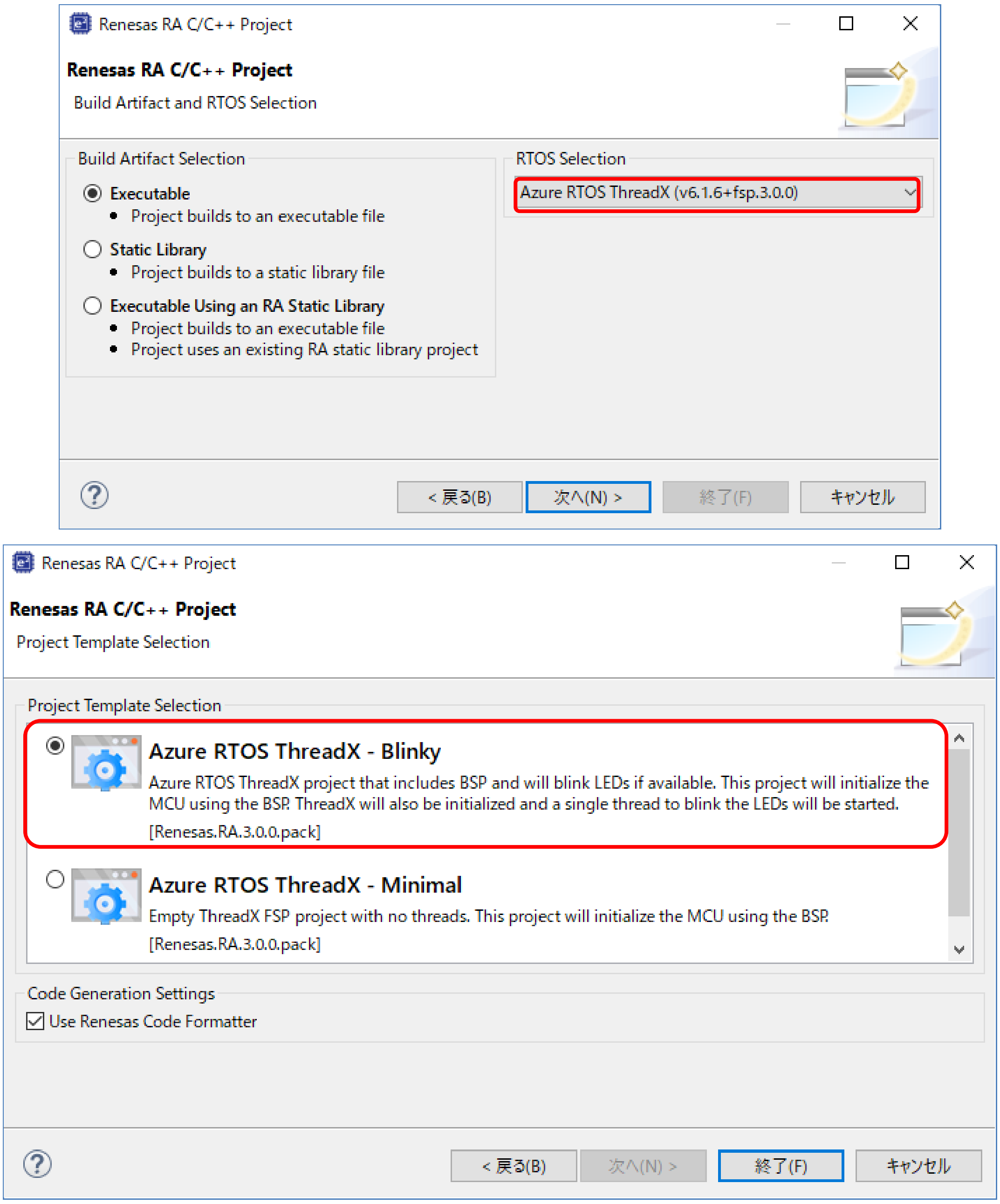


図 129　Azure RTOS アプリケーションの設定 ー 新規プロジェクトの作成

(2) [RA プロジェクトコンフィグレーションエディタ] ビューの [Stacks Configuration] ページを開きます。「3.5.5　スタック構成（Stacks Configuration）ページ」を参照してください。

(3) [Threads] ペインで "Blinky Thread" を選択し、[Stacks] ペインで [New Stack] → [Driver] → [Timers] → [Timer Driver on r_gpt] の順に選択して、Blinky スレッドに GPT モジュールを追加します。

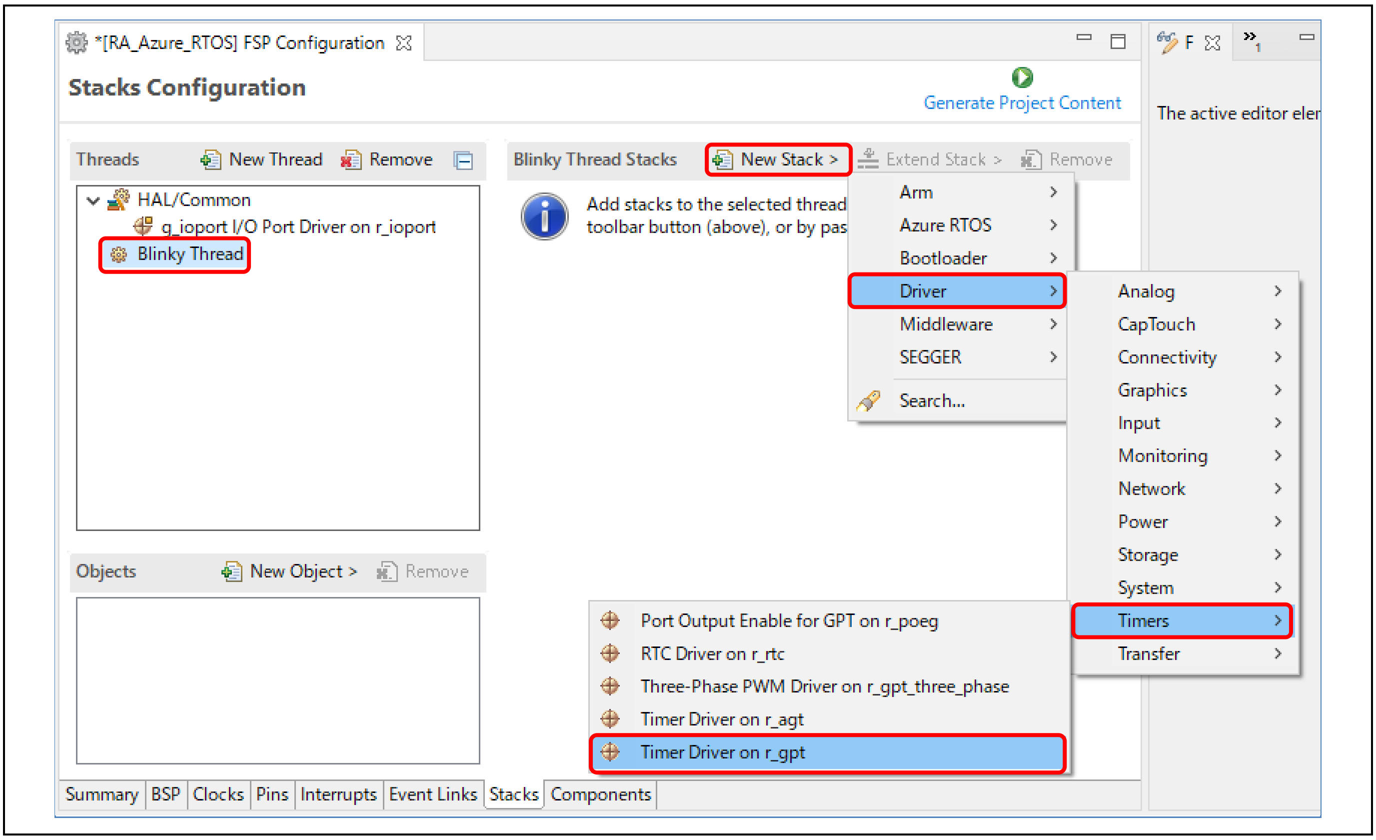


図130 Azure RTOSアプリケーションの設定 ― GPTモジュールの追加

(4) GPT モジュールを次のように設定します。

- Name：g_timer
- Mode：Periodic
- Period：1
- Period Unit：Seconds
- Callback：gpt_callback
- Overflow/Crest Interrupt priority：Priority 2

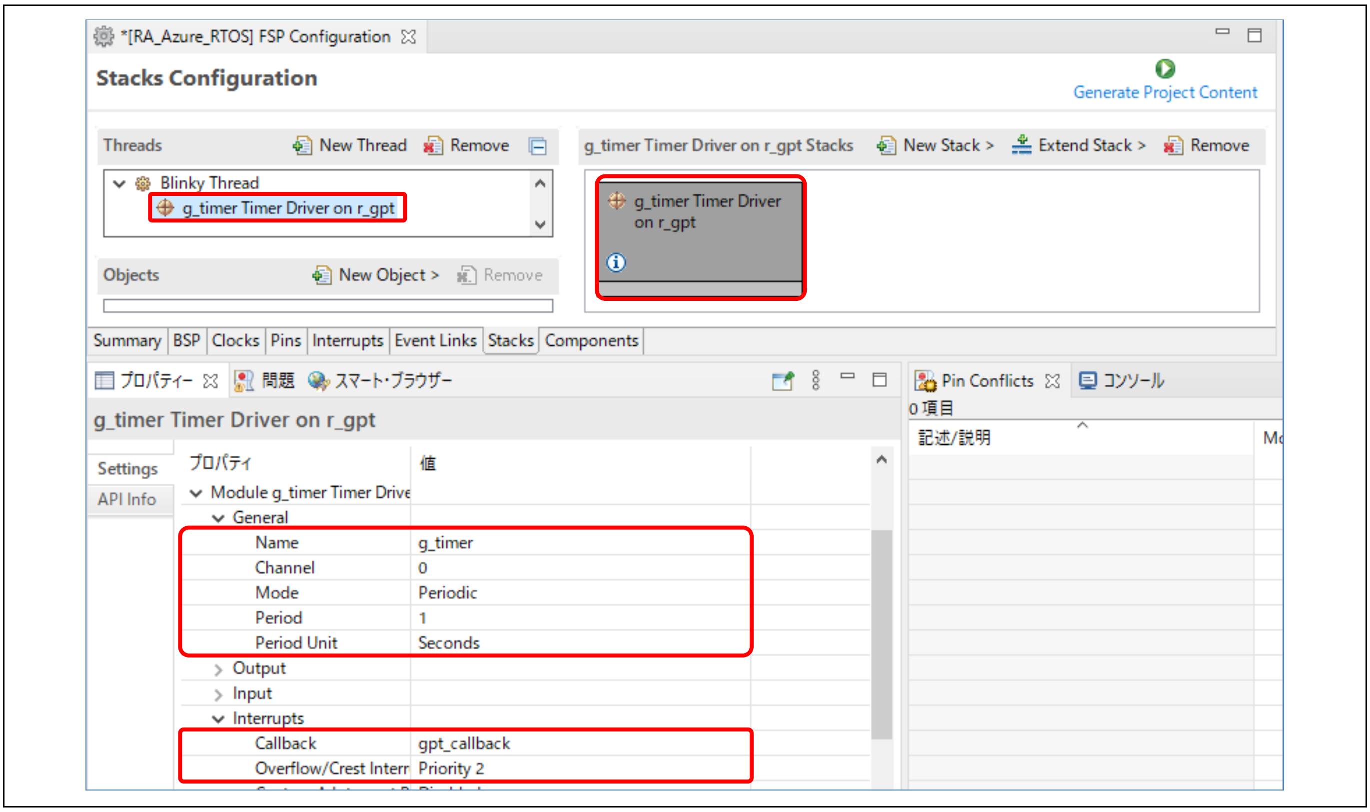


図 131　Azure RTOS アプリケーションの設定 ― GPT モジュールの設定

(5) [Threads] ペインで “Blinky Thread” を選択し、[Objects] ペインで [New Object] → [Semaphore] の順に選択して、Blinky スレッドにセマフォオブジェクトを追加します。

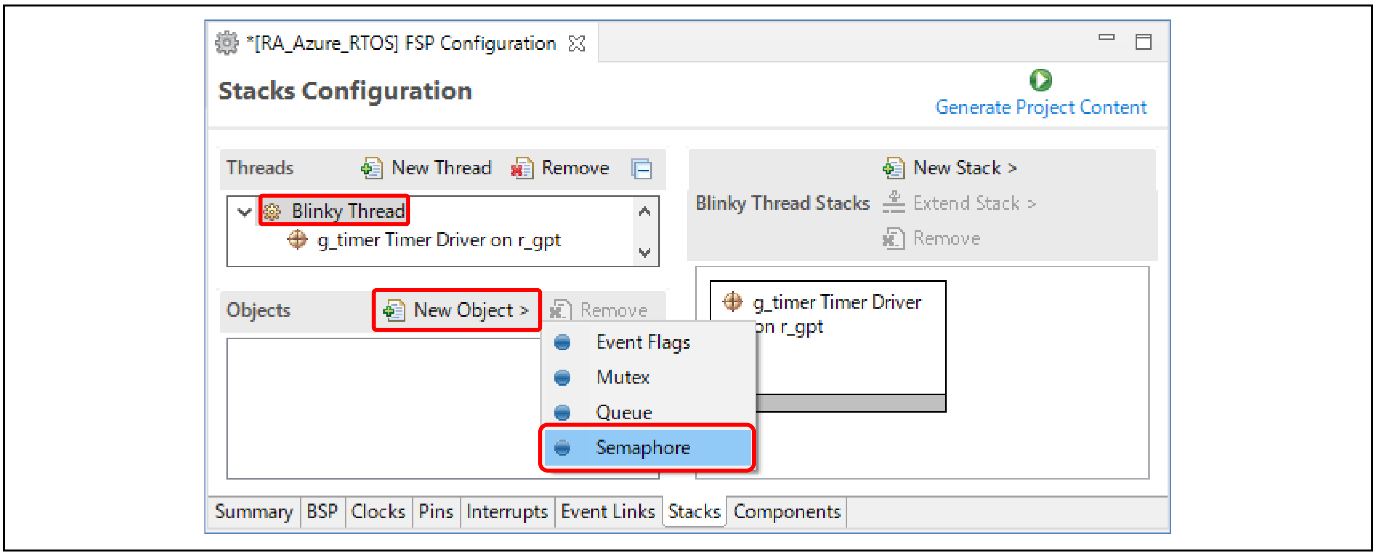


図132　Azure RTOSアプリケーションの設定 ― セマフォオブジェクトの追加

(6) 作成したセマフォを次のように設定します。

- Name：Blinky Semaphore
- Symbol：g_blinky_semaphore

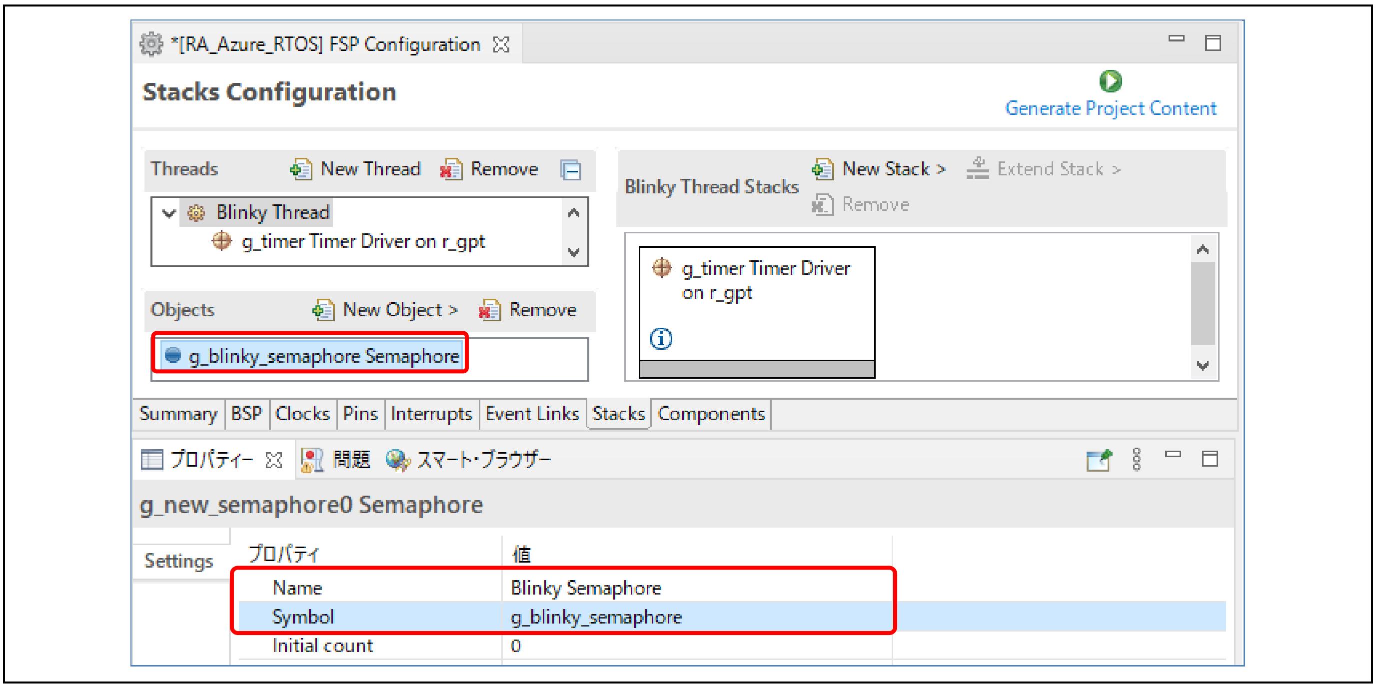


図 133 Azure RTOS アプリケーションの設定 ー セマフォオブジェクトの設定

(7) [Ctrl + S] キーで設定を保存し、Generate Project Content ボタンでソースコードを生成します。

(8) “src\blinky_thread_entry.c” ファイルを開き、次のように変更します。

- blinky_thread_entry() 関数の “while(1)” ループの前に、GPT モジュール初期化のコードを追加します。

```
g_timer.p_api->open(g_timer.p_ctrl, g_timer.p_cfg);
g_timer.p_api->start(g_timer.p_ctrl);
```

- blinky_thread_entry() 関数のタスクディレイ命令を削除し、セマフォを待つコードを追加します。

```
tx_semaphore_get(&g_blinky_semaphore, TX_WAIT_FOREVER);
```

- Blinky スレッドにセマフォを通知する関数 gpt_callback() を追加します。

```
void gpt_callback(timer_callback_args_t *p_args) {
(void)p_args;
tx_semaphore_put(&g_blinky_semaphore);
}
```

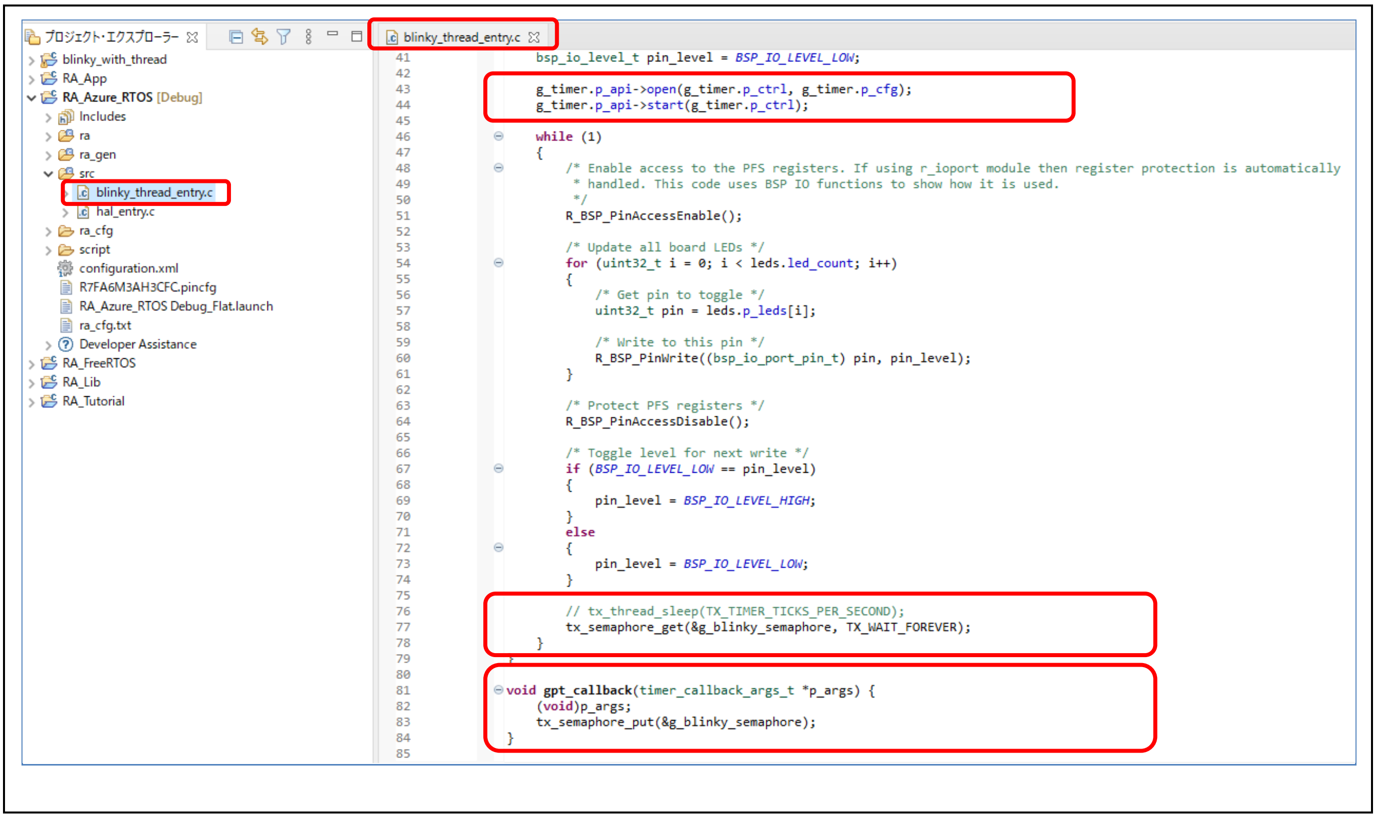


図 134　Azure RTOS アプリケーションの設定 ー ユーザソースコードの追加

(9) プロジェクトをビルドして、EK-RA6M3 ボードで実行します。 1 秒間隔で LED が点滅するのを確認してください。

8. ヘルプ

ヘルプシステムによって、ユーザはワークベンチ内の各ヘルプウィンドウやヘルプ画面から、ヘルプドキュメントのブラウズ、検索、ブックマーク、印刷が可能です。また、ヘルプメニューから e² studio 専用のオンラインフォーラムにアクセスできます。

[ヘルプ] をクリックしてヘルプメニューをプルダウンしてください。

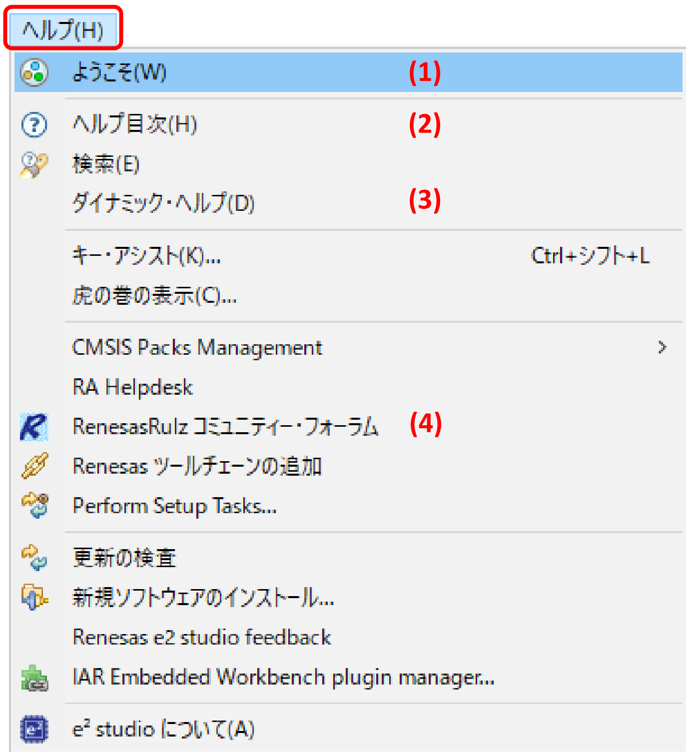


図 135　ヘルプメニュー

ヘルプメニューの機能：

(1) [ようこそ] をクリックすると、e² studio の概要とリリースノートを表示します。

(2) [ヘルプ目次] をクリックすると、新たにヘルプウィンドウが開きヘルプを検索できます。

(3) [ダイナミック・ヘルプ] をクリックすると、ワークベンチにヘルプ画面を開きます。

(4) [RenesasRulz コミュニティー・フォーラム] をクリックすると、e² studio 関連のディスカッション参加型オンラインフォーラムにアクセスします。インターネット接続が必要です。

[ヘルプ目次] 内には多くの有用なコンテンツが入っています。

- “デバッグに関する機能” にはデバッガの設定、ブレークポイントの使用可能数、などが書かれています。

  [ヘルプ] メニューから [ヘルプ目次] → [e² studio ユーザーガイド] → [デバッグに関する機能] の順に選択してください。

- “RA トピックス” には RA プロジェクトの作成方法、RA コンフィグレーションエディタの使い方、FAQ、などが書かれています。

  [ヘルプ] メニューから [ヘルプ目次] → [RA トピックス] の順に選択してください。

改訂記録	Renesas e² studio 2021-04 以上 ユーザーズマニュアル：クイックスタートガイド

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2021.9.24	ー	初版発行

Renesas e^2 studio 2021-04以上
ユーザーズマニュアル：クイックスタートガイド

発行年月日　2021年9月24日　　Rev.1.00

発行　　　　ルネサス エレクトロニクス株式会社
　　　　　　〒135-0061　東京都江東区豊洲3-2-24（豊洲フォレシア）

統合開発環境 e² studio 2021-04 以上
ユーザーズマニュアル：クイックスタートガイド

ルネサス エレクトロニクス株式会社

R20UT4989JJ0100