ModusToolbox™用 IAR Embedded Workbench ユーザー ガイド

ModusToolbox™ツールパッケージ バージョン 3.5.0

本書について

適用範囲と目的

この文書の新しいバージョンは、こちらのウェブで入手可能です。

ModusToolbox™ソフトウェアは、デバイス設定とアプリケーション開発をサポートするためのツールとライブラリのセットです。これらのツールは、インフィニオンのデバイスを既存の開発方法に統合できるようにするツールセットです。本書では、IAR Embedded Workbench を ModusToolbox™ソフトウェアと共に使用する方法について説明します。

IAR Embedded Workbench でインフィニオンのデバイスを扱う一般的なフローは以下のとおりです。

- ソフトウェアのダウンロード/インストール
- Project Creator を使用して ModusToolbox™アプリケーションを作成
- IAR Embedded Workbench でプロジェクトを作成
- プロジェクトの設定とビルド
- デバイスのプログラム
- アプリケーションのデバッグ

本書の表記法

- **太字** - 見出しレベル、列見出し、メニューとサブメニューを強調します。
- *斜体* - ファイル名とパスを表します。
- `Monospace` - API、関数、割込みハンドラ、イベント、データ型、エラーハンドラ、ファイル/フォルダ名、ディレクトリ、コマンドライン入力、プログラムコードを表します。
- **File > New** - メニュー項目を選択するとカスケード サブ メニューが開くことを示します。

参考資料

必要に応じて、以下の資料を参照してください。

- ModusToolbox™ software installation guide - Windows、Linux、macOS へのソフトウェアのインストールに関する情報と手順を提供します。
- ModusToolbox™ tools package user guide - ModusToolbox™ツールパッケージに含まれるすべてのツールに関する情報を提供します。
- Project Creator user guide - Project Creator ツールに関する具体的な情報を提供します。

目次

本書について 1
目次 2
1 ソフトウェアのダウンロード/インストール 4
1.1 ModusToolbox™ソフトウェア 4
1.2 IAR Embedded Workbench (Windows のみ) 4
1.3 J-Link 4
2 IAR Embedded Workbench 用アプリケーションの作成/エクスポート 5
2.1 ModusToolbox™で新しいアプリケーションを作成/エクスポート 5
2.2 IAR ワークスペースおよびプロジェクトの作成 7
3 その他の項目 9
3.1 対応デバッガー プローブ 9
3.2 C++ファイルでアプリケーションを設定 9
3.3 ビルド オプション、プリビルド/ポストビルドの設定 9
3.4 RTOS 設定 11
3.5 パッチ適用済みフラッシュローダー 11
3.6 ETM/ITM トレースの実行 11
3.7 マルチコアツールバーと CTI の使用 (I-Jet および CMSIS-DAP のみ) 11
4 PSOC™ 4 および PSOC™ 6 シングルコア アプリケーション 12
4.1 設定およびビルド 12
4.2 KitProg3/MiniProg4 (CMSIS-DAP) でのプログラム/デバッグ 12
4.3 J-Link を使ったプログラミング/デバッグ 14
4.4 外部メモリのプログラム 16
4.5 外部メモリを有効にした PSOC™ 6 MCU の消去 16
5 PSOC™ 64 Secure シングルコア アプリケーション 18
5.1 設定およびビルド 18
5.2 プログラミングおよびデバッグ 21
6 AIROC™ CYW20829 アプリケーション 23
6.1 設定およびビルド 23
6.2 プログラミングおよびデバッグ 27
7 PSOC™ Control C3 セキュア アプリケーション 29
7.1 デフォルトのポリシーが適用されるデバイス 29
7.2 プロビジョニングされたデバイス 29
7.3 プログラミングおよびデバッグ 32
8 PSOC™ 6 および XMC7000/TRAVEO™ II マルチコア アプリケーション 34
8.1 CM4/CM7 (スレーブ) コアの設定 34
8.2 CM0+ (マスター) コアの設定とビルド 37

目次

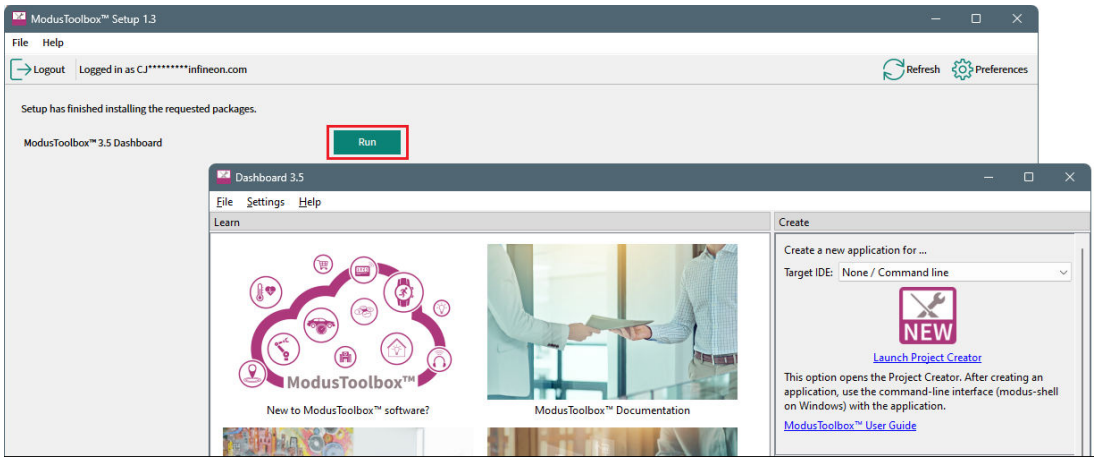
8.3 CMSIS-DAP/i-Jet デバッグオプションの設定 41
8.4 CMSIS-DAP/i-Jet によるマルチコア デバッグ セッションの開始 42
8.5 J-Link によるマルチコア デバッグ セッションの開始 43
改訂履歴 44
免責事項 45

1 ソフトウェアのダウンロード/インストール

1.1 ModusToolbox™ソフトウェア

https://softwaretools.infineon.com/tools/com.ifx.tb.tool.modustoolboxsetup から ModusToolbox™ Setup プログラムをダウンロードしてください。必要な ModusToolbox™ツールとパッケージのインストール方法については、ModusToolbox™ software installation guide の説明を参照してください。

すべてのツールのインストールが完了したら、**Run** をクリックしてダッシュボードを起動してください。



1.2 IAR Embedded Workbench (Windows のみ)

インフィニオンは、PSOC™ Control C3 デバイス用に IAR Embedded Workbench バージョン 9.60.3 以降を推奨し、テストを実施済みです。このバージョンは、ModusToolbox™ がサポートする全デバイスで使用できます。

IAR コンパイラの既定のインストール先は *C:\iar\[version]\arm* です。環境変数 `CY_COMPILER_ARM_DIR` を、スラッシュ (/) で正しいインストールパスに設定してください。あるいは、各アプリケーションの *Makefile* に、この変数を設定することもできます。ビルドシステム変数の詳細については、ModusToolbox™ tools package user guide を参照してください。

1.3 J-Link

J-Link のデバッグには、J-Link ソフトウェアをダウンロードし、インストールしてください。

https://www.segger.com/downloads/jlink/#J-LinkSoftwareAndDocumentationPack

2 IAR Embedded Workbench 用アプリケーションの作成/エクスポート

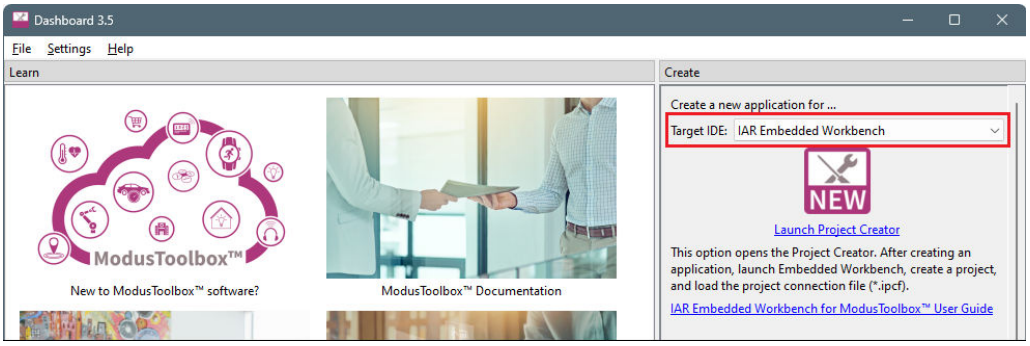
ここでは、IAR Embedded Workbench IDE で使用する ModusToolbox™アプリケーションを作成またはエクスポートする方法について説明します。

- ModusToolbox™で新しいアプリケーションを作成/エクスポート
- IAR ワークスペースおよびプロジェクトの作成

2.1 ModusToolbox™で新しいアプリケーションを作成/エクスポート

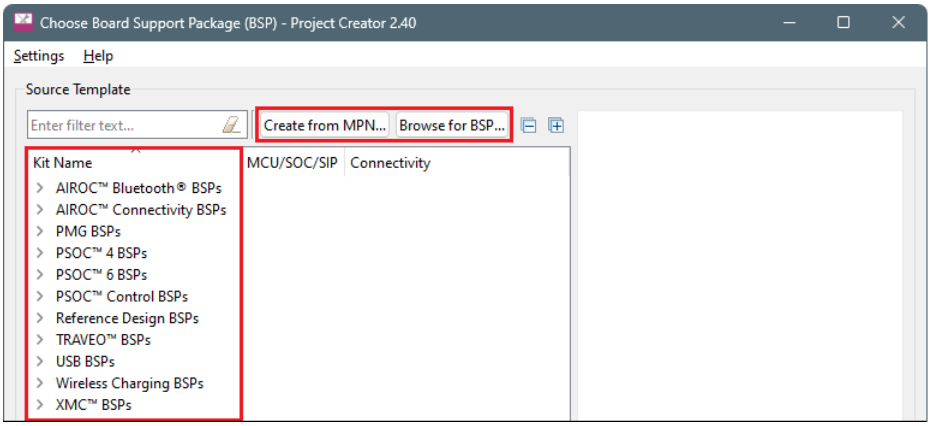
新しいアプリケーションの作成

1. ダッシュボードを使って Project Creator ツールを開き、IAR Embedded Workbench 用の ModusToolbox™ アプリケーションを作成してください。



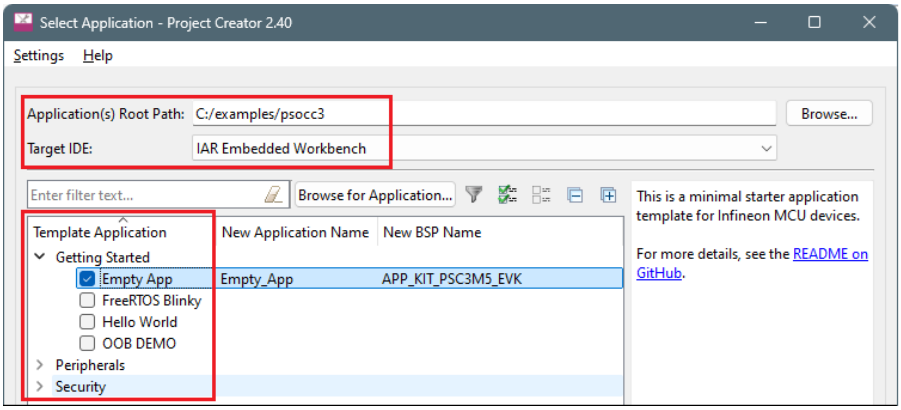
詳細は、Project Creator user guide を参照してください。

2. リストから BSP を選択するか、いずれかのボタンで MPN から BSP を作成するか、ディスク上の BSP を選択してください。

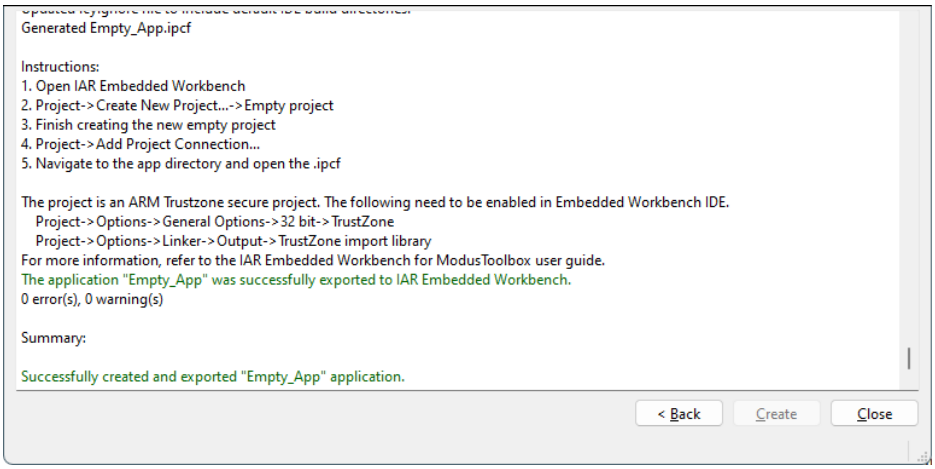


3. 次のページで、作成する場所とアプリケーションを選択してください。ターゲット IDE がダッシュボードからすでに選択されていることを確認してください。

2 IAR Embedded Workbench 用アプリケーションの作成/エクスポート



4. **Create** をクリックしてください。プロセスが完了すると、コンソールにメッセージが表示されます。



5. **Close** をクリックしてください。

既存アプリケーションのエクスポート

他の IDE やコマンドライン用に作成された ModusToolbox™アプリケーションがある場合、新しいアプリケーションを作成する代わりに、そのアプリケーションを IAR で使用できるようにエクスポートできます。ターミナル ウィンドウ (Windows では modus-shell) を開き、次のように入力してください。

```
make ewarm CY_IDE_PRJNAME=[project-name] TOOLCHAIN=IAR
```

ここで、`[project-name]`はルートアプリケーション/プロジェクトフォルダ名です。

注: *core-make-3.0 またはそれ以前のバージョンで作成されたアプリケーションでは、代わりに `make ewarm8` コマンドを使う必要があります。*

これにより、Embedded Workbench の設定ファイルでは `TOOLCHAIN` が IAR に設定されますが、ModusToolbox™アプリケーションの *Makefile* には**設定されません**。そのため、IAR Embedded Workbench 内でのビルドは IAR ツールチェーンを使用し、ModusToolbox™ 環境でのビルドは 、*Makefile* で以前に指定したツールチェーンを引き続き使用します。ModusToolbox™ビルドでも IAR ツールチェーンを使用したい場合は、`TOOLCHAIN` 変数を編集してください。

2.2 IAR ワークスペースおよびプロジェクトの作成

アプリケーションを作成またはエクスポートすると、ModusToolbox™のアプリケーション/プロジェクトディレクトリに IAR プロジェクト接続ファイル (.ipcf) が表示されます。オリジナルの ModusToolbox™アプリケーションのすべてのファイルとディレクトリの階層を含む XML ファイルです。例: *Hello_World.ipcf*。IAR で ModusToolbox™アプリケーションを使用するためには、次のようにしてください。

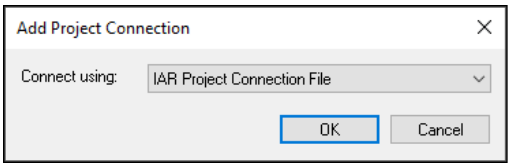
1. IAR Embedded Workbench を起動してください。

2. **File > New Workspace** の選択後、**File > Save Workspace** を選択し、ModusToolbox™ワークスペースを含むディレクトリに任意のワークスペース名 (*.eww) を入力してください。

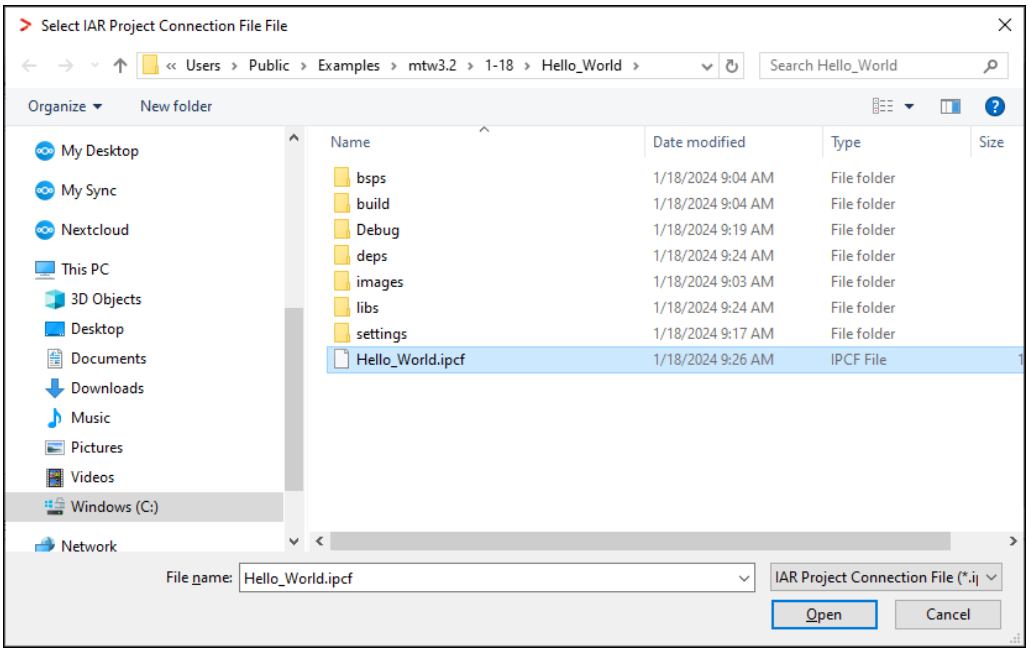
3. **Project > Create New Project > Empty project** を選択し、**OK** をクリックしてください。

4. ModusToolbox™プロジェクトディレクトリをブラウズし、任意のプロジェクト名(*.ewp)を入力し、**Save** をクリックしてください。

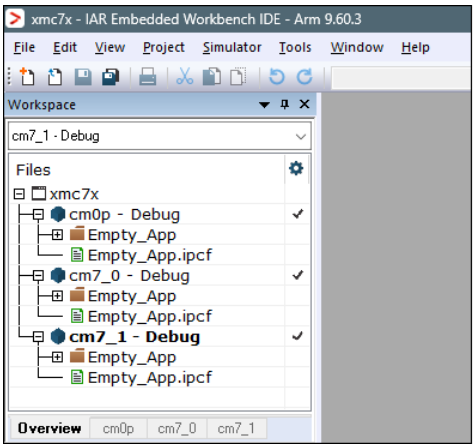
5. **Project > Add Project Connection** を選択し、ダイアログで「IAR Project Connection File」が選択されていることを確認し、**OK** をクリックしてください。



6. Select IAR Project Connection File ダイアログで、*.ipcf* ファイルを選択し、**Open** をクリックしてください。



7. ユーザーアプリケーションに複数のコア/プロジェクトがある場合は、各コアの IAR プロジェクトを作成し、各*.ipcf* ファイルを追加するプロセスを繰り返してください。以下に例を示します。

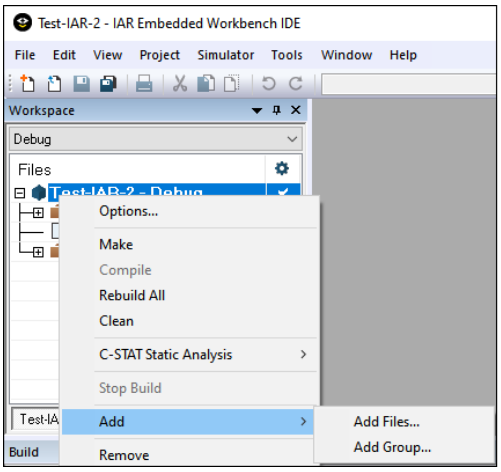


8. すべてのプロジェクトが作成され、接続されたら、ワークスペースとプロジェクトを保存してください。

***注**:* *その後、ModusToolbox™環境でさらにファイルやライブラリを追加した場合は、「make ewarm」を再度実行して、IAR Embedded Workbench プロジェクトの.ipcf ファイルを更新する必要があります。例:*

```
make ewarm TOOLCHAIN=IAR CY_IDE_PRJNAME=Helloo_World
```

- *CY_IDE_PRJNAME オプションを使用しない場合、生成された ipcf ファイルは元のサンプルコード名に戻り、アプリケーションは更新されません。*
- *プロジェクトファイルを生成する ModusToolbox™ ツールに接続したままでも構わない場合は、ワークスペースから.ipcf ファイルを削除し、IAR Embedded Workbench を再起動してください。*
- *IAR Embedded Workbench で変更を加えたい場合は、**Add** オプションを使ってワークスペース レベルで行う必要があります。こうした追加は.ipcf ファイルには含まれません。*



次のステップ

最も単純なケースでは、.ipcf ファイルに必要なビルド設定がすべて含まれているため、通常はビルド設定を構成する必要はありません。しかし、TrustZone モードの使用、ビルド後のバイナリイメージの処理 (複数の追加バイナリイメージの結合など)、バイナリイメージの署名やリマップを別のメモリ領域にリビルドするなど、ビルド設定の構成が必要なケースやデバイスがいくつかあります。ビルド設定の構成、アプリケーションのビルド、デバイスが接続されている場合のプログラム、アプリケーションのデバッグの手順については、該当するデバイスのセクションを参照してください。

3 その他の項目

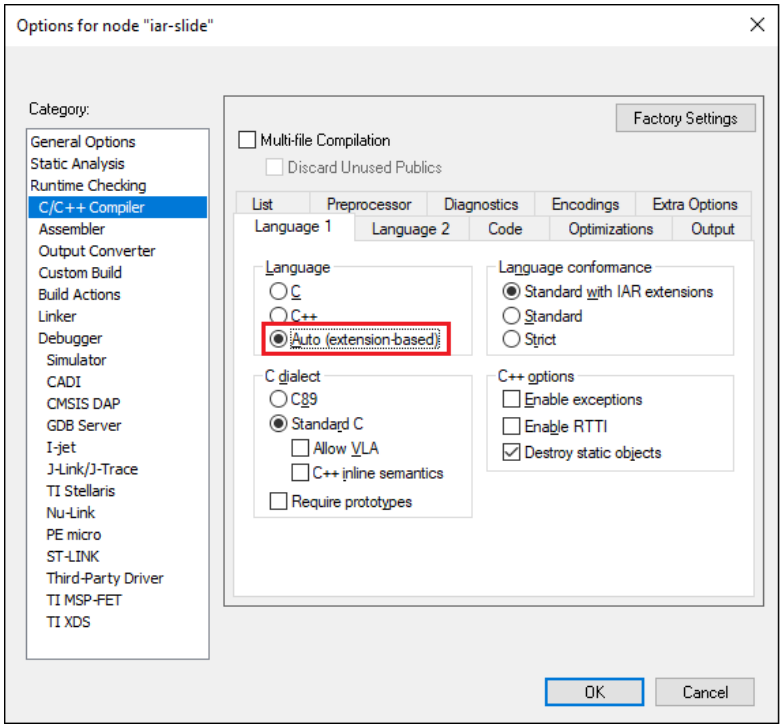
ここでは、ご使用のデバイスやアプリケーションに該当するかどうかを問わない、一般的な説明を記します。

3.1 対応デバッガー プローブ

- KitProg3 オンボード プログラマ
- MiniProg4
- IAR I-jet
- J-Link

3.2 C++ファイルでアプリケーションを設定

ユーザーアプリケーションに C++ファイルが含まれる場合は、C/C++コンパイラー オプションを設定する必要があります。プロジェクトを右クリックし、**Options > C/C++ Compiler > Language 1** を選択し、**Auto (extension-based)**を選択後、**OK** をクリックしてください。



3.3 ビルド オプション、プリビルド/ポストビルドの設定

様々なビルドオプションを設定するためには、IAR Embedded Workbench の Project Options ダイアログのコマンドラインオプションを入力するセクションを使用してください。

- **C / C++ Compiler > Extra Options**
- **Assembler > Extra Options**
- **Linker > Extra Options**

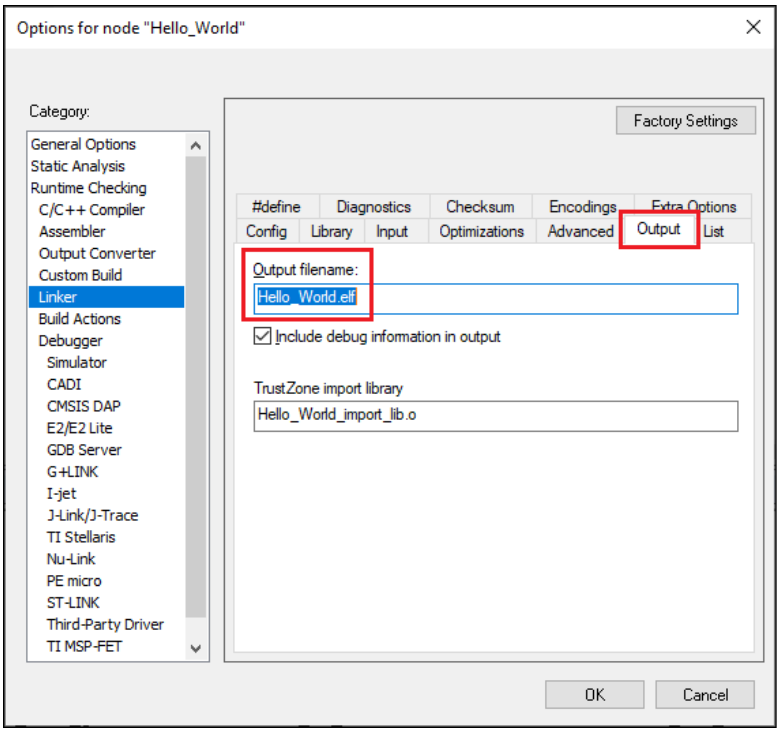
ModusToolbox アプリケーションでのプリビルドとポストビルドのステップは、デフォルトではエクスポートプロセスに含まれません。オリジナルのアプリケーションにこれらのステップがある場合、以下のように手動で IAR Embedded Workbench に含める必要があります。

1. IAR Embedded Workbench で Options ダイアログを開き、**Build Actions** カテゴリに移動してください。

2. **New** ボタンをクリックして、New Build Action ダイアログを開いてください。

3. New Build Action ダイアログで、コマンドライン引数、出力ファイルや入力ファイル、作業ディレクトリ、ビルド順序を入力してください。

4. .elf ファイルで動作するポストビルドスクリプトを使用している場合、**Linker > Output** タブに切り替え、**Output filename** フィールドのファイル拡張子を「.out」から「.elf」に変更してください。



5. **OK** をクリックして、オプション ダイアログを閉じてください。
詳細は、IAR-Click user guide を参照してください。

6. IAR のメインメニューで、**Project > Make** を選択し、アプリケーションをビルドしてください。以下のように表示されます。

```
Hello_World.elf

Total number of errors: 0
Total number of warnings: 0
Resolving dependencies...
Build succeeded
```

3.4 RTOS 設定

ユーザーアプリケーションに RTOS が含まれる場合、IAR Embedded Workbench のコンパイラとリンカの設定を更新する必要があります。これは、C/C++ライブラリの実装をスレッドセーフにするのに必要です。このプロセスは、IAR C/C++開発ガイドに記載されています。

1. **Project > Options > General Options > Library Configuration** を開いてください。
2. **Library** メニューで、「Full」を選択してください。
3. **Enable thread support in library** チェックボックスを選択してください。
4. **OK** をクリックしてください。

有効化後には、インフィニオンが clib-support ライブラリから提供する、ロックを実行するための関数を実装する必要があります。https://github.com/Infineon/clib-support/blob/master/README.md

注: これはインフィニオンのデバイスに限られたものではなく、IAR のコンパイラ/リンカと C ライブラリの実装がどのように動作するかを示しています。どのコンパイラも、同様の機能を備えています。

3.5 パッチ適用済みフラッシュローダー

異なる QSPI 設定のサポートを有効にするために、ModusToolbox™ QSPI Configurator はフラッシュローダー ファイルにパッチを適用し、アプリケーション ディレクトリに保存します。こうしたアプリケーションを IAR Embedded Workbench にエクスポートする場合、これらのパッチを適用したフラッシュローダー ファイルを適切なディレクトリにコピーする必要があります。

*<app-dirbsps><Kit-Name><configuration><GeneratedSource>*ディレクトリにある*.out* ファイルをコピーしてください。

その flashloader ファイルを*[install-path]•flashloader•Infineon•[device]*ディレクトリに貼り付けてください。

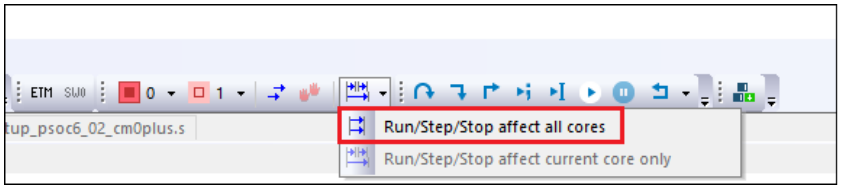
3.6 ETM/ITM トレースの実行

詳細については、アプリケーションノート AN235279 - Performing ETM and ITM trace on PSOC™ 6 MCU を参照してください。

3.7 マルチコアツールバーと CTI の使用 (I-Jet および CMSIS-DAP のみ)

I-Jet または CMSIS-DAP ドライバによってマルチコアデバッグが確立されると、マルチコアツールバーが利用可能になります。これにより、単一の IDE インスタンス内から、全コア / 単一コアの停止と再開ができるようになります。

また、クロストリガー インターフェース (CTI) と呼ばれる機能もあります。これにより、他のコアが停止 /再開されたときに、あるコアを即座に停止/再開できます。例えば、他のコアがブレークポイントをヒットしたときに、そのコアの 1 つがどのコードを実行しているかをチェックしなければいけない場合に便利です。CTI を使用するためには、マルチコア アプリケーションで使用可能な **Run/Step/Stop affect all cores** オプションを選択してください。



4 PSOC™ 4 および PSOC™ 6 シングルコア アプリケーション

IAR Embedded Workbench 用アプリケーションの作成/エクスポートの手順に従ってください。

次に、このセクションの手順を使用して、IAR Embedded Workbench でシングルコア PSOC™ 4 および PSOC™ 6 デバイス用のアプリケーションをビルド、プログラム、デバッグしてください。

4.1 設定およびビルド

多くの場合、シンプルなシングルコアアプリケーションの設定とビルドに必要なものはほとんどありません。設定が必要な場合については、以下を参照してください。

- C++ファイルでアプリケーションを設定
- ビルド オプション、プリビルド/ポストビルドの設定
- RTOS 設定

IAR のメインメニューで、**Project > Make** を選択し、アプリケーションをビルドしてください。以下のように表示されます。

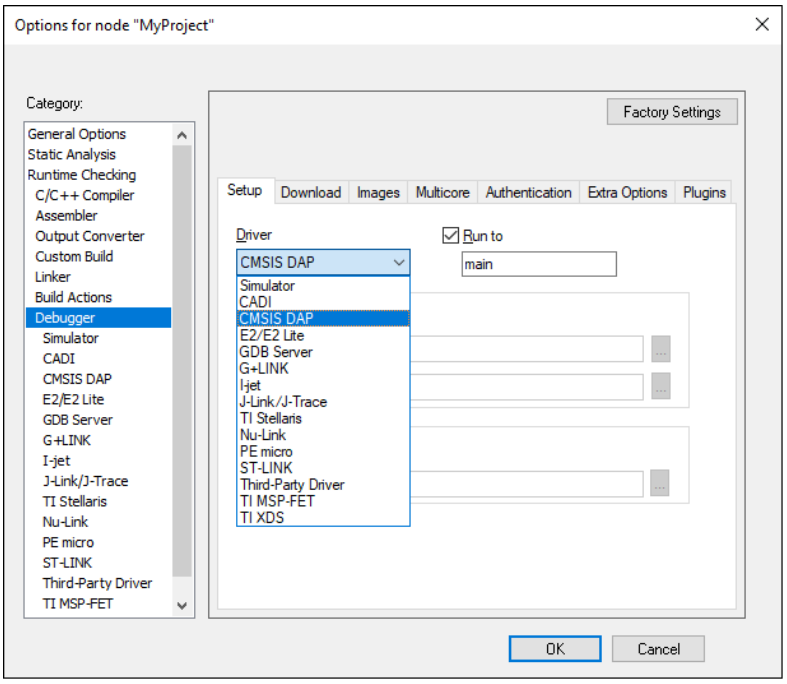
```
Hello_World.out

Total number of errors: 0
Total number of warnings: 0
Resolving dependencies...
Build succeeded
```

4.2 KitProg3/MiniProg4 (CMSIS-DAP) でのプログラム/デバッグ

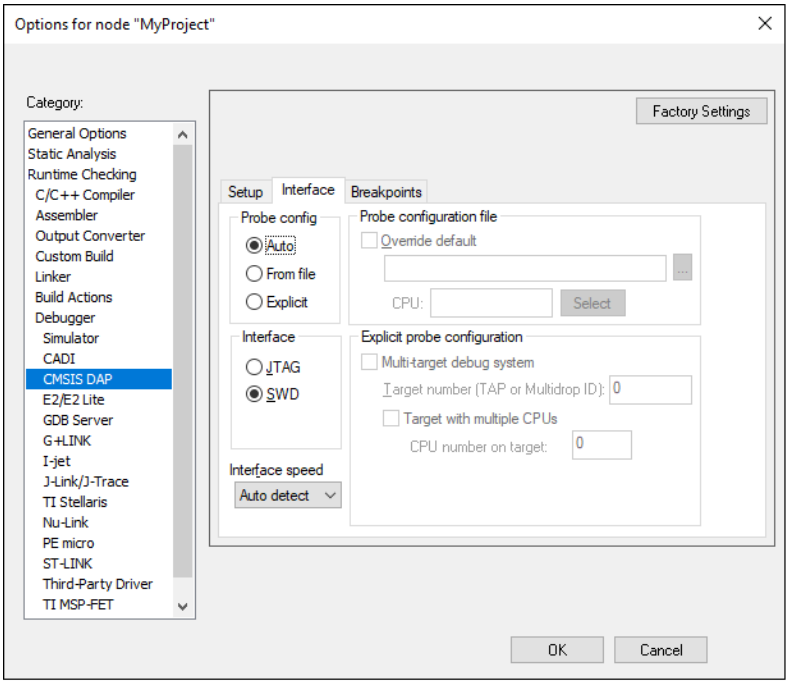
必要に応じて、fw-loader ツールを実行し、ボードのファームウェアが KitProg3 にアップグレードされているのを確認してください。詳細については、KitProg3 ユーザーガイドを参照してください。ツールはデフォルトで以下のディレクトリにあります。

C:\Infineon\Tools\ModusToolboxProgtools-[version]\fw-loader\bin

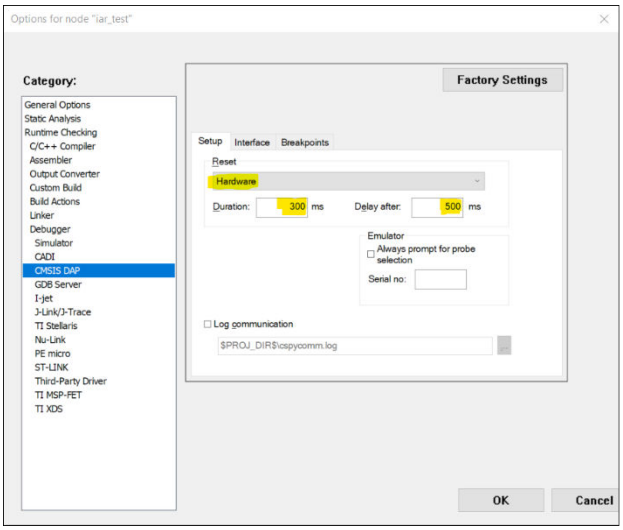
1. IAR Embedded Workbench で、オプション ダイアログを開き、**Debugger** のドライバリストから **CMSIS-DAP** を選択してください。



2. **CMSIS-DAP** ノードを選択し、インターフェースを **JTAG** から **SWD** に切り替え、インターフェース速度を **2MHZ** に設定してください。



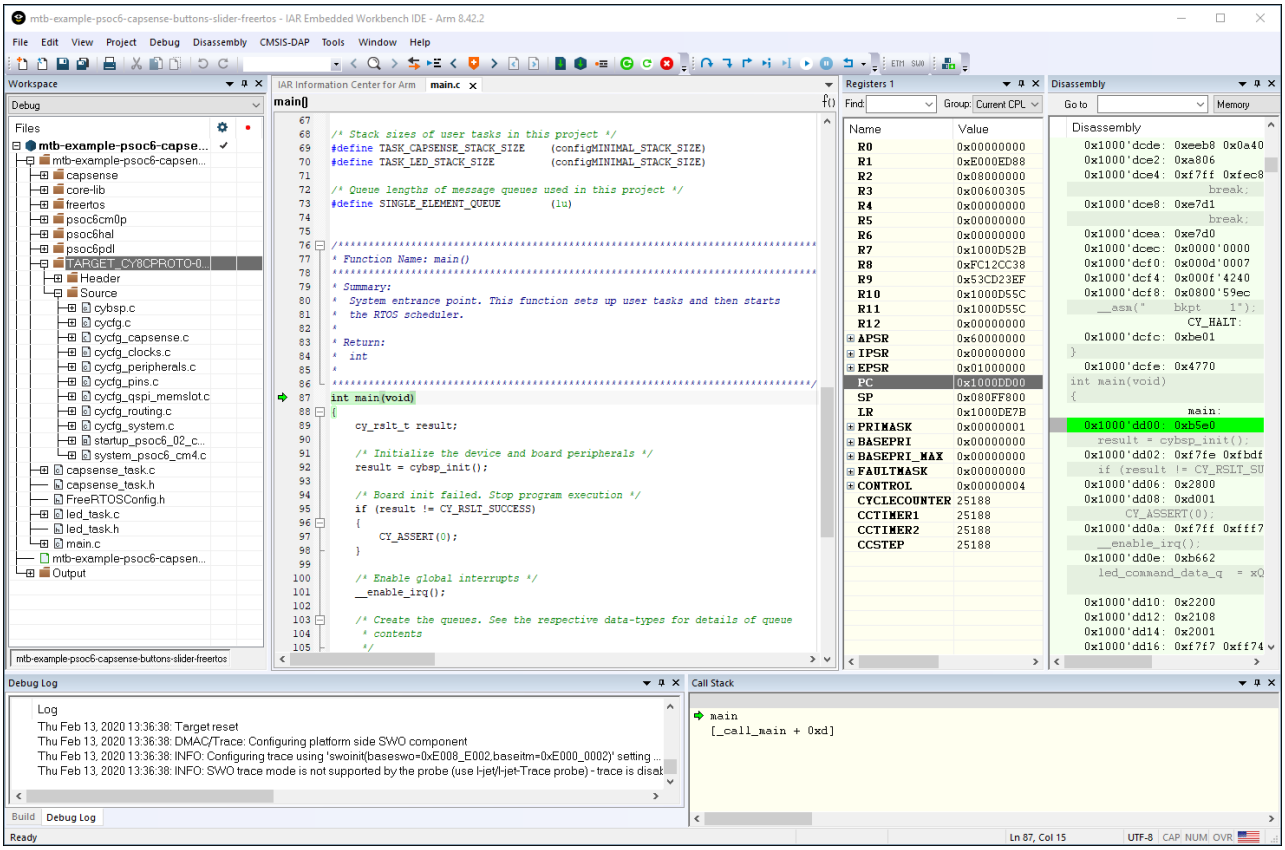
*注: シングルコア PSOC™ 6 マイコンで MiniProg4 を使用する場合、次のように **Setup** タブで **Reset** の特殊なタイプを指定する必要があります。*



3. **OK** をクリックしてください。

4. **Project > Download and Debug** を選択してください。

IAR Embedded Workbench はデバッグセッションを開始し、main 関数にジャンプします。

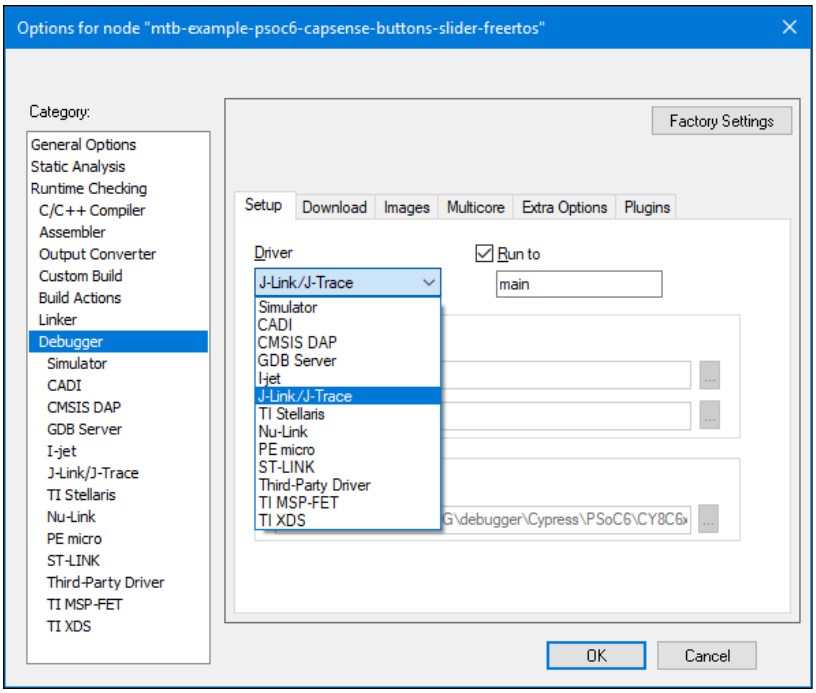


4.3 J-Link を使ったプログラミング/デバッグ

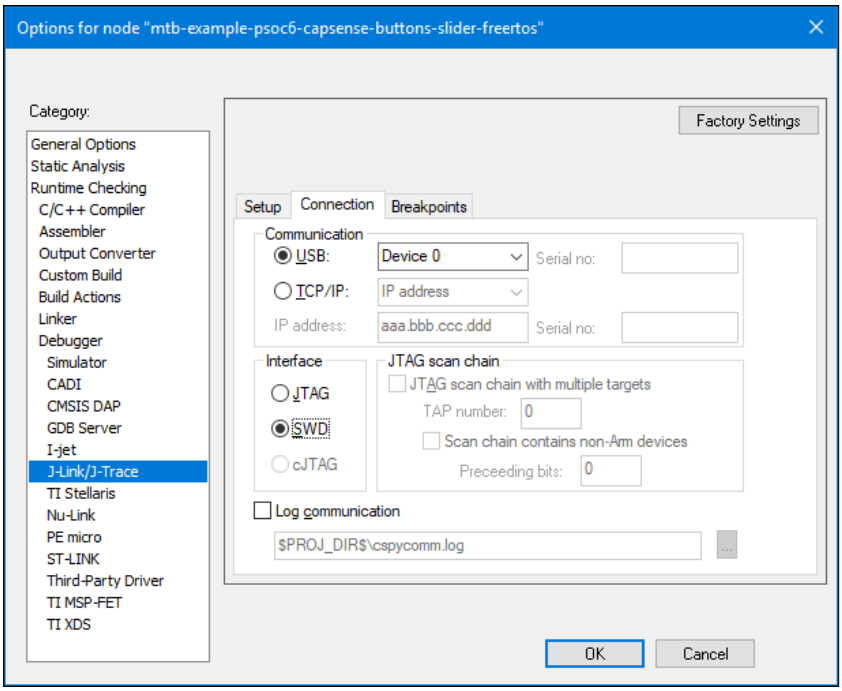
アプリケーションのデバッグには、J-Link デバッガー プローブを使用できます。

1. オプションダイアログを開き、**Category** から **Debugger** の項目を選択してください。

2. 次に **J-Link/J-Trace** をアクティブ ドライバとして選択してください。



3. **カテゴリの J-Link/J-Trace** の項目を選択し、**Connection** タブでインターフェースを **SWD** に切り替えてください。



4. J-Link デバッグプローブを 10 ピンアダプターに接続してください (プロトタイピングキットでは半田付けが必要)。

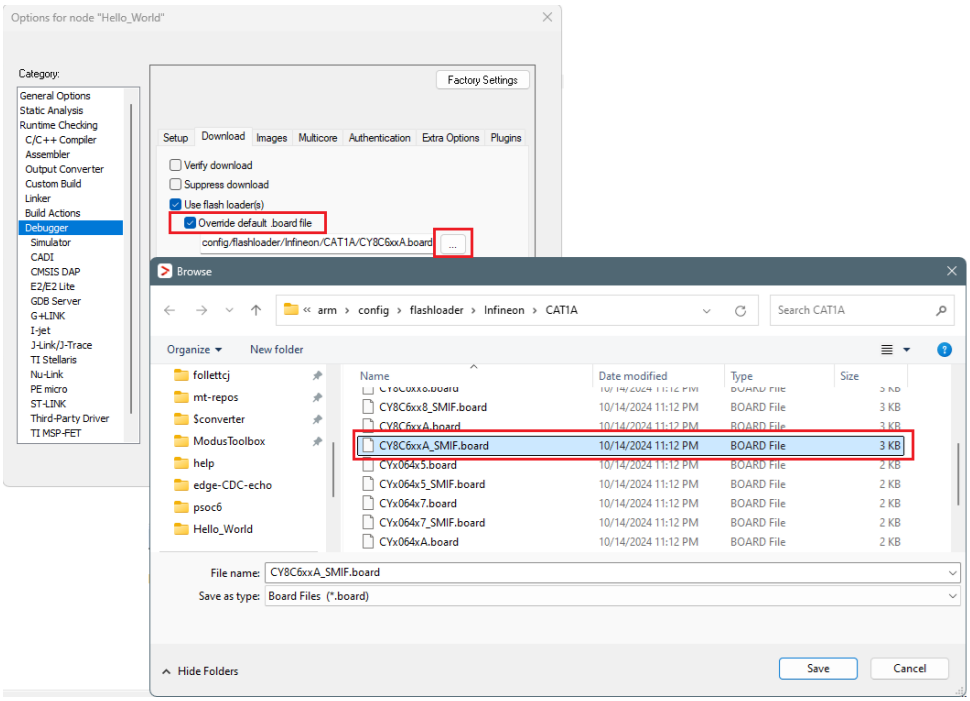
5. **Project > Download and Debug** を選択してデバッガーを起動してください。

4.4 外部メモリのプログラム

IAR Embedded Workbench では、外部メモリプログラミングがデフォルトで無効になっています。PSOC™ 6 デバイスでは、*.board ファイル内の SMIF 領域を手動で有効にする必要があります。手順は次のとおりです。

1. オプションダイアログを開き、**Category** から **Debugger** の項目を選択してください。

2. **Download** タブをクリックし、**Override default .board file** チェックボックスを選択してください。



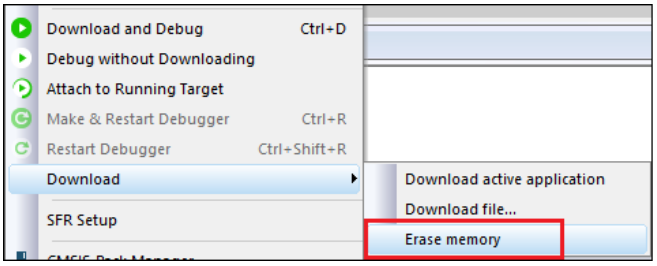
- このプロジェクトで現在使用されているデフォルトの*.board* ファイルを特定してください。
- **Browse [...]**ボタンをクリックし、"SMIF"を含む同じ*.board* ファイルに移動して選択してください。
- **Save** をクリックしてください。

3. **OK** をクリックして、オプション ダイアログを閉じてください。

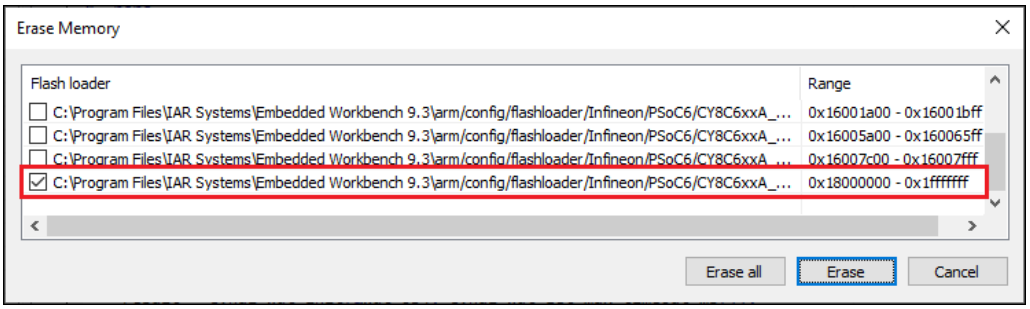
4.5 外部メモリを有効にした PSOC™ 6 MCU の消去

PSOC™ 6 MCU のフラッシュローダーを使用して外部メモリを正常に消去するためには、デバイスの内部フラッシュに有効な QSPI コンフィギュレーション データが含まれていなければいけません。QSPI_XIP の例のように、以前にプログラムされたアプリケーションの一部の場合があります。詳細については、アプリケーションノート AN228740 の 7 章を参照してください。

1. **Project > Download > Erase memory** を選択してください。



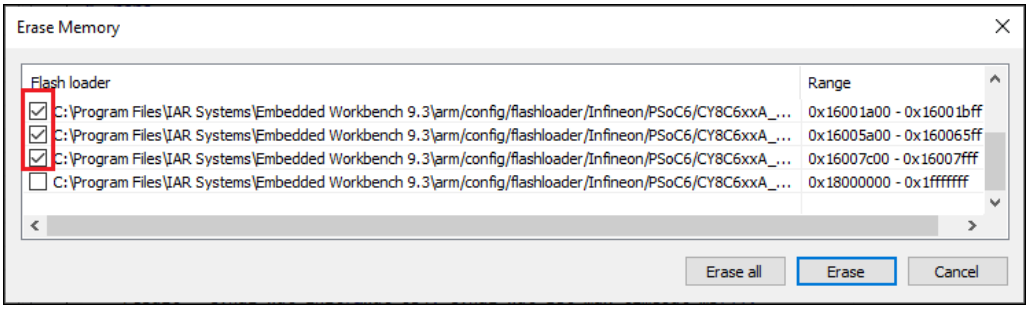
2. 0x18000000-0x1fffffff を除き、選択されたリージョンのチェックボックスをすべて外してください。



3. **Erase** をクリックしてください。

4. **Project > Download > Erase memory** をもう一度選択してください。

5. 他のすべてのリージョンを選択し、0x18000000-0x1fffffff の選択を解除してください。



6. **Erase** をクリックしてください。

5 PSOC™ 64 Secure シングルコア アプリケーション

IAR Embedded Workbench 用アプリケーションの作成/エクスポートの手順に従ってください。PSOC™ 64 Secure MCU の場合、Secure Boot SDK ユーザー ガイド の指示に従って、デバイスのプロビジョニングとキーを生成する必要もあります。

注: PSOC™ 64 アプリケーションには、python と cysecuretools のインストールが必要です。

次に、このセクションの手順を使用して、IAR Embedded Workbench で PSOC™ 64 Secure MCU 用のアプリケーションを設定、ビルド、プログラミング、デバッグしてください。

5.1 設定およびビルド

アプリケーションの作成後、ビルドを試みる前にいくつかの変更を加えます。

1. プロジェクトフォルダにある *Makefile* を以下のように編集してください。

```
APPNAME=Hello_World
# Use the same App_Name used to create the project

TOOLCHAIN=IAR
```

2. ModusToolbox™インストールディレクトリから modus-shell ターミナルを開き、IAR プロジェクト ディレクトリに移動し、次のコマンドを実行してください。

```
make build -j8
```

3. ログからポストビルド コマンドをコピーしてください。以下に例を示します。

```
../mtb_shared/core-make/release-v3.6.1/make/scripts/python.bash C:/examples/psoc64-iar/
Hello_World/bsps/TARGET_APP_CY8CPROTO-064S1-SB/psoc64_postbuild.py --core CM4 --secure-
boot-stage single --policy policy_single_CM0_CM4 --target cyb06xx7 --toolchain-path C:/
Users/follettcj/Infineon/Tools/mtb-gcc-arm-eabi/11.3.1/gcc --toolchain IAR --build-dir C:/
examples/psoc64-iar/Hello_World/build/APP_CY8CPROTO-064S1-SB/Debug --app-name Hello_World
--cm0-app-path ../mtb_shared/cat1cm0p/release-v1.8.0/COMPONENT_CAT1A/
COMPONENT_CM0P_SECURE --cm0-app-name psoc6_01_cm0p_secure
```

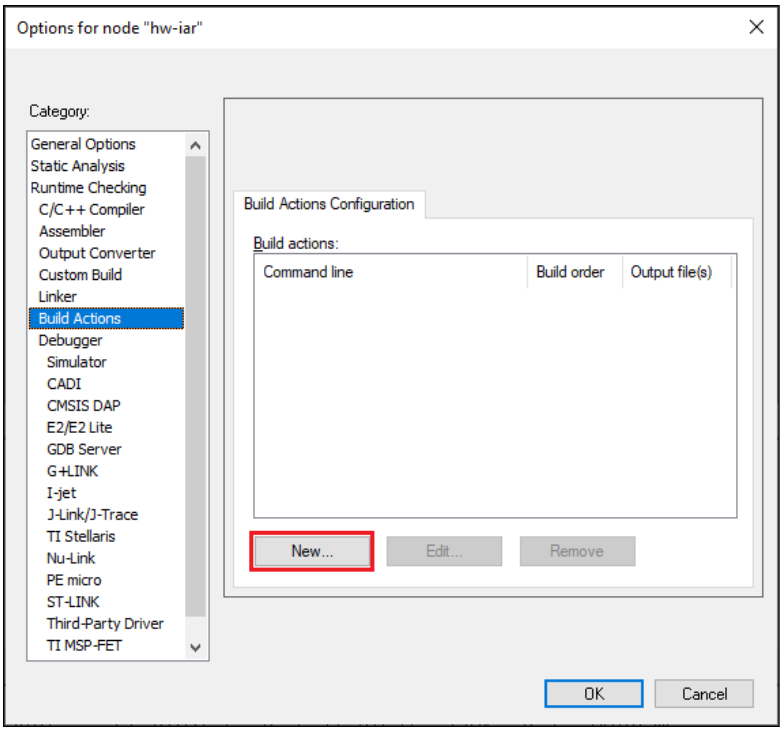
4. このコマンドを適当なエディターに貼り付け、以下のように編集してください。
   - python へのパスを更新
   - ポリシーファイルへのパスを追加。例: `--policy-path ./policy`
   - `--build-dir` を `Debug/Exe` に変更
   - 必要であれば、`--app-name` を IAR 名のプロジェクトに変更

編集後のコマンド例:

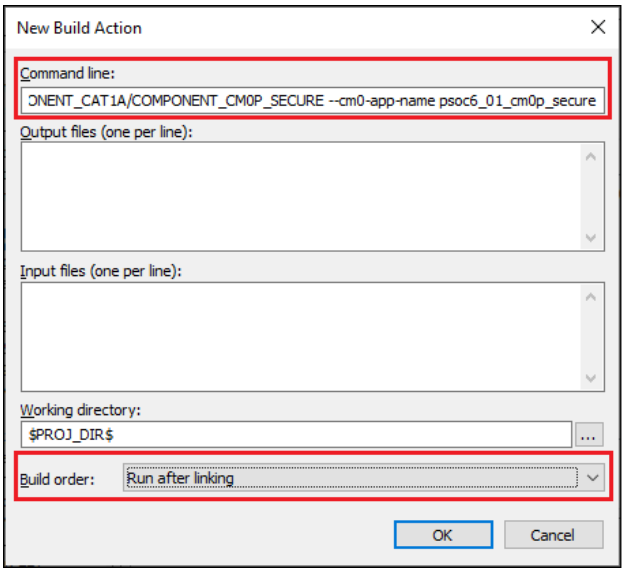
```
[path to python.exe] C:/examples/psoc64-iar/Hello_World/bsps/TARGET_APP_CY8CPROTO-064S1-SB/psoc64_postbuild.py --core CM4 --secure-boot-stage single --policy-path ./policy --policy policy_single_CM0_CM4 --target cyb06xx7 --toolchain-path C:/Users/follettcj/Infineon/Tools/mtb-gcc-arm-eabi/11.3.1/gcc --toolchain IAR --build-dir Debug/Exe --app-name Hello_World --cm0-app-path ../mtb_shared/cat1cm0p/release-v1.8.0/COMPONENT_CAT1A/COMPONENT_CM0P_SECURE --cm0-app-name psoc6_01_cm0p_secure
```

5. 編集したコマンドをコピーしてください。

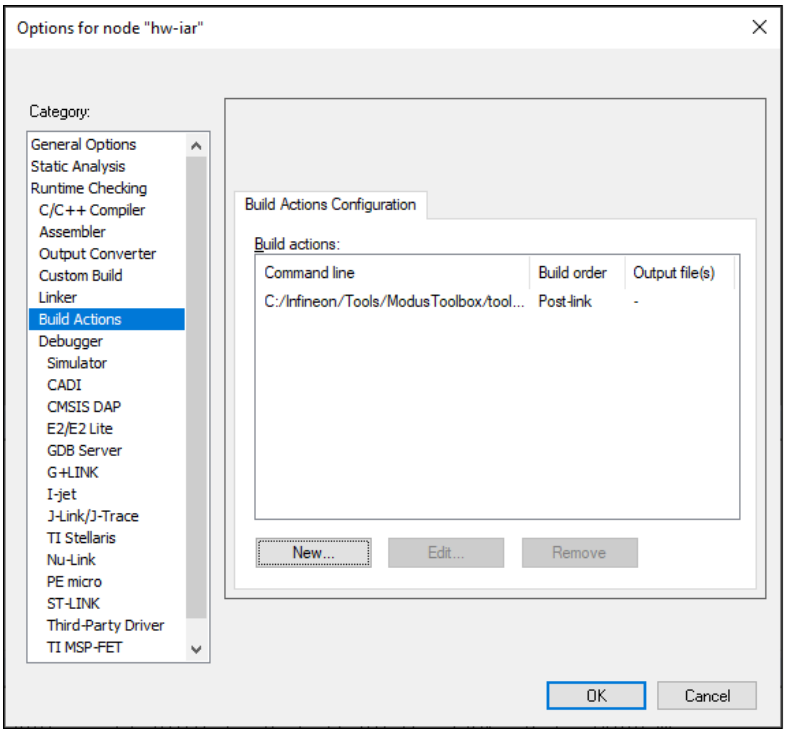
6. IAR embedded Workbench で、Options ダイアログを開き、**Build Actions** に進み、**New** をクリックしてください。



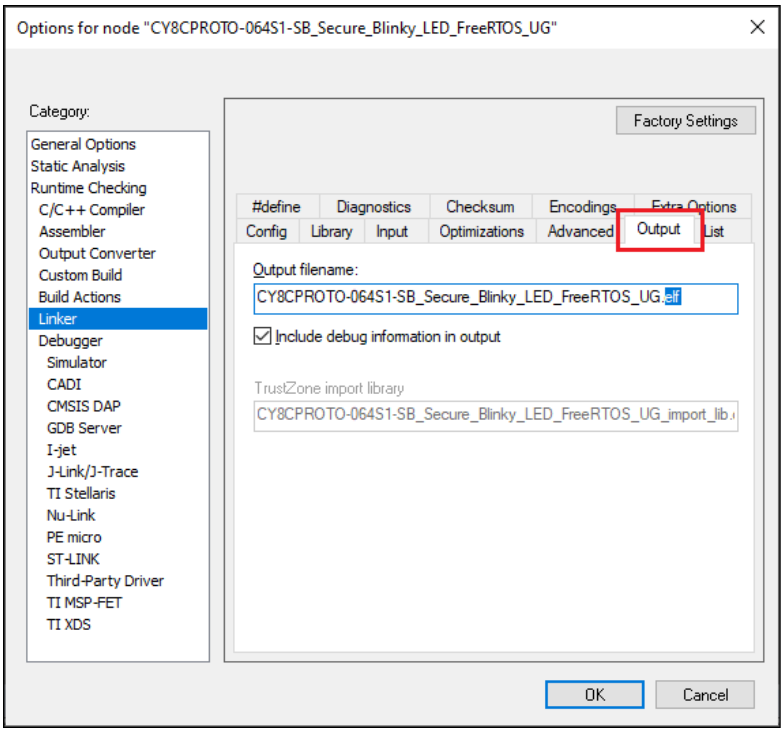
7. New Build Action ダイアログで、編集したコマンドを **Command line** フィールドに貼り付け、**Build order** フィールドで「Run after linking」を選択し、**OK** をクリックしてください。



8. **OK** をクリックしてダイアログを閉じてください。



9. **Linker > Output** に切り替え、**Output filename** フィールドのファイル拡張子を「.out」から「.elf」に変更してください。



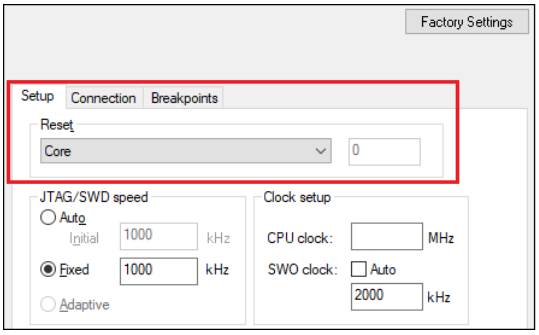
10. **OK** をクリックしてダイアログを閉じてください。

11. IAR のメインメニューで、**Project > Make** を選択し、アプリケーションをビルドしてください。

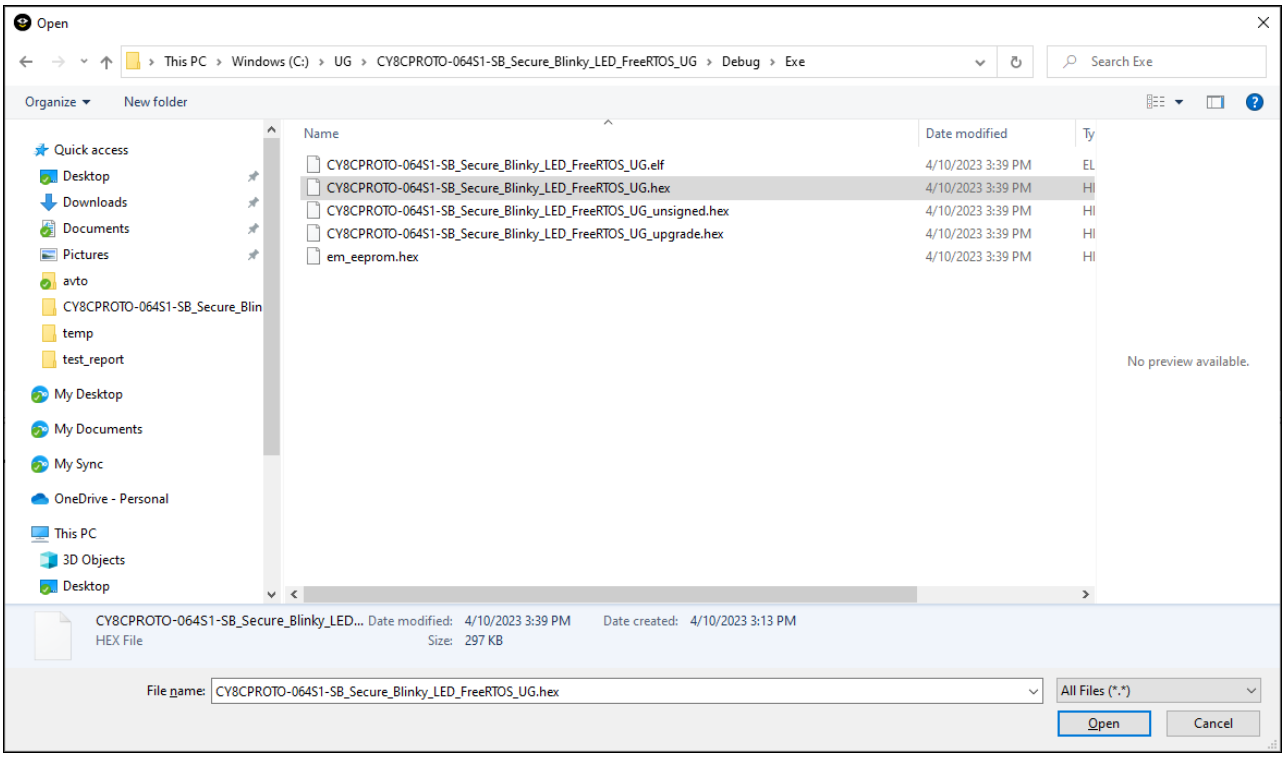
5.2 プログラミングおよびデバッグ

1. CMSIS-DAP または J-Link のデフォルトのデバッガー オプションを設定できます。KitProg3/MiniProg4 (CMSIS-DAP) でのプログラム/デバッグまたは J-Link を使ったプログラミング/デバッグを参照してください。

   ***注**: PSOC™ 64「Secure Boot」MCU で J-Link を使用する場合、**J-Link > Setup** タブで **Reset**「Core」という特別なタイプを指定する必要があります。*



2. **Project > Download > Download file...**を選択し、*<project_root>\Debug\Exe* 内の*<project_name>.hex* ファイルを選択してください。



3. **Project > Debug without Downloading** を選択してください。

6 AIROC™ CYW20829 アプリケーション

IAR Embedded Workbench 用アプリケーションの作成/エクスポートの手順に従ってください。

次に、このセクションの説明に従って、IAR Embedded Workbench で AIROC™ CYW20829 デバイス用のアプリケーションを設定、ビルド、プログラム、デバッグしてください。

6.1 設定およびビルド

アプリケーションの作成後、ビルドを試みる前にいくつかの変更を加えます。

1. プロジェクトフォルダにある *Makefile* を以下のように編集してください。

```
APPNAME=Hello_World
# Use the same App_Name used to create the project

TOOLCHAIN=IAR
```

2. 任意のコードエディターを使って、IAR プロジェクトのディレクトリに *postbuild.bat* ファイルを作成してください。

3. ファイルの先頭に `SET PATH=%PATH%;"/usr/bin"`を追加してください。

4. ModusToolbox™インストールディレクトリから modus-shell ターミナルを開き、IAR プロジェクト ディレクトリに移動し、次のコマンドを実行してください。

```
make build -j8
```

5. IAR 関連のビルド後の出力を modus-shell ターミナルからコピーしてください。以下に例を示します。

```
C:/iar/ewarm-9.60.3/arm/bin/ielftool C:/examples/20829/Hello_World/build/
APP_CYW920829M2EVK-02/Debug/Hello_World.elf --bin-multi C:/examples/20829/Hello_World/
build/APP_CYW920829M2EVK-02/Debug/Hello_World.bin; C:/iar/ewarm-9.60.3/arm/bin/
ielfdumparm  -a C:/examples/20829/Hello_World/build/APP_CYW920829M2EVK-02/Debug/
Hello_World.elf > C:/examples/20829/Hello_World/build/APP_CYW920829M2EVK-02/Debug/
Hello_World.dis; "[install-path]/ModusToolbox/tools_[version]/modus-shell/bin/bash.exe" --
norc --noprofile ../mtb_shared/recipe-make-cat1b/release-v2.8.0/make/scripts/20829/
flash_postbuild.sh "IAR" "C:/examples/20829/Hello_World/build/APP_CYW920829M2EVK-02/
Debug" "Hello_World" "[install-path]/Infineon/Tools/mtb-gcc-arm-eabi/11.3.1/gcc/bin" "C:/
Users/follettcj/ModusToolbox/tools_3.5/srecord/bin/srec_cat.exe" "0x00003A00"; "[install-
path]/ModusToolbox/tools_[version]/modus-shell/bin/bash.exe" --norc --noprofile ../
mtb_shared/recipe-make-cat1b/release-v2.8.0/make/scripts/20829/run_toc2_generator.sh
"NON_SECURE" "C:/examples/20829/Hello_World/build/APP_CYW920829M2EVK-02/Debug"
"Hello_World" "flash" "bsps/TARGET_APP_CYW920829M2EVK-02" "NONE" "[install-path]/Infineon/
Tools/mtb-gcc-arm-eabi/11.3.1/gcc" "" 0x20000 0 "" "0x00003A00" "256"; [install-path]/
ModusToolbox/tools_[version]/srecord/bin/srec_cat.exe C:/examples/20829/Hello_World/build/
APP_CYW920829M2EVK-02/Debug/Hello_World.final.bin -Binary -offset 0x60000000 -o C:/
examples/20829/Hello_World/build/APP_CYW920829M2EVK-02/Debug/Hello_World.final.hex -Intel
-Output_Block_Size=16;
```

6. ビルド後の出力を *postbuild.bat* ファイルに貼り付け、以下の編集を行い、ファイルを保存してください。

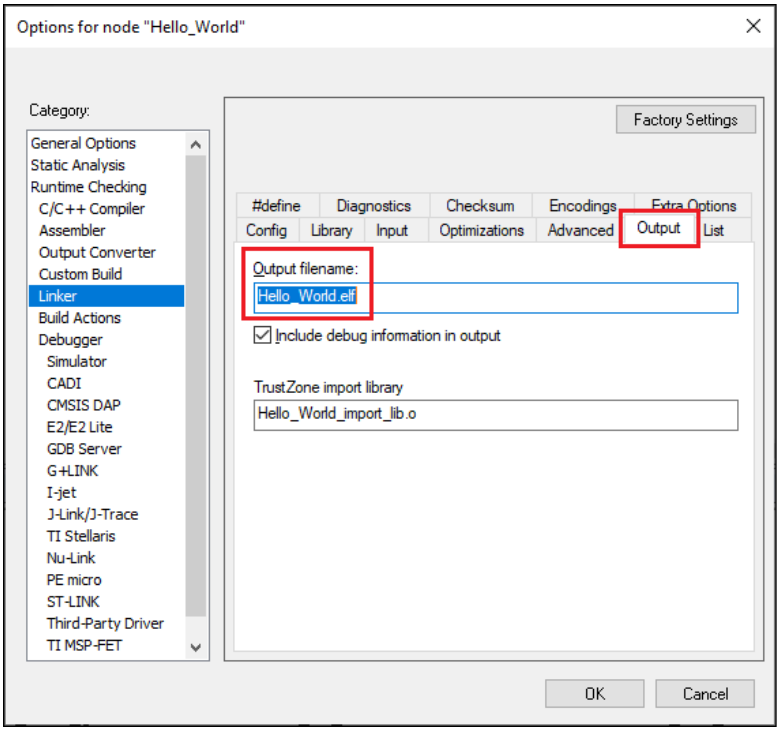
- パスを「build/APP_CYW920829M2EVK-02/Debug/...」から「Debug/Exe/...」に変更してください。
- セミコロンをすべて取り除いてください。セミコロンごとに改行することを推奨します。
- 不要なコマンド (elfdumparm、rm -rf、cp など) を削除してください。

編集ファイルの例:

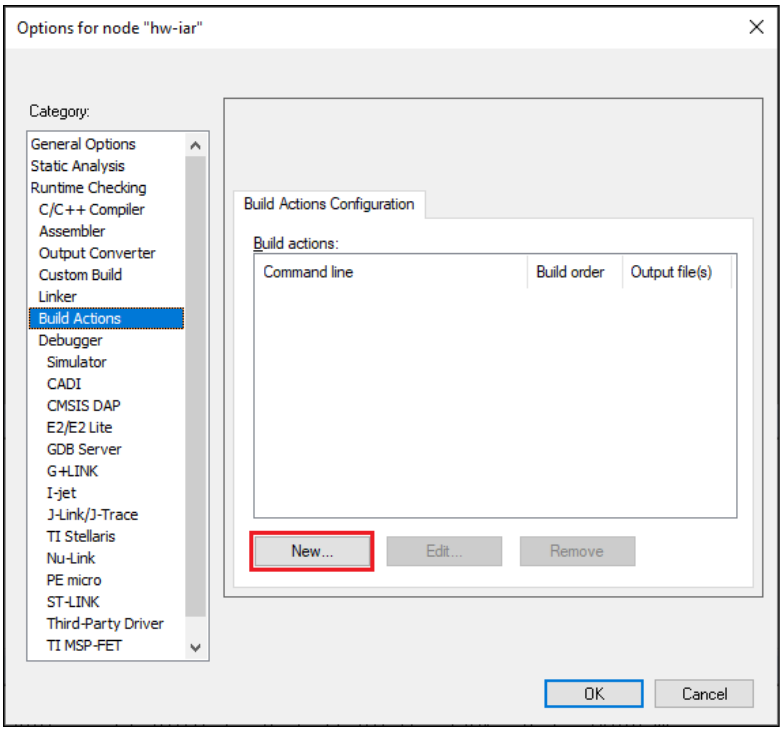
```
PATH=%PATH%;"/usr/bin"

C:/iar/ewarm-9.60.3/arm/bin/ielftool C:/examples/20829/Hello_World/Debug/Exe/
Hello_World.elf --bin-multi C:/examples/20829/Hello_World/Debug/Exe/Hello_World.bin

"[install-path]/ModusToolbox/tools_[version]/modus-shell/bin/bash.exe" --norc --
noprofile ../mtb_shared/recipe-make-cat1b/release-v2.8.0/make/scripts/20829/
flash_postbuild.sh "IAR" "C:/examples/20829/Hello_World/Debug/Exe" "Hello_World"
"[install-path]/Infineon/Tools/mtb-gcc-arm-eabi/11.3.1/gcc/bin" "[install-path]/
ModusToolbox/tools_[version]/srecord/bin/srec_cat.exe" "0x00003A00"

"[install-path]/ModusToolbox/tools_[version]/modus-shell/bin/bash.exe" --norc --
noprofile ../mtb_shared/recipe-make-cat1b/release-v2.8.0/make/scripts/20829/
run_toc2_generator.sh "NON_SECURE" "C:/examples/20829/Hello_World/Debug/Exe"
"Hello_World" "flash" "bsps/TARGET_APP_CYW920829M2EVK-02" "NONE" "[install-path]/Infineon/
Tools/mtb-gcc-arm-eabi/11.3.1/gcc" "" 0x20000 0 "" "0x00003A00" "256"

[install-path]/ModusToolbox/tools_[version]/srecord/bin/srec_cat.exe "Debug/Exe/
Hello_World.final.bin" -Binary -offset 0x60000000 -o "Debug/Exe/Hello_World.final.hex"
-Intel -Output_Block_Size=16
```

注: お使いのシステムに合わせてパスを更新してください。

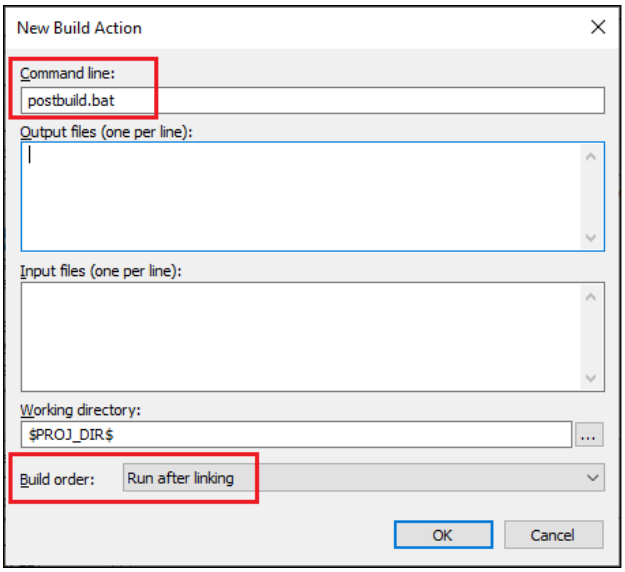
7. IAR Embedded Workbench のオプションダイアログを開き、**Linker > Output** に進み、**Output filename** フィールドのファイル拡張子を「.out」から「.elf」に変更してください。

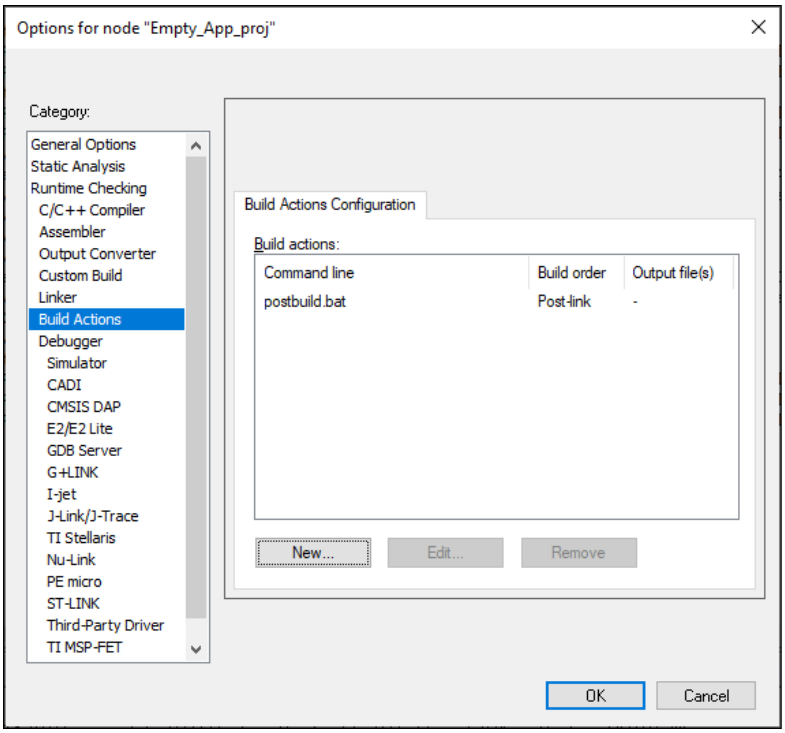


8. **Build Actions** に切り替え、**New** をクリックしてください。



9. 新しいビルド アクション ダイアログで、*postbuild.bat* ファイルを **Command line** フィールドに入力し、**Build order** フィールドで「Run after linking」を選択し、**OK** をクリックしてください。





10. **OK** をクリックして、オプション ダイアログを閉じてください。

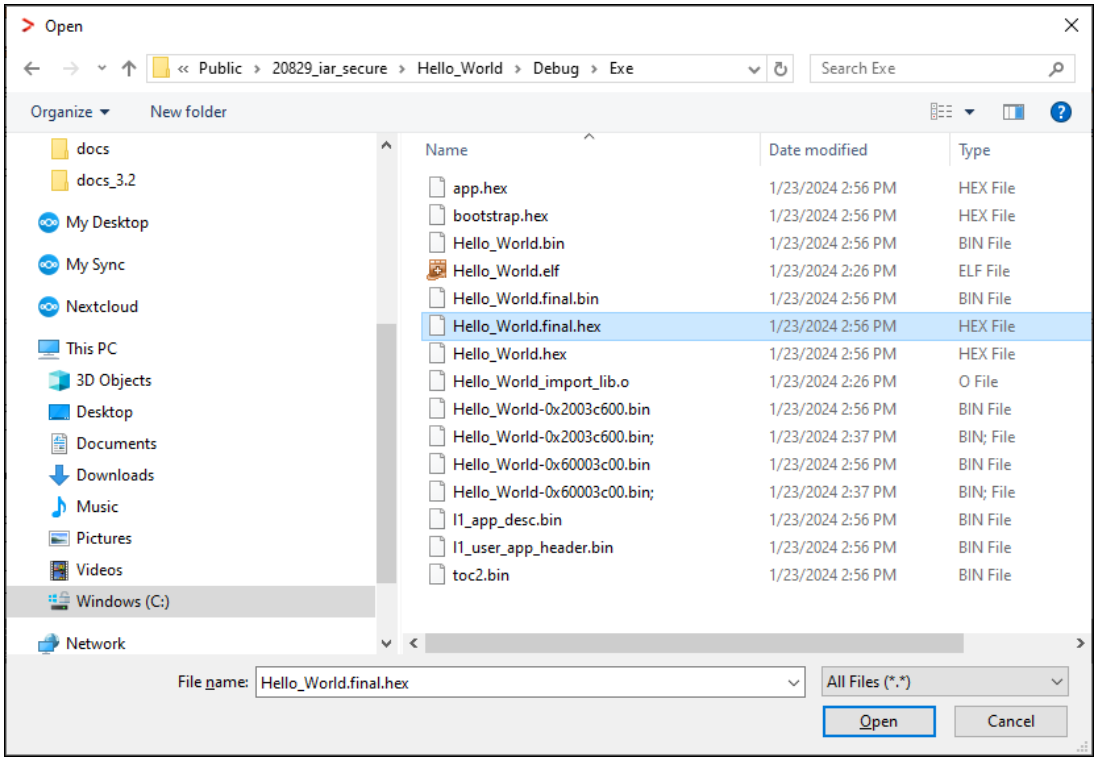
11. IAR のメインメニューで、**Project > Make** を選択し、アプリケーションをビルドしてください。以下のように表示されます。

```
Total number of errors: 0
Total number of warnings: 0
Resolving dependencies...
Build succeeded
```

6.2 プログラミングおよびデバッグ

Project > Download > Download file...を選択し、*<project_root>DebugExe* 内の*<project_name>.final.hex* ファイルを選択してください [All Files (*.*) への切り替えが必要な場合があります]。

6 AIROC™ CYW20829 アプリケーション



Project >Debug without Downloading を選択してください。

7 PSOC™ Control C3 セキュア アプリケーション

IAR Embedded Workbench 用アプリケーションの作成/エクスポートの手順に従ってください。プロセスの最後には、コンソールにメッセージが表示されます。PSOC™ Control C3 デバイスの場合、メッセージの 1 つが以下のように表示されます。

```
The project is an ARM Trustzone secure project. The following need to be enabled in Embedded
Workbench IDE.
    Project->Options->General Options->32 bit->TrustZone
```

ほとんどの場合、PSOC™ Control C3 のアプリケーションは、デフォルトで TrustZone セキュアに設定されています。TrustZone テクノロジーの詳細については、Arm®のウェブサイトを参照してください。https://www.arm.com/technologies/trustzone-for-cortex-m.

IAR Embedded Workbench でアプリケーションを開く際、TrustZone の設定を確認する必要があります。オプション ダイアログを開き、**General Options > 32-bit** に進み、TrustZone **Mode** が「Secure」に設定されていることを確認してください。

アプリケーションを保存し、**Project > Make** を選択してビルドしてください。Output は進捗状況を表示し、次のようなテキストで終わります。

```
Total number of errors: 0
Total number of warnings: 0
Resolving dependencies...
Build succeeded
```

Arm 社の TrustZone 技術に加え、PSOC™ Control C3 デバイスには様々なセキュリティ ライフサイクル ステージ (LCS) があります。セキュリティの詳細については、アプリケーションノート AN240106 - Getting started with PSOC™ Control C3 security を参照してください。

以下のセクションでは、アウトオブボックス (out-of-the-box) というポリシーを持つデバイスと、プロビジョニングされたデバイスの操作について詳しく説明します。

***注**: デフォルトでは、サンプルコードから作成されたアプリケーションのデフォルトは TrustZone の「セキュア」モードです。PSOC™ Control C3 セキュリティの詳細については、アプリケーションノート AN240106 - Getting started with PSOC™ Control C3 security を参照してください。*

7.1 デフォルトのポリシーが適用されるデバイス

デバイスはデフォルトのポリシーで出荷されるため、セキュリティやコード署名に関する知識がなくても、アプリケーションの開発とデバッグを繰り返すことができます。これは開発 LCS と呼ばれます。この状態では、プログラミングやデバッグの前に何も設定する必要はありません。プログラミングおよびデバッグ セクションに進んでください。

7.2 プロビジョニングされたデバイス

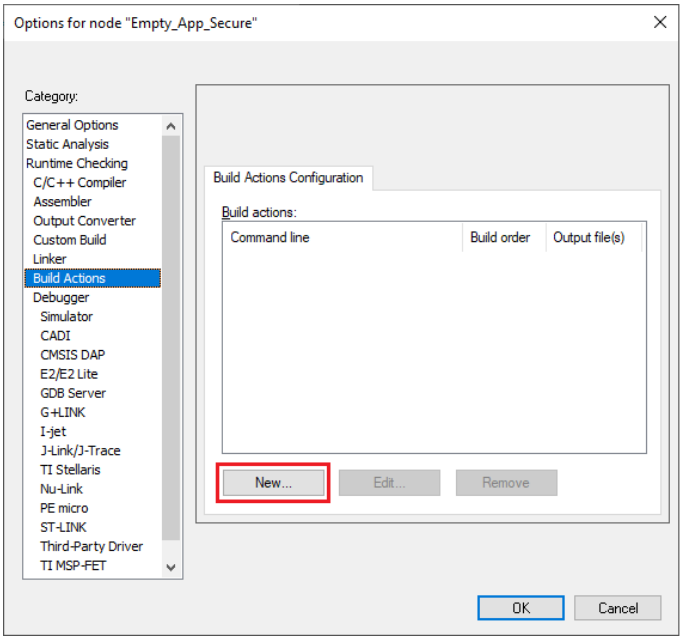
デバイスをプロビジョニングした場合は、ModusToolbox™ Edge Protect Security Suite を使用して、プロビジョニング時に使用したのと同じキーで HEX ファイルに署名する必要があります。これを行うには、*postbuild.bat* ファイ

ルに以下のようなコマンドすべてを 1 行で追加してください。設定に合わせてパス、ファイル名、キーを更新してください。

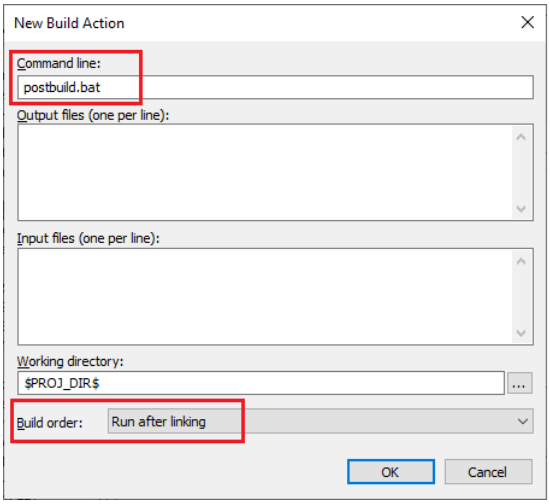
```
[path-to-edgeprotecttools]\bin\edgeprotecttools sign-image --image "Debug/Exe/Empty_App.hex" --output "Debug/Exe/Empty_App.hex" --header-size 0x400 --slot-size 0x20000 --key keys/oem_rot_priv_key_0.pem --hex-addr 0x32000000 --align 1 --pad --security-counter 0 --erased-val 0 --public-key-format full --pubkey-encoding raw --signature-encoding raw --min-erase-size 0x200 --overwrite-only
```

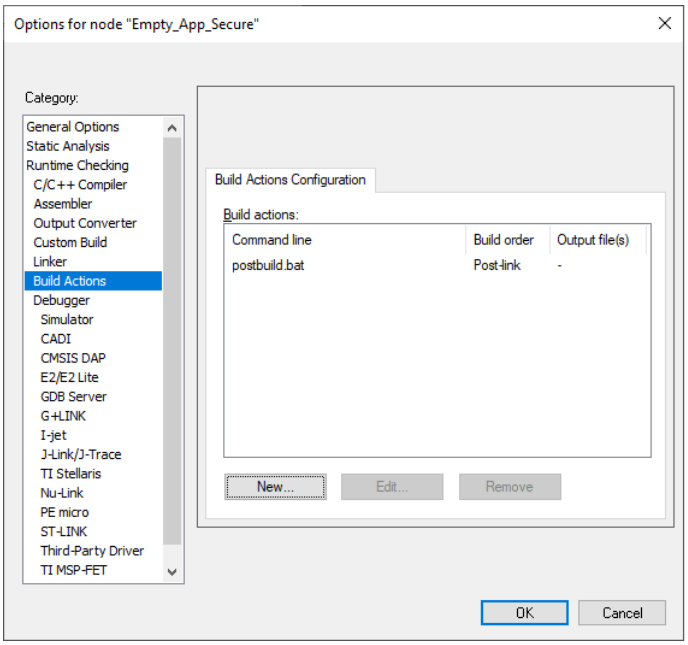
次に、プログラミングとデバッグを行う前に、デバイスを設定するためにいくつかの設定を行います。

1. オプションダイアログで **Build Actions** を開き、**New** をクリックしてください。

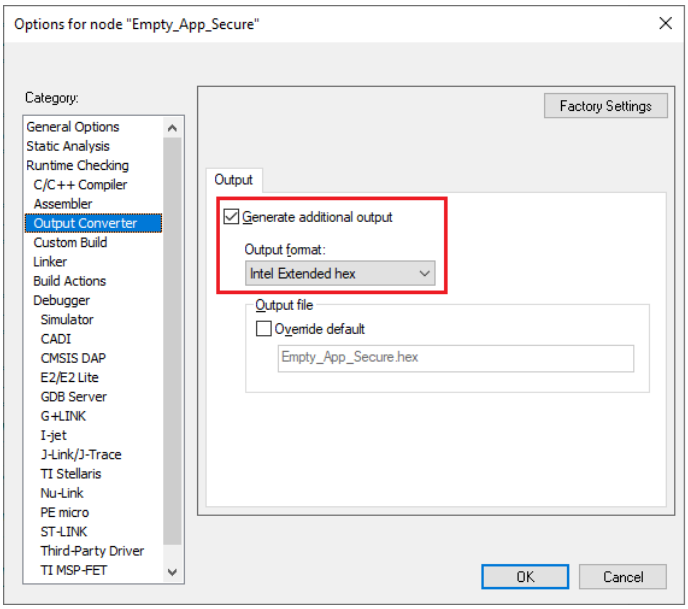


2. 新しいビルド操作ダイアログで、*postbuild.bat* ファイルを **Command line** フィールドに入力し、**Build order** フィールドで「Run after linking (リンク後に実行)」を選択し、**OK** をクリックしてください。





3. **Output Converter** に切り替え、**Generate additional output** を有効にし、**Output format** で「Intel Hex ファイル」を選択してください。



4. **OK** をクリックして、オプション ダイアログを閉じてください。

5. IAR のメインメニューで、**Project > Make** を選択し、アプリケーションをビルドしてください。

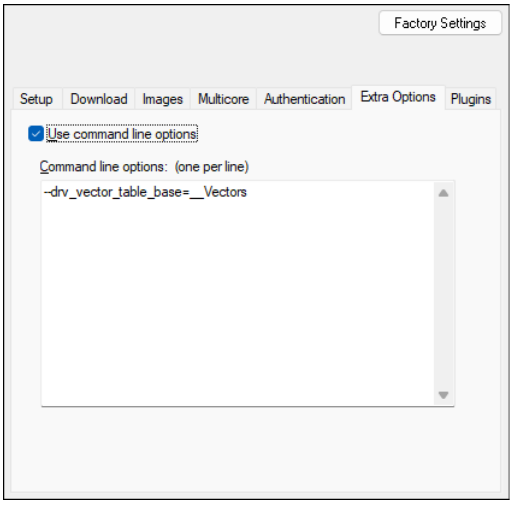
プログラミングおよびデバッグセクションに進んでください。

7.3 プログラミングおよびデバッグ

プログラミングとデバッグを行う前に、目的のプローブのデフォルトのデバッガー オプションを設定します。KitProg3/MiniProg4 (CMSIS-DAP) でのプログラム/デバッグまたは J-Link を使ったプログラミング/デバッグを参照してください。

1. オプションダイアログの **Extra Options** タブを開き、**Use command line options** チェックボックスを選択し、以下のコマンドラインオプションを貼り付けてください。

```
--drv_vector_table_base=__Vectors
```

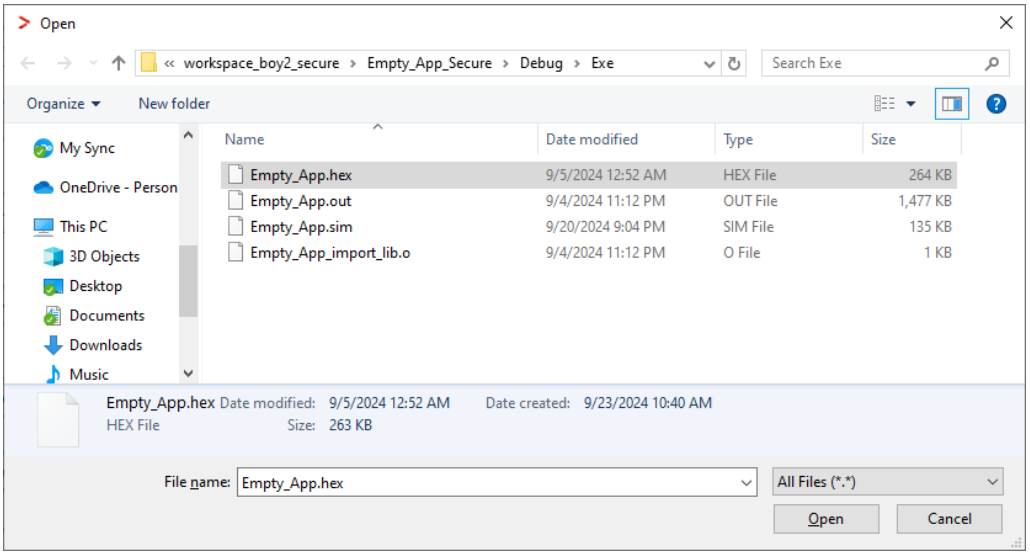


これは、C スタートアップコードのベクタテーブルの IAR 名が無効という問題を解決するのに必要です。

2. **OK** をクリックして、オプション ダイアログを閉じてください。

3. **Project > Build Target** を選択してアプリケーションをビルドしてください。

4. **Project > Download > Download file...**を選択し、*<project_root>DebugExe* 内の*<project_name>.hex* ファイルを選択してください。[**All Files (*.*)**に切り替えが必要な場合があります]。



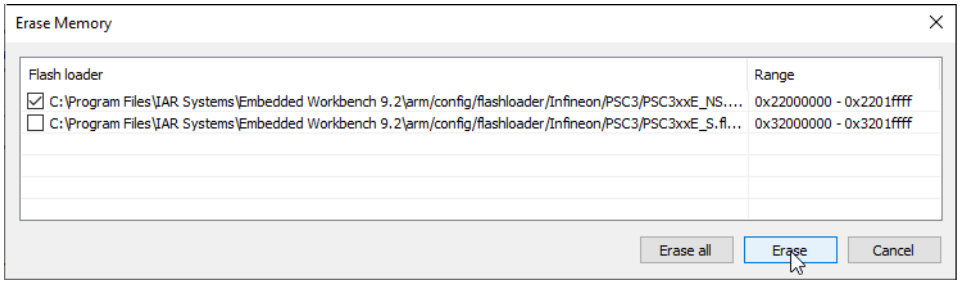
5. **Project >Debug without Downloading** を選択してください。

7 PSOC™ Control C3 セキュア アプリケーション

ターゲットのプロビジョニングに使用されたポリシーに基づいて、フラッシュにアクセス制限が発生することがあります。同じ物理フラッシュ メモリは、非セキュア アドレス (0x22000000-0x2203FFFF) またはセキュア アドレス (0x32000000-0x3203FFFF) 上で S-BUS を介して消去/プログラムできます。

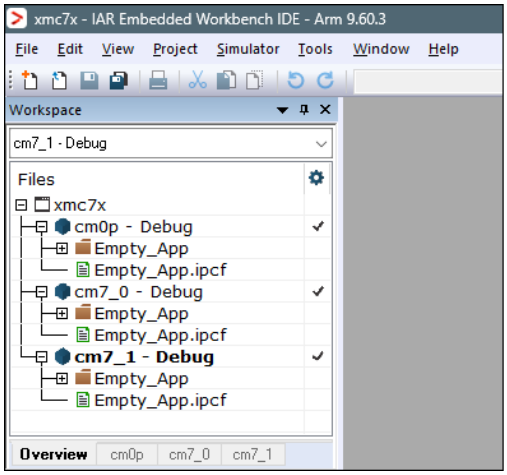
フラッシュ プログラミング動作中、CPU コアは Secure world で動作します。この状態では、非セキュアアドレスには常にアクセス可能ですが、特定のリージョンが非セキュアアクセス用に設定されている場合、セキュアアドレスを利用できないことがあります (PSOC™コントロール C3 では、非セキュアアクセスのみを意味します)。

フラッシュ メモリ全体を消去するためには、セキュアでない範囲のアドレスだけを消去することが重要です。セキュアアドレスの範囲を選択解除するだけです。



8 PSOC™ 6 および XMC7000/TRAVEO™ II マルチコア アプリケーション

IAR Embedded Workbench 用アプリケーションの作成/エクスポートの手順に従ってください。完了すると、以下のようなマルチコアのワークスペースができます。

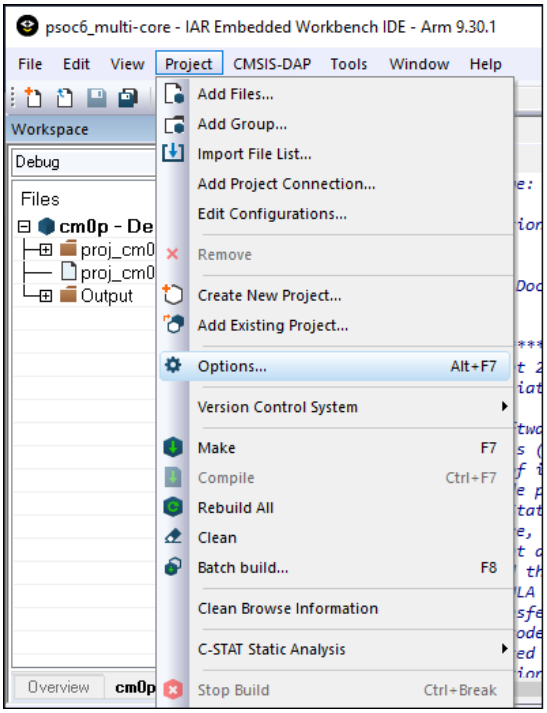


次に、このセクションの手順を使用して、IAR Embedded Workbench で PSOC™ 6 および XMC7000 マルチコア アプリケーションの設定、ビルド、プログラミング、デバッグを行います。

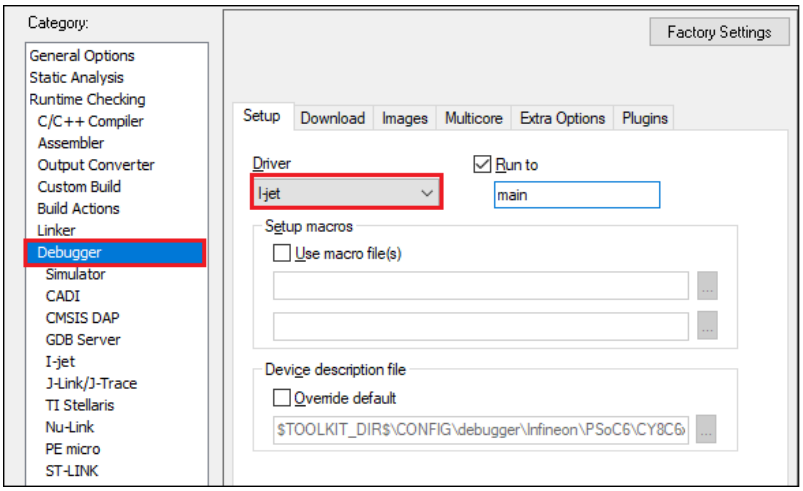
マルチコア デバッグ セッションを開始する前に、ワークスペース内のすべてのプロジェクトが適切に設定されていなければいけません。IAR には「マスター」と「スレーブ」という概念があります。CM0+コア プロジェクトをマスター プロジェクトとして設定し、他のコア (PSOC™ 6 の CM4 および XMC7000 の CM7) をスレーブ プロジェクトとして設定してください。

8.1 CM4/CM7 (スレーブ) コアの設定

1. CM4/CM7 コアプロジェクトを選択し、**Project > Options** に進んでください。



2. ダイアログで、**Setup** タブの **Debugger** カテゴリを選択し、適切なドライバ (I-jet、CMSIS-DAP、J-Link) を選択してください。



3. HEX ファイル生成を有効にしてください。
   - **Runtime Checking > Output Converter** カテゴリで、**Generate additional output** チェックボックスを選択してください。
   - **Output format** が確実に **Intel Extended hex** に設定されているようにしてください。
   - **OK** をクリックしてください。

4. CM4/CM7 (トリプルコア MCU 用) のすべてのプロジェクトについて、これらの手順を繰り返してください。

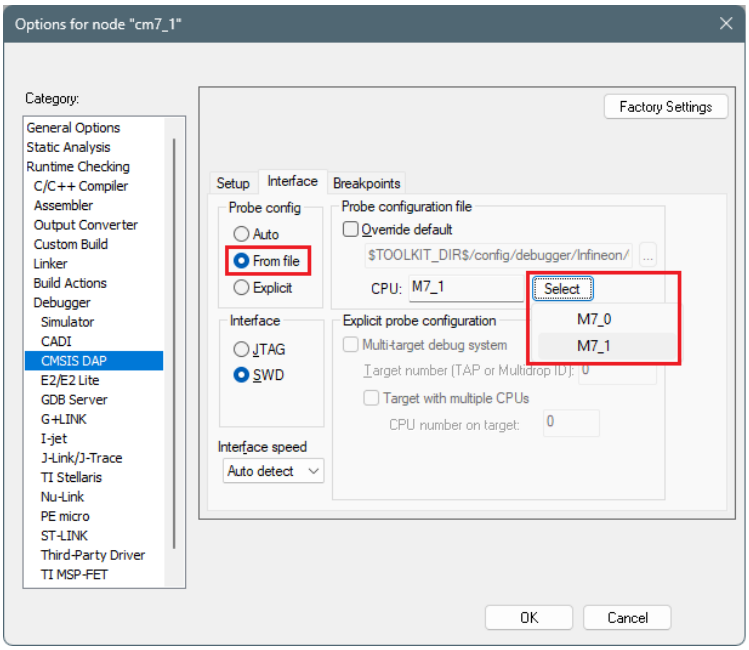
8.1.1 XMC7000/TRAVEO™ II の具体的な手順

一部の XMC7000 MCU はトリプルコア デバイスです。IAR ワークスペースで 2 番目の CM7 コアを使用する場合、このプロジェクトが 2 つ目の CM7 コアを ターゲットにしているのを IAR が理解できるように、プロジェクト設定でターゲットコアを暗黙的に設定する必要があります。デフォルトでは、IAR は最初の CM7 コアに接続するので、ターゲットコアの指定は省略できます。

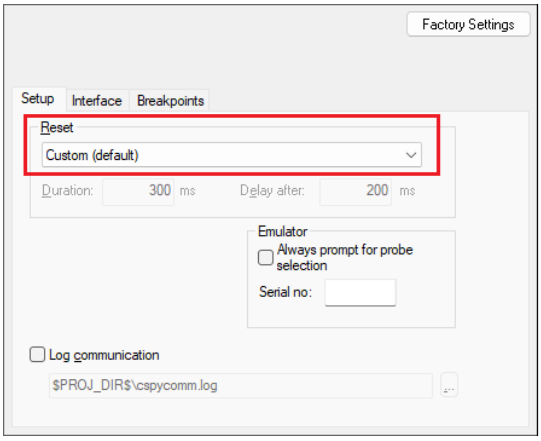
1. 2 番目の CM7 コアのプロジェクトを選択し、**Project > Options** に進んでください。

2. **Debugger** カテゴリーでプローブを選択し、**Interface** タブに切り替えてください。

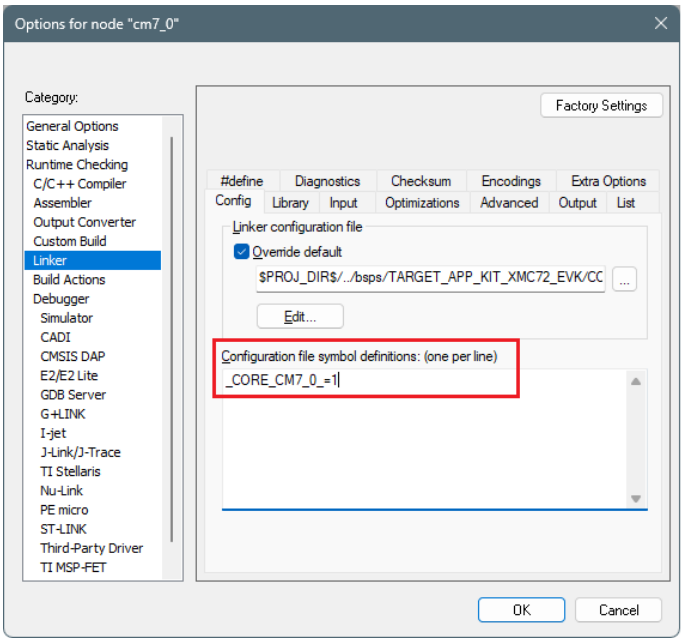
3. **From file** ラジオボタンを選択し、**CPU** ラベルの横にある **Select** をクリックし、**M7_1** を選択してください。



4. **Setup** タブに切り替え、**Reset** プルダウンメニューから「カスタム」を選択してください。



5. さらに、2 つの CM7 コア用に 1 つのリンカースクリプトがあるため、プロジェクト設定で特別なリンカースクリプトシンボルを指定し、CM7_0 と CM7_1 を区別してください。
   a. 最初の CM7 コアのプロジェクトを選択し、**Project > Options > Linker** に進んでください。
   b. **Configuration file symbol definitions** フィールドに `_CORE_CM7_0_=1` を追加し、**OK** をクリックしてください。



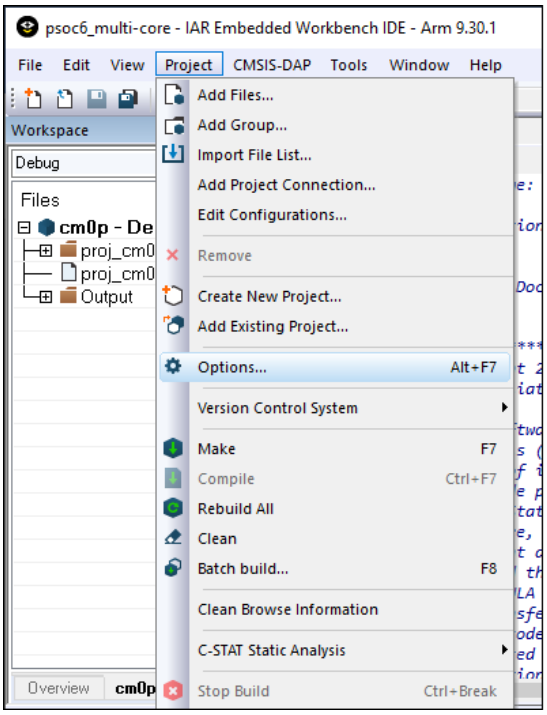
6. 2 番目の CM7 コアに対しても同様です。
   a. 2 番目の CM7 コアのプロジェクトを選択し、**Project > Options > Linker** に進んでください。
   b. **Configuration file symbol definitions field** に `_CORE_CM7_1_=1` を追加し、**OK** をクリックしてください。

   ***注**: CM4/CM7 コアをスタンドアロンでデバッグする場合、デバッグセッションを起動しても CM0+ イメージがロードされるだけで、CM0+ プロジェクトはビルドされないため、変更があった場合は必ず CM0+ プロジェクトをリビルドしてください。*

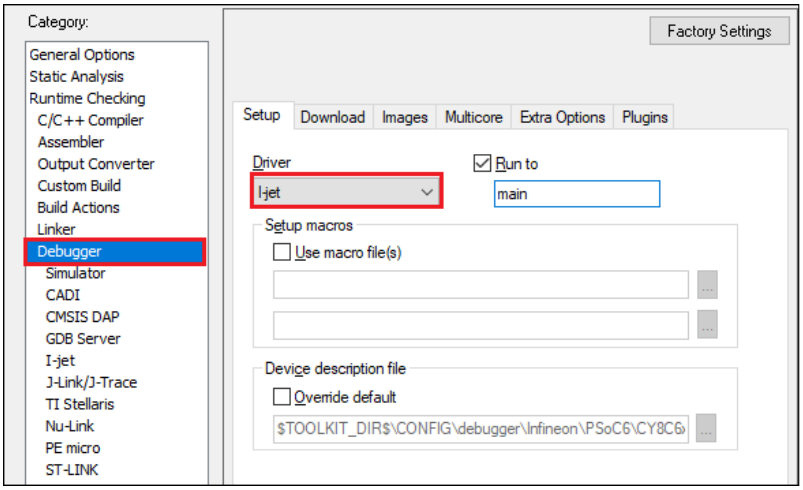
7. 次に進む前に、CM7 プロジェクトをビルドしてください。

8.2 CM0+ (マスター) コアの設定とビルド

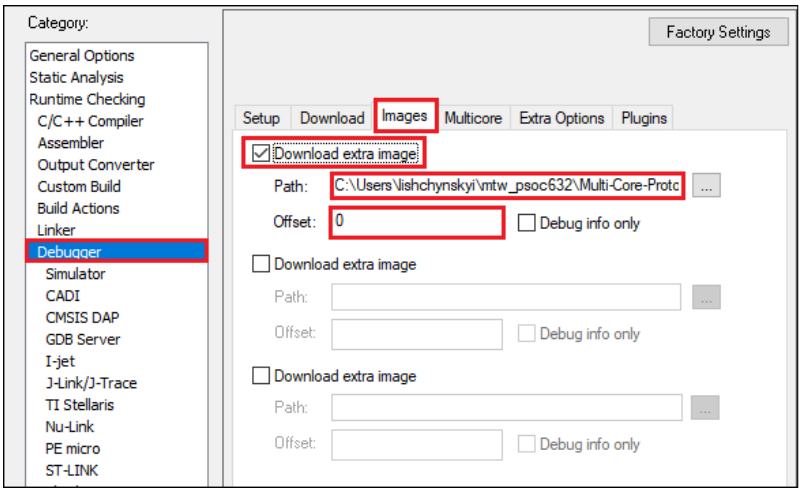
1. CM0+プロジェクトを選択し、**Project > Options** に進んでください。



2. ダイアログで、**Setup** タブ内の **Debugger** カテゴリを選択し、該当する **Driver** (I-jet、CMSIS-DAP、J-Link) を選択してください。



3. **Images** タブに切り替えて、デバッグの前にダウンロードする追加バイナリイメージを指定すると、すべてのプロジェクトのバイナリイメージを一度にダウンロードできます。
   - **Download extra image** チェックボックスを選択してください。
   - CM4/CM7 の **HEX** イメージへの **Path** を指定してください。
   - **Offset** に 0 を入力してください。

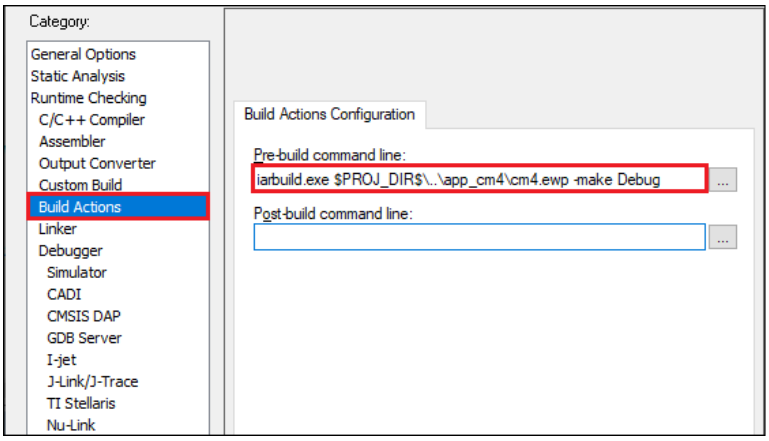


HEX ファイルの代わりに OUT ファイルを指定した場合、CM0+と CM4/CM7 の両 OUT ファイルに main 関数が存在するため、IAR IDE は `main()`の先頭で停止に失敗します。

注: トリプルコア マイコンの場合は、追加イメージを 2 つダウンロードする必要があります。

4. プログラミング/デバッグの前にすべてのプロジェクトをビルドするプリビルドコマンドを追加してください。**Build Actions** カテゴリで、**Pre-build command line** を設定します。

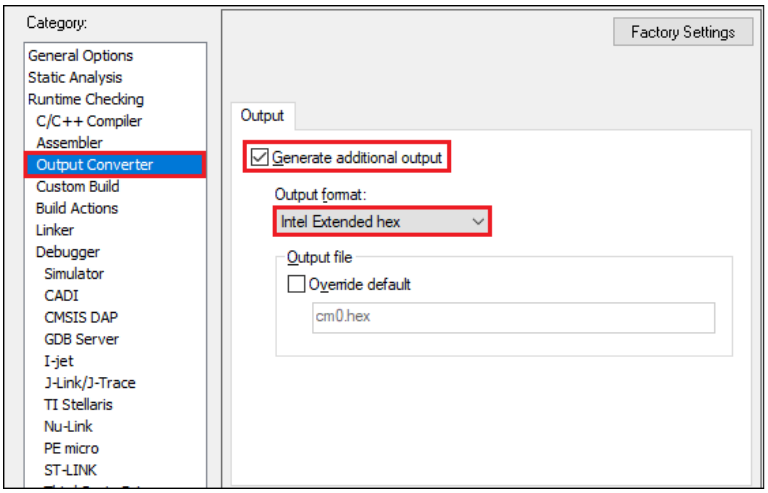
```
iarbuild.exe "<cm4/cm7_proj_loc>.ewp" -make Debug
```



5. マイコンに 3 つのコアが搭載されている場合、3 つ目のコア用にプロジェクトをビルドするポストビルドアクションも同様に指定するとよいでしょう。

6. HEX ファイル生成を有効にしてください。**Runtime Checking > Output Converter** カテゴリで次のように行います。

- **Generate additional output** チェックボックスを選択してください。
- **Output format** が **Intel Extended hex** に設定されていることを確認してください。



7. **OK** をクリックし、**File > Save All** を選択し、すべての変更を保存してください。

8. プロジェクトをビルドしてください。

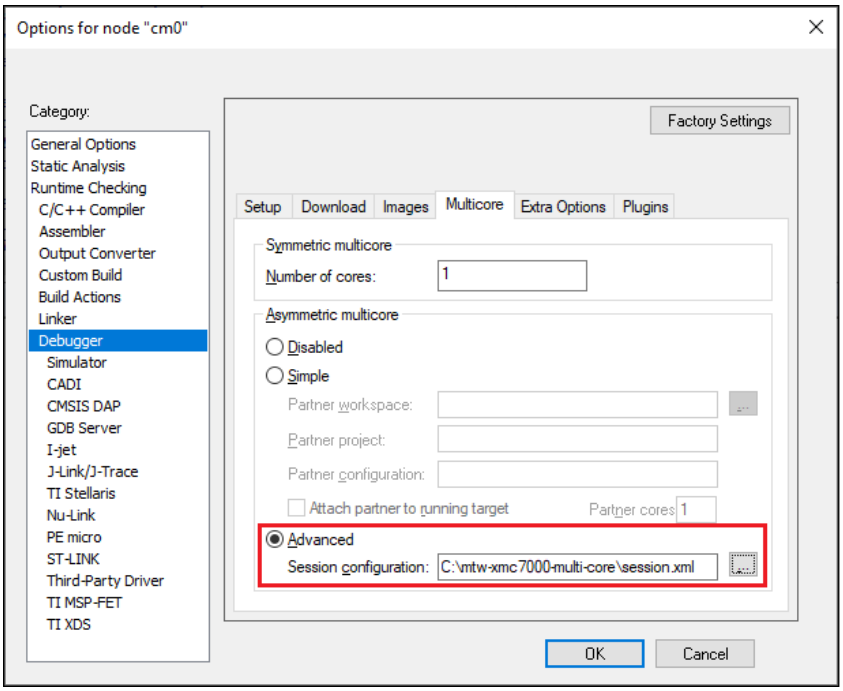
8.3 CMSIS-DAP/i-Jet デバッグオプションの設定

1. セッション設定ファイルを作成してください。

   これは、マルチコア デバッグ セッションで起動されるべきプロジェクト リストを含む xml ファイルです。以下はトリプルコア デバイスの例です。デュアルコアデバイスの場合、3 番目のパートナーノードを削除してください。

```
<?xml version="1.0" encoding="utf-8"?>

<sessionSetup>
    <partner>
        <name>cm0</name>
        <workspace>C:\Users\mtw-multi-core\Multicore_App\multi-core_workspace.eww</
workspace>
        <project>cm0</project>
        <config>Debug</config>
        <numberOfCores>1</numberOfCores>
        <attachToRunningTarget>false</attachToRunningTarget>
    </partner>
    <partner>
        <name>cm7_0</name>
        <workspace>C:\Users\mtw-multi-core\Multicore_App\multi-core_workspace.eww</
workspace>
        <project>cm7_0</project>
        <config>Debug</config>
        <numberOfCores>1</numberOfCores>
        <attachToRunningTarget>true</attachToRunningTarget>
    </partner>
    <partner>
        <name>cm7_1</name>
        <workspace>C:\Users\mtw-multi-core\Multicore_App\multi-core_workspace.eww</
workspace>
        <project>cm7_1</project>
        <config>Debug</config>
        <numberOfCores>1</numberOfCores>
        <attachToRunningTarget>true</attachToRunningTarget>
    </partner>
</sessionSetup>
```

2. CM0+プロジェクトのマルチコアデバッグを設定してください。
   a. **Project > Options -> Debugger** に移動してください。
   b. **Multicore** のタブに切り替えてください。
   c. **Advanced** ラジオボタンを選択し、**Session configuration** フィールドにセッション設定ファイルへのパスを指定してください。
   d. **OK** をクリックしてください。

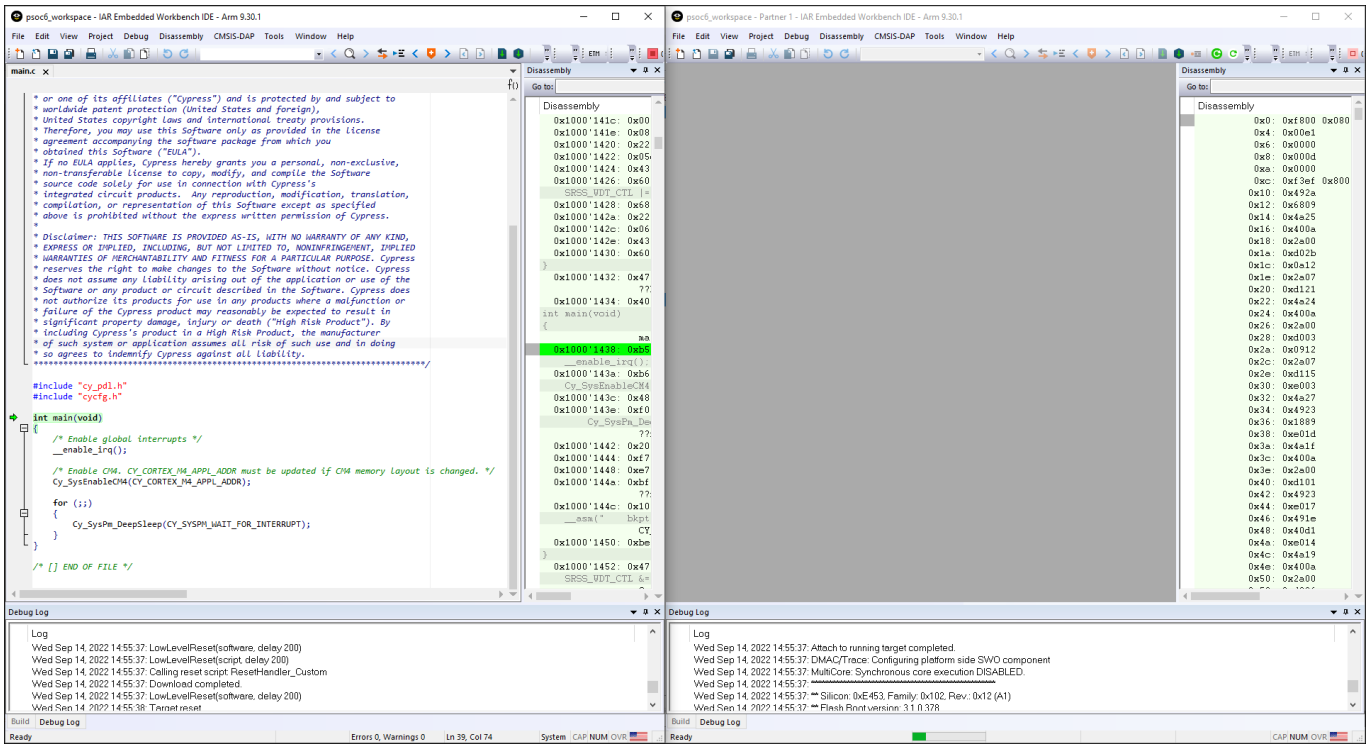


3. ワークスペースを保存してください。

8.4 CMSIS-DAP/i-Jet によるマルチコア デバッグ セッションの開始

CM0 プロジェクトを選択し、**Download and debug** ボタンをクリックしてください。

IAR はすべてのプロジェクトをビルドし、すべての個別イメージをプログラムして、マルチコアデバッグセッションを起動します。IAR は、セッションファイルで指定されたプロジェクトごとに、個別の IDE インスタンスを開きます。デュアルコア MCU の場合、以下のようになります。



画面の左側には、CM0+コアに接続された IAR IDE インスタンスが表示されています。右側は CM4 (または CM7) コアがまだ起動していないことを示しています。CM0+コアで `Cy_SysEnableCM4()` 関数が実行されると、CM4/CM7 はアプリケーションの実行を開始します。

2 つの IAR IDE インスタンス間を行き来することで、コードをステップスルーできます。

8.5 J-Link によるマルチコア デバッグ セッションの開始

IAR IDE は J-Link ドライバをネイティブにサポートしていないため、いくつかの制限があります。

IAR は、J-Link プローブを使用する場合、ネイティブのマルチコアデバッグサポートは提供しません。これは、マルチコアデバッギングを起動するために、いくつかの IAR IDE インスタンスを手動で開く必要があることを意味します (コアごとに 1 つのインスタンス)。また、J-Link プローブによるマルチコアデバッグには、CMSIS-DAP や I-Jet プローブで利用可能ないくつかの機能が欠けています。したがって、ターゲット プローブによって、プロジェクトの設定を若干変える必要があります。

- IAR は、コアごとに別々の IDE インスタンスを自動的にオープンしないので、手動で行わなければいけません。
- 一部の拡張機能は使用できません。詳細については、「マルチコアツールバーと CTI の使用」のセクションを参照してください。

J-Link でマルチコアデバッグを起動する手順

1. マルチコアの IAR ワークスペースを別々の IDE インスタンスで開いてください (IDE インスタンスの数は、マイコンのコア数と同じでなければいけません)。

2. 最初の IDE インスタンスで CM0+プロジェクトを選択し、**Download and Debug** をクリックしてください。デバッガーはすべてのイメージをダウンロードし、ターゲットをリセットし、CM0+プロジェクトの `main()`の先頭で停止します。

3. 他の IDE インスタンスに切り替え、**Project > Attach to Running Target** を選択してください。

改訂履歴

版数	日付	変更内容
英語版(**)	-	この版は英語版のみです。英語版の改訂内容: New document.
英語版(*A)	-	この版は英語版のみです。英語版の改訂内容: Removed obsolete instructions for customizing linker scripts.
英語版(*B)	-	この版は英語版のみです。英語版の改訂内容: Updated for ModusToolbox™ version 3.2. Updated recommended version. Removed Python. Added instructions for projects with C++ files. Added instructions for AIROC™ CYW20829 devices.
英語版(*C)	-	この版は英語版のみです。英語版の改訂内容: Updated for ModusToolbox™ version 3.3.
英語版(*D)	-	この版は英語版のみです。英語版の改訂内容: Included instructions for configuring a PSOC™ Control C3 device.
英語版(*E)	-	この版は英語版のみです。英語版の改訂内容: Updated for ModusToolbox™ version 3.4.
英語版(*F)	-	この版は英語版のみです。英語版の改訂内容: Updated for ModusToolbox™ version 3.5.
**	2025-07-17	これは英語版 002-37598 Rev. *G を翻訳した日本語版 Rev. **です。英語版の改訂内容: Updated document to separate the instructions per each device type.

Trademarks

All referenced product or service names and trademarks are the property of their respective owners.

Edition 2025-07-17

Published by

Infineon Technologies AG

81726 Munich, Germany

© 2025 Infineon Technologies AG

All Rights Reserved.

Do you have a question about any aspect of this document?

Email: erratum@infineon.com

Document reference
IFX-efq1743008065249

重要事項

本手引書に記載された情報は、本製品の使用に関する手引きとして提供されるものであり、いかなる場合も、本製品における特定の機能性能や品質について保証するものではありません。本製品の使用の前に、当該手引書の受領者は実際の使用環境の下であらゆる本製品の機能及びその他本手引書に記された一切の技術的情報について確認する義務が有ります。インフィニオンテクノロジーズはここに当該手引書内で記される情報につき、第三者の知的所有権の不侵害の保証を含むがこれに限らず、あらゆる種類の一切の保証および責任を否定いたします。

本文書に含まれるデータは、技術的訓練を受けた従業員のみを対象としています。本製品の対象用途への適合性、およびこれら用途に関連して本文書に記載された製品情報の完全性についての評価は、お客様の技術部門の責任にて実施してください。

警告事項

技術的要件に伴い、製品には危険物質が含まれる可能性があります。当該種別の詳細については、インフィニオンの最寄りの営業所までお問い合わせください。

インフィニオンの正式代表者が署名した書面を通じ、インフィニオンによる明示の承認が存在する場合を除き、インフィニオンの製品は、当該製品の障害またはその使用に関する一切の結果が、合理的に人的傷害を招く恐れのある一切の用途に使用することはできないこと予めご了承ください。