RL78/G24

フレキシブル・アプリケーション・アクセラレータ(FAA) ツールガイド e2 studio 編

要旨

本ガイドでは、RL78/G24 に搭載のフレキシブル・アプリケーション・アクセラレータ（FAA）のプログラムのビルドおよびデバッガ操作について説明します。

対象デバイス

RL78/G24

RL78/G24 Fast Prototyping Board

構成

1 章：フレキシブル・アプリケーション・アクセラレータ（FAA）の概要

FAA および、FAA のプログラム作成の概要について説明しています。

2 章：FAA プログラムのビルド、デバッグの概要

新規プロジェクト作成手順、FAA のプログラムのビルドおよびデバッグ時に設定が必要なツールオプションについて説明しています。また、デバッガの基本操作について説明しています。

3 章：サンプルプロジェクトによるデバッガ操作

サンプルコードを使用して、FAA プログラムのデバッグ操作について説明しています。

関連ドキュメント

- RL78/G24 ユーザーズマニュアル　ハードウェア編（R01UH0961）
- RL78/G24 Fast Prototyping Board ユーザーズマニュアル（R20UT5091）

目次

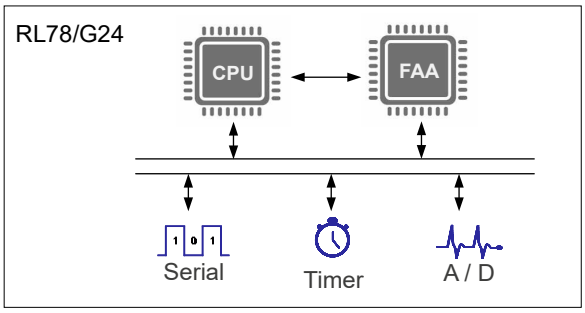
1.　概要

1.1　フレキシブル・アプリケーション・アクセラレータ（FAA）

RL78/G24 に搭載されるフレキシブル・アプリケーション・アクセラレータ（FAA）は、ハーバード・アーキテクチャを採用したルネサス エレクトロニクス株式会社のオリジナルのアプリケーションアクセラレータです。32 ビット乗算、加算、減算を 1 サイクルで実行します。

また、アドレス・バス選択機能により FAA から一部の周辺機能を直接アクセスできます。CPU と FAA を組み合わせて動作させることができ、システムの特性を向上させることができます。

図 1-1　RL78/G24　FAA のイメージ



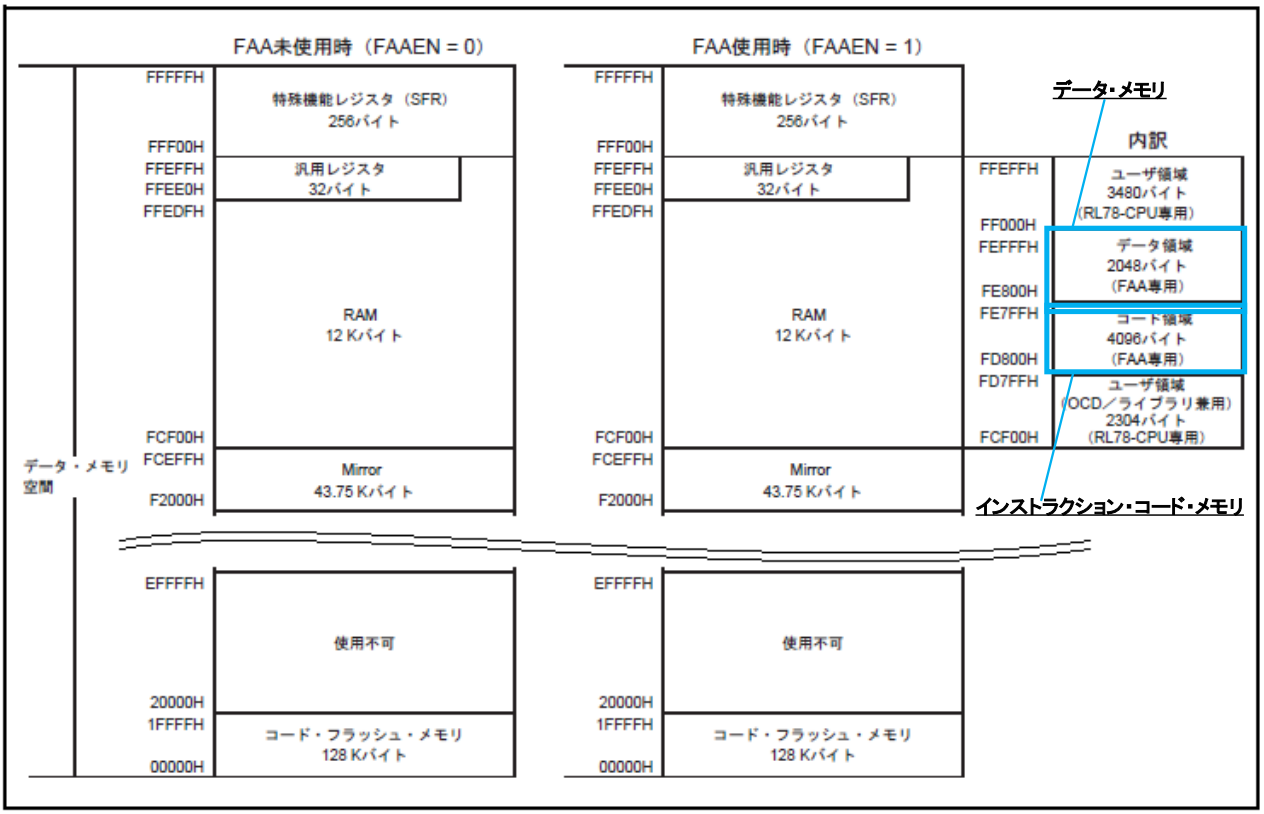
1.2　FAA のメモリ領域

FAA を使用時、RL78/G24 の内部 RAM の一部が FAA 専用のメモリとなります。

インストラクション・コード・メモリ：　FAA が実行するプログラムを格納します。

データ・メモリ：　FAA のプログラム実行に必要なデータを格納します。

図 1-2　インストラクション・コード・メモリ、データ・メモリのメモリマップ



1.3　RL78/G24 のプログラム

1.3.1　プログラム構成

CPU のプログラムと FAA のプログラムは別ファイルに記述します。また、FAA のプログラムは、FAA 専用の命令セットを使用します。CPU プログラムと FAA プログラムを一緒にビルドし、RL78/G24 で実行可能なオブジェクト・ファイル（ロード・モジュール・ファイル）にします。

図 1-3　RL78/G24　FAA 使用時のプログラム構成のイメージ

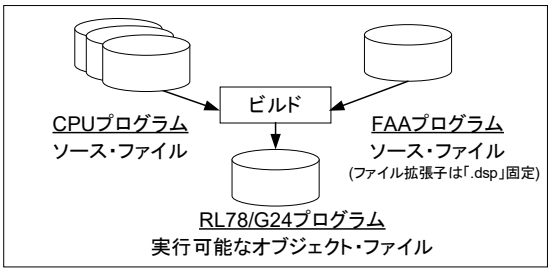
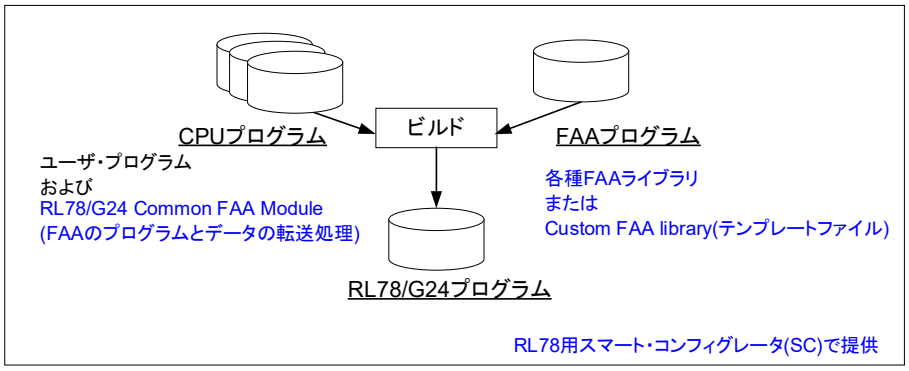


図 1-4　RL78 用スマート・コンフィグレータ（SC）で提供する FAA 関連のファイル



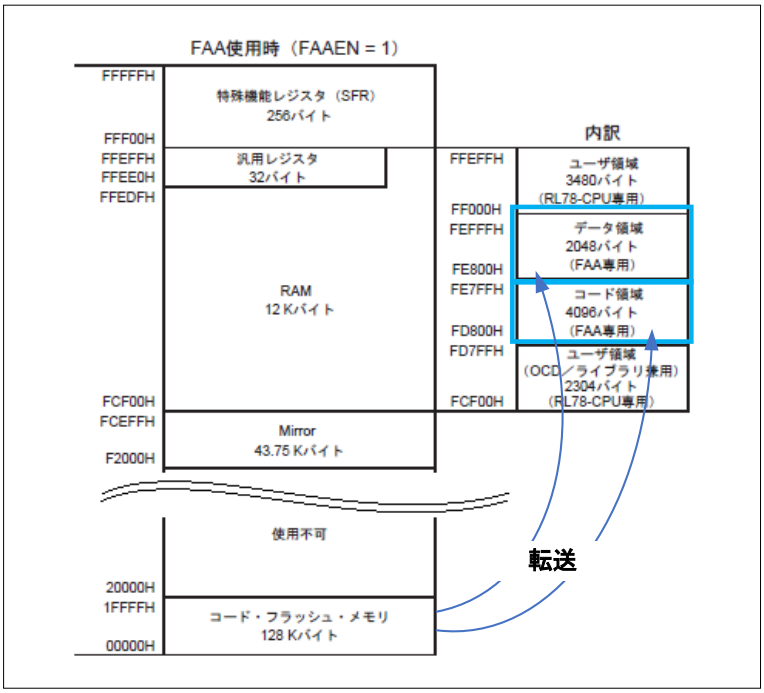
備考. SC 提供の FAA のプログラムとデータの転送処理および FAA ライブラリを使用しない場合、ご自身でプログラムを用意して頂く必要があります。

1.3.2　FAA のプログラムとデータの転送

実行可能なオブジェクト・ファイルは、RL78/G24 のコード・フラッシュ・メモリに書き込みます。しかし、FAA のプログラムはインストラクション・コード・メモリに、データはデータ・メモリに配置する必要があります。そのため、FAA のプログラムを実行する前（FAA 動作許可前）に、コード・フラッシュ・メモリに格納されている FAA のプログラムとデータを、インストラクション・コード・メモリ、データ・メモリへ転送する必要があります。

FAA のプログラムとデータの転送処理は、RL78 用スマート・コンフィグレータ（SC）の FAA コンポーネントーFAA ライブラリ「RL78/G24 Common FAA Module」で提供しています。SC で、転送処理を出力する手順については、2.3 FAA プログラムの追加 を参照してください。

図 1-5　FAA のプログラムとデータの転送



1.3.3　FAA プログラム

FAA プログラムは、次のいずれかの方法で作成します。

- 用途に応じて用意された FAA ライブラリを使用する。ライブラリはソース・ファイルで提供されますがソース変更は不可です。（各種機能の FAA ライブラリ）
- テンプレートファイルを使用して、ユーザ自身でコーディングする。（テンプレート（Custom FAA Library））

どちらの方法も、スマート・コンフィグレータ（SC）でプログラムのプロジェクトに追加します。

SC で、FAA プログラムのファイル（ライブラリまたはテンプレート）を出力する手順については、2.3 FAA プログラムの追加 を参照してください。

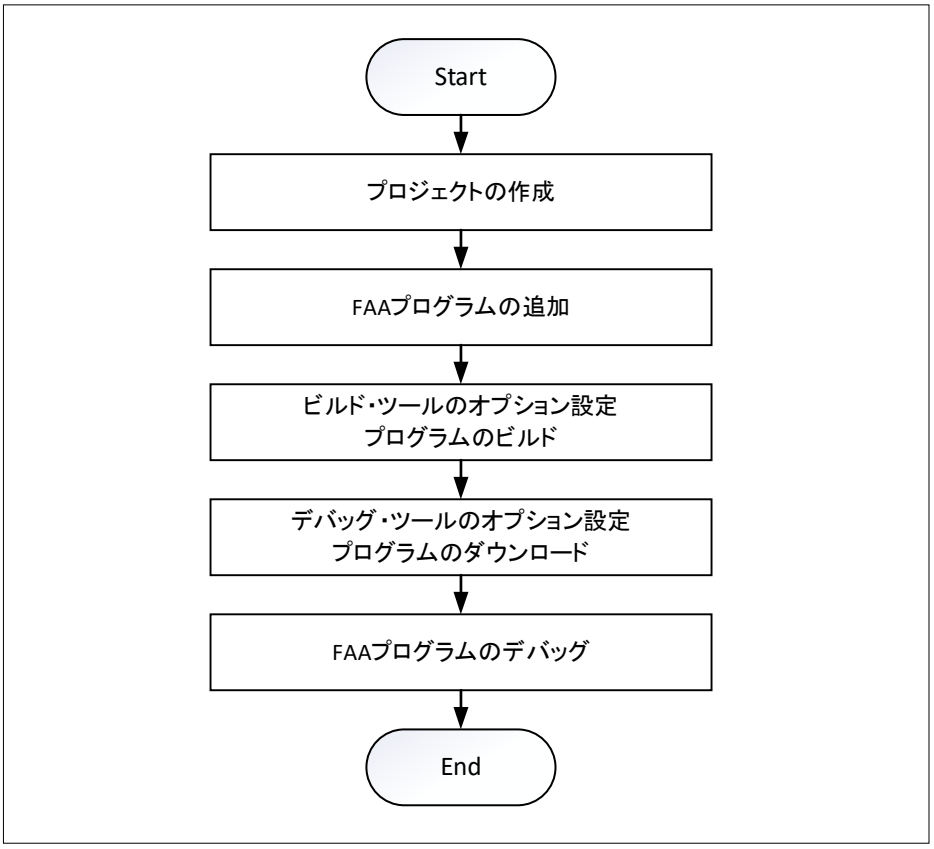
備考. FAA の命令セットは、RL78/G24 ユーザーズマニュアル　ハードウェア編（R01UH0961）のフレキシブル・アプリケーション・アクセラレータ（FAA）の章を参照してください。

1.3.4　FAA プログラムのビルドとデバッグ

FAA のプログラムのビルドおよびデバッグには、いくつかのオプションを設定する必要があります。本ガイドでは、図 1-6 に示す各処理において、設定が必要なオプションについて説明します。また、FAA プログラムをデバッグする際のデバッガ操作方法について説明します。

なお、本ガイドでは、スマート・コンフィグレータ（SC）が生成する FAA プログラム（ライブラリまたはテンプレート）を使用することを前提としています。

図 1-6　本ガイド 2 章での操作説明



2.　オプション設定と操作

e2 studio 環境で、FAA プログラムのビルドおよびデバッグに必要なオプション設定とデバッガ操作について説明します。

本ガイドで説明のないオプションについては、必要に応じて適宜設定してください。また、オプション、操作の詳細、および制限事項については、e2 studio のヘルプまたはドキュメントを参照してください。

2.1　動作環境

本ガイドでは、次のツールで説明をします。

表 2-1　ソフトウェア・ツール

統合開発環境	項目	バージョン
e2 studio	ルネサス エレクトロニクス製　e2 studio	v2025-12
	ルネサス エレクトロニクス製 RL78 ファミリ用 C コンパイラ CC-RL	V1.15.01
	ルネサス エレクトロニクス製 FAA/GREEN_DSP　構造化アセンブラ　DSPASM	V1.05.00
	ルネサス エレクトロニクス製 RL78 スマート・コンフィグレータ　プラグイン	e2 studio v2025-12 に同梱のバージョン

表 2-2　ハードウェア・ツール

ボード/ エミュレータ	項目
ボード[注1]	ルネサス エレクトロニクス製　RL78/G24　Fast Prototyping Board
エミュレータ[注2]	ルネサス エレクトロニクス製　E2 エミュレータ Lite
	ルネサス エレクトロニクス製　E2 エミュレータ

注1.　COM Port 接続／エミュレータ接続で、ジャンパ設定等が異なります。詳細は RL78/G24 Fast Prototyping Board ユーザーズマニュアル（R20UT5091）を参照してください。

注 2.　デバッガと RL78/G24 Fast Prototyping Board を COM Port 接続する場合、エミュレータは必要ありません。

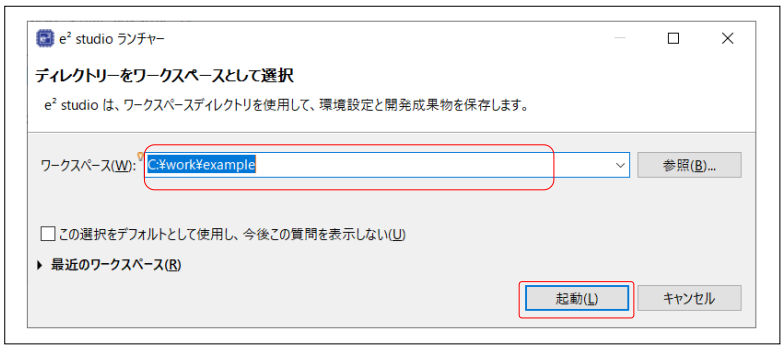
2.2　プロジェクトの作成

使用するマイクロコントローラに RL78/G24 製品を選択し、プログラムのプロジェクトを作成します。

手順

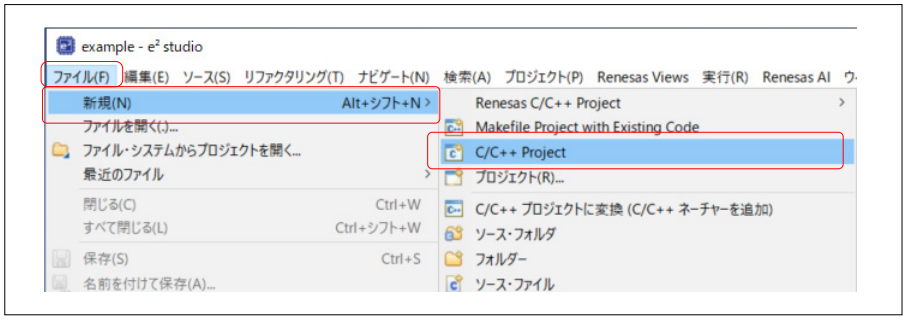
1. e2 studio を起動します。
2. 「e2 studio ランチャー」の「ワークスペース」で、ワークスペースディレクトリを指定し、「起動」をクリックします。

図 2-1　e2 studio ランチャー



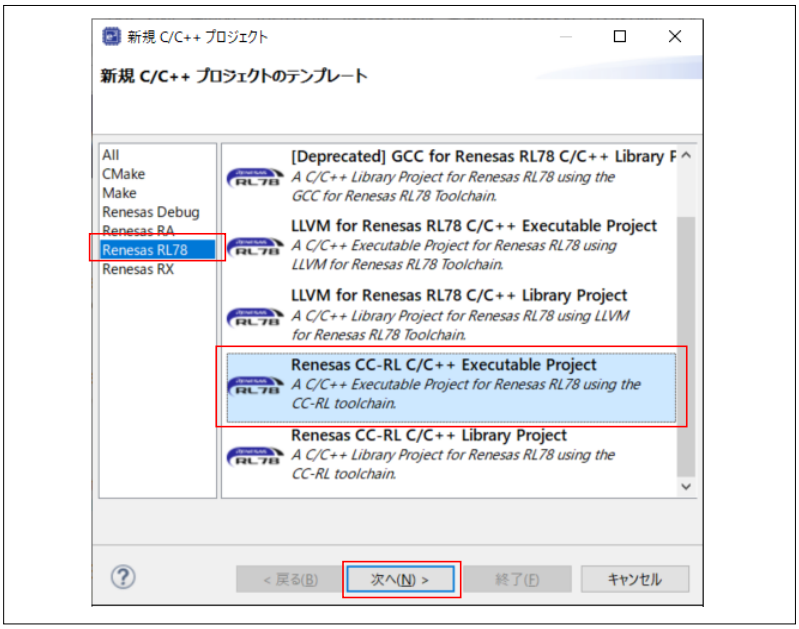
3. 「ファイル」メニュー→「新規」→「C/C++ Project」を選択します。

図 2-2　「ファイル」メニュー→「新規」→「C/C++ Project」



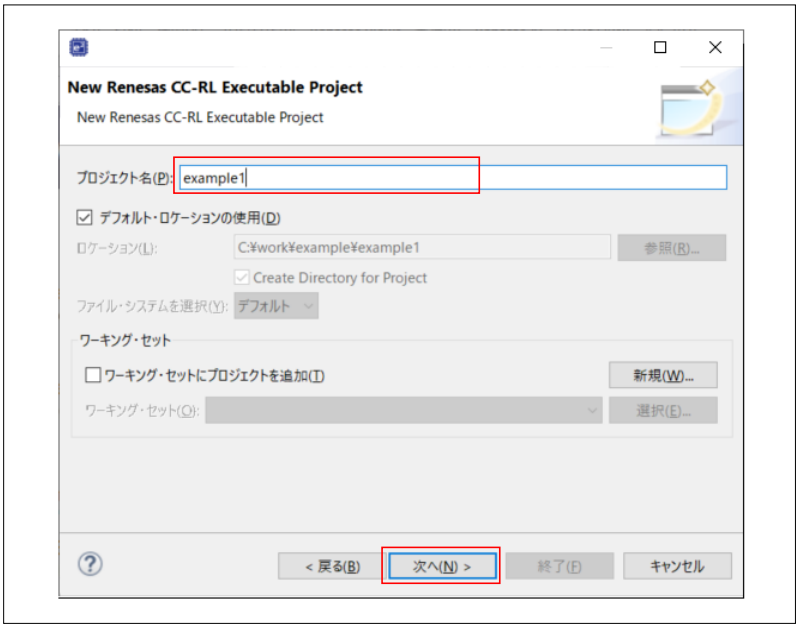
4. 「新規 C/C++プロジェクト」ダイアログで、「Renesas RL78」、「Renesas CC-RL C/C++ Executable Project」を選択し、「次へ」をクリックします。

図 2-3　新規 C/C++ プロジェクト ダイアログ（テンプレート選択）



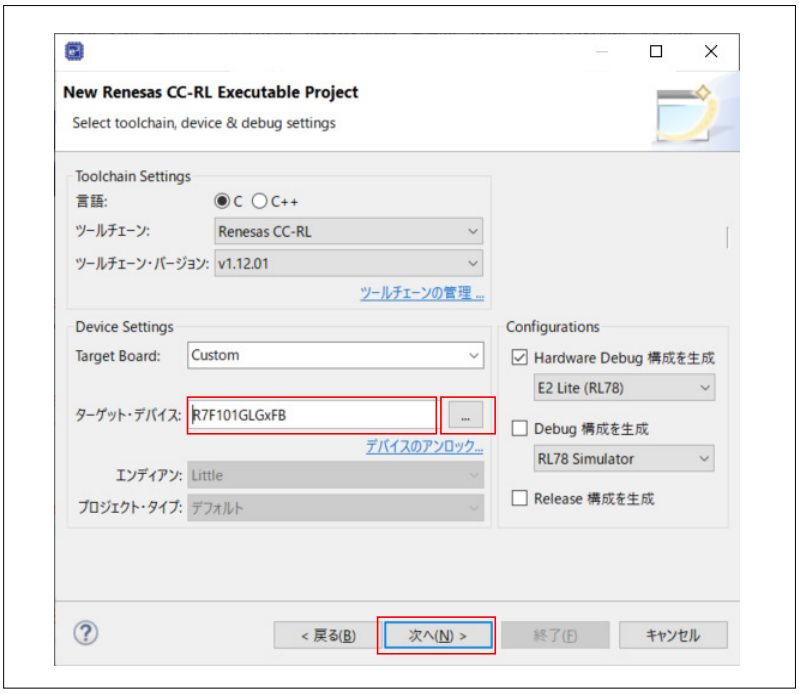
5. 「プロジェクト名」にプロジェクト名を入力し、「次へ」をクリックします。

図 2-4　新規 C/C++ プロジェクト ダイアログ（プロジェクト・ファイル名指定）



6. 「ターゲット・デバイス」に”R7F101GLGxFB”と入力し、「次へ」をクリックします。
（「…」をクリックし、デバイス名一覧からデバイスを選択することもできます。）

図 2-5　新規 C/C++ プロジェクト ダイアログ（ターゲット・デバイス選択）



7. 「Use Smart Configurator」をチェックし、「終了」をクリックします。

図 2-6　新規 C/C++ プロジェクト ダイアログ（スマート・コンフィグレータ選択）

8. 「関連付けられたパースペクティブを開きますか？」ダイアログで、「パースペクティブを開く」をクリックします。「ようこそ」タブが最前面に表示されている場合、「ようこそ」タブの「隠す」をクリックします。

図 2-7　関連付けられたパースペクティブを開きますか？ ダイアログ

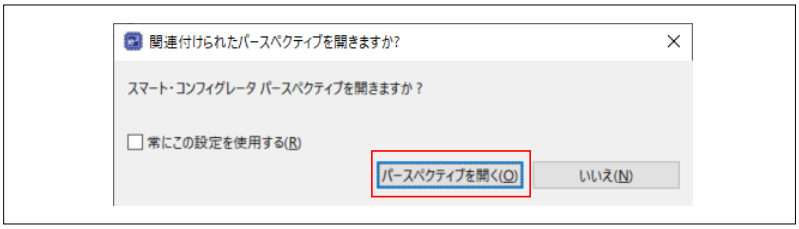
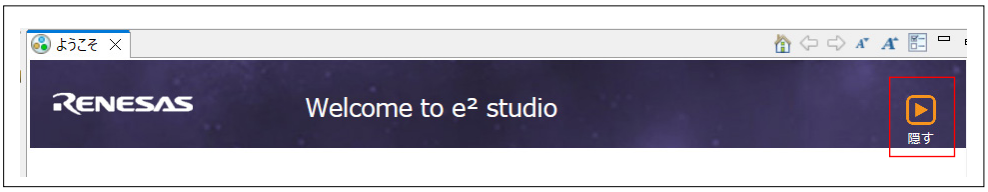
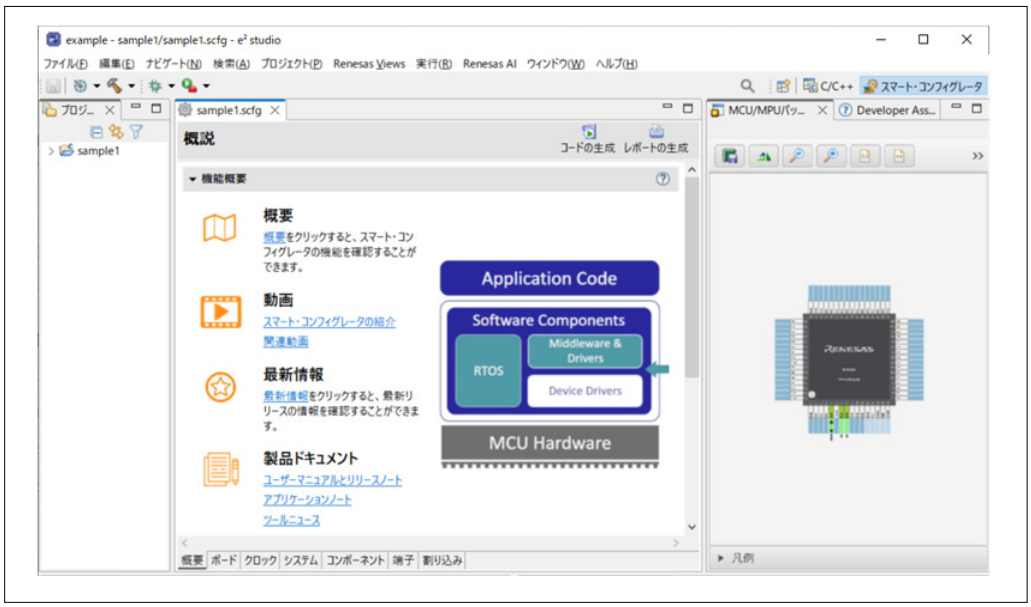


図 2-8　ようこそ タブ



9. スマート・コンフィグレータのパースペクティブが表示されます。

図 2-9　スマート・コンフィグレータのパースペクティブ



2.3　FAA プログラムの追加

スマート・コンフィグレータ（SC）から FAA のプログラム（ライブラリまたはテンプレート）をプロジェクトへ追加します。

本ガイドでは、SC の FAA コンポーネントの追加と、CPU のプログラムで最低限設定が必要な「クロック」、「システム」、「電源検出回路」の設定のみを説明します。その他の周辺機能はシステムに合わせて適宜設定してください。

2.3.1　FAA コンポーネントの追加

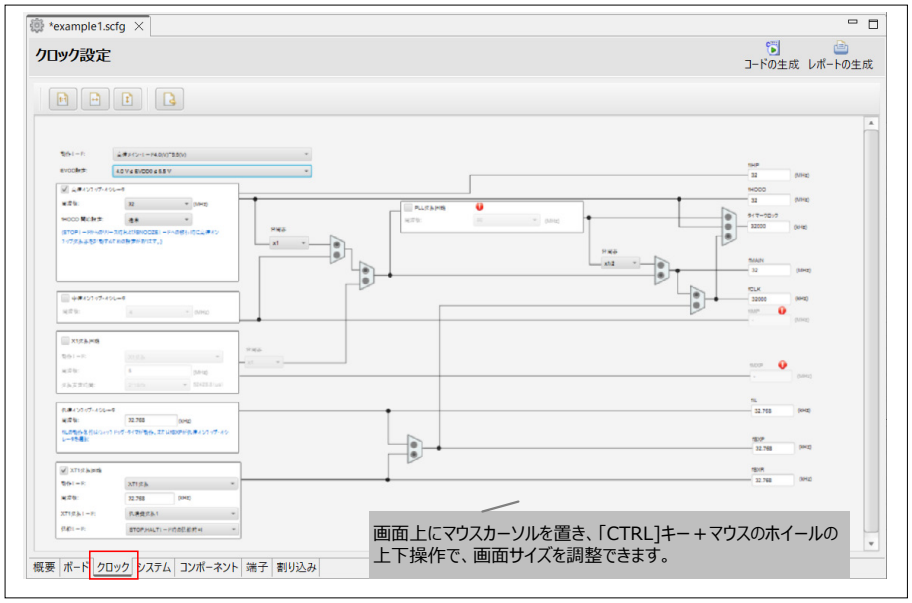
手順

1. スマート・コンフィグレータで、「クロック」をクリックします。

図 2-10　スマート・コンフィグレータ　「クロック」タブの選択

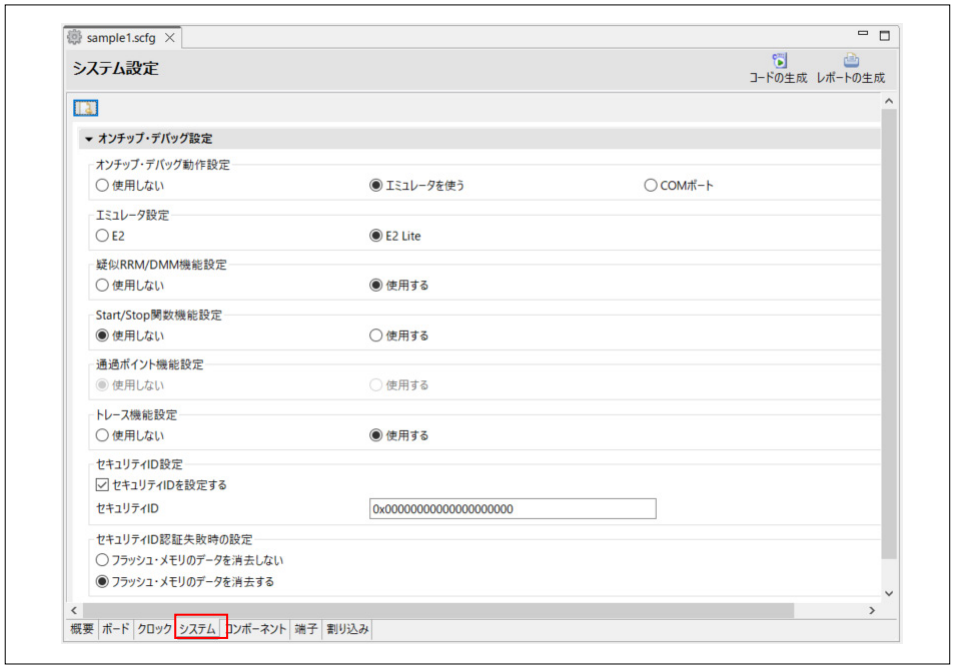
2. システムに合わせて各種クロック、動作モードを設定します。

図 2-11　スマート・コンフィグレータ　「クロック」タブ



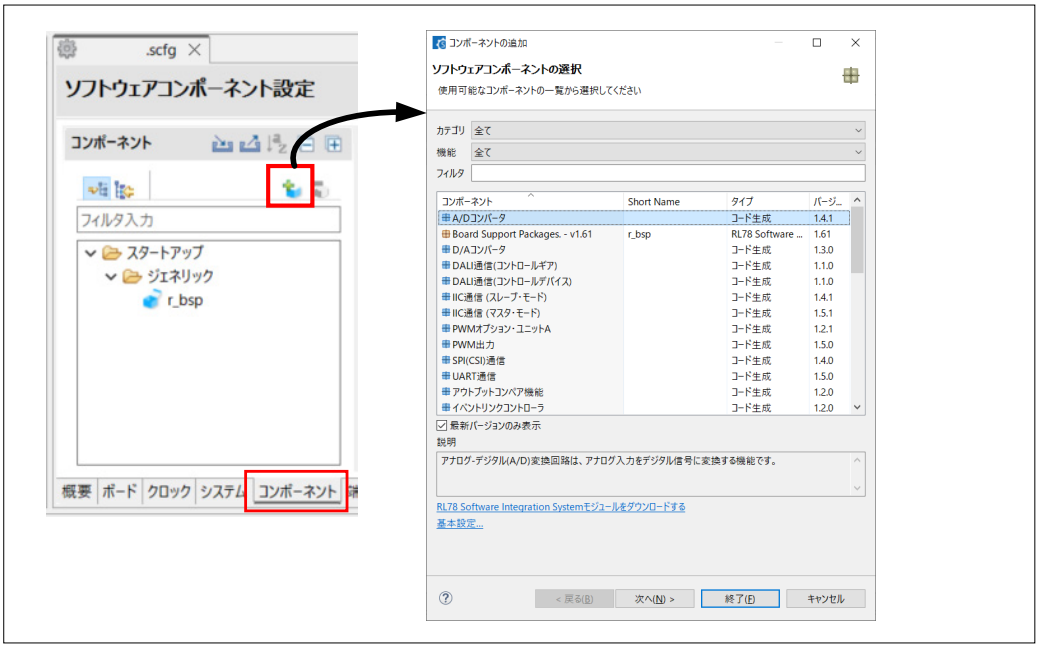
3. 「システム」をクリックします。「システム」タブで、使用するデバッグ・ツール、機能、セキュリティ ID を設定します。

図 2-12　スマート・コンフィグレータ　「システム」タブ



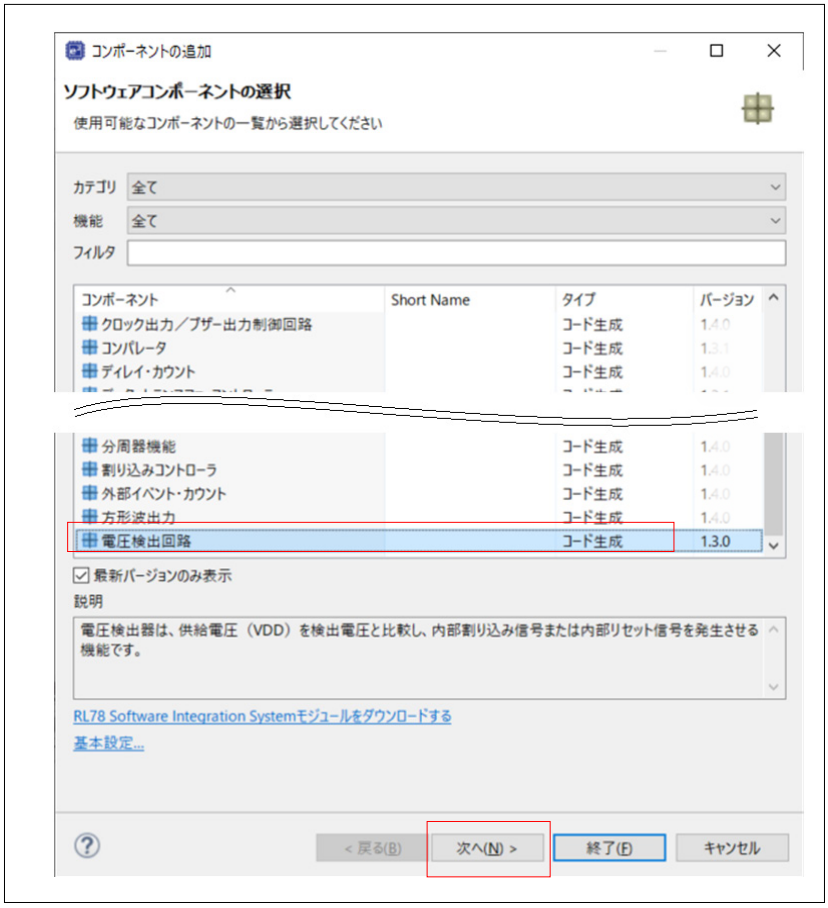
4. 「コンポーネント」をクリックします。次に「コンポーネントの追加」をクリックします。「コンポーネントの追加」ダイアログが表示されます。

図 2-13　スマート・コンフィグレータ「コンポーネント」タブ



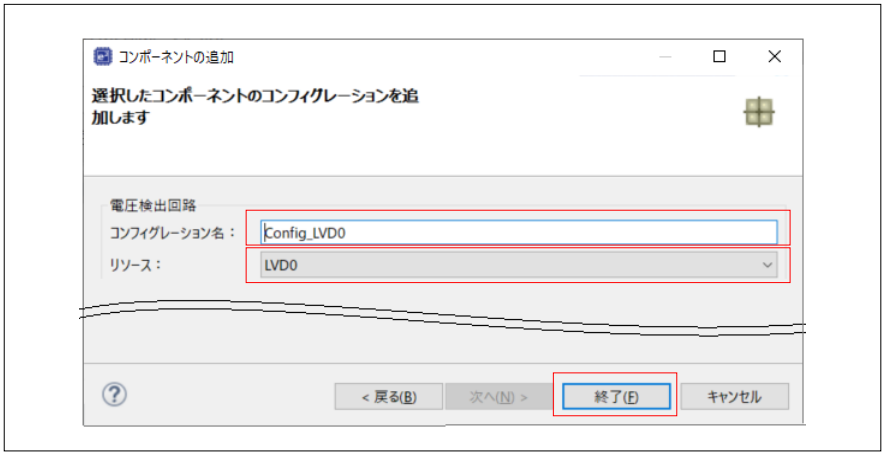
5. 「コンポーネントの追加」ダイアログで、「電圧検出回路」を選択し、「次へ」をクリックします。

図 2-14　「電圧検出回路」の選択



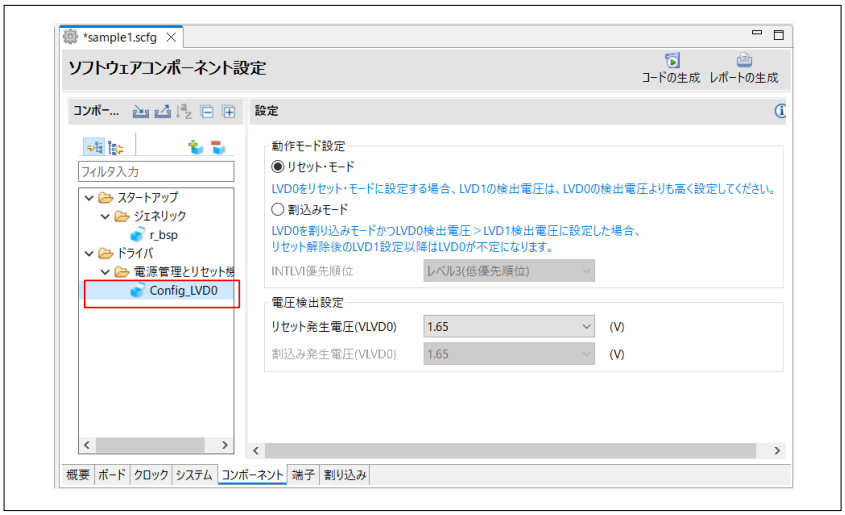
6. 「リソース」で、LVD0 を選択します。また、コンフィグレーション名を確認し、「終了」をクリックします。（コンフィグレーション名は任意の名前に変更できます。）

図 2-15　リソース選択とコンフィグレーション名の確認（電圧検出回路）



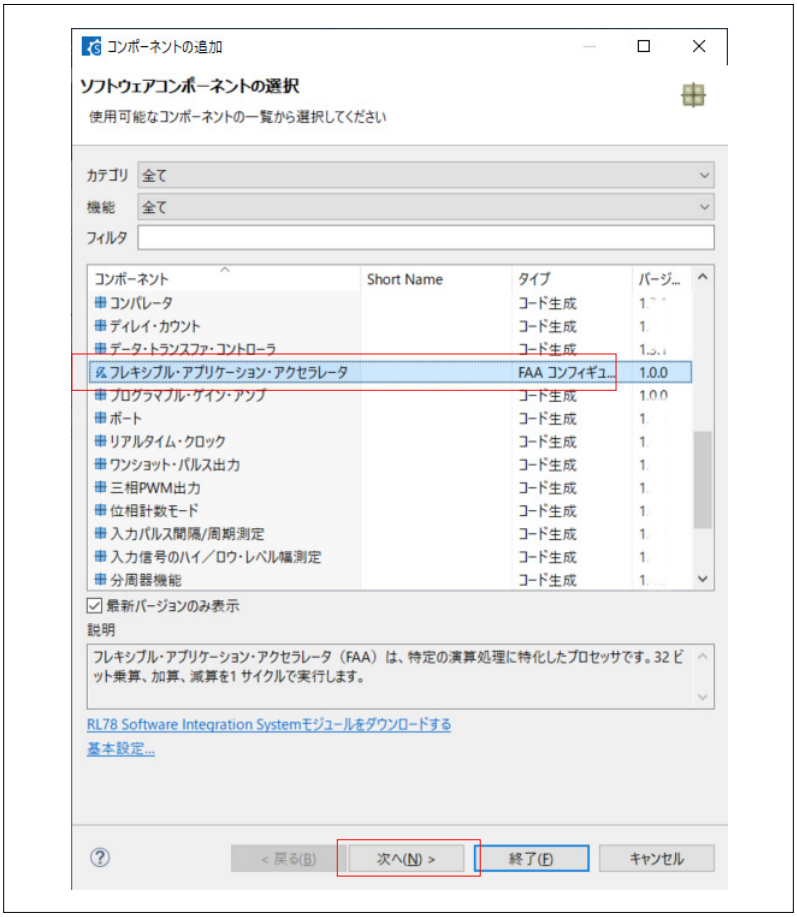
7.　使用するコンポーネントのツリーに電圧検出回路が追加されます。設定画面で、システムに合わせて電圧検出回路を設定します。

図 2-16　スマート・コンフィグレータ　「電圧検出回路」設定画面



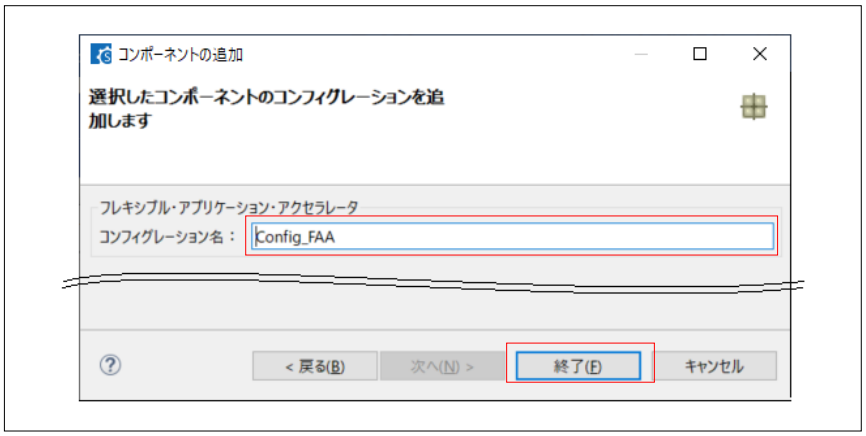
8.　再度「コンポーネントの追加」ダイアログを開き、「フレキシブル・アプリケーション・アクセラレータ」を選択し、「次へ」をクリックします。

図 2-17　「フレキシブル・アプリケーション・アクセラレータ」の選択



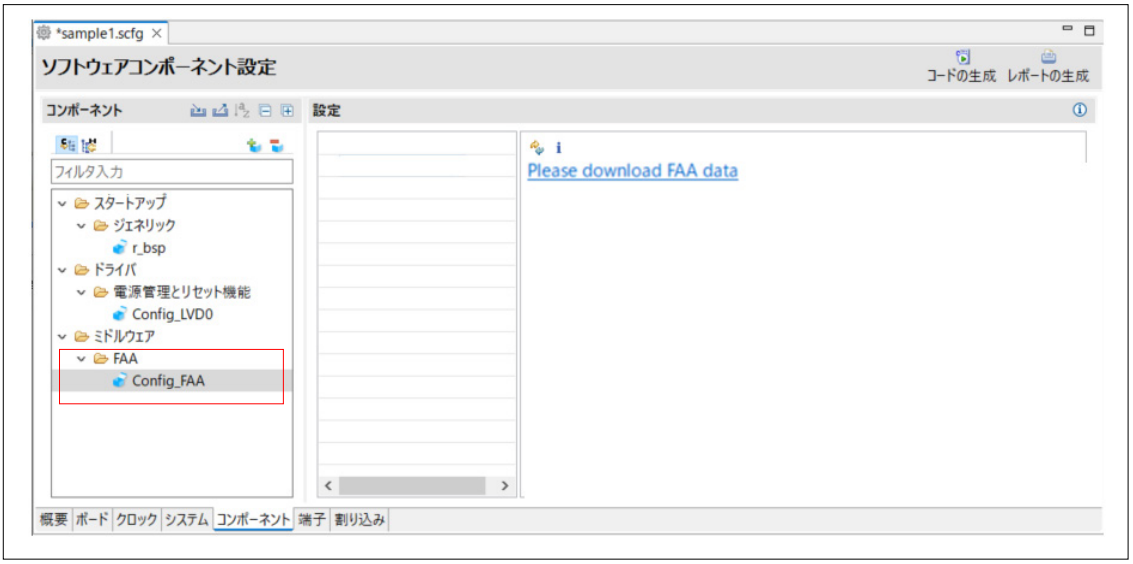
9. コンフィグレーション名を確認し、「終了」をクリックします。（コンフィグレーション名は任意の名前に変更できます。）

図 2-18　コンフィグレーション名の確認（フレキシブル・アプリケーション・アクセラレータ）



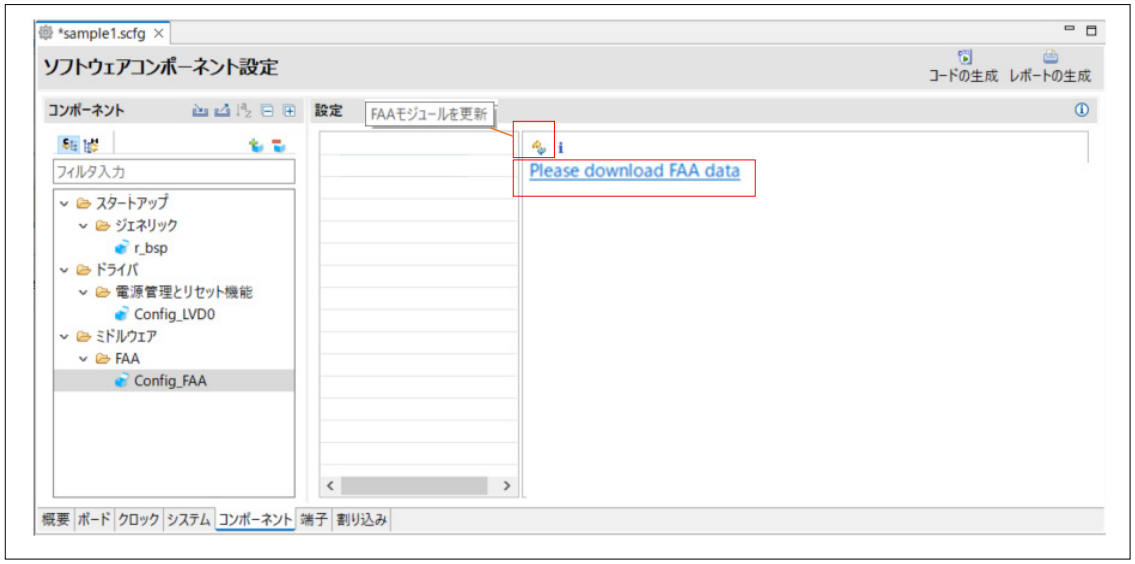
10. 使用するコンポーネントのツリーに FAA が追加されます。

図 2-19　FAA（フレキシブル・アプリケーション・アクセラレータ）コンポーネントの追加



11. FAA コンポーネントを初めて使用する場合、FAA ライブラリまたはテンプレートをスマート・コンフィグレータ専用のサーバからダウンロードする必要があります。ダウンロードするために「FAA モジュールを更新」または「Please download FAA data」をクリックします。（「FAA モジュールを更新」は、新規 FAA ライブラリや最新バージョンの確認・取得時にも利用してください。）

図 2-20　FAA モジュール（ライブラリ）を更新／ダウンロード



12. ダウンロードするライブラリを選択し、「ダウンロード」をクリックします。続いて表示される「免責事項」ダイアログで、「Accept」をクリックします。

図 2-21　FAA モジュール（ライブラリ）のダウンロード

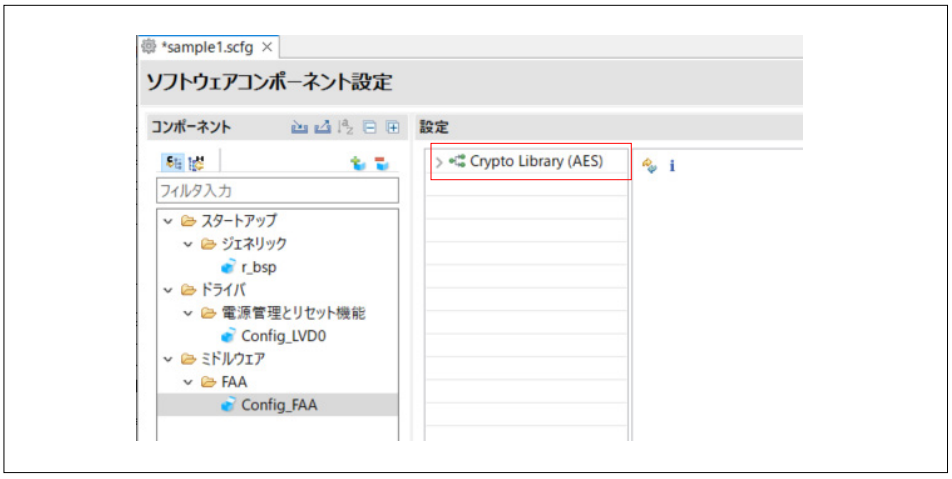
備考：実際のダウンロード画面で表示される内容と異なります。

表 2-3　FAA ライブラリ

タイトル	概要
RL78/G24 Common FAA Module	1.3.2 FAA のプログラムとデータの転送 に記載の FAA のプログラムとデータの転送ルーチンです。 FAA ライブラリ／テンプレートを使用する場合、必ずダウンロードします。
Custom FAA Library	FAA プログラムを記載するテンプレートです。
その他	各種機能の FAA ライブラリです。

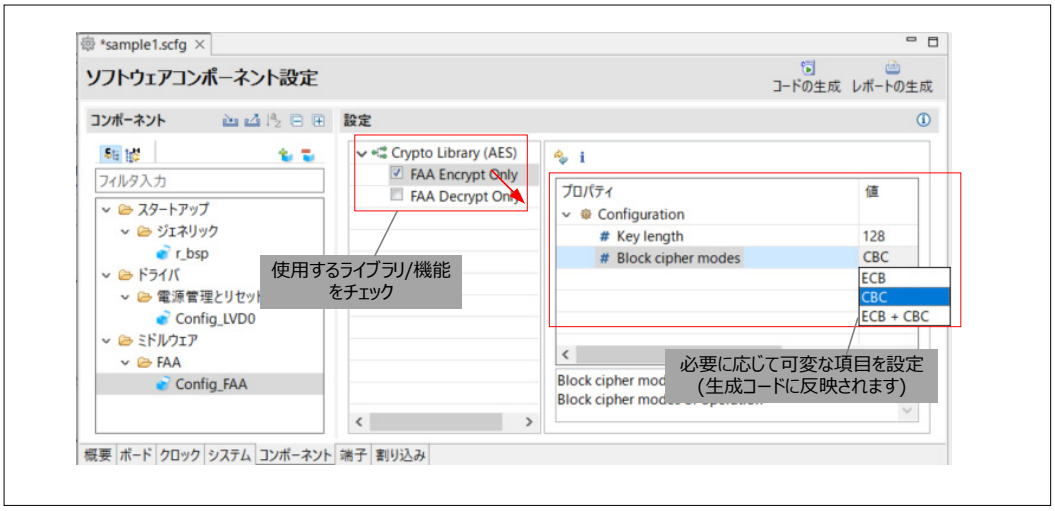
13. ダウンロードしたライブラリが追加されます。（RL78/G24 Common FAA Module は表示されません。）

図 2-22　FAA ライブラリの追加



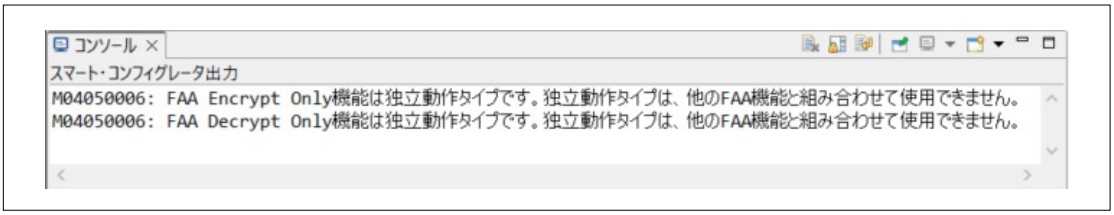
14. ダウンロードしたライブラリのうち、実際に使用するライブラリ／機能をチェックします。チェックした機能のプロパティで、設定項目がある場合は適宜設定します。

図 2-23　FAA ライブラリの選択・設定



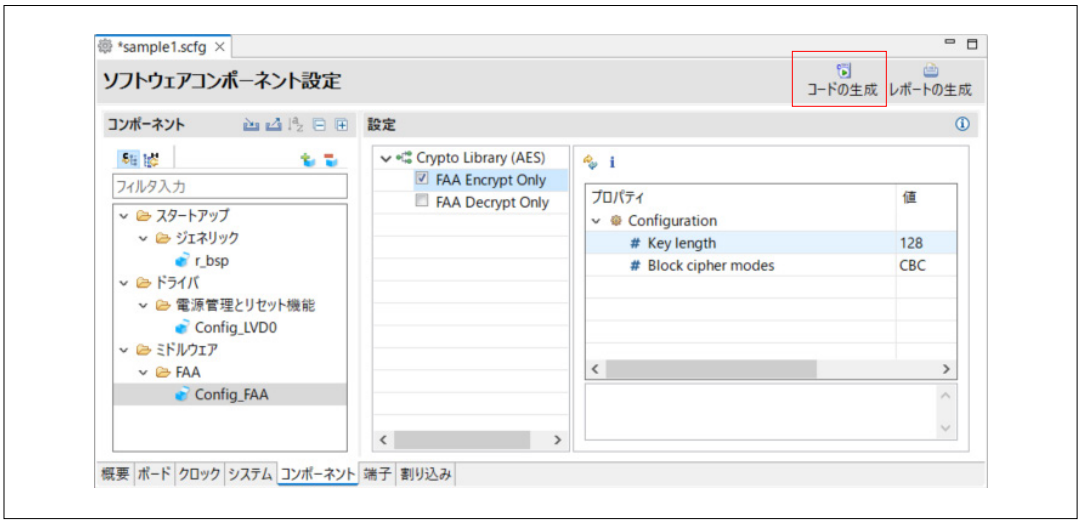
備考：ライブラリ／機能は、他と組み合わせて使用可能なタイプ（サブプロセッサ型）と組み合わせ使用不可のタイプ（独立型）があります。独立型と他のライブラリ／機能を同時に使用しないでください。独立型を選択後に他のライブラリ／機能を選択すると、「コンソール」タブに下記のようなメッセージが表示されます。

図 2-24　ワーニング



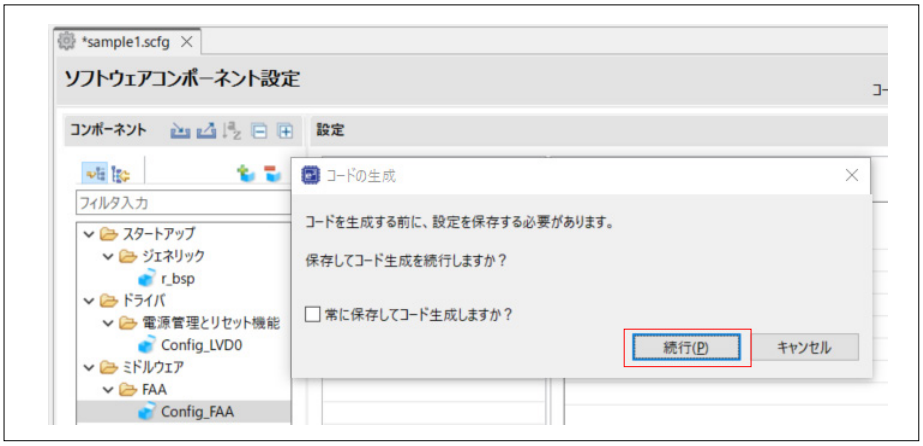
15. 「コードの生成」をクリックし、FAA ライブラリおよび追加した周辺機能のソース・ファイルを生成します。

図 2-25　コード生成



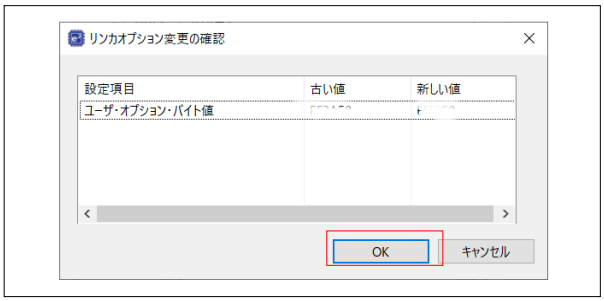
16. 「コード生成」ダイアログが表示された場合、「続行」をクリックします。

図 2-26　「コード生成」ダイアログ



17. 「リンカオプション変更の確認」ダイアログが表示された場合、「OK」をクリックします。

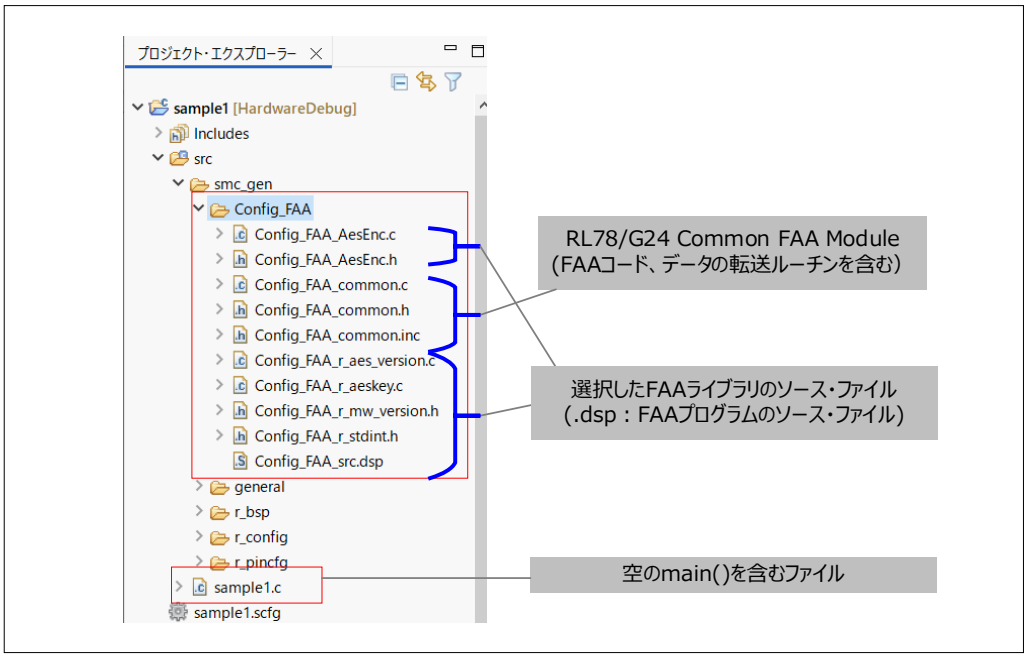
図 2-27　「リンカオプション変更の確認」ダイアログ



備考. スマート・コンフィグレータ（SC）の「クロック」、「システム」、「電圧検出回路」（LVD0）で設定した一部の項目は、ビルド・ツール（CC-RL）のリンカオプションの設定に反映されます。

18. FAA ライブラリおよび追加した周辺機能のソース・ファイルが生成されプロジェクトに登録されます。選択する FAA ライブラリによって生成されるファイルは異なります。FAA ライブラリのソース・ファイル例について以下に示します。

図 2-28　FAA ライブラリのソース・ファイル（例）



備考 1. スマート・コンフィグレータ（SC）の「クロック」、「システム」、「電圧検出回路」（LVD0）で設定した一部項目の内容は、ビルド・ツール（CC-RL）のリンカオプションの設定に反映されます。

備考 2. 上記赤枠以外のファイルについては、RL78 スマート・コンフィグレータ ユーザーガイド: e² studio 編（R20AN0579）を参照してください。

19. FAA を制御する関数が FAA ライブラリのソース・ファイル内に定義されています。CPU プログラムでこれらの関数をコールし FAA を動作させます。システムに合わせて CPU プログラムを作成してください。
関数については、各 FAA ライブラリのドキュメントを参照してください。
（https://www.renesas.com/rl78g24　「ドキュメント」にて検索）

2.3.2 FAA ライブラリの構成概要

FAA ライブラリのファイル構成の概要について以下に示します。

表 2-4 FAA ライブラリ構成概要

タイトル	構成ファイル	説明
RL78/G24 Common FAA Module	<Config_FAA>_common.c <Config_FAA>_common.h	転送処理、FAA 制御用共通関数を定義。 転送処理(R_Config_FAA_Create)は、SC が生成する周辺機能の初期化関数（R_Systeminit）内で実行されるため、ユーザプログラムでコールする必要はありません。
	<Config_FAA>_common.inc	FAA 用の SFR を定義
Custom FAA Library	<Config_FAA>_src.dsp	FAA ソース・ファイルのテンプレート
その他	<Config_FAA>_XXX.c / asm / s <Config_FAA>_XXX.h /inc <Config_FAA>_src.dsp	各種機能の FAA ライブラリ 各ライブラリのドキュメントを参照してください。

- <Config_FAA>は、手順 9 で設定したコンフィグレーション名です。
- XXX は、各ライブラリにより異なります。
- 各種機能の FAA ライブラリおよびテンプレート（Custom FAA Library）で提供する FAA ソース・ファイル（.dsp）では、コードセクション名を FAACODE,データセクション名を FAADATA で定義しています。
- 構造化アセンブラ DSPASM V1.50.0 以前を使用時、Custom FAA Library を選択した場合、テンプレートにユーザコードおよびデータを追記してください。テンプレートのままでビルドをするとエラーとなります。

2.4 ビルド・ツールのオプション設定

ビルドの前に、FAA プログラムのビルドに必要なビルド・ツールのオプションの設定をします。一部のオプションは、2.3.1 FAA コンポーネントの追加 で、スマート・コンフィグレータ（SC）が設定します。表 2-5 の「SC による設定」で「設定しない」と記載されているオプションを手動で設定してください。また、本章で説明のないビルド・ツールのオプションについては、必要に応じて適宜設定してください。

ビルド・ツールのプロパティ表示方法：
プロジェクト・ツリーにおいてプロジェクトを選択したのち、「プロジェクト」メニュー→「C/C++ Project Settings」を選択、またはコンテキストメニュー→「C/C++ Project Settings」を選択

ダイアログのクローズ方法：
オプションの変更内容を有効にするため、ダイアログの「適用して閉じる」をクリック

表 2-5 に FAA プログラムのビルドに必要なビルド・ツールのオプションを示します。

表 2-5 ビルド・ツールの設定オプション

ツール名	大項目	小項目	設定内容	SC による設定
FAA Assembler	プリプロセス処理	マクロを識別する方法 (-macro_exact)	exact	設定する
	コード生成	DSP ソース内で定義したセクション名 (-dsp_section)	FAACODER,FAADATAR	設定する
		ROM から RAM へマッピングするセクション (-rom)	FAACODE=FAACODER,FAADATA=FAADATAR	設定する
Linker	セクション	デバイス・ファイルの情報からセクションを自動的に配置する (-auto_section_layout)	チェックする または チェックを外す	設定しない
		セクション (-start)	FAACODE,FAADATA/XXXX XXXX（0x なしの 16 進数）はコード・フラッシュ・メモリの 0xD8 番地以降の偶数番地を指定	設定しない
		FAA メモリ領域を自動的に割り当てる (-dsp_memory_area)	はい または、 はい（FAA メモリ領域をまたいでセクションを自動配置する）注 2	設定する注 1
	出力	ROM から RAM へマップするセクション (-rom)	FAACODE=FAACODER FAADATA=FAADATAR	設定する

注 1. SC は「はい」を設定します。

注 2. ユーザプログラム（CPU のプログラム）で使用する RAM サイズが 2304 バイト（RAM 上の FAA コード領域より前のユーザ RAM 領域）より大きい場合は、手動で「はい（FAA メモリ領域をまたいでセクションを自動配置する）」に設定してください。
また、「はい（FAA メモリ領域をまたいでセクションを自動配置する）」を指定した場合、「デバイス・ファイルの情報からセクションを自動的に配置する」をチェックしてください。

2.4.1　FAA Assembler オプション

図 2-29　FAA Assembler オプションープリプロセス処理

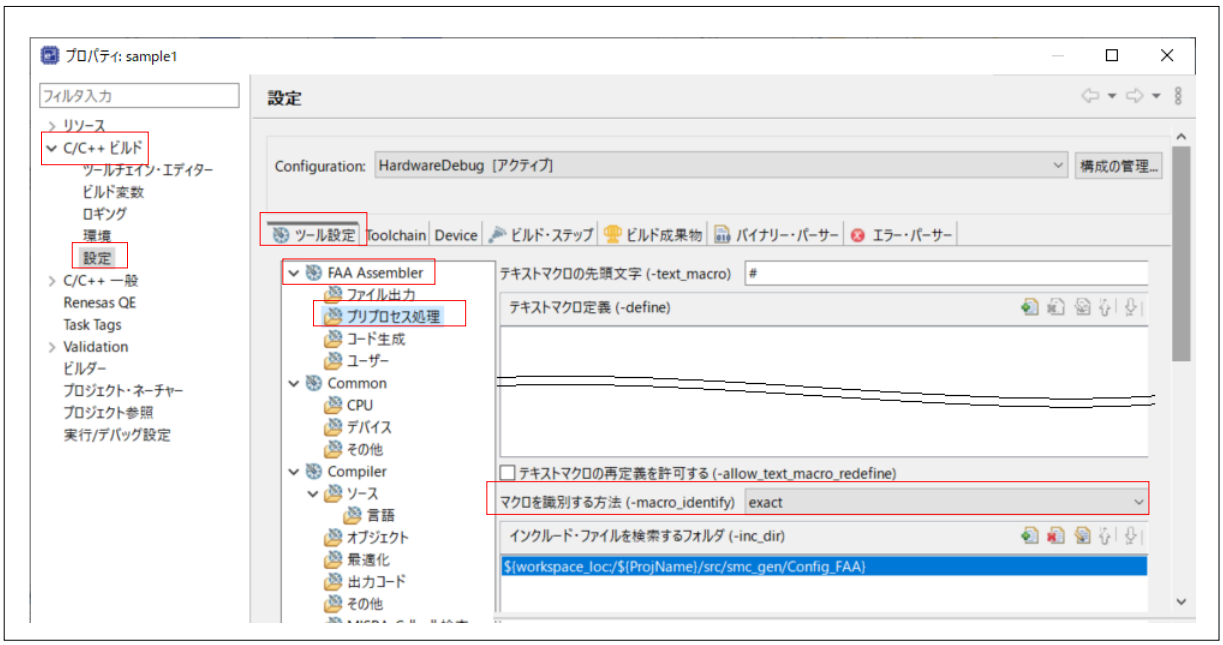


表 2-6　FAA Assembler オプションープリプロセス処理　設定内容の概要

大項目	小項目	概要
プリプロセス処理	マクロを識別する方法 (-macro_exact)	「exact」を設定します。 FAA ソース・ファイル内のテキスト・マクロを置換時に、トークン単位で行います。Exact を指定しないと、置き換え対象となる識別子が、他の識別子の一部に含まれている場合でも置き換えが行われます。

図 2-30　FAA Assembler オプション－コード生成

表 2-7　FAA Assembler オプション－コード生成　設定内容の概要

大項目	小項目	概要
コード生成	DSP ソース内で定義したセクション名 (-dsp_section)	FAACODER,FAADATAR を設定します。 スマート・コンフィグレータ（SC）で生成する FAA プログラムのファイル（ライブラリ／テンプレート）では、コードセクション名は FAACODE、データセクション名は FAADATA で定義していますが、RAM 領域に再配置するセクション名 FAACODER,FAADATAR を指定します。
	ROM から RAM へマッピングするセクション (-rom)	FAACODE＝FAACODER、FAADATA=FAADATAR を設定します。 コード・フラッシュ・メモリに配置されている FAA プログラムとデータの定義シンボルを、内蔵 RAM（インストラクション・コード・メモリ、データ・メモリ）に再配置します。再配置をしないと、FAA のプログラムとデータのシンボルのアドレスがコード・フラッシュ・メモリ領域のままとなり、デバッグ時にシンボル情報を正しく扱えません。 左辺は、コード・フラッシュ・メモリに配置された FAA のプログラムとデータのセクションを指定します。右辺は転送先の RAM のセクションを指定します。 FAA のプログラムとデータをインストラクション・コード・メモリおよびデータ・メモリへ転送する処理（SC 生成の Config_FAA_Common_c 内）では、転送先の RAM セクションを FAACODER,FAADATAR としたコードとなっているため、右辺は FAACODER、FAADATAR を指定します。

2.4.2　Linker オプション

図 2-31　Linker オプション－セクション

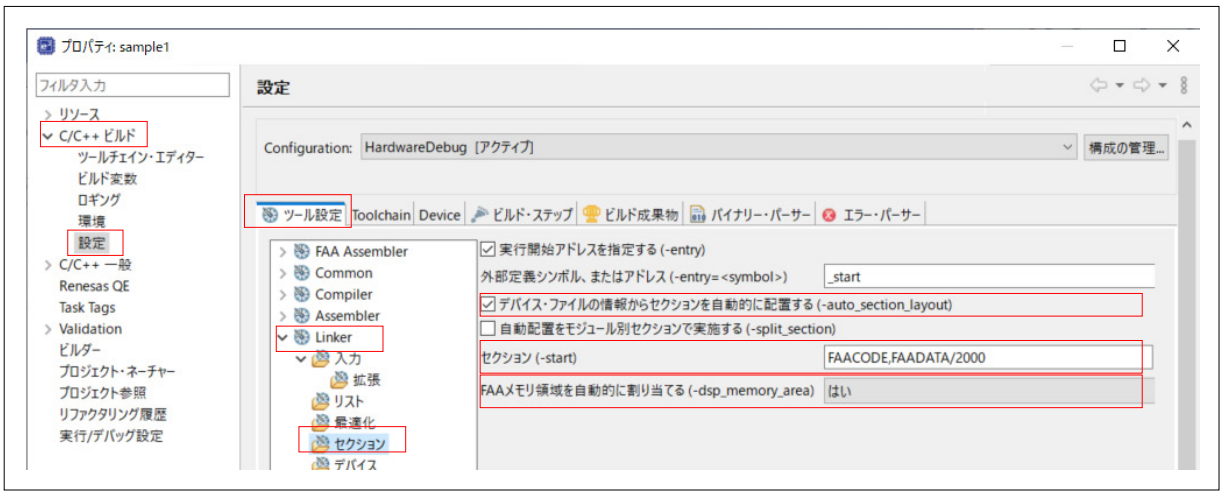


表 2-8　Linker オプション－セクション　設定内容の概要

大項目	小項目	概要
セクション	デバイス・ファイルの情報からセクションを自動的に配置する (-auto_section_layout)	チェックを付けます。 デバイス・ファイルの情報からセクションが自動的に配置されます。チェックしない場合、「セクション(-start)」でプログラムで使用する各セクションのアドレスを指定する必要があります。
	セクション (-start)	FAACODE,FAADATA/アドレス を設定します。 FAA のプログラムとデータを格納するコード・フラッシュ・メモリのアドレスを指定します。スマート・コンフィグレータ（SC）が生成する FAA プログラムのファイル（ライブラリ／テンプレート）では、FAA のプログラムのセクション名は FAACODE、データのセクション名は FAADATA で定義されているため、FAACODE、FAADATA を指定します。 また、FAA のプログラムとデータをインストラクション・コード・メモリおよびデータ・メモリへ転送する処理（SC 生成の Config_FAA_Common_c 内）は、2 バイト単位で転送を行っています。そのため FAACODE と FAADATA は、2 バイト境界にアラインする必要があります。D8H 以降の偶数番地を指定します。（例では 2000H 番地以降に配置）
	FAA メモリ領域を自動的に割り当てる (-dsp_memory_area)	「はい」を設定します。 内蔵 RAM の FAA 専用領域を確保します。FAA インストラクション・コード・メモリ領域（FD800H-FE7FFH）およびデータ・メモリ領域（FE800H-FEFFFH）に、CPU プログラムの変数を配置しないようにします。

図 2-32　Linker オプション－出力

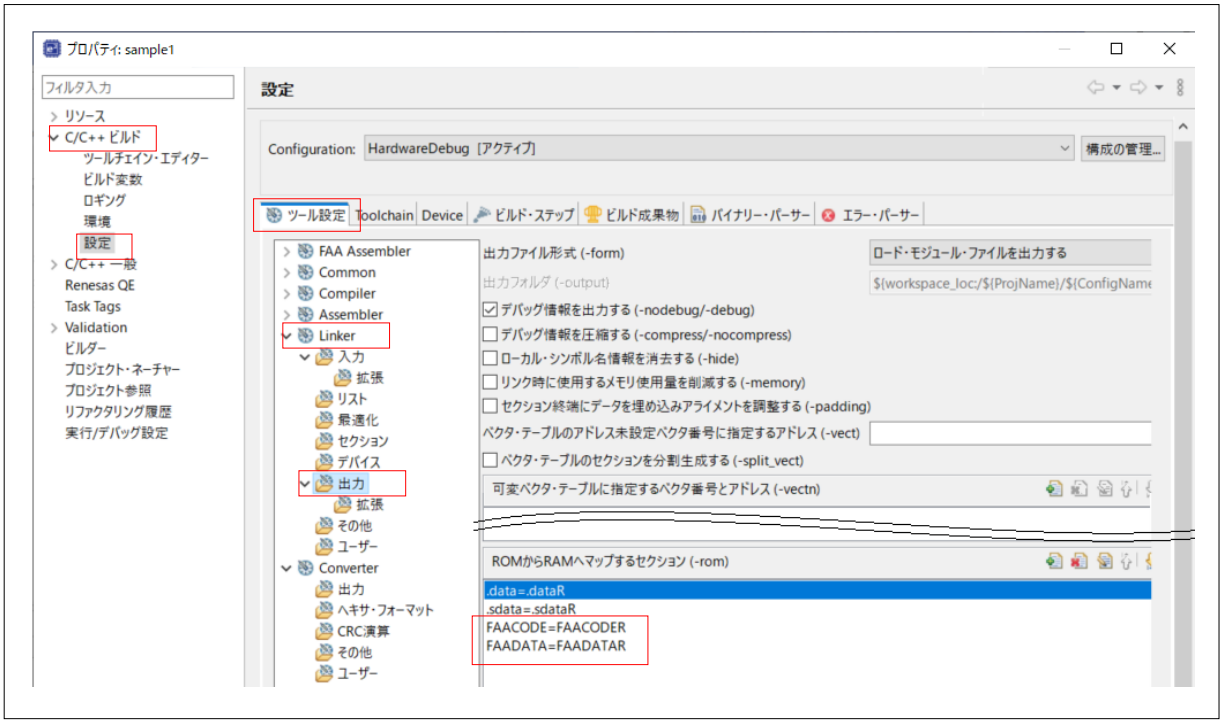
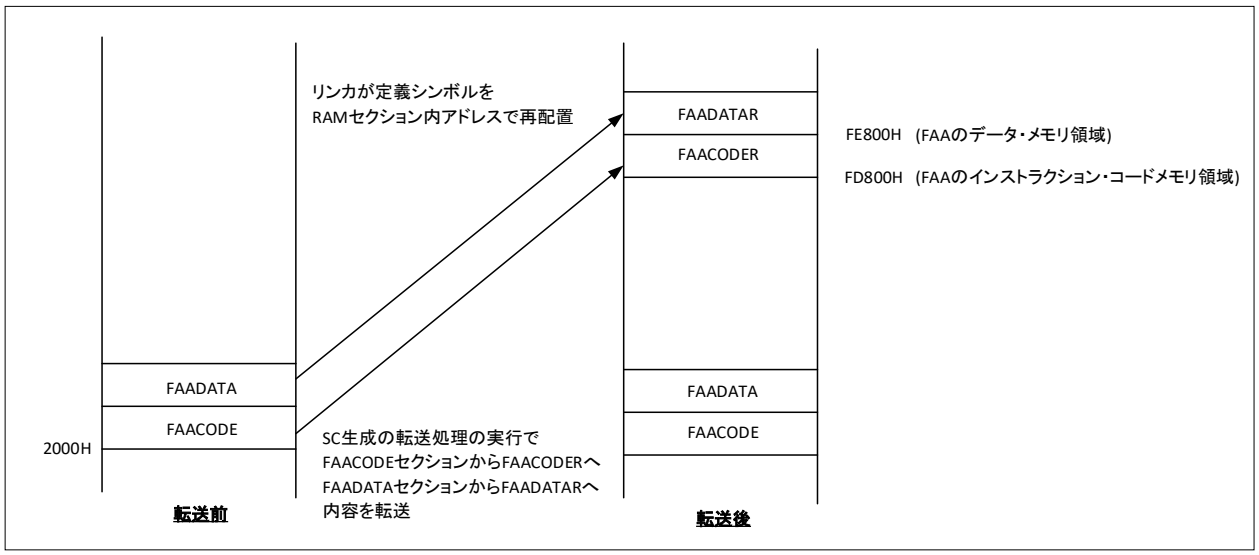


表 2-9　Linker オプション－出力　設定内容の概要

大項目	小項目	概要
出力	ROM から RAM へマップするセクション (-rom)	FAACODE＝FAACODER、FAADATA=FAADATAR を設定します。 コード・フラッシュ・メモリに配置されている FAA プログラムとデータの定義シンボルを、内蔵 RAM（インストラクション・コード・メモリ、データ・メモリ）に再配置します。再配置をしないと、FAA のプログラムとデータのシンボルのアドレスがコード・フラッシュ・メモリ領域のままとなり、デバッグ時にシンボル情報を正しく扱えません。 左辺は、コード・フラッシュ・メモリに配置された FAA のプログラムとデータのセクションを指定します。右辺は転送先の RAM のセクションを指定します。 FAA のプログラムとデータをインストラクション・コード・メモリおよびデータ・メモリへ転送する処理（SC 生成の Config_FAA_Common_c 内）では、転送先の RAM セクションを FAACODER,FAADATAR としたコードとなっているため、右辺は FAACODER、FAADATAR を指定します。

図 2-33　転送処理前後の配置イメージ例



2.4.3　プログラムのビルド

FAA プログラムのビルドに必要なビルド・ツールのオプションを設定後、ビルドを行います。ビルドの実行にはいくつかの方法があります。ここでは 2 つの方法を示します。

- e2 studio の「プロジェクト」メニュー→「プロジェクトのビルド」を選択する。（図 2-34）
- e2 studio のツールバー上のビルドボタンをクリックする。（図 2-35）

図 2-34　「プロジェクト」メニュー

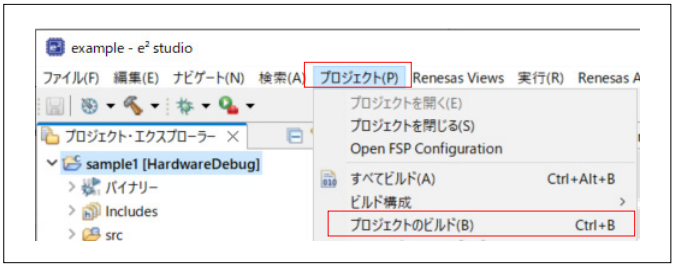
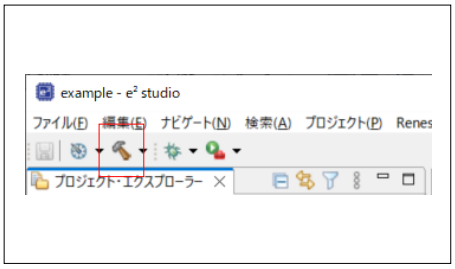
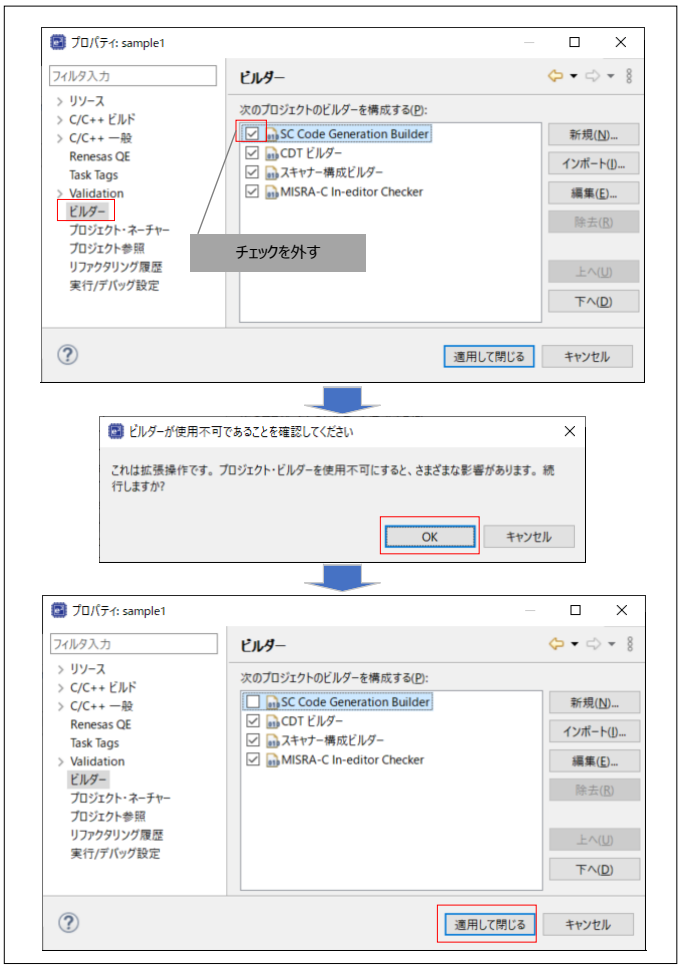


図 2-35　ツールバー



備考. スマート・コンフィグレータ（SC）は、SC の GUI 上の設定とファイル内のコードの不一致を防ぐために、ビルド前またはプロジェクトのクリーン後に自動でコード生成を行います。この機能を停止するには、プロジェクトのプロパティで「SC Code Generation Builder」のチェックを外してください。

図 2-36　SC Code Generation Builder



2.5　デバッグ・ツールのオプション設定

実行可能なオブジェクトを RL78/G24 Fast Prototyping Board へダウンロードする前に、FAA プログラムのデバッグに必要なデバッグ・ツールのオプションの設定をします。一部のオプションは、2.3.1 FAA コンポーネントの追加 で、スマート・コンフィグレータ（SC）が設定します。表 2-10 の「SC による設定」で「設定しない」と記載されているオプションを手動で設定してください。また、本章で説明のないデバッグ・ツールのオプションについては、必要に応じて適宜設定してください。

必要なオプションを設定後、オブジェクトのダウンロードを行います。

デバッグ・ツールの構成表示方法：

1. プロジェクト・ツリーにおいてプロジェクトを選択したのち、「実行」メニュー→「デバッグの構成」を選択、またはコンテキストメニュー→［デバッグ］－「デバッグの構成」を選択
2. 「デバッグ構成」ダイアログで、「Renesas GDB Hardware Debugging」以下の「プロジェクト名 HardwareDebug」をクリック

ダイアログのクローズ方法：

オプションの変更内容を有効にするため、ダイアログの「適用」をクリック後に「閉じる」をクリック

図 2-37　デバッグ構成

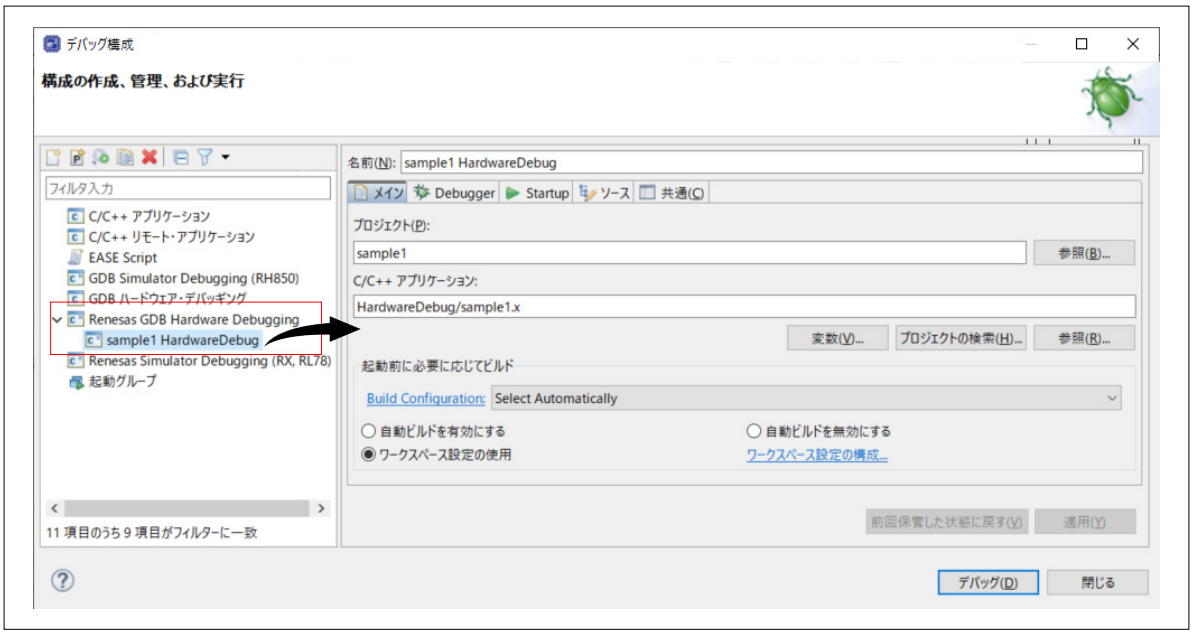


表 2-10 に FAA プログラムのデバッグに必要なデバッグ・ツールのオプションを示します。

表 2-10　デバッグ・ツールの設定オプション

タブ	下位のタブ	項目	設定内容	SC による設定
Debugger	マルチコア設定	Core State – FAA	Enabled	設定する[注1]
Startup	－	イメージとシンボルをロード	ファイル名： “プロジェクト名”GreenDSP_Core.x ロード・タイプ：　シンボルのみ オフセット：　0 接続時：　Yes コア：　FAA	設定しない

注 1. e2 studio のデバッグの構成の設定と SC の「システム」の設定で選択しているデバッガのが一致している場合のみ

2.5.1　Debugger オプション

図 2-38　Debugger オプション

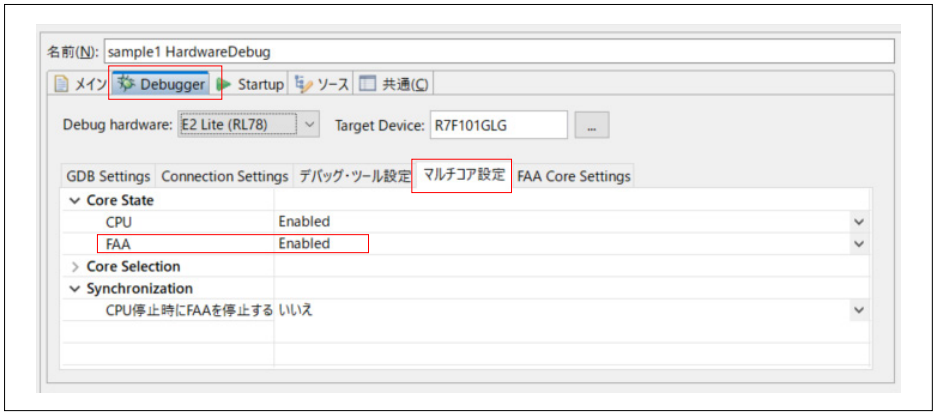


表 2-11　Debugger オプション　設定内容の概要

下位のタブ	項目	概要
マルチコア設定	Core State - FAA	「Enabled」を設定します。 デバッガで FAA プログラムのデバッグを可能にします。

2.5.2　Startup オプション

図 2-39　Startup オプション

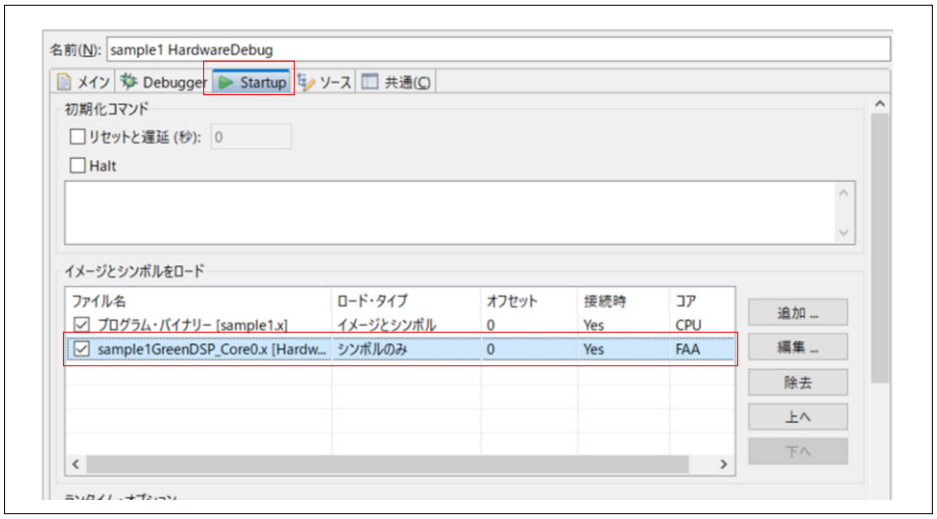


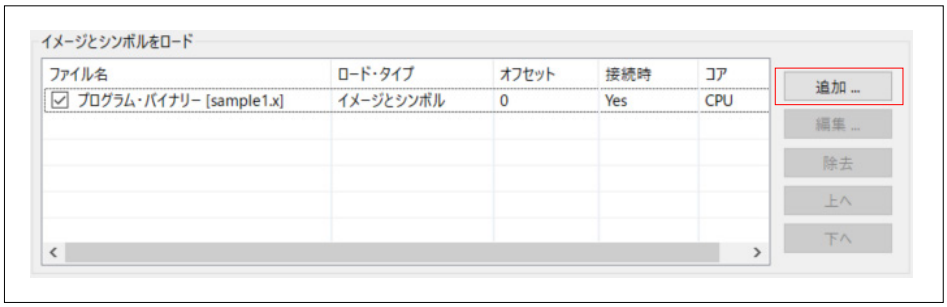
表 2-12　Startup オプション　設定内容の概要

下位のタブ	項目	概要
―	イメージとシンボルをロード	FAA プログラムのバイナリ・モジュールを指定します。 ファイル名：　“プロジェクト名”GreenDSP_Core.x ロード・タイプ：　シンボルのみ オフセット：　0 接続時：　Yes コア：　FAA FAA プログラムのバイナリ・モジュールのシンボルをダウンロードし、FAA プログラムのソース・デバッグを可能にします。

設定方法：
（プロジェクトのビルド完了後に行います。）

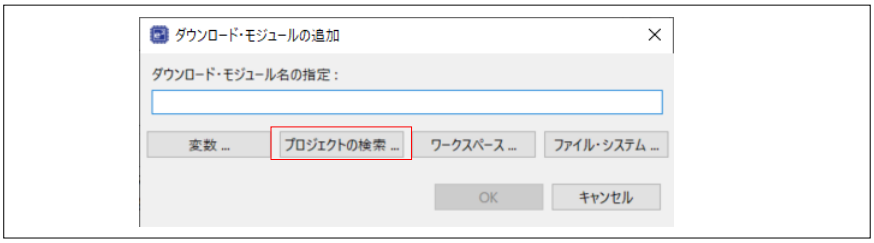
1. 「追加」をクリックします。

図 2-40　モジュールの追加（1/6）



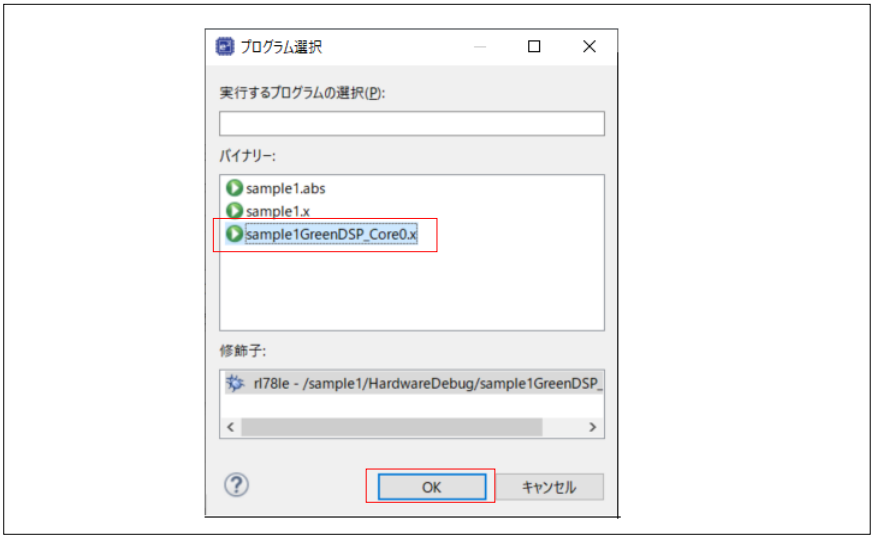
2. 「ダウンロードモジュールの追加」ダイアログで、「プロジェクトの検索」をクリックします。

図 2-41　モジュールの追加（2/6）



3. 「”プロジェクト名”GreenDSP_Core0.x」をクリックし、「OK」をクリックします。

図 2-42　モジュールの追加（3/6）

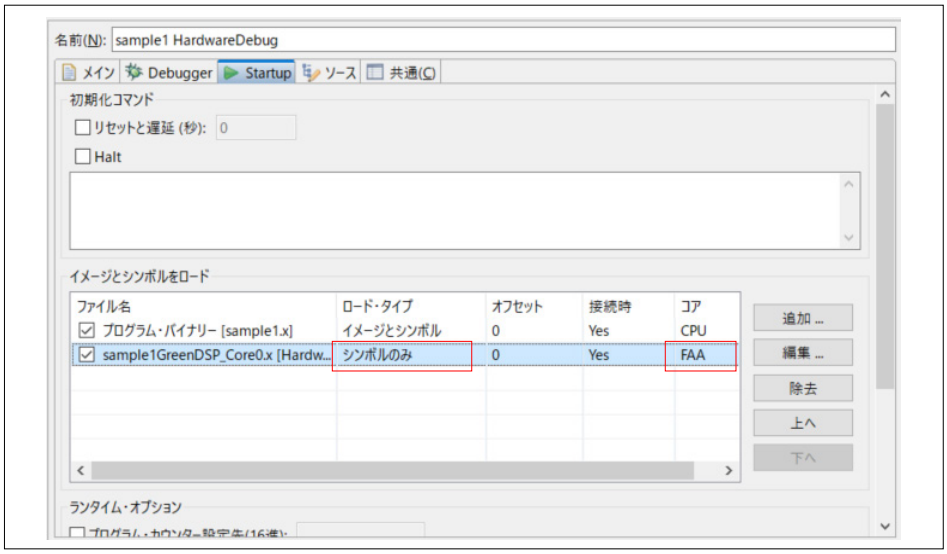


4.　「OK」をクリックします。

図 2-43　モジュールの追加（4/6）

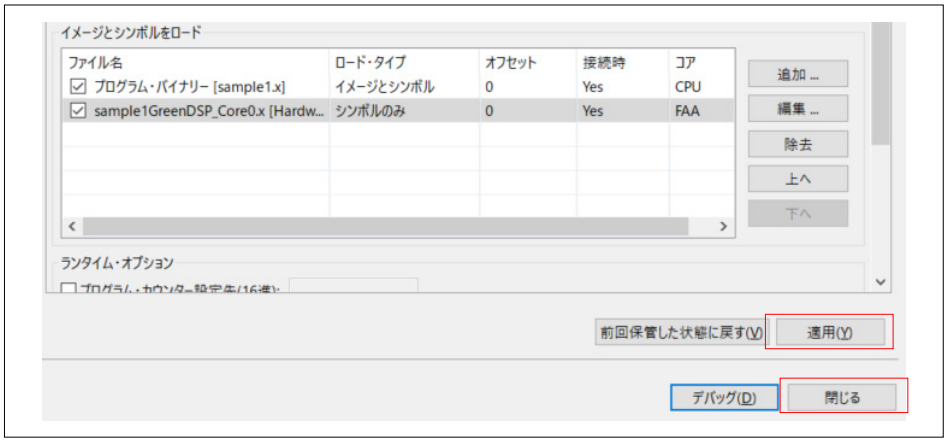
5.　「ロード・タイプ」を「シンボルのみ」に、「コア」を「FAA」に設定します。

図 2-44　モジュールの追加（5/6）



6.　「適用」をクリック後、「閉じる」をクリックします。

図 2-45　モジュールの追加（6/6）



2.5.3　プログラムのダウンロード

FAA プログラムのデバッグに必要なデバッグ・ツールのオプションを設定後、PC と RL78/G24 Fast Prototyping Board を接続し、オブジェクトをダウンロードします。ダウンロードにはいくつかの方法があります。ここでは 2 つの方法を示します。

- e2 studio の「実行」メニュー→「デバッグ」を選択する。（図 2-46）
- e2 studio のツールバー上のデバッグボタンをクリックする。（図 2-47）

注意 1.　デバッグ・ツールにエミュレータを使用する場合、ダウンロード前にデバッグ構成ダイアログでボードへの電源供給を確認してください。
・「Debugger」タブ→「Connection Settings」タブ→「ターゲット・ボードとの接続」

注意 2.　オブジェクトをダウンロードしただけでは、FAA プログラムはインストラクション・コード・メモリに配置されません。CPU プログラムで、FAA のプログラムとデータをコード・フラッシュ・メモリからインストラクション・コード・メモリ、データ・メモリへ転送する必要があります。スマート・コンフィグレータ（SC）の FAA コンポーネントを使用することで、転送処理の関数（R_Config_FAA_Create）が生成され main 関数実行前の初期化ルーチン（R_Systeminit）で転送が行われます。

図 2-46　「デバッグ」メニュー

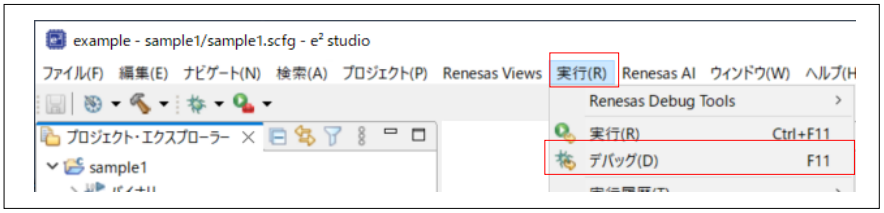
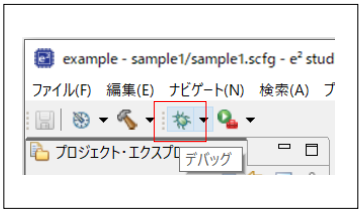


図 2-47　デバッグ・ツールバー



2.6　FAA プログラムのデバッグ

2.6.1　デバッグ対象

RL78/G24 のプログラムをデバッグする場合、デバッグ対象を CPU とするか、FAA とするか選択します。選択は、デバッグビューで行います。

- CPU の選択方法：(CPU) [core: 0] 以下に表示されるソース名を選択する。（図 2-48）
- FAA の選択方法：(FAA) [core: 1] 以下に表示されるソース名を選択する。（図 2-49）

図 2-48　デバッグ対象に CPU を選択

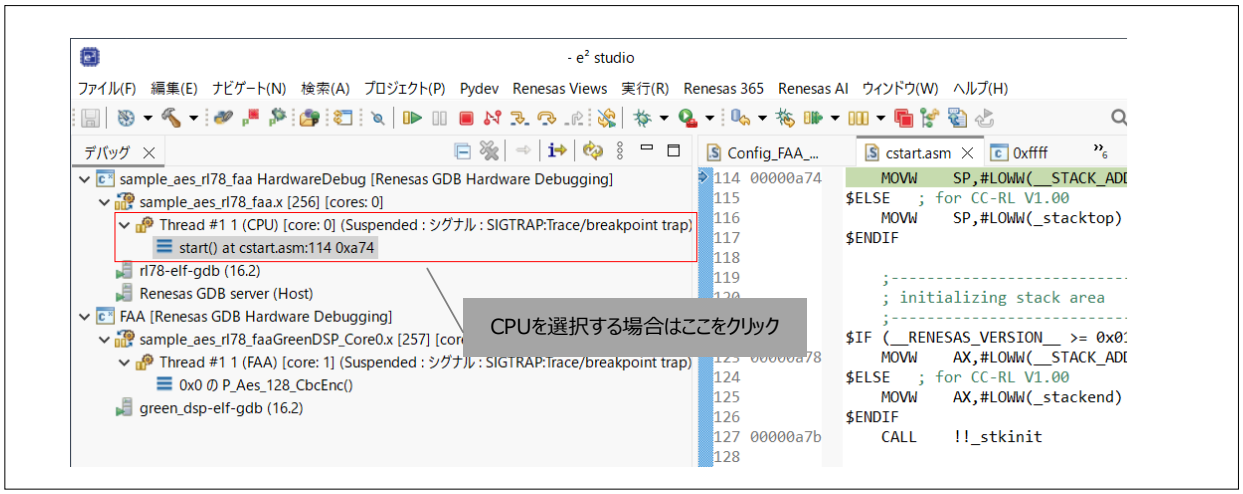
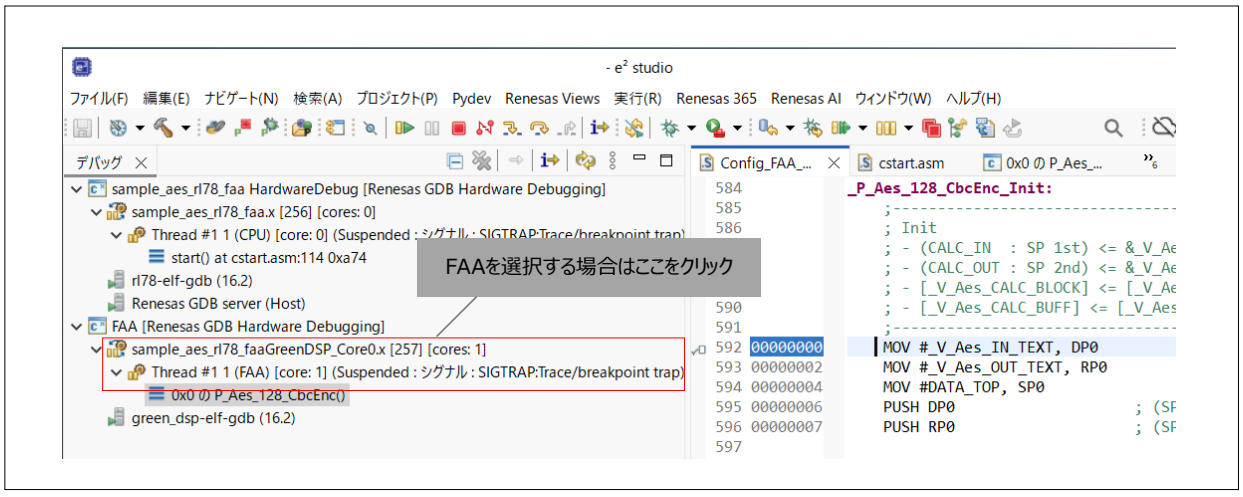


図 2-49　デバック対象に FAA を選択



デバッグ対象のソース・ファイルのみにアドレスカラムにアドレス情報が表示され、ソース・レベルでステップ実行等のデバッグ操作を行うことができます。

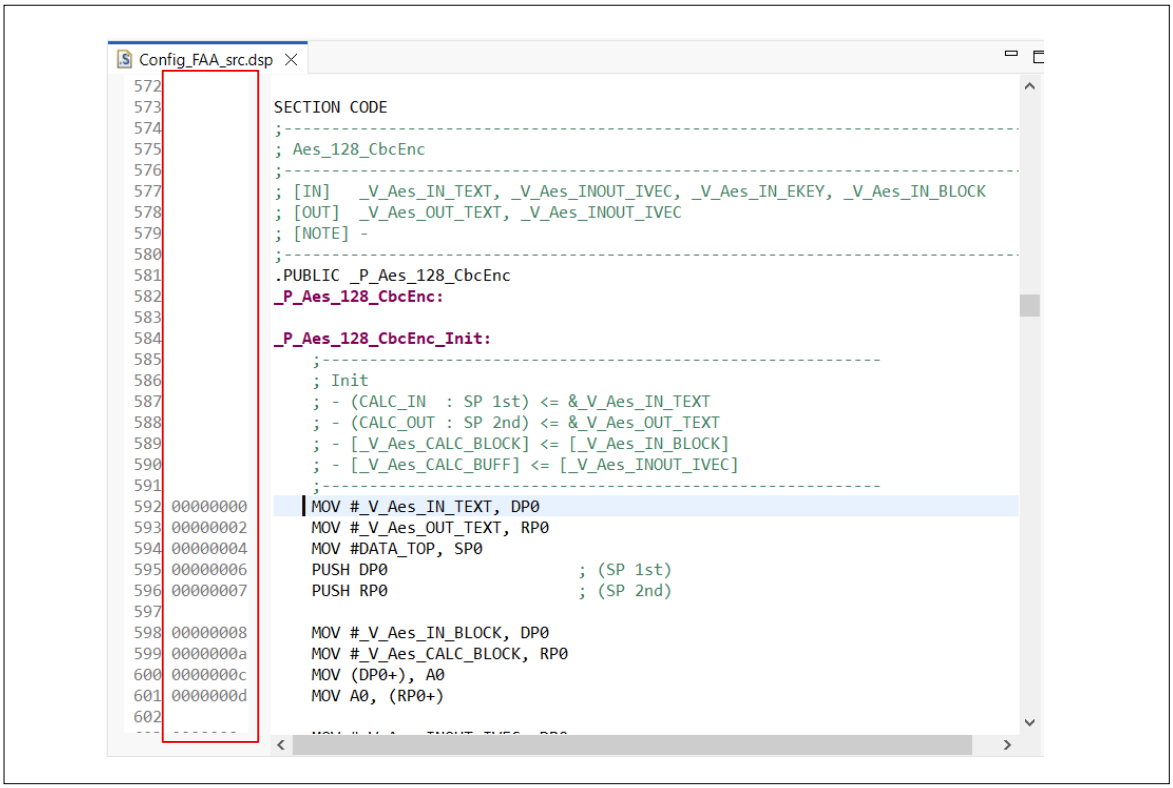
デバッグ対象は、プログラムの実行中にも変更が可能です。

2.6.2　ソース表示

デバッグ対象に FAA を選択後、FAA プログラムを記述したソース・ファイル（.dsp）をエディタで表示すると、アドレスカラムにアドレス情報が表示され、FAA ソース・レベルでステップ実行等のデバッグ操作を行うことができます。

アドレスカラムに表示されるアドレスは、FAA のインストラクション・コード・メモリ空間のアドレスです。デバッグ対象が CPU の場合、FAA ソース・ファイルのアドレスカラムにアドレスは表示されません。

図 2-50　FAA ソース・ファイル表示



2.6.3　実行／停止

デバッグ対象に FAA を選択時、FAA ソース・デバッグが可能となります。FAA プログラムの実行／停止にはいくつかの方法があります。ここでは 2 つの方法を示します。

- e2 studio の「実行」メニュー→「再開」／「中断」を選択する。（図 2-51）
- e2 studio のデバッグ・ツールバー上の実行／停止ボタンをクリックする。（図 2-52）

図 2-51　「デバッグ」メニュー

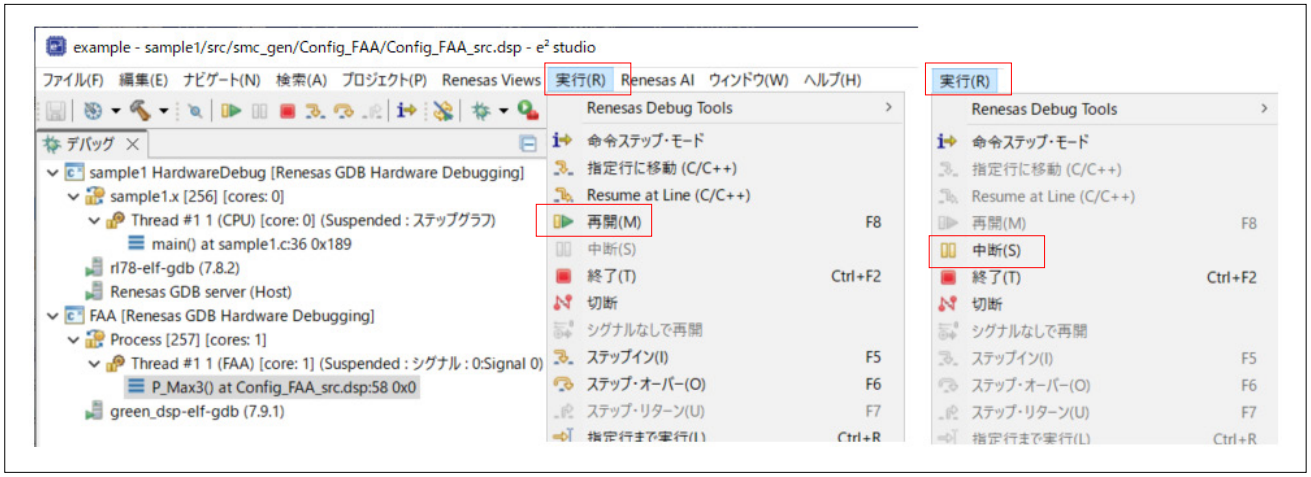


図 2-52　デバッグ・ツールバー

FAA プログラムの制御は次のとおりです。

✓　FAA のステータスが次の場合はプログラム実行を開始できません。また、ステップ実行等、他のデバッグ操作もできません。

・ FAA にクロックが供給されていない（FAAEN ビット= 0）

・ FAA 動作禁止（ENB ビット= 0）

CPU のプログラムで FAA を動作許可状態（FAAEN＝1、ENB＝１）にしてください。

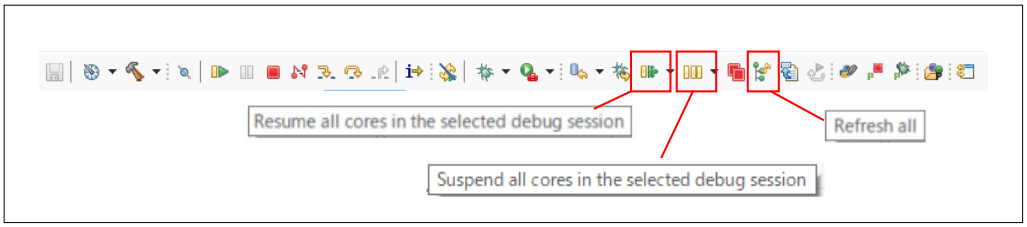
✓　FAA がデバッグ対象時にプログラムの実行／停止操作を行った場合、FAA プログラムのみを実行／停止します。CPU プログラムは同期して実行／停止しません。ただし、デバッグ・ツールオプションの「Debugger」タブ→「マルチコア設定」タブ→「Synchronization」カテゴリの「CPU 停止時に FAA を停止する」で「はい」を設定した場合、CPU がデバッグ対象時に CPU プログラムを停止する操作を行うと FAA プログラムも停止します。

✓　ステップ実行では、FAA のみステップ実行を行います。

✓　リセット操作では、FAA に対してソフトウェア・リセットを行います。MCU（CPU＋周辺機能）はリセットされません。デバッグ対象が CPU 時にリセット操作を行うと、MCU をリセットし、FAA はリセットされません。

- ✓ CPU が WIND レジスタを操作するプログラムを実行中は、FAA に対してデバッグ操作を行わないでください。FAA のデバッグ操作によりデバッガが一時的に WIND レジスタを書き換えるため、CPU で実行中のプログラムの動作が不正となる場合があります。
- ✓ デバッグ中に CPU または FAA プログラムのソース・ファイルを変更してビルドした場合、デバッグ中に変更したプログラムをダウンロードしても正しくダウンロードできません。プログラムを変更した場合は一度デバッグ接続を切断し、再接続してください。
- ✓ CPU と FAA をデバッグするプロジェクトでは、ツールバーの「Resume All」 と「Suspend All」ボタンは動作しません。
- ✓ FAA が CPU プログラムによって開始または停止された場合、デバッグビュー上の FAA の状態は更新されません。 デバッグビューで FAA を選択しても、FAA の情報 （ステータスやレジスター ビューなど） は更新されません。 ツールバーの「Refresh All」 ボタンを押すと、各ビューが更新されます。

図 2-53　デバッグ・ツールバー　－ Resume all, Suspend all, Refresh all



- ✓ デバッグの終了時は、「Terminate all」ボタンを押すと、CPU 側と FAA 側の両方のデバッグ・セッションが終了できます。

図 2-54　デバッグ・ツールバー　－ Terminate all

2.6.4　ブレークポイント

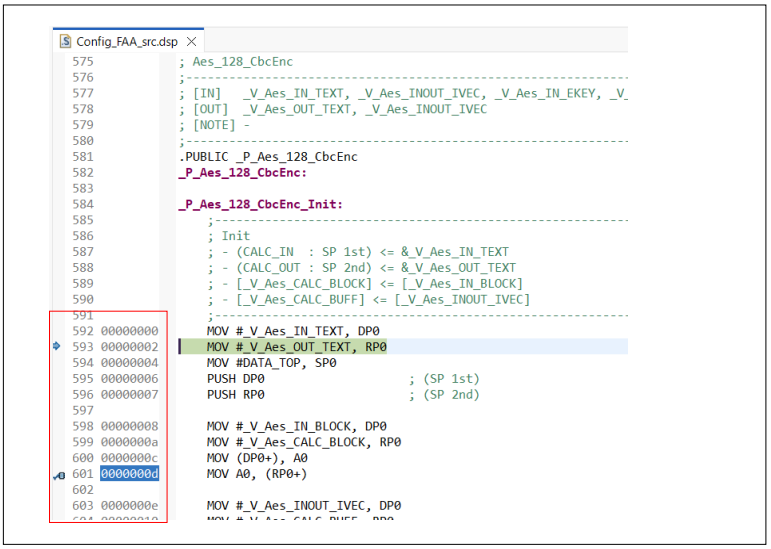
デバッグ対象に FAA を選択後、エディタに FAA ソースを表示し、ブレークポイントを設定したい行のソース欄外をダブルクリックすることでブレークポイントを設定します。設定済みのアイコンをダブルクリックするとブレークポイントが解除されます。

FAA プログラムのブレークポイントの制御は次のとおりです。

- ✓　ハードウエア・ブレーク　最大 4 点設定可能です。（実行後ブレーク）
- ✓　FAA がハードウエア・ブレークを検出して停止した場合、CPU は同期して停止しません。

注. FAA 領域に転送する前の FAA プログラムとデータが含まれる領域（コード・フラッシュ・メモリ）にソフトウェアブレークポイントは設定しないでください。設定した場合、FAA のコードを転送する前にブレークポイントを削除してください。

図 2-55　FAA プログラム　ブレークポイントの設定



2.6.5　メモリの表示

FAA がデバッグ対象時、メモリービューに FAA のインストラクション・コード・メモリとデータ・メモリを表示します。

FAA のメモリ表示の制御は次のとおりです。

✓　FAA 領域を表示するには、表示アドレスとして次のように指定します。（図 2-56, 図 2-57）

- FAA インストラクション・コード・メモリ領域：
  「FAA インストラクション・コード・メモリ領域のアドレス+0x10000000」
- FAA データ・メモリ領域：
  「FAA データ・メモリ領域のアドレス+0」

✓　デバッグ対象が CPU の場合、メモリービューの表示は CPU のメモリが表示されます。

✓　FAA プログラム実行中の表示更新はできません。

✓　FAA のステータスが次の場合は、表示内容は不定となります。

- FAA にクロックが供給されていない（FAAEN ビット= 0）
- FAA 動作禁止（ENB ビット= 0）

図 2-56　メモリービュー（FAA インストラクション・コード・メモリ領域）

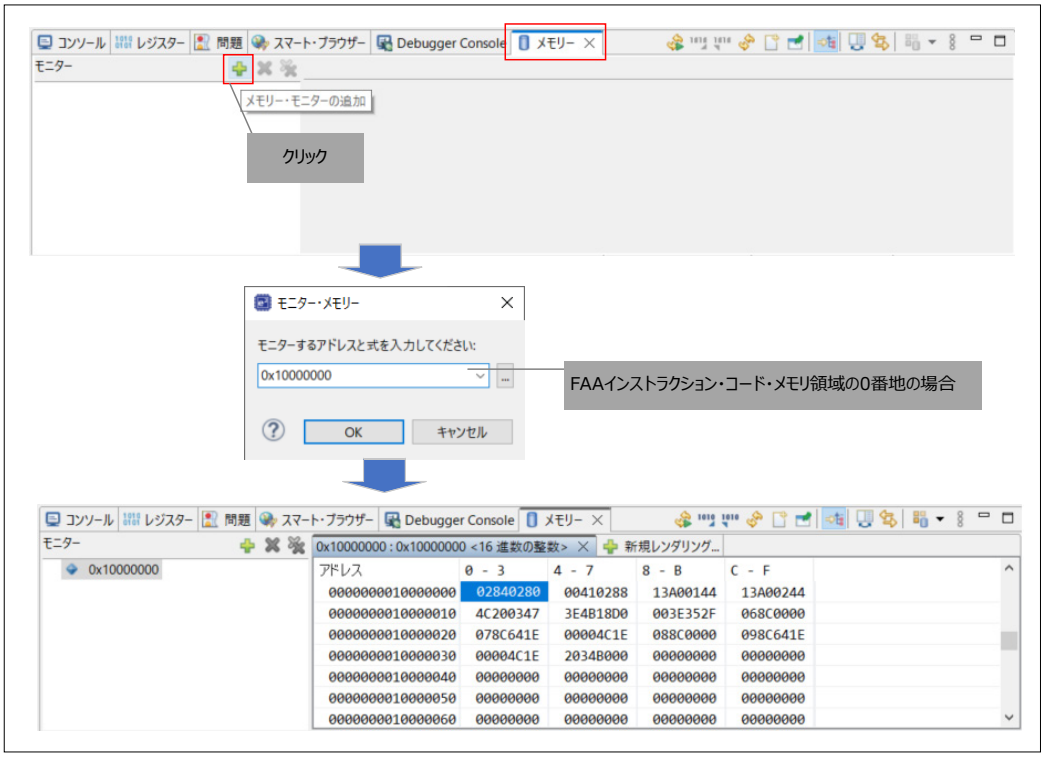
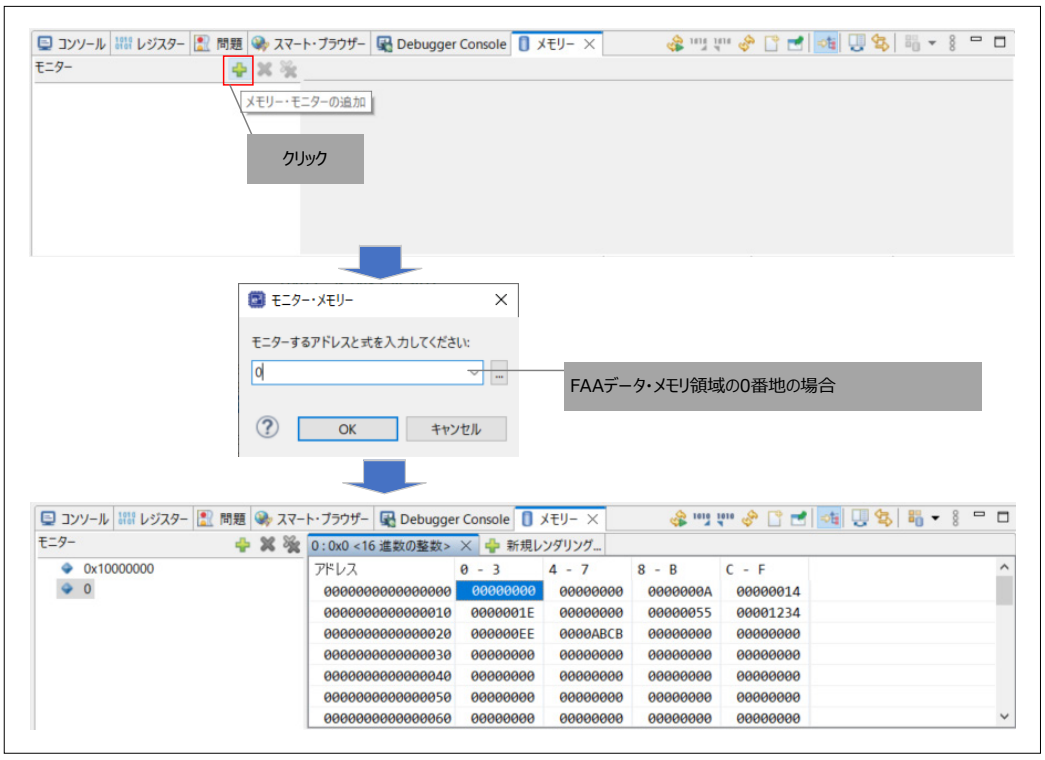
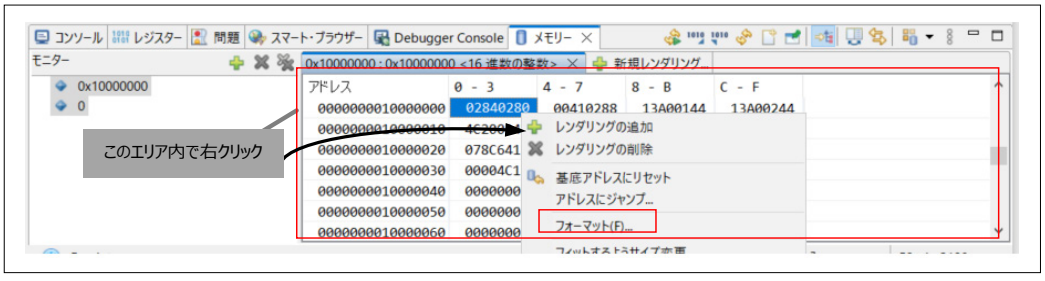


図 2-57　メモリービュー（FAA データ・メモリ領域）



備考. メモリービューの表示フォーマットは、コンテキストメニューの「フォーマット」メニューで変更できます。

図 2-58　メモリービュー（表示フォーマット変更）



2.6.6　シンボル（ラベル）の表示

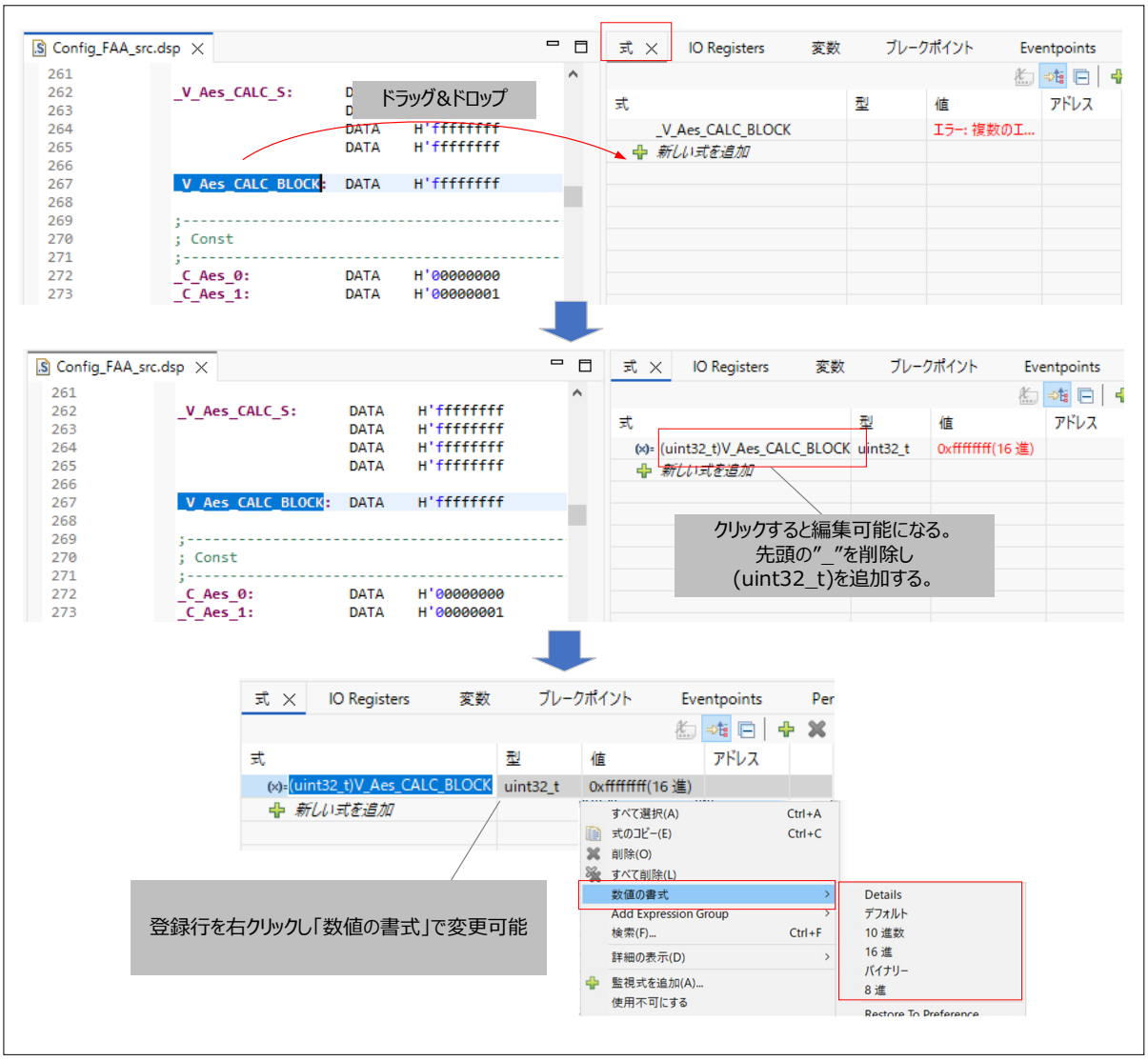
FAA がデバッグ対象時、FAA プログラム内で定義したシンボル（ラベル）を式ビューに表示できます。

FAA の式ビュー表示の制御は次のとおりです。

- ✓　式ビューにシンボル／ラベルを登録時、シンボル（ラベル）名の先頭の“_”を削除してください。
  また、シンボル（ラベル）名の前に”(uint32_t)”を追加してください。
- ✓　アドレス欄に表示するアドレスは、FAA 空間のアドレスです。
- ✓　デバッグ対象が CPU の場合、表示内容は不定となります。
- ✓　FAA のステータスが次の場合は、表示内容は不定となります。
  - ・ FAA にクロックが供給されていない（FAAEN ビット= 0）
  - ・ FAA 動作禁止（ENB ビット= 0）

備考：シンボルを CPU プログラムから参照可能にするには、FAA プログラム内で”_”で始まる名前で定義し、かつ public 宣言をする必要があります。

図 2-59　式ビュー



2.6.7　レジスタ表示

FAA がデバッグ対象時、レジスタービューに演算パラメータ・レジスタ・セット、アドレス・ポインタ・セット、制御レジスタなどを表示します。

図 2-60　レジスタービュー

2.6.8　IO Registers(SFR)表示

FAA がデバッグ対象時、IO Registers ビューには FAA がアクセス可能な SFR（Special Function Register）のみを表示します。FAA がアクセス可能な SFR は、2 種類あります。

- FAA の SFR
  アドレス・バス選択レジスタ（ADBSEL）の設定に影響を受けず、FAA バスを介してアクセス可能なレジスタ
- 周辺機能の SFR
  アドレス・バス選択レジスタ（ADBSEL）で FAA からのアクセスを選択時、FAA バスを介してアクセス可能なレジスタ

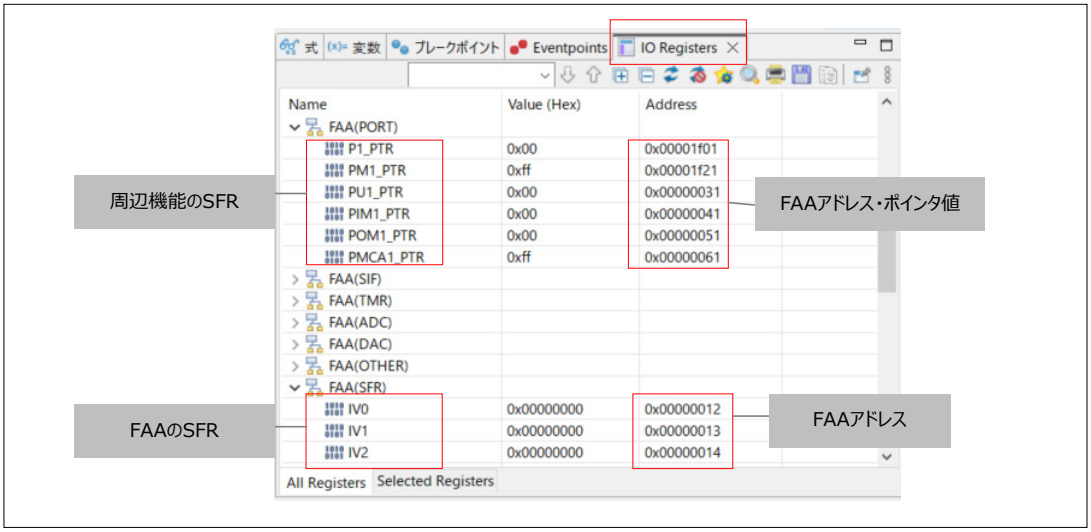
  参考：アクセス方法は 2 通りあります。
  ・ FAA アドレスによるアクセス
  ・ FAA・アドレス・ポインタ（FAAAP）を用いたアクセス

  アドレス・バス選択レジスタ（ADBSEL）およびアクセスについては、RL78/G24 ユーザーズマニュアル ハードウェア編（R01UH0961）を参照してください。

FAA の IO Register 表示の制御は次のとおりです。

- ✓ FAA の SFR のアドレス欄に表示されるアドレスは FAA アドレスとなります。
- ✓ アドレス・バス選択機能で FAA からのバスアクセス許可をすることでアクセス可能になる周辺機能の SFR は、レジスタ名の末尾に「_PTR」を追加した名前で表示します。
  アドレス欄に表示されるアドレスは、FAA・アドレス・ポインタ（FAAAP レジスタ）でアクセス時に FAAAP レジスタに設定する値です。

図 2-61　IO Registers ビュー



3. サンプルプロジェクト

3.1 サンプルコード仕様

3.1.1 仕様概要

本サンプルコードでは、16 ビット・タイマ KB30（TKB30）を使用し、2 つの PWM 出力を行います。PWM 出力は LED1 と LED2 に接続します。

CPU プログラムで TKB30 の初期設定を行い、TKB30 のタイマ割り込み（INTTKB00）の発生回数をカウントして一定周期（500ms）のタイミングを作り、一定周期で FAA の動作を開始します。

FAA プログラムでは、PWM 出力のデューティ比を変更し LED の輝度を制御します。デューティ比を変更後、動作を停止します。

表 3-1 使用する周辺機能と用途

周辺機能	用途
16 ビット・タイマ KB30（TKB30）	TKBO00 端子と TKBO01 端子から PWM 出力する
フレキシブル・アプリケーション・アクセラレータ（FAA）	TKBO00 端子と TKBO01 端子から出力する PWM 出力のデューティ比を変更する

図 3-1 PWM 出力の動作概要

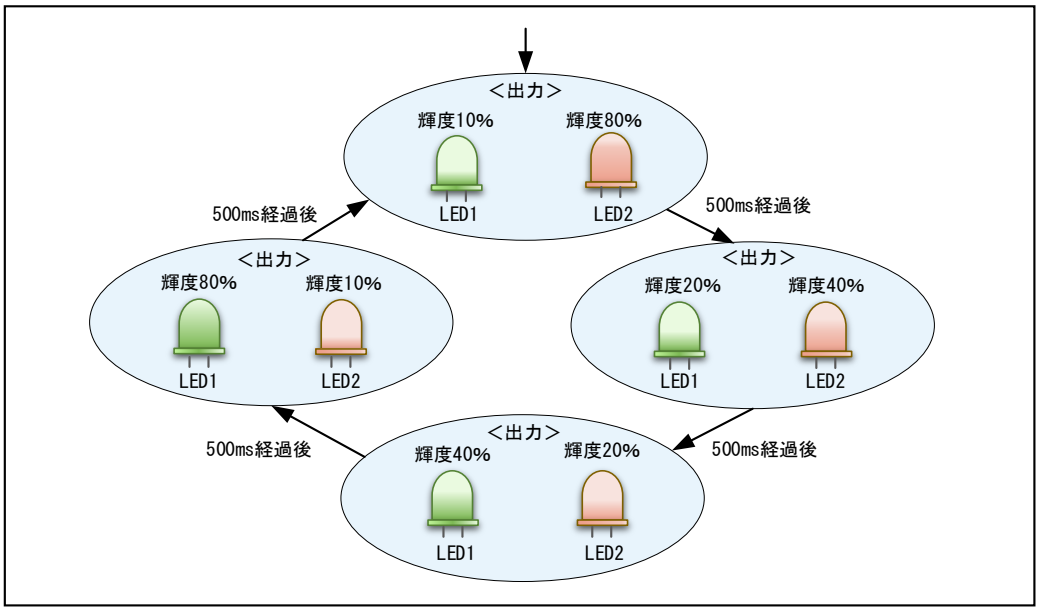


表 3-2 PWM 出力のデューティ比と LED 輝度の関係

デューティ比	輝度
10%	10%
20%	20%
40%	40%
80%	80%

3.1.2　動作概要

本サンプルコードでは、16 ビット・タイマ KB30（TKB30）を単体動作モード（TKBCR00 レジスタによる周期制御）で使用し、P12/TKBO00 と P13/TKBO01 から PWM 出力を行います。

TKB30 の PWM パルス周期を 2ms とし、周期ごとに発生する割り込み（INTTMKB30）を 250 回カウントします。500ms ごとに CPU から FAA を起動し、FAA で PWM 出力のデューティ比を変更します。

1. ［CPU プログラム］デューティ値確認用の変数に、TKBCR01 レジスタ、TKBCR03 レジスタの初期値を格納します。
2. ［CPU プログラム］TKB30 の動作を許可します。
3. ［CPU プログラム］TKB30 の SFR アクセスを FAA バスに設定します。
4. ［CPU プログラム］TKB30 の割り込み（INTTMKB30）が 250 回発生（500ms 経過）するのを待ちます。
5. ［CPU プログラム］タイマ動作開始後、2ms ごとに TKB30 のタイマ割り込み（INTTKB30）が発生します。
6. ［CPU プログラム］TKB30 の割り込み（INTTMKB30）では、割り込みの発生回数をカウントします。
7. ［CPU プログラム］TKB30 の割り込み（INTTMKB30）が 250 回発生（500ms 経過）すると、FAA へクロック供給を許可し、FAA の動作を許可します。
8. ［CPU プログラム］FAA のスタックポインタ、FAA プログラムの開始アドレスを設定し、FAA の動作を開始させます。その後、FAA のプログラム完了を待ちます。
9. ［FAA プログラム］コンペア・レジスタ（TKBCR01）を更新して TKBO00 出力のデューティ比を更新します。また、コンペア・レジスタ（TKBCR03）を更新して TKBO01 出力のデューティ比を更新します。500ms 経過 するごとに、TKBO00 出力のデューティ比は 10%→20%→40%→80%の順に 2 倍ずつ更新され、デューティ比が 80%の後は再び 10%に設定されます。TKBO01 出力のデューティ比は 80%→40%→20%→10%の順に 1/2 倍ずつ更新され、デューティ比が 10%の後は再び 80%に設定されます。
10. ［FAA プログラム］更新したデューティ比（TKBCR01 レジスタ値、TKBCR03 レジスタ値）をグローバル変数に格納し、FAA は動作を停止します。
11. ［CPU プログラム］FAA のプログラム実行が完了すると、FAA へクロック供給を停止し、FAA の動作を禁止します。
12. ［CPU プログラム］デューティ値確認用変数に、更新されたデューティ比（TKBCR01 レジスタ値、TKBCR03 レジスタ値）を格納します。
13. ［CPU プログラム］4 へ戻り、再び TKB30 の割り込み（INTTMKB30）が 250 回発生（500ms 経過）を待ちます。

3.2 動作確認条件

表 3-3 動作確認条件

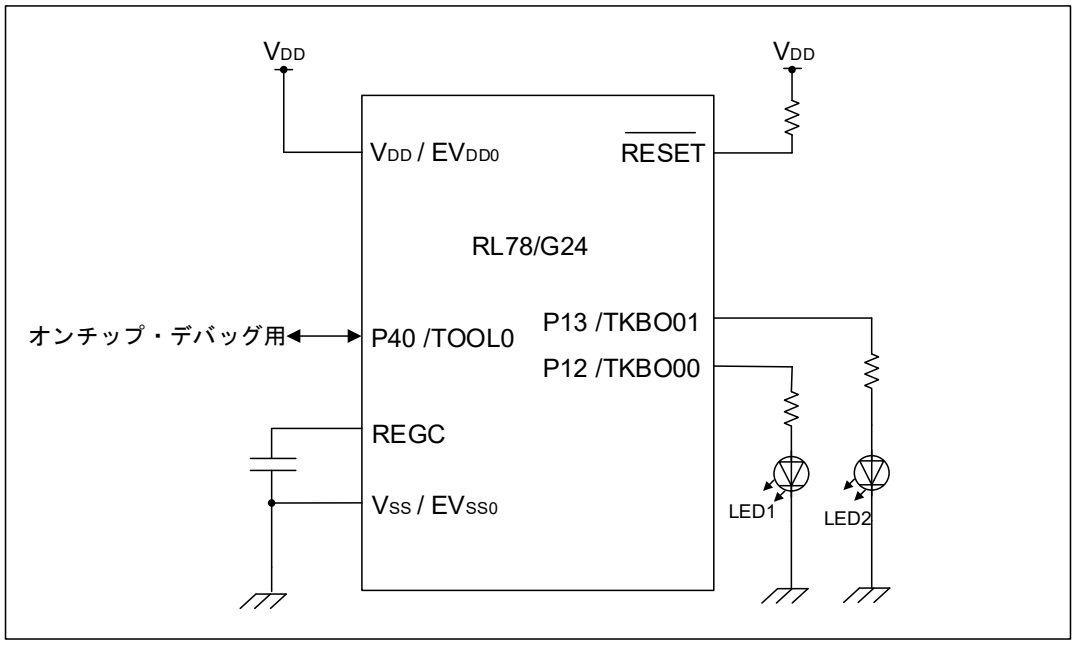
項目	内容
使用マイコン	RL78/G24（R7F101GLG）
動作周波数	・ 高速オンチップ・オシレータ・クロック: 32MHz ・ CPU／周辺ハードウエア・クロック: 32MHz
動作電圧	・ 3.3V（2.7V～5.5V で動作可能） ・ LVD0 動作（VLVD0）：リセット・モード 立ち上がり時 TYP. 2.97V 立ち下がり時 TYP. 2.91V
統合開発環境 (e2 studio)	ルネサス エレクトロニクス製 v2025-12
C コンパイラ (e2 studio)	ルネサス エレクトロニクス製 CC-RL V1.15.01
FAA/GREEN_DSP 構造化アセンブラ	ルネサス エレクトロニクス製 DSPASM V1.05.0
スマート・コンフィグレータ (SC) プラグイン	ルネサス エレクトロニクス製 e2 studio v2025-12に同梱のバージョン
ボードサポートパッケージ (BSP)	ルネサス エレクトロニクス製 V1.92
エミュレータ	E2 エミュレータ Lite
使用ボード	RL78/G24 Fast Prototyping Board (RTK7RLG240C00000BJ)

3.3 ハードウェア説明

3.3.1 ハードウェア構成例

本サンプルコードで使用するハードウェア構成例を示します。

図 3-2 ハードウェア構成例



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

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

注意 3. VDD は LVD0 にて設定したリセット解除電圧（VLVD0）以上にしてください。

3.3.2 使用端子

表 3-4 に使用端子と機能を示します。

表 3-4 使用端子と機能

端子名	入出力	内容
P12 / TKBO00	出力	PWM 出力 (LED1 の点灯制御)
P13 / TKBO01	出力	PWM 出力 (LED2 の点灯制御)

注意 本アプリケーションノートは、使用端子のみを端子処理しています。実際に回路を作成される場合は、端子処理などを適切に行い、電気的特性を満たすように設計してください。

3.4　ソフトウェア説明

3.4.1　スマート・コンフィグレータの設定

本サンプルコードにおけるスマート・コンフィグレータ（SC）の設定を示します。スマート・コンフィグレータの設定における各表の項目、設定内容は設定画面の表記で記載しています。

3.4.1.1　クロック設定

本サンプルコードで使用しているクロック設定を以下に示します。

動作モード：高速メイン・モード 2.7(V)~5.5(V)

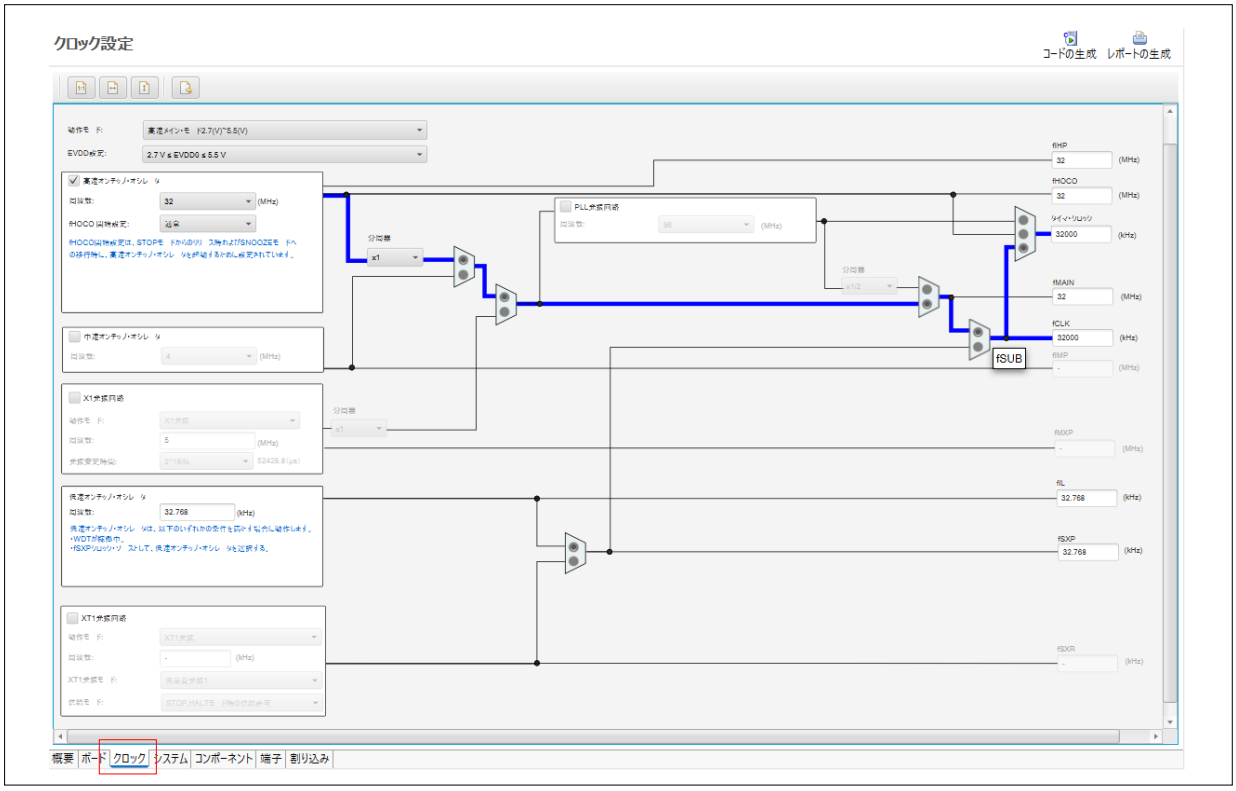
EVDD 設定：2.7 V ≤ EVDD0 ≤ 5.5V

高速オンチップ・オシレータ：32MHz

fCLK：32000kHz

タイマークロック：32000kHz

図 3-3　クロック設定



3.4.1.2　システム設定

本サンプルコードで使用しているシステム設定を以下に示します。

図 3-4　システム設定

3.4.1.3　コンポーネントの設定

本サンプルコードで使用しているコンポーネントの設定を以下に示します。

表 3-5　コンポーネントの設定（LVD0）

項目	内容
コンポーネント	電圧検出回路
コンフィグレーション名	Config_LVD0
リソース	LVD0

図 3-5　LVD0 の設定

表 3-6　コンポーネントの設定（TKB30）

項目	内容
コンポーネント	PWM 出力
動作	単体動作モード（TKBCRn0 レジスタによる周期制御）
コンフィグレーション名	Config_TKB0
リソース	TKB0

図 3-6　TKB30 の設定

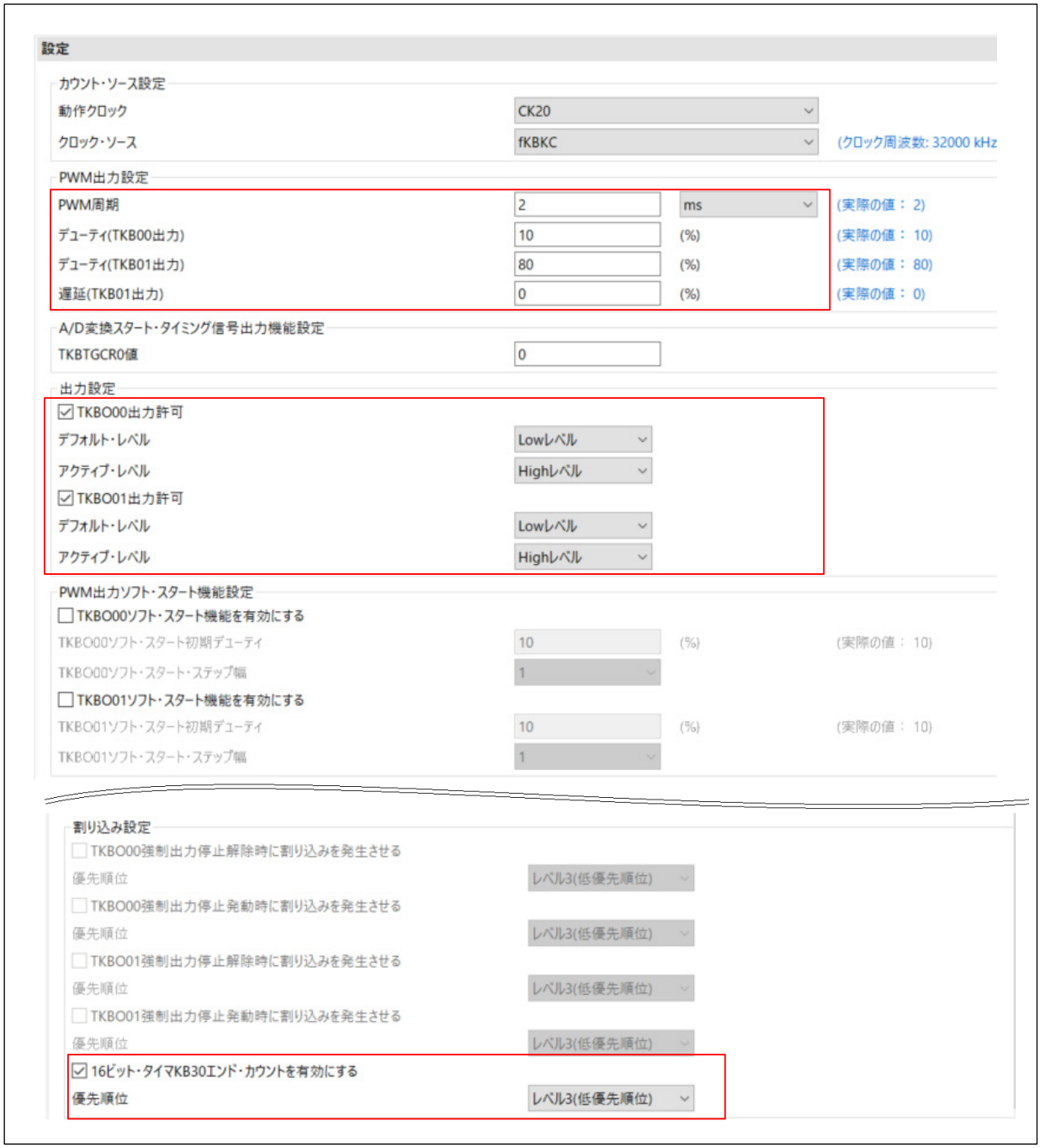
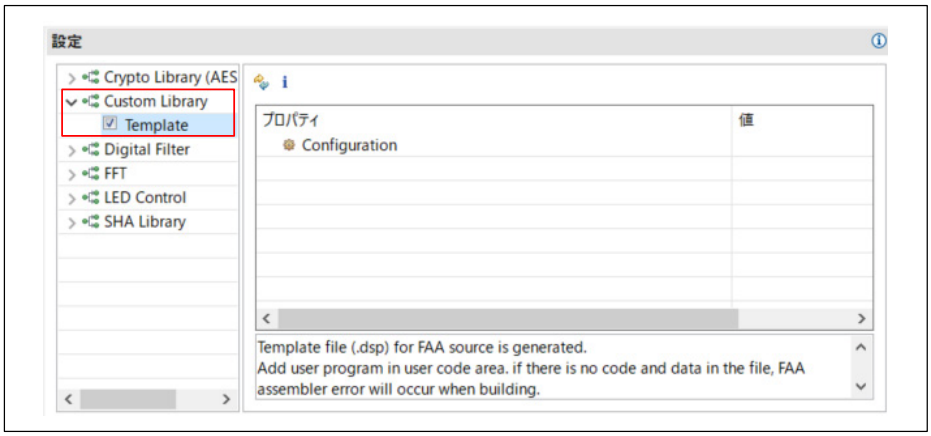


表 3-7　コンポーネントの設定（FAA）

項目	内容
コンポーネント	フレキシブル・アプリケーション・アクセラレータ
コンフィグレーション名	Config_FAA

図 3-7　FAA の設定



備考：サンプルプロジェクトを読み込み後に FAA ライブラリが表示されない場合、2.3.1 FAA コンポーネントの追加 の No.11 を参照して FAA ライブラリをダウンロードしてください。

3.4.2 フォルダ構成

表 3-8 にサンプルプロジェクトで使用しているソース・ファイル／ヘッダファイルの構成を示します。

表 3-8 フォルダ構成

フォルダ、ファイル名	説明	SC が生成
¥sample_project<DIR>	サンプルプロジェクトのフォルダ	
¥src<DIR>	プログラム格納用フォルダ	√
sample_project.c	サンプル・ソース・ファイル	√注 1
¥smc_gen<DIR>	スマート・コンフィグレータ生成フォルダ	√
¥Config_FAA<DIR>	FAA 用プログラム格納フォルダ	√
Config_FAA_common.c	Common FAA module のソース・ファイル	√
Config_FAA_common.h	Common FAA module のヘッダファイル	√
Config_FAA_common.inc	FAA 用インクルードファイル	√
Config_FAA_src.dsp	FAA 用アセンブラ・ソース・ファイル	√注 2
¥Config_TKB0<DIR>	TKB30 用プログラム格納フォルダ	√
Config_TKB0.c	TKB30 用ソース・ファイル	√
Config_TKB0.h	TKB30 用ヘッダファイル	√
Config_TKB0_user.c	TKB30 用割り込みソース・ファイル	√注 3
¥general<DIR>	初期化、共通プログラム格納フォルダ	√
¥r_bsp<DIR>	BSP 用プログラム格納フォルダ	√
¥r_config<DIR>	コンフィグレーションヘッダ格納フォルダ	√

注 ”<DIR>”は、ディレクトリを意味します。
注1. サンプルプロジェクト用にコードを追加しています。
注2. 本サンプルプロジェクトでは、FAA ライブラリの Custom Library を選択しているため、ファイル生成直後はテンプレートのみでコードは記載されていません。サンプルプロジェクト用にコードを追加しています。
注3. SC のユーザコードエリアに、サンプルプロジェクト用にコードを追加しています。

3.4.3 オプション・バイトの設定

表 3-9 にオプション・バイト設定を示します。

表 3-9 オプション・バイト設定

アドレス	設定値	内容
000C0H/040C0H	1110 1111B (EFH)	ウォッチドッグ・タイマ動作停止 （リセット解除後、カウント停止）
000C1H/040C1H	1111 1011B (FBH)	LVD0 リセット・モード 検出電圧：立ち上がり 2.97V／立下り 2.91V
000C2H/040C2H	1110 1000B (E8H)	フラッシュ動作モード：高速メインモード 高速オンチップ・オシレータの周波数：32MHz
000C3H/040C3H	1000 0100B (84H)	オンチップ・デバッグ動作許可

3.4.4 定数一覧

表 3-10、表 3-11 に本サンプルコードで使用する定数を示します。

表 3-10 定数一覧（CPU プログラム）

定数名	設定値	内容	使用関数
FAA_BUS_ACCESS	0200H	FAA から TKB30 のレジスタへのアクセス許可（ADBSEL 設定値）	main

表 3-11 定数一覧（FAA プログラム）

定数名	設定値	内容
_C_TKBO00_DUTY_INIT	1900H	TKBO00 出力の初期デューティ比（TKBCR01 設定値）
_C_TKBO01_DUTY_INIT	C800H	TKBO01 出力の初期デューティ比（TKBCR03 設定値）
_C_TKBTRG_TKBRDT_REQ	1H	TKB30 コンペアレジスター斉書き換え要求（TKBRDT0 設定値）

3.4.5 変数一覧

表 3-12、表 3-13 に本サンプルコードで使用する変数を示します。

表 3-12 変数一覧（CPU プログラム）

型	変数名	内容	使用関数
uint32_t	g_work_tkbo00	現在の TKBO00 出力のデューティ比（TKBCR01 設定値）確認用変数	main
uint32_t	g_work_tkbo01	現在の TKBO01 出力のデューティ比（TKBCR03 設定値）確認用変数	main
uint8_t	g_tkb_interrupt_flag	500ms 経過フラグ	r_Config_TKB0_end_count_interrupt

表 3-13 変数一覧（FAA プログラム）

サイズ	変数名	内容
4 バイト	_V_TKBO00_DUTY	変更後の TKBO00 出力のデューティ比（TKBCR01 設定値）格納変数
4 バイト	_V_TKBO01_DUTY	変更後の TKBO01 出力のデューティ比（TKBCR03 設定値）格納変数

3.4.6　関数一覧

表 3-14、表 3-15 に本サンプルコードで使用する関数、処理を示します。ただし、スマート・コンフィグレータで生成された関数のうち、コード追加を行っていないものは記載しません。

表 3-14 関数一覧（CPU プログラム）

関数名	概要	ソース・ファイル
main	メイン処理	main.c
r_Config_TKB0_end_count_interrupt	TKB30 のタイマ割り込み（INTTKB00）の発生回数のカウント	Config_TKB0_user.c

表 3-15 処理一覧（FAA プログラム）

ラベル名	概要	ソース・ファイル
_P_TKB_PWM	TKBO00,TKBO01 出力のデューティ比の更新	Config_FAA_src.dsp

3.4.7　関数仕様

サンプルコードの関数仕様を示します。

CPU プログラム

[関数名]　main()

概　要	メイン処理
ヘッダ	r_smc_entry.h、Config_TKB0.h
宣　言	void main(void)
説　明	タイマ TKB30 の動作を開始し、500ms 経過するごとに FAA を動作させます。
引　数	なし
リターン値	なし

CPU プログラム

[関数名]　r_Config_TKB0_end_count_interrupt()

概　要	TKB30のタイマ割り込み処理
ヘッダ	r_cg_macrodriver.h、r_cg_userdefine.h、Config_TKB0.h
宣　言	static void __near r_Config_TKB0_end_count_interrupt(void)
説　明	INTTMKB0 割り込み発生回数をカウントし、割り込みが 250 回発生（500ms 経過）するごとに 500ms 経過フラグをセットします。
引　数	なし
リターン値	なし

FAA プログラム

[ラベル名]　_P_TKB_PWM

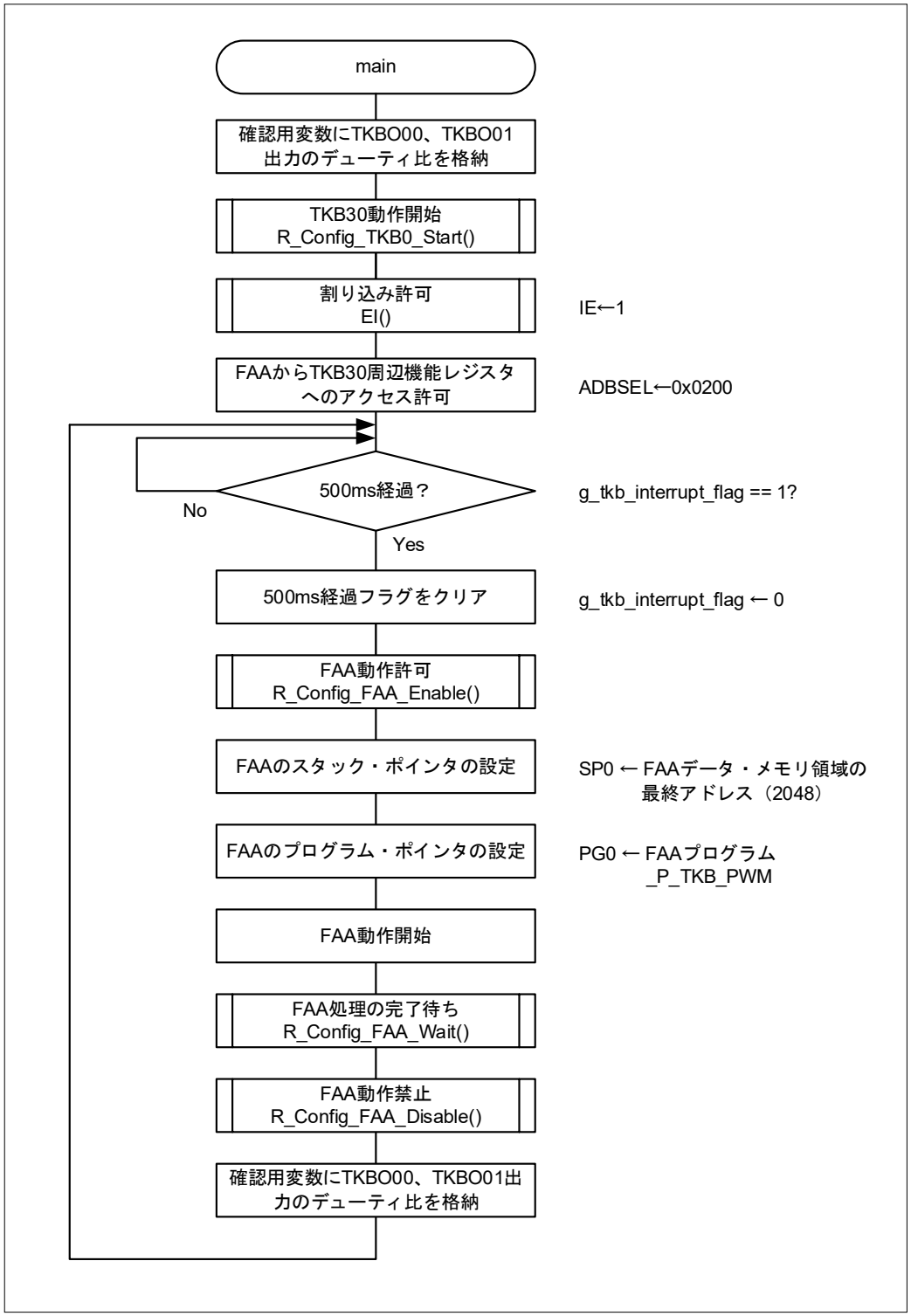
概　要	TKBO00、TKBO01出力のデューティ比更新処理
ヘッダ	Config_FAA_common.inc
宣　言	ー
説　明	TKBO00,TKBO01 の PWM 出力のデューティ比を更新します。
引　数	なし
リターン値	なし

3.4.8　フローチャート

3.4.8.1　メイン処理

図 3-8 にメイン処理のフローチャートを示します。

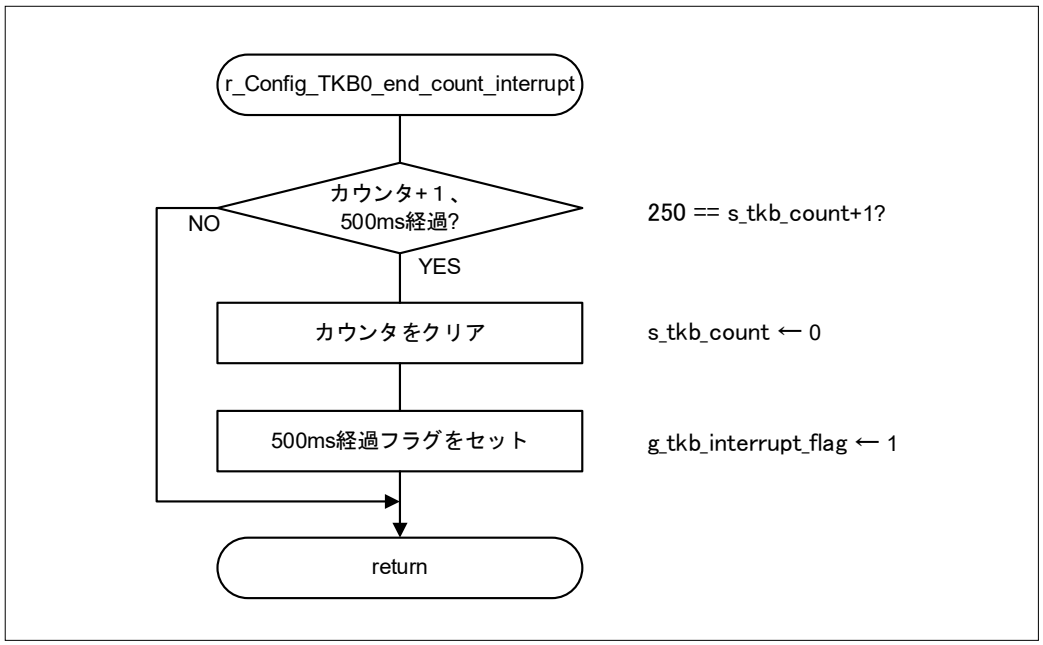
図 3-8 メイン処理



3.4.8.2　r_Config_TKB0_end_count_interrupt 関数

図 3-9 に r_Config_TKB0_end_count_interrupt 関数のフローチャートを示します。

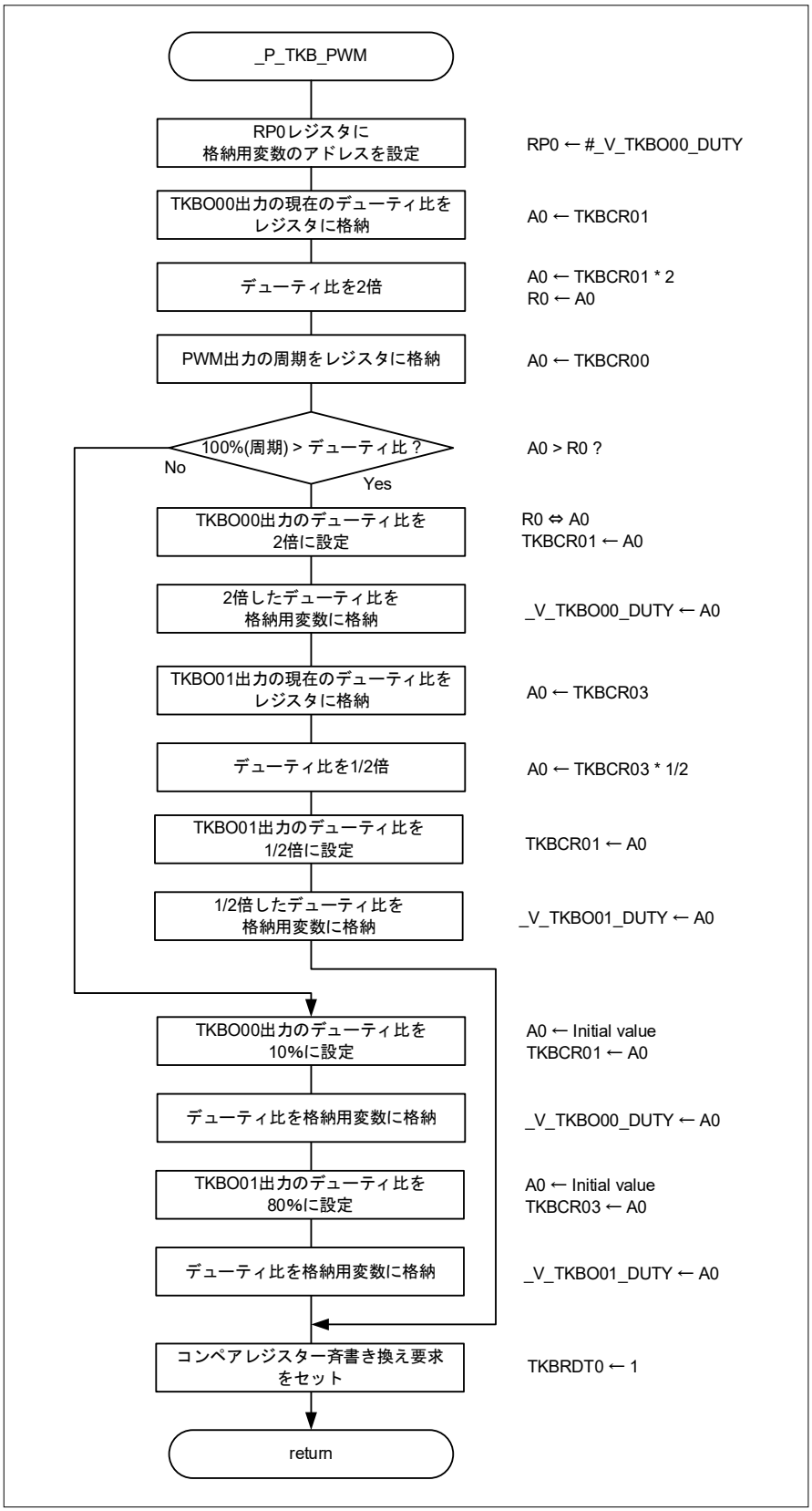
図 3-9 r_Config_TKB0_end_count_interrupt 関数



3.4.8.3　FAA 処理

図 3-10 に FAA 処理のフローチャートを示します。

図 3-10 FAA 処理



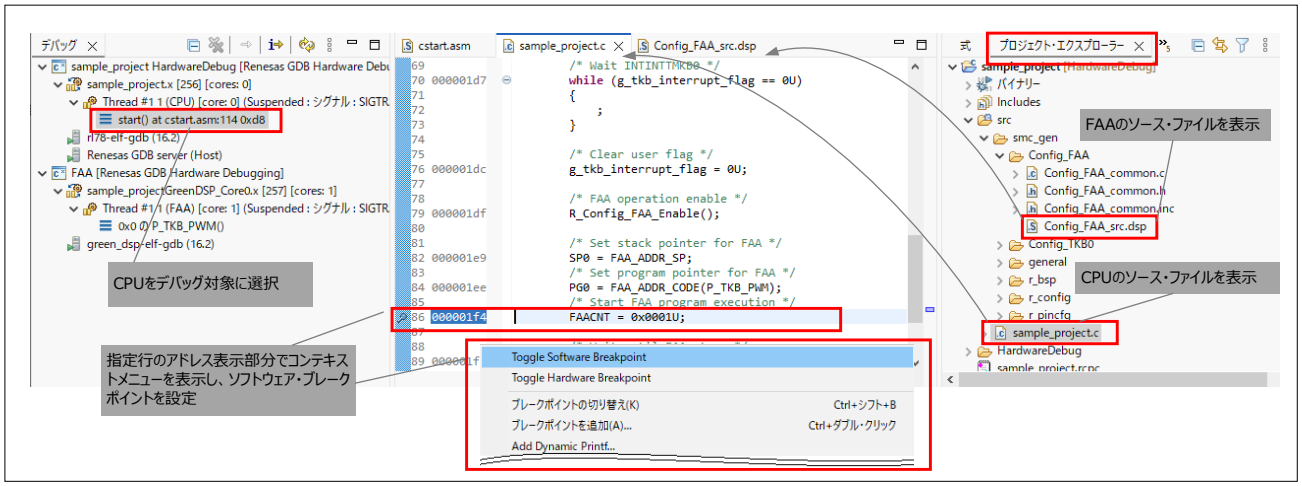
3.5　デバッグ基本操作

サンプルコードを使用し、FAA プログラムのデバッグの基本操作について説明します。

手順

1. RL78/G24 Fast Prototyping Board を PC に接続します。
2. e2 studio を起動し、サンプルプロジェクトをインポートします。サンプルプロジェクトでは、E2 エミュレータ Lite 接続の設定になっています。COM ポート接続で使用する場合は e2 studio の「実行」―「デバッグの構成」メニューで設定を変更してください。
3. プログラムをビルドします。（2.4.3 プログラムのビルド　参照）
4. プログラムを RL78/G24 Fast Prototyping Board にダウンロードします。（2.5.3 プログラムのダウンロード　参照）
5. デバッグ対象に CPU を選択します。（2.6.1 デバッグ対象 参照）
6. FAA プログラムをデバッグするには、FAA が動作許可状態（FAAEN=1、ENB=1）になっている必要があります。CPU プログラムを FAA の動作を許可するコードまで実行します。
   sample_project.c を開き、「FAACNT = 0x0001U;」のアドレス表示位置でコンテキストメニューを表示し、「Toggle Software Breakpoint」をクリックしてソフトウェアブレークポイントを設定します。
   また、あらかじめ、FAA プログラムのソース・ファイル Config_FAA_src.dsp を開いておきます。

図 3-11　sample_project.c（デバッグ対象：CPU）



備考 1.　FAA の動作許可は、R_Config_FAA_Enable()で行っています。

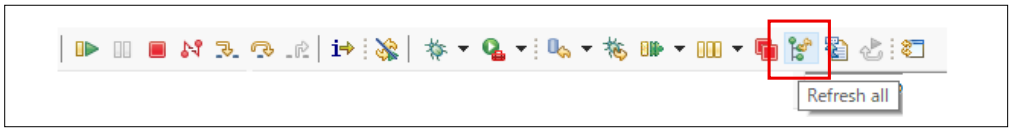
備考 2.　ブレークポイントにはハードウェア・ブレーク（実行後ブレーク）とソフトウェア・ブレーク（実行前ブレーク）の 2 種類があります。

7. ツールバーで、リセット、実行開始のボタンをクリックします。main 関数の先頭まで実行されるので、再度実行開始ボタンをクリックします。

図 3-12　デバッグ・ツールバー

8. ブレークポイントで停止後、デバッグ対象を FAA へ変更します。変更後、FAA プログラムのソース・ファイルが表示されない場合、「Refresh all」ボタンを押してください。

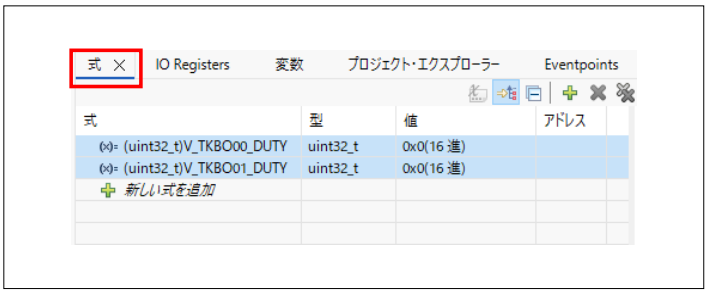
図 3-13　デバッグ・ツールバー



9. [変数表示]
ウォッチする変数（_V_TKBO00_DUTY、_V_TKBO01_DUTY）を式ビューに登録します。

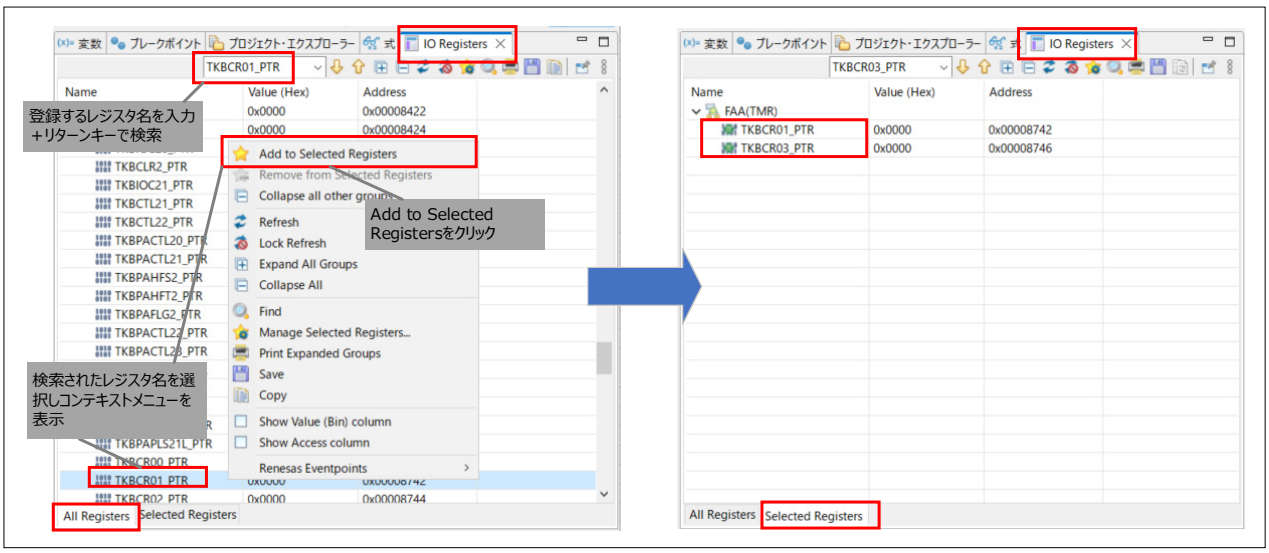
・登録後に先頭の"_"の削除と"(uint32_t)"を追加し、必要に応じて 16 進表記に変更してください。
（2.6.6 シンボル（ラベル）の表示 参照）

図 3-14　式ビュー



10. [SFR 表示]
ウォッチする SFR レジスタ（TKBCR01_PTR、TKBCR03_PTR）を IO Register ビューで、Selected Register 表示に登録します。

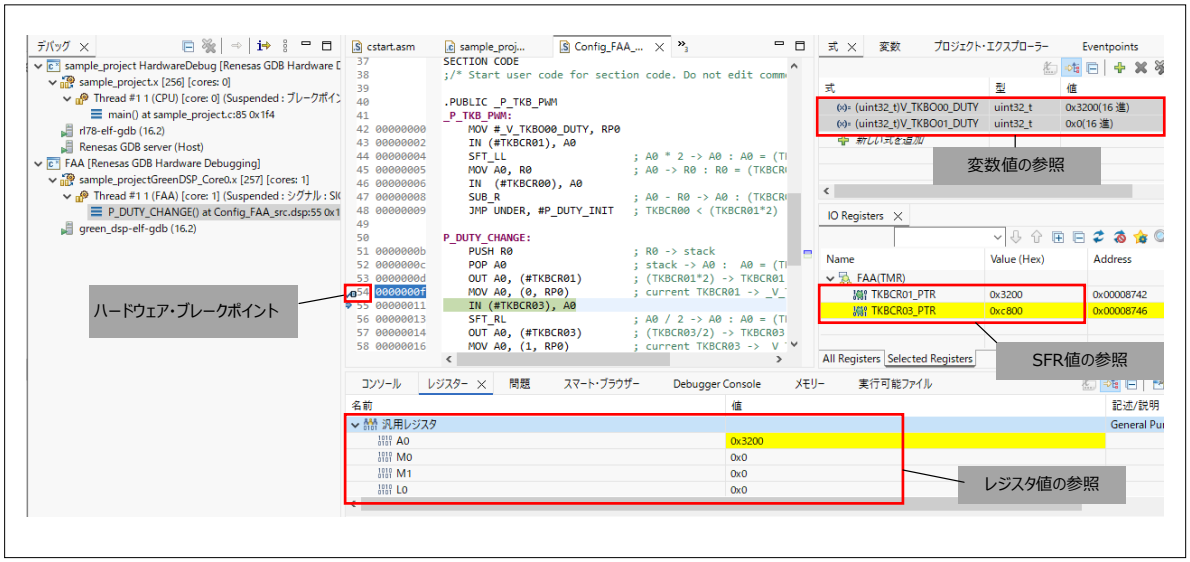
図 3-15　IO Registers ビュー



11. FAA プログラムをステップ実行／実行し、変数、SFR、レジスタの値を確認しながらデバッグを行います。

・ FAA プログラムのソースのアドレス行をダブルクリックし、ブレークポイントを設定できます。（2.6.4 ブレークポイント　参照）

図 3-16　FAA プログラムのデバッガ画面例



4. サンプルコード

サンプルコードは、ルネサス エレクトロニクスホームページから入手してください。

5. 参考ドキュメント

RL78/G24 ユーザーズマニュアル ハードウェア編（R01UH0961）

RL78 ファミリ ユーザーズマニュアル ソフトウェア編（R01US0015）

DSPASM FAA/GREEN_DSP 構造化アセンブラ ユーザーズマニュアル（R20UT3911）

RL78/G24 Fast Prototyping Board ユーザーズマニュアル（R20UT5091）

RL78 スマート・コンフィグレータ ユーザーガイド：e² studio 編（R20AN0579）

（最新版をルネサス エレクトロニクスホームページから入手してください。）

テクニカルアップデート／テクニカルニュース

（最新版の情報をルネサス エレクトロニクスホームページから入手してください。）

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	Nov.14.23	-	First edition
1.10	Jan. 13.26	5	図 1-4　追加
		6	1.3.2 説明追加 1.3.3 備考を追加
		8	表 2-1　ツールのバージョンを変更 表 2-2　注 1 を追加
		18	図 2-21　変更
		19	図 2-22、図 2-23、図 2-24　変更
		20	図 2-25　変更
		21	図 2-28　変更、備考 1　追加 19. にリンクを追加
		22	表 2-4　転送処理の関数名を追加 表 2-4　表下の説明を一部変更
		23	注 2　説明を変更
		30	表 2-10 注 1 を追加
		34	注意 2 に転送処理の関数名を追加
		37	2.6.3 実行／停止の説明を変更
		38	2.6.3 実行／停止の説明を追加
		39	2.6.4 ブレークポイントの説明を追加
		42	図 2-59　変更
		44	2.6.8 IO Registers(SFR)表示の説明を一部削除
		47	表 3-3　ツールのバージョンを変更
		49	図 3-3　変更
		53	表 3-8　フォルダ構成からサンプルスクリプトを削除
		ー	3.5 サンプルスクリプト仕様　を削除
		59	3.5 デバッグ基本操作　の手順を変更
		60	図 3-14　変更

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。