PSoC™ 4 入門

About this document

Scope and purpose

AN79953 では、Arm® Cortex®-M0/M0+ベースのプログラム可能なシステムオンチップである PSoC™ 4 についてご紹介します。本資料は、開発者のお客様が PSoC™ 4 アーキテクチャとその開発ツールを探究するために役立つほか、PSoC™ Creator および ModusToolbox™という PSoC™ 4 用の開発ツールを使用して新規のプロジェクトを作成する際に有用な案内をご提供します。本アプリケーション ノートは PSoC™ 4 の徹底的な研究を促進するリソースもご提供します。

Intended audience

このアプリケーションノートは、PSoC™および ModusToolbox™を初めて使用するエンジニア、および組込みマイクロコントローラーの使用経験があるエンジニアを対象とします。

関連製品ファミリ

すべての **PSoC™ 4** 製品

ソフトウェアバージョン

PSoC™ Creator 4.3 SP2 以降, **ModusToolbox™** 2.3 以降

その他のサンプルコードが必要な場合は、以下を参照してください。

PSoC™のサンプルコード リストにアクセスするためには**サンプルコード ウェブページ**をご覧ください。
ビデオ ライブラリについては**ここ**からご覧ください。

Table of contents

Table of contents

1 はじめに

PSoC™ 4 デバイスは、単一のチップ上にカスタム アナログとデジタル ペリフェラル機能、メモリおよび Arm® Cortex®-M0 または Cortex®-M0+マイクロコントローラーを集積した、真のプログラマブル組込みシステムオンチップです。

このようなシステムは、マイクロコントローラー ユニット (MCU) と外付けアナログおよびデジタルペリフェラルの組合わせを使用するほとんどのミックスド シグナルの組込みシステムとは異なります。通常、このようなシステムは MCU に加えて、オペアンプ、ADC、Application-specific Integrated Circuit (ASIC) など多くの集積回路が必要です。

PSoC™ 4 は、MCU と外部 IC の組み合わせに代わるものとして低コストなソリューションを提供します。システム全体のコスト削減に加えて、プログラマブル アナログとデジタル サブシステムにより高い柔軟性、設計のインフィールド チューニングおよび製品化までの時間縮小をご提供します。

PSoC™ 4 の CAPSENSE™ (静電容量タッチ センシング機能) は、比類のない信号対雑音比、クラス最高の耐水性およびボタンやスライダー、トラック パッド、近接センサーなど多種多様なセンサー タイプを提供します。PSoC™ 4 は SRAM、プログラマブル ロジックおよび割込みからウェイクアップする機能を保持しながら、クラス最高の低消費電流 150 nA を実現します。また、非保持の電力モードでウェイクアップ機能を保持し、わずか 20 nA の消費電流です。PSoC™ 4 ファミリ デバイスは、Bluetooth® Low Energy (Bluetooth® LE) 無線システムを搭載する PSoC™ 4 Bluetooth® LE も含んでいます。PSoC™ 4 Bluetooth® LE の詳細については **AN91267** を参照してください。

本ドキュメントを使用するにあたって

この後の数ページでは、PSoC™ 4 自身と、PSoC™、ModusToolbox™、および PSoC™ Creator を使用して設計する利点について説明します。スキップして簡単なデザインを作成する場合は、ModusToolbox™では **ModusToolbox™を使用するはじめての PSoC™ 4 設計**に、PSoC™ Creator では **PSoC™ Creator を使用するはじめての PSoC™ 4 設計**に進んでください。

2 PSoC™リソース

ここで利用できる豊富なデータは、適切な PSoC™デバイスを選択し、デバイスを設計に迅速かつ効果的に統合するのを支援します。以下は、PSoC™ 4 リソースの要約です。

- **概要: PSoC™ポートフォリオ**、**PSoC™ロードマップ**
- **製品セレクタ: PSoC™ 4**。さらに、**PSoC™ Creator** にはデバイス選択ツールを含みます。
- **データシート**は、各ファミリの電気的仕様を掲載し説明します。
- **アプリケーション ノート**は、基本的なレベルから高度なレベルまでの幅広いトピックに触れており、以下のものが含まれます。
  - **AN88619**: PSoC™ 4 Hardware Design Considerations
  - **AN73854**: Introduction to Bootloaders
  - **AN89610**: Arm® Cortex® Code Optimization
  - **AN86233**: PSoC™ 4 Low-Power Modes and Power Reduction Techniques
  - **AN57821**: PSoC™ 3, PSoC™ 4 および PSoC™ 5LP のアナログ/デジタル混在回路基板レイアウトの注意事項
  - **AN89056**: PSoC™ 4 – IEC 60730 Class B and IEC 61508 SIL Safety Software Library
  - **AN64846**: Getting Started with CAPSENSE™
  - **AN85951**: PSoC™ 4 および PSoC™ 6 MCU CAPSENSE™デザインガイド
- **サンプルコード**は、製品の機能と使用法を示します。PSoC™ Creator の**サンプルコード**と ModusToolbox™の**サンプルコード**を参照してください。
- **テクニカル リファレンス マニュアル (TRM)**: 各 PSoC™ 4 デバイス ファミリのアーキテクチャとレジスタの詳細な説明を掲載します。
- **PSoC™ 4 Programming Specification** は、PSoC™ 4 不揮発性メモリのプログラムに必要な情報を提供します。
- **開発キット**:
  - **CY8CKIT-040**, **CY8CKIT-041**, **CY8CKIT-042**, **CY8CKIT-044**, **CY8CKIT-046**, **CY8CKIT-042-BLE**, **CY8CKIT-045S**, および **CY8CKIT-041S-MAX** PSoC™ 4 Pioneer Kits は使いやすい安価な開発プラットフォームです。これらのキットには、Arduino 準拠シールドおよび Digilent® Pmod™ドーター カードの専用コネクタを搭載します。
  - **CY8CKIT-043**、**CY8CKIT-145**、**CY8CKIT-147** および **CY8CKIT-149** は PSoC™ 4 デバイスをサンプリングするための低コスト プロトタイピング プラットフォームです。
  - **MiniProg3** または **MiniProg4** キットは、フラッシュプログラムとデバッグのためのインターフェースを提供します。
  - 統合開発環境 (IDE): PSoC™ 4 を使用したアプリケーション開発に使用できる開発プラットフォームには、**ModusToolbox™**と **PSoC™ Creator** の 2 つがあります。
  - **PSoC™ 4 CAD ライブラリ**は、一般的なツールのフットプリントと回路図のサポートを提供します。 **IBIS モデル**もご利用いただけます。
- **トレーニングビデオ**は、**PSoC™ 4 101 シリーズ**を含む幅広いトピックで利用できます。
- **Cypress Developer Community** は、世界中の仲間の PSoC™開発者との接続を 24 時間年中無休で可能にし、専用の **PSoC™ 4 MCU** コミュニティをホストします。

2.1 ファームウェア/アプリケーション開発

PSoC™ 4 を使用したアプリケーション開発に使用できる開発プラットフォームは 2 つあります。

PSoC™リソース

- **ModusToolbox™**: ModusToolbox™ソフトウェアには、構成ツール、低レベルドライバー、ミドルウェアライブラリ、オペレーティングシステムのサポート、および MCU とワイヤレスアプリケーションの作成を可能にするその他のパッケージが含まれます。オプションの Eclipse IDE も含まれます。

ModusToolbox™は、EclipseIDE から起動できるスタンドアロンのデバイスおよびミドルウェアコンフィギュレーターをサポートします。コンフィギュレーターを使用して、デバイス内のさまざまなブロックの構成を設定し、ファームウェア開発で使用できるコードを生成します。ModusToolbox™は、すべての PSoC™ 6 MCU および最新の PSoC™ 4 MCU デバイスをサポートします。**Table 1** に、サポートされている PSoC™ 4 デバイスを示します。サポートされている PSoC™ 4 デバイスのすべてのアプリケーション開発には ModusToolbox™を使用することを推奨します。詳細については、**ModusToolbox™のリソース**を参照してください。

Table 1　　ModusToolbox™でサポートされる PSoC™ 4 デバイスのリスト

デバイス [1]	ModusToolbox™	PSoC™ Creator
PSoC™ 4000S, PSoC™ 4100S, PSoC™ 4100S Plus, PSoC™ 4100S Plus 256K	有	有
PSoC™ 4100S Max	有	無
他のすべての PSoC™ 4 シリーズ	無	有

ライブラリと有効化ソフトウェアは **GitHub** で入手できます。

ModusToolbox™のツールとリソースもコマンドラインで使用できます。詳細なドキュメントについては、**Running ModusToolbox™ from the command Line** を参照してください。

- **PSoC™ Creator**: **PSoC™ Creator** は無料の Windows ベースの IDE です。これにより PSoC™ 3、PSoC™ 4、PSoC™ 5LP、および PSoC™ 6 MCU システムのハードウェアとファームウェアの同時設計が可能になります。アプリケーションは、回路図キャプチャと 150 を超える事前検証済みの本番環境対応の周辺機器コンポーネントを使用して作成されます。

[1] 完全な PSoC™ 4 ポートフォリオについては、**PSoC™ 4 の機能セット**を参照してください。

2.2 IDE の選択

Figure 1 は、適切な IDE を選択するために役立ちます。

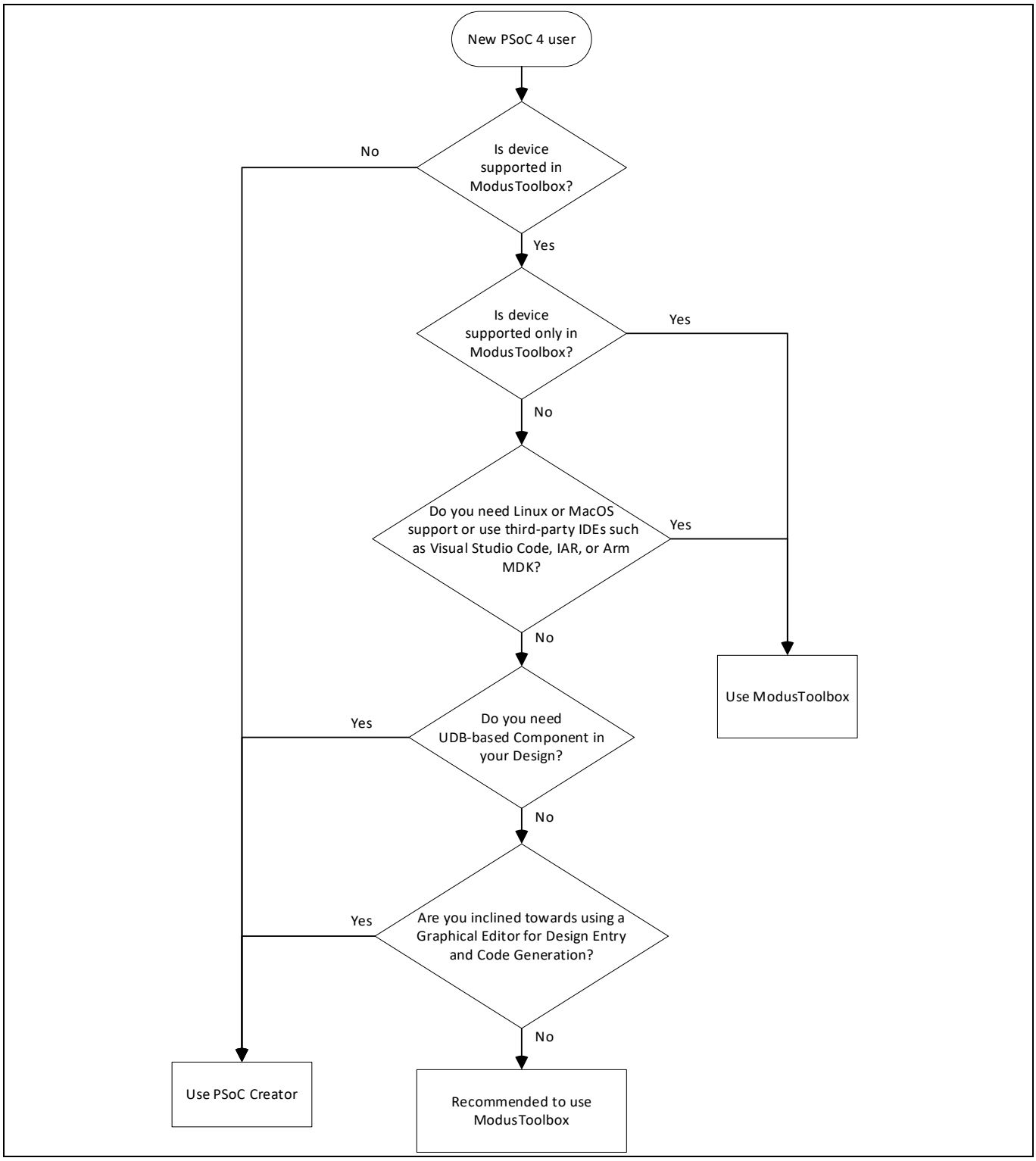

Figure 1 IDE の選択

2.2.1 ModusToolbox™を選ぶ理由

- 包括的: 必要なリソースがあります。
- 柔軟性: 独自のワークフローでリソースを使用できます。
- アトミック：必要なリソースだけを取得できます。
- **GitHub** 上のコードリポジトリの大規模なコレクション。これには以下を含みます。
  - インフィニオンキットと連携したボードサポートパッケージ (BSP)
  - Hardware Abstruction Layer (HAL) や Peripheral Driver Library (PDL) などの低レベルのリソース
  - CAPSENSE™などの業界をリードする機能を可能にするミドルウェア
  - 徹底的にテストされたサンプルコードアプリケーションの広範なセット
- Windows、Linux、および macOS プラットフォームに対応します。
- IDE に依存せず、Visual Studio Code、Arm® MDK (µVision)、IAR Embedded Workbench などのサードパーティ IDE に対応します。
- Eclipse ベースのツールの使用経験があり、ModusToolbox™用のオプションの Eclipse ベースの IDE の機能と拡張性を利用したい場合は、ModusToolbox™を選択してください。

2.3 ModusToolbox™のリソース

ModusToolbox™は、統合された MCU およびワイヤレスシステムを作成するための没入型開発経験を可能にし、インフィニオンデバイスを既存の開発方式に統合できるようにするツールとソフトウェアのセットです。

ModusToolbox™用の Eclipse IDE は、アプリケーションの構成と開発をサポートするマルチプラットフォーム開発環境です。ModusToolbox™インストーラーには、デザインコンフィギュレーターとツール、およびビルドシステムインフラストラクチャが含まれます。ビルドシステムインフラストラクチャには、Eclipse IDE、メイクインフラストラクチャ、およびその他のツールから独立して実行できる新しいプロジェクト作成ウィザードが含まれます。

2.3.1 PSoC™ 4 ソフトウェアリソース

ModusToolbox™には、**GitHub** を介して配信されるサイプレスが提供するソフトウェアリソースも含みます。これらには以下が含まれます。

- BSP – BSP は、ボード固有のドライバーおよびその他の機能を含むファームウェアのレイヤーです。BSP は、ボードを初期化し、ボードレベルの周辺機器へのアクセスを提供する API を提供するライブラリのセットです。これには、PSoC™ 4 用の PDL ライブラリなどの低レベルのリソースが含まれ、ボード周辺機器用のマクロがあります。カスタム BSP を作成して、エンドアプリケーションボードのサポートを有効にできます。詳細については、**ModusToolbox™ Library Manager ユーザーガイド**を参照してください。
- HAL – HAL は、インフィニオン MCU でハードウェアブロックを構成および使用するための高レベルのインターフェースを提供します。これは、複数の製品ファミリで使用できる汎用インターフェースです。使いやすさと移植性に重点を置いているということは、HAL がすべての低レベルの周辺機能を公開しているわけではないことを意味します。HAL は、低レベルのドライバー (PSoC™ 4 PDL など) を同梱し、MCU への高レベルのインターフェースを提供します。インターフェースは、任意の MCU で動作するように抽象化されています。これは、ターゲット MCU から独立したアプリケーションファームウェアを作成するために役立ちます。
- HAL は、単一のアプリケーション内でプラットフォーム固有のライブラリ (PSoC™ 4 PDL など) と組み合わせられます。ある部分でよりきめ細かい制御が必要な場合でも、ほとんどのアプリケーションで HAL のよりシンプルで汎用的なインターフェースを活用できます。

PSoC™リソース

- PSoC™ 4 PDL – PDL は、デバイスヘッダーファイル、スタートアップコード、および周辺機器ドライバーを 1 つのパッケージに統合します。PDL は、PSoC™ 4 デバイスファミリに対応します。ドライバーは、ハードウェア機能を一連の使いやすい API に抽象化します。これらは、PDL API リファレンスに完全に文書化されています。
- PDL は、レジスタの使用法とビット構造を理解する必要性を減らし、PSoC™ 4 シリーズの広範な周辺機器セットのソフトウェア開発を容易にします。アプリケーションのドライバーを構成してから、API 呼び出しを使用して周辺機器を初期化して使用します。
- アプリケーションに特定の機能を提供する広範なミドルウェアライブラリ。**利用可能なミドルウェア**は、接続 (Bluetooth®、AWS IoT、Bluetooth® LE、セキュアソケット) から PSoC™固有の機能 (CAPSENSE™) にまたがります。ミドルウェアは、**GitHub** リポジトリを介してライブラリとして提供されます。
- ユーティリティ、Makefile、スクリプト、およびその他の構成ソフトウェア。

2.3.2 コンフィギュレーター

ModusToolbox™は、ハードウェアブロックの構成を容易にするコンフィギュレーターと呼ばれるグラフィカルアプリケーションを提供します。たとえば、ドキュメントを検索してシリアル通信ブロック (SCB) を目的の構成の UART として構成する代わりに、適切なコンフィギュレーターを開き、ボーレート、パリティ、およびストップビットを設定します。ハードウェア構成を保存すると、ツールは「C」コードを生成して、ハードウェアを目的の構成で初期化します。

コンフィギュレーターは互いに独立していますが、一緒に使用して柔軟な構成オプションを提供できます。これらは、スタンドアロンとして、他のツールと組み合わせて使用することも、Eclipse IDE からの起動もできます。コンフィギュレーターは次の目的で使用されます。

- オプションを設定し、ドライバーを構成するためのコードを生成する
- 周辺機器のピンやクロックなどの接続を設定する
- ミドルウェアを構成するためのオプションの設定とコードの生成

PSoC™ 4 アプリケーションの場合、使用可能なコンフィギュレーターは次のとおりです。

- **DeviceConfigurator:** システム (プラットフォーム) 機能と基本的な周辺機器 (UART、タイマー、PWM など) をセットアップします。
- **CAPSENSE™ Configurator および Tuner:** CAPSENSE™を構成し、必要なコードを生成します。
- **Smart I/O Configurator:** スマート I/O を構成します。

これらの各コンフィギュレーターは独自のファイルを作成します (例えば、CAPSENSE™の場合は *design.cycapsense*)。コンフィギュレータファイル (*design.modus* または *design.cycapsense*) は通常、BSP で提供されます。BSP に基づいてアプリケーションが作成されると、ファイルがアプリケーションにコピーされます。アプリケーション用のカスタムデバイスコンフィギュレータファイルを作成し、BSP によって提供されるファイルの上書きもできます。詳細については、**ModusToolbox™のヘルプ**を参照してください。

2.3.3 PSoC™ 4 のソフトウェア開発

PSoC™ 4 デバイスのソフトウェア開発を可能にするために、重要なソースコードとツールが提供されます。ツールを使用して、ハードウェアの構成方法を指定したり、ファームウェアで使用できるコードを生成したり、追加機能のために CAPSENSE™などのさまざまなミドルウェアライブラリを組み込んだりできます。このソースコードにより、サポートされているデバイスのファームウェアの開発が容易になります。レジスタセットを理解しなくても、ファームウェアをすばやくカスタマイズしてビルドするために役立ちます。

PSoC™リソース

ModusToolbox™環境では、コンフィギュレーターを使用して、デバイスまたは CAPSENSE™機能などのミドルウェアライブラリを構成できます。

ドライバーコードは、*mtb-pdl-cat2* ライブラリとして提供されます。ミドルウェアは、機能ごとに個別のライブラリとして提供されます。

Eclipse IDE、サードパーティ IDE、またはコマンドラインのいずれを使用しているかに関係なく、レジスタレベルで作業している場合は、PDL のドライバーソースコードを参照してください。PDL には、プロジェクトに必要なすべてのデバイス固有のヘッダーファイルとスタートアップコードが含まれます。また、各ドライバーのリファレンスとしても機能します。PDL はソースコードとして提供されるため、レジスタレベルでハードウェアにアクセスする方法を確認できます。

一部のデバイスは特定の周辺機器をサポートしていません。PDL は、サポートされているデバイスのすべてのドライバーのスーパーセットです。このスーパーセットの設計は、以下のことを意味します。

- 周辺機器の初期化、構成、および使用に必要なすべての API 要素が利用可能です。
- PDL は、使用可能な周辺機器に関係なく、さまざまな PSoC™ 4 デバイスで役立ちます。
- PDL には、ターゲットの周辺機器が選択したデバイスに存在することを確認するためのエラーチェックを含みます。

これにより、周辺機器が利用可能である限り、コードは PSoC™ 4 デバイスファミリ全体で互換性を維持できます。デバイスヘッダーファイルは、デバイスで使用可能な周辺機器を指定します。コードがサポートされていないペリフェラルを使用しようとすると、コンパイル中にエラーが発生します。コードを記述する前に、デバイスのデータシートを参照して、周辺機器がサポートされているかどうかを確認してください。

Figure 2 に、ModusToolbox™用の Eclipse IDE を使用して以下のことができることを示します。

1. Project Creator (**File > New > ModusToolbox Application**) を使用して、BSP (キット) 用の新しいアプリケーションを作成します。
2. 構成ツールを使用して周辺機器とミドルウェアライブラリを構成します。
3. ライブラリマネージャーを使用して、ライブラリと BSP を簡単かつ迅速に追加、更新、または削除します。
4. オプションで、Eclipse ベースの IDE でコードを開発します。

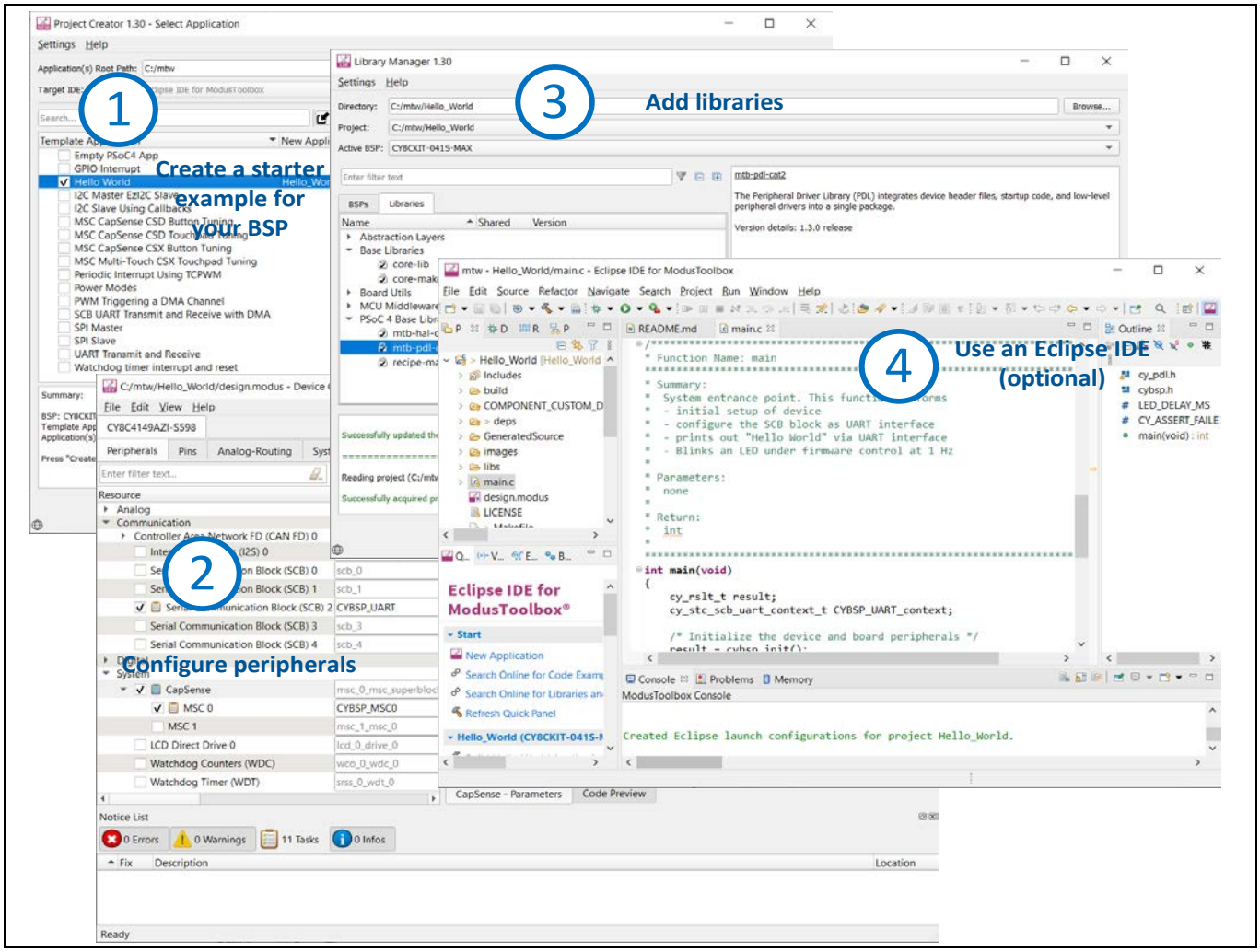

Figure 2 **ModusToolbox™リソースおよびミドルウェア用の Eclipse IDE**

2.3.4 サンプルコード

Project Creator ツールは、ModusToolbox™ソフトウェアとともに使用され、BSP とサンプルコードに基づいてアプリケーションを作成します。Project Creator は、任意のソフトウェア環境で使用できるアプリケーションを作成するためのグラフィカルユーザーインターフェース (GUI) およびコマンドラインインターフェース (CLI) として提供されます。

1. プロジェクトクリエーターツールを開いてください。

メニューの **File > New > ModusToolbox Application** に従ってください。

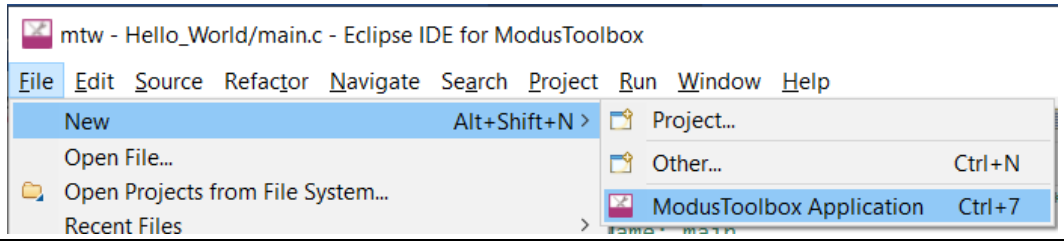

Figure 3 **プロジェクトクリエーターツールを開く**

2. ターゲットの PSoC™ 4 開発キットを選択してください。

Choose Board Support Package (BSP) ダイアログで、リストからキットを選択してください。

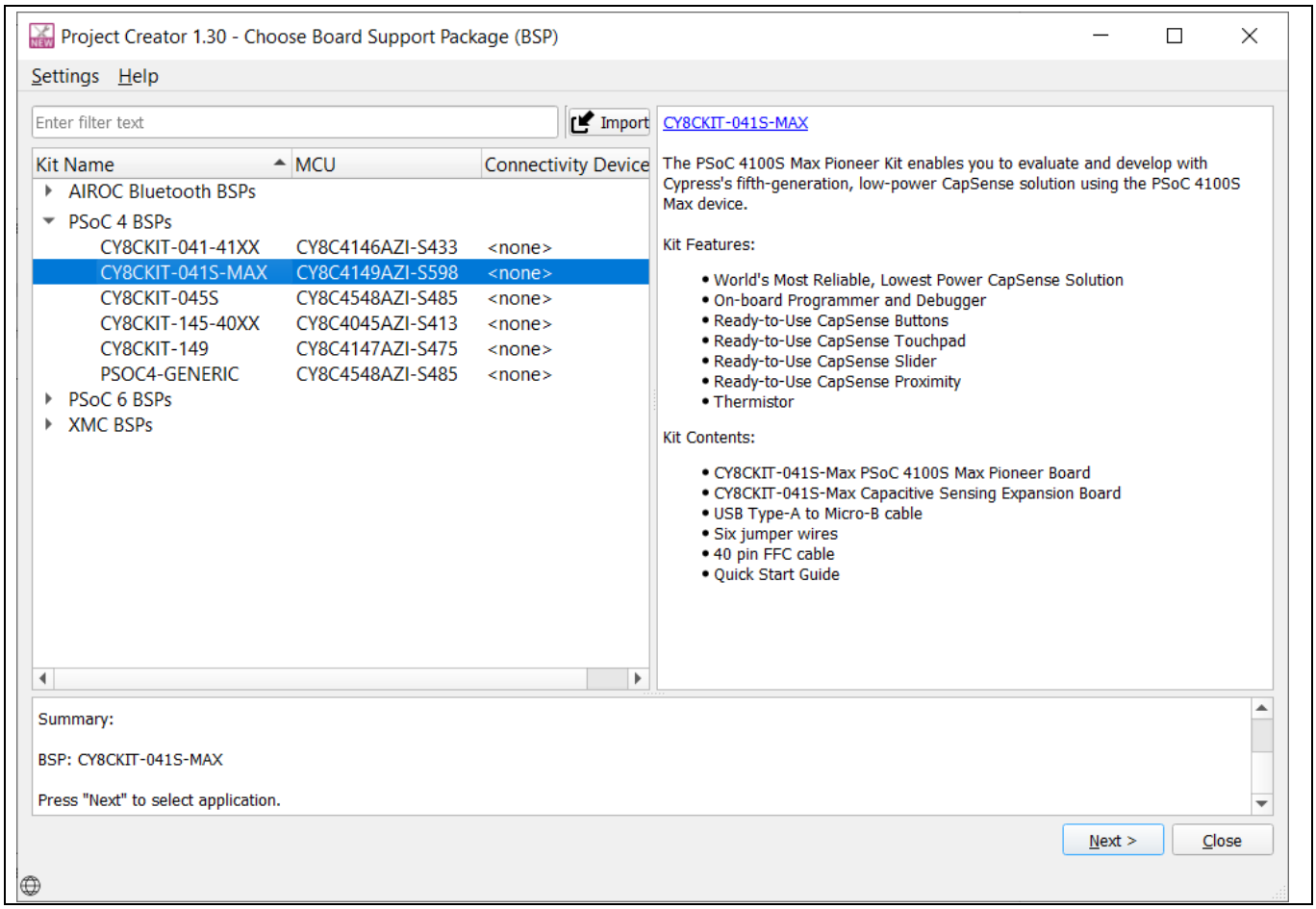

Figure 4　　ターゲットハードウェアを選択

3. アプリケーションを選択してください。

Select Application ダイアログには、選択した BSP に適用可能な **Application** が一覧表示されます。必要なアプリケーションを選択し、**Create** をクリックしてください。選択したアプリケーションが作成されます。

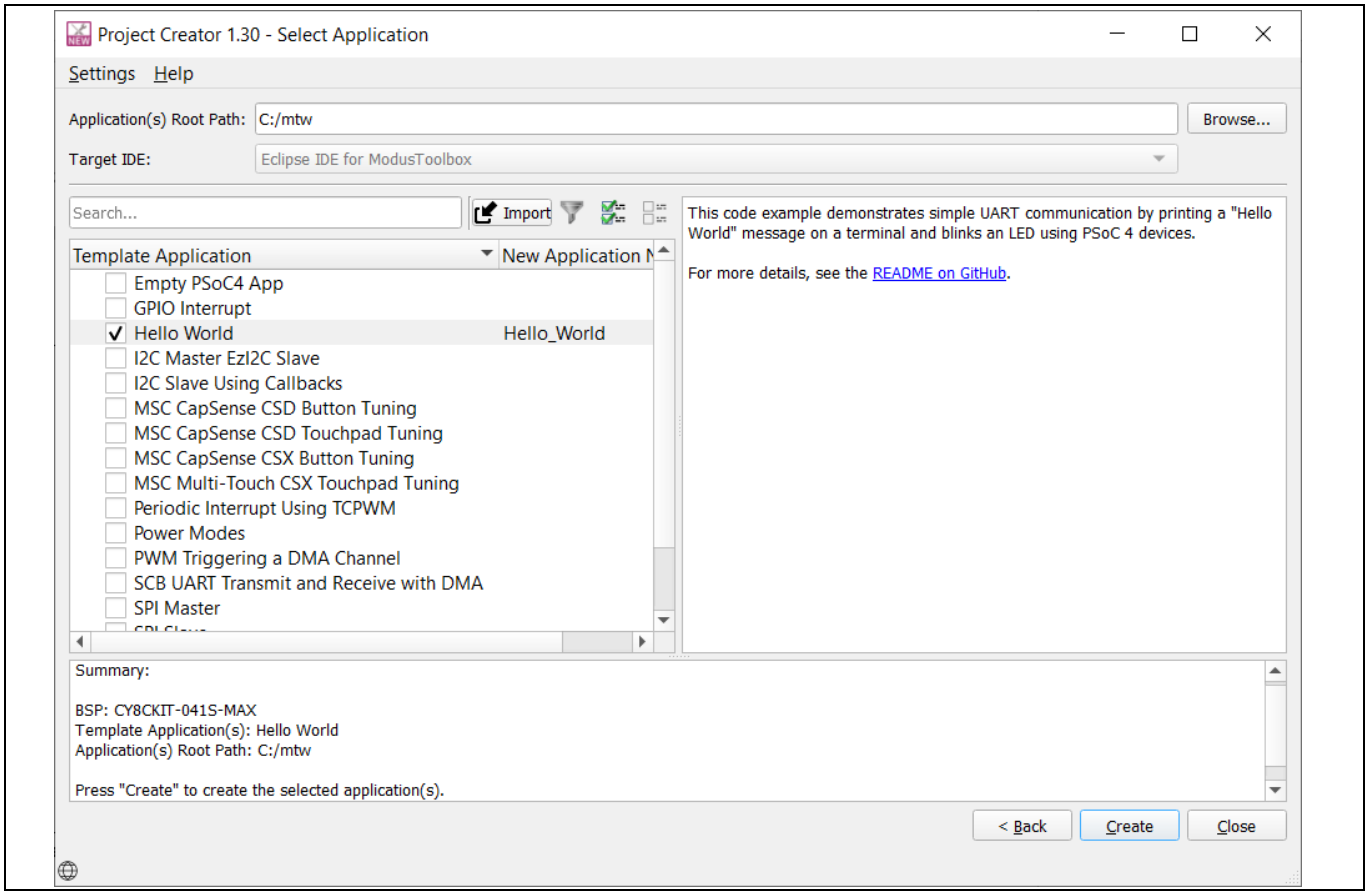

Figure 5　　アプリケーションを選択

2.3.5　ModusToolbox™のヘルプ

ModusToolbox™ user guide は、ModusToolbox™ソフトウェアの高レベルの概要を提供します。

ModusToolbox™の最新バージョンをダウンロードしてインストールしてください。ModusToolbox™用の Eclipse IDE を起動し、**Help > ModusToolbox General Documentation** と操作してください。

- **User Guide**: アプリケーションの構築、プログラミング、およびデバッグの側面について説明します。また、IDE とともにインストールされるツールのさまざまな側面についても説明します。
- **ModusToolbox Documentation Index**: ModusToolbox™ソフトウェアに含まれるさまざまなドキュメントへの簡単な説明とリンクを提供します。

Release Note: ModusToolbox™の対応するリリースの機能について説明し、知っておくべき既知の問題、回避策、および設計への影響を示します。ModusToolbox™用の Eclipse IDE に関するドキュメントについては、**Help > Eclipse IDE for ModusToolbox Documentation** と操作してください。

- **Quick Start Guide**: ModusToolbox™用の Eclipse IDE を使用するための基本を提供します。
- **User Guide**: Eclipse IDE を使用してアプリケーションを作成し、それらをビルド、プログラミング、およびデバッグする方法について説明します。
- **Eclipse IDE Survival Guide**: Eclipse IDE の使用中に発生する可能性のある一般的な問題への回答を提供します。ほとんどの質問は、ModusToolbox™ではなく Eclipse IDE に関連します。

2.4 PSoC™ Creator

PSoC™ Creator は、PSoC™システムのハードウェアとファームウェアの同時編集、コンパイル、およびデバッグを可能にする IDE です。**Figure 6** に示すように、PSoC™ Creator を使用すると、以下のことができます。

1. コンポーネントをドラッグ&ドロップして、ハードウェア システム設計の構築
2. アプリケーションのファームウェアと PSoC™ハードウェアの共同設計
3. コンフィギュレーション ツールを使ったコンポーネントの構成
4. 100 以上のコンポーネントを含むライブラリの利用
5. コンポーネント データシートの閲覧

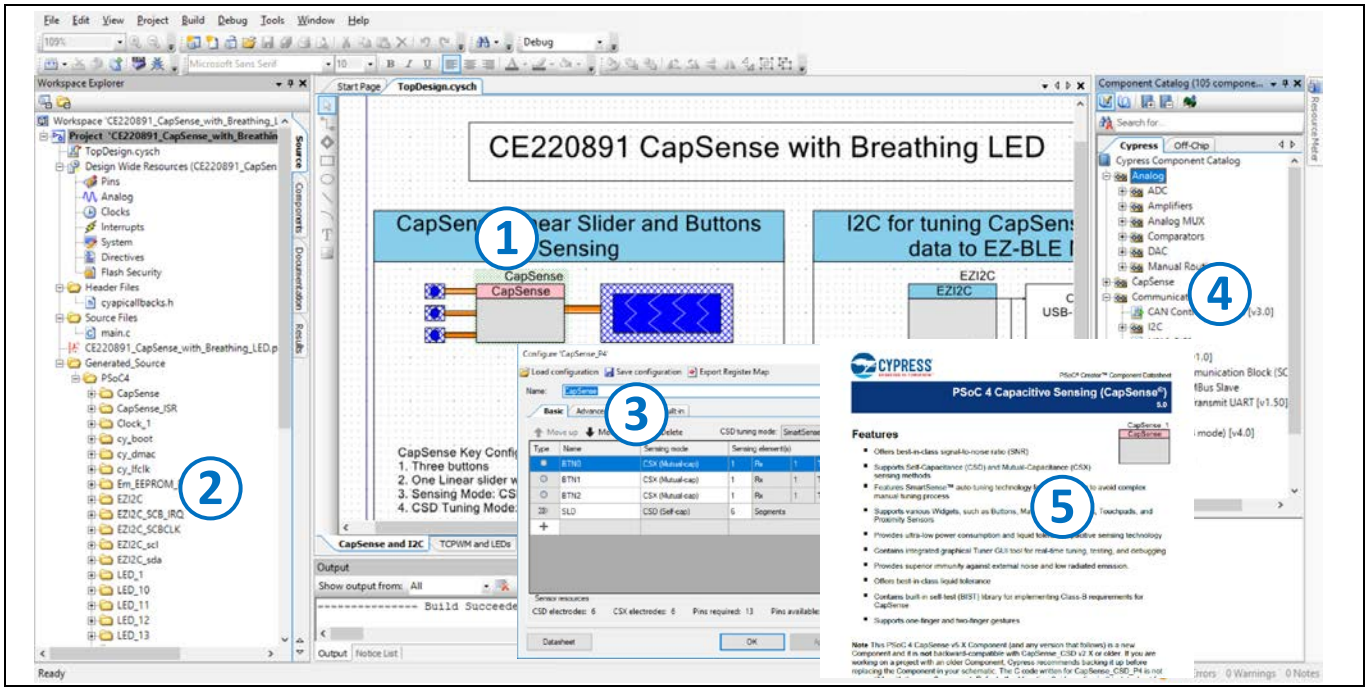

Figure 6 PSoC™ Creator の特長

2.4.1 サンプルコード

PSoC™ Creator は多数のサンプルコード プロジェクトを提供します。これらのプロジェクトは、**Figure 7** に示すように、PSoC™ Creator のスタート ページからアクセスできます。

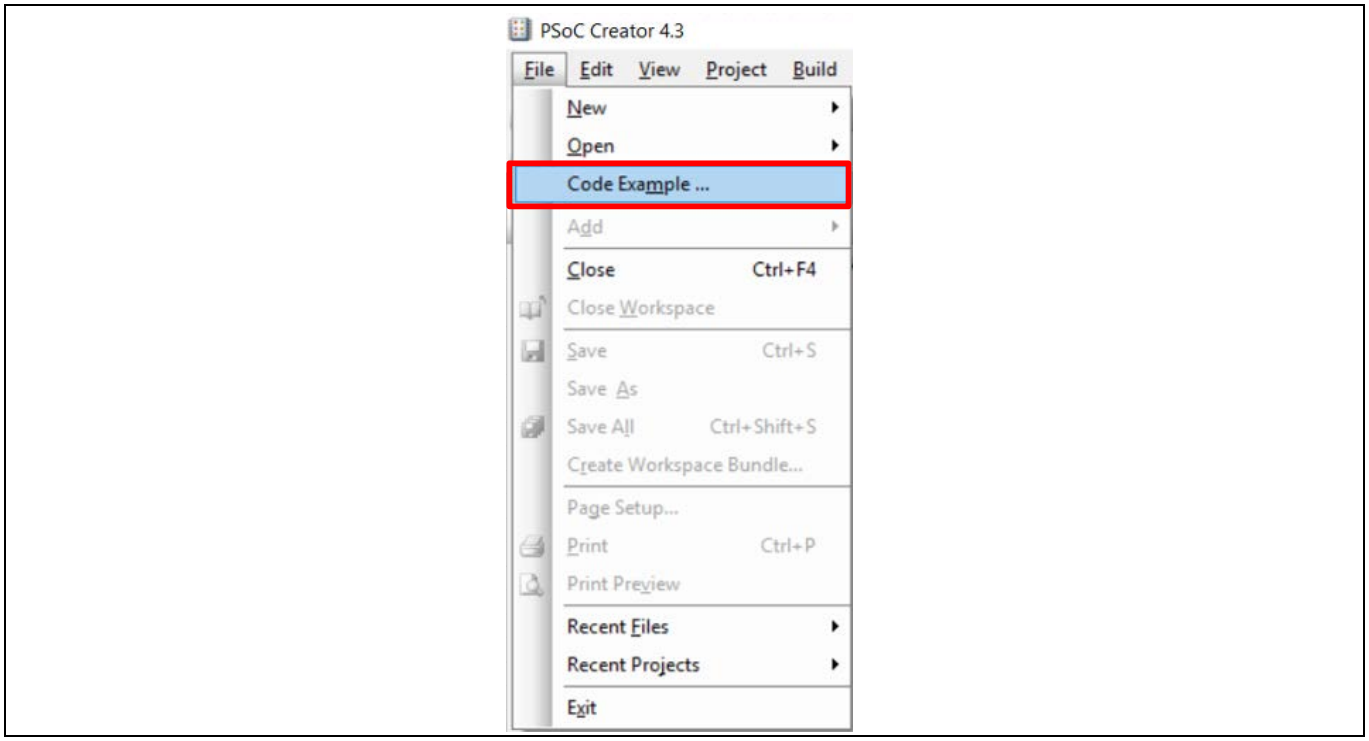

Figure 7　　PSoC™ Creator サンプルコード

サンプルプロジェクトは、空のページの代わりに完了した設計で始まり設計時間を短縮でき、PSoC™ Creator Components が様々な用途にどのように利用できるかを示します。**Figure 8** に示すようにサンプルコードおよびデータシートを含みます。

Figure 8 に示す Find Example Project ダイアログで、以下のことができます。

a) アーキテクチャまたはデバイス ファミリ (PSoC™ 3、PSoC™ 4、PSoC™ 5LP、または PSoC™ 6 MCU) またはカテゴリやキーワードに基づいてサンプルをフィルタします。

b) フィルタリングされたサンプルリストから選択します。

c) 選択したデータシートをレビューします (**Documentation** タブ)。

d) 選択したサンプルコードをレビューします。このウィンドウからコードをコピーしてプロジェクトに貼り付けると、コード開発の時間短縮ができます。または、選択したものに基づいて新規プロジェクト (または必要に応じて新規ワークスペース) も作成できます。これにより、完成した基本的設計から開始することで設計プロセスを時間短縮できます。その後、その設計を所望のアプリケーションに適用できます。

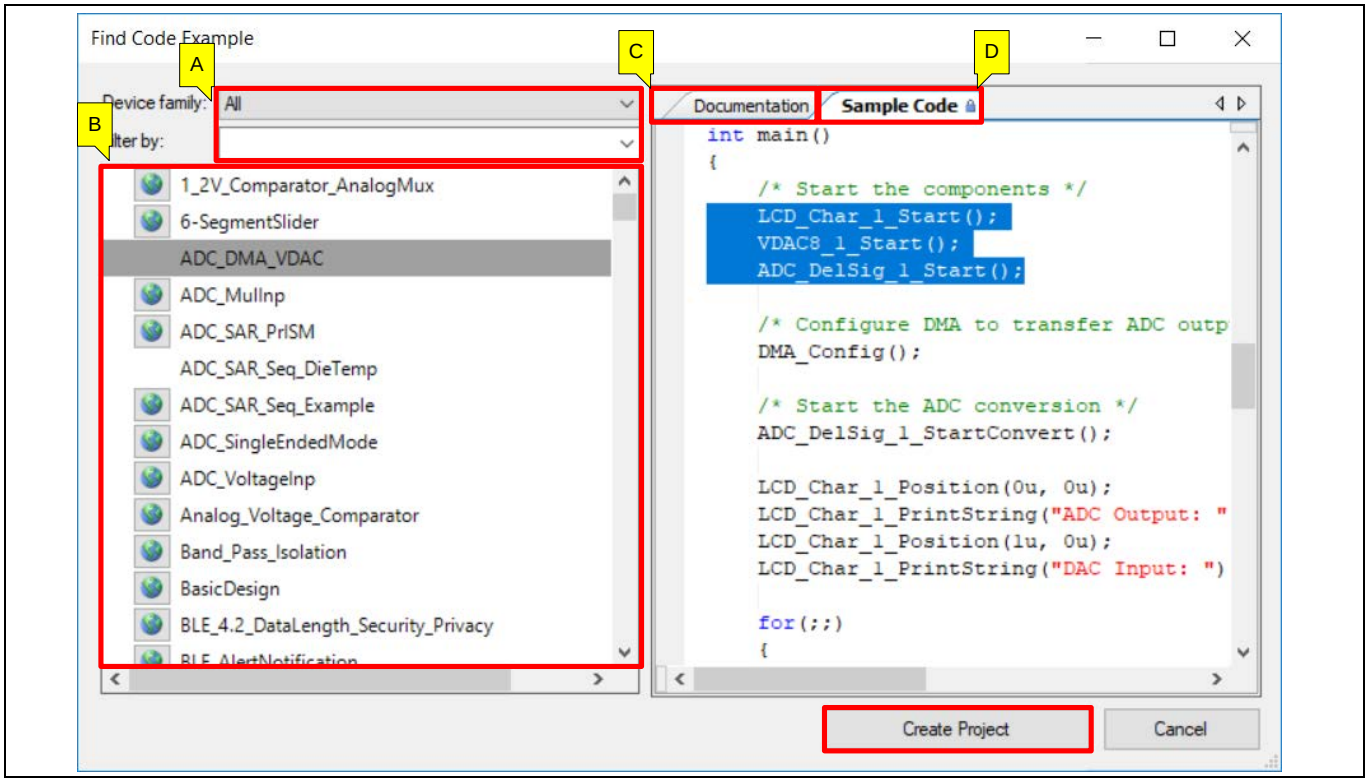

Figure 8 サンプルコードのあるサンプルコードプロジェクト

2.4.2 PSoC™ Creator ヘルプ

PSoC™ Creator ホームページへアクセスし、PSoC™ Creator の最新版をダウンロードしてください。次に、PSoC™ Creator を起動し、以下の項目へ操作してください。

- **Quick Start Guide** (クイック スタート ガイド): **Help > Documentation > Quick Start Guide** を選択します。このガイドは PSoC™ Creator プロジェクトを開発するための基礎を提供します。
- **Simple Component example project**: **File > Open > Example projects** を選択します。これらのサンプル プロジェクトは PSoC™ Creator コンポーネントの設定と使用方法を示します。
- **System Reference Guide**: **Help > System Reference > System Reference Guide** を選択します。このガイドは PSoC™ Creator により提供されるシステム機能を一覧で説明します。
- **Component datasheet**: コンポーネントを右クリックして **Open Datasheet** を選択します。すべての PSoC™ 4 コンポーネント データシートの一覧は、**PSoC™ 4 Component datasheets** ページに掲載されています。
- **PSoC™ Creator Training Video**: これらの動画は PSoC™ Creator 入門のステップ バイ ステップ方法を提供します。
- **Document Manager**: PSoC™ Creator が提供するドキュメント マネージャーにより、ドキュメント リソースを容易に検索し、確認できます。ドキュメント マネージャーを開くためには、メニューアイテムの **Help > Document Manager** を選択します。

2.5 テクニカル サポート

ご質問については弊社のテクニカル サポート チームが対応します。**テクニカルサポート**ページでサポートリクエストを作成できます。サポートチームは、**GitHub** リポジトリに投稿された質問、問題、バグレポートを確認して対応します。

緊急サポートが必要な場合は、以下のサポート リソースを利用してください。

- **セルフ ヘルプ**
- **各地の販売代理店**

3 PSoC™ 4 の機能セット

PSoC™ 4 は **Figure 9** に示すように、CPU とメモリ サブシステム、デジタル サブシステム、アナログ サブシステム、システム リソースを含む幅広い機能を備えます。次の節は各機能について簡単に説明します。詳細については、PSoC™ 4 ファミリのデバイス データシート、テクニカル リファレンス マニュアル (TRM) および **PSoC™リソース**にリストアップされたアプリケーション ノートを参照してください。

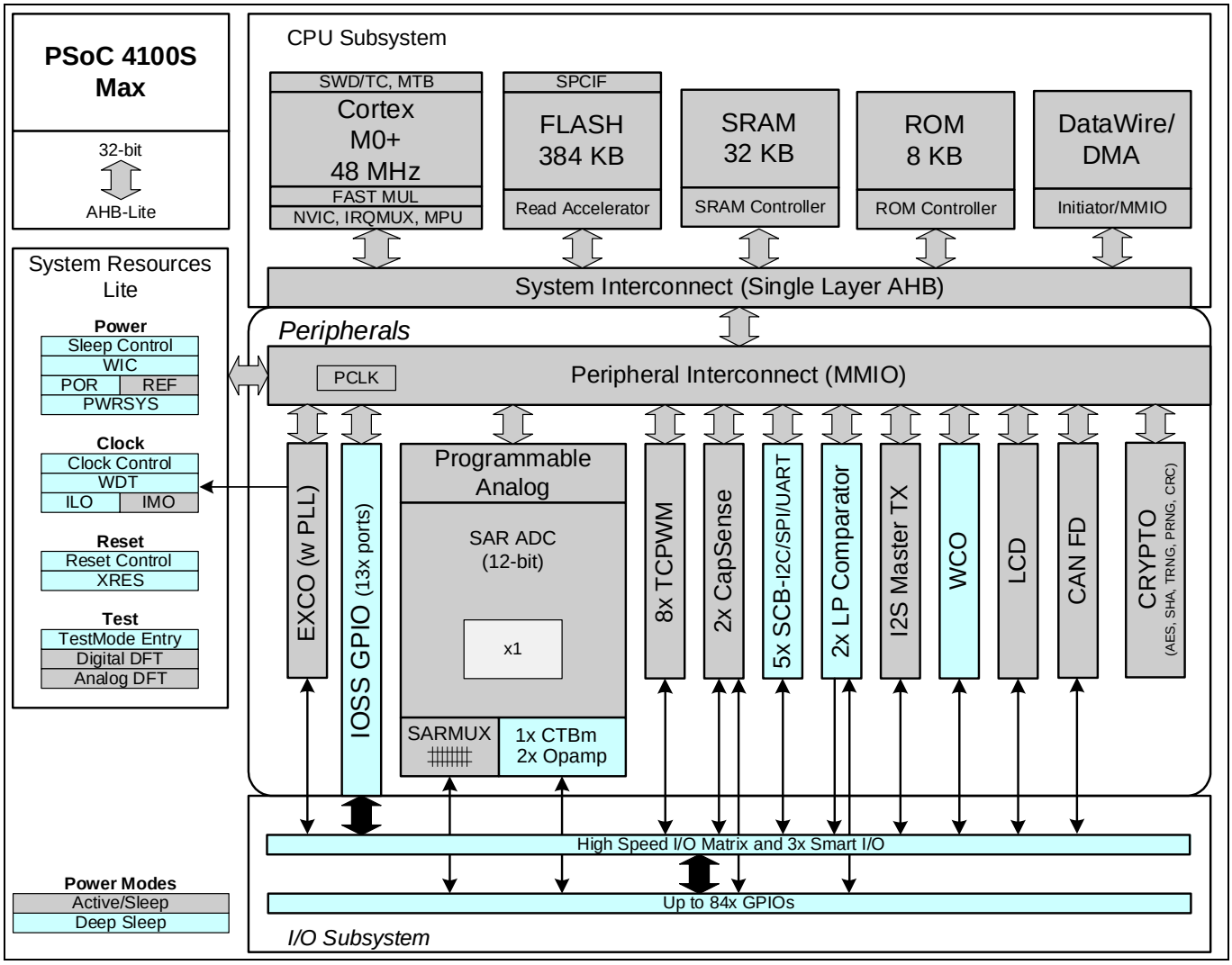

Figure 9 PSoC™ 4 アーキテクチャ (PSoC™ 4100S Max)

PSoC™ 4 ポートフォリオは、Arm® CM0 および CM0+マイクロコントローラーのいくつかのファミリで構成されます。ポートフォリオ内のほとんどのデバイスには、静電容量センシングアプリケーション用の CAPSENSE™テクノロジーが搭載されます。PSoC™ 4 ポートフォリオの他の主要な機能には、プログラム可能なアナログブロックを介したカスタマイズ可能なアナログフロントエンドと、USB、コントローラーエリアネットワーク (CAN)、Bluetooth® LE などの有線および無線接続オプションが含まれます。これらの独自の機能により、PSoC™ 4 は業界で最も柔軟でスケーラブルな低電力ミックスドシグナルアーキテクチャを実現しました。PSoC™ 4 デバイスは、**Table 2** に示すように、さまざまな機能に基づいてさまざまなファミリに分類されます。

PSoC™ 4 の機能セット

Table 2　　PSoC™ 4 ファミリ

分類	ファミリ	特長	詳細
入門レベル	PSoC™ 4000 ファミリ	CAPSENSE™	Table 3
インテリジェントアナログ	PSoC™ 4100 ファミリ	CAPSENSE™ + プログラム可能なアナログ	Table 4
プログラム可能なデジタル	PSoC™ 4200 ファミリ	CAPSENSE™ + プログラム可能なアナログ + プログラム可能なデジタルブロック	Table 5
特定用途向け	PSoC™ 4500 ファミリ	CAPSENSE™ + モーター制御	Table 6
	PSoC™ 4700 ファミリ	CAPSENSE™ + Inductive Sensing	
アナログコプロセッサ [2]	PSoC™ 4A00 ファミリ	CAPSENSE™ + プログラム可能なアナログブロック	AN211293

Note:　　Table 3 および Table 4 で、青色で強調表示されている列のファミリは ModusToolbox™でサポートされていることを示します。

Table 3　　PSoC™ 4000 ファミリの特長

特長		PSoC™ 4000	PSoC™ 4000S
CPU		16 MHz Cortex®-M0	48 MHz Cortex®-M0+
フラッシュメモリ		16 KB	32 KB
SRAM		2 KB	4 KB
GPIO		20	36
CAPSENSE™		16 センサー	35 センサー
シングルスロープ ADC (10 ビット 46 ksps)		なし	1
コンパレータ		固定閾値の 1 つの CSD コンパレータ (1.2 V)	ウェイクアップ機能付き 2 つの低電力コンパレータ
IDAC [3]		1 つの 7 ビットと 1 つの 8 ビット	2 つの 7 ビット
スマート I/O ポート		なし	2
電源電圧範囲		1.71 V ～ 5.5 V	1.71 V ～ 5.5 V
低消費電力モード		2.5 µA でのディープスリープ	2.5 µA でのディープスリープ
セグメント LCD 駆動		なし	4 COM セグメント LCD 駆動
シリアル通信		1 つの I²C	プログラム可能な I²C, SPI, または UART を備えた 2 つの SCB
タイマーカウンターパルス幅変調器 (TCPWM)		1	5
クロック	内部メイン発振器 (IMO)	24 MHz/32 MHz	24 MHz ～ 48 MHz
	内部低速発振器 (ILO)	32 kHz 内部 ILO	40 kHz

[2] PSoC™ 4 アナログコプロセッサファミリデバイスの使用を開始するためには、AN211293 を参照してください。

[3] IDAC は、CAPSENSE™を使用しない場合にのみ利用可能です。詳細は、各 PSoC™ 4 アーキテクチャ TRM を参照してください。

PSoC™ 4 の機能セット

特長		PSoC™ 4000	PSoC™ 4000S
	時計水晶発振器 (WCO)	なし	32 kHz
電源監視		パワーオンリセット (POR) 電圧低下検出 (BOD)	POR, BOD
対応するキット		**CY8CKIT-040 Pioneer kit**	**CY8CKIT-041 Pioneer kit**
対応する IDE		**PSoC™ Creator**	**PSoC™ Creator, ModusToolbox™**

Table 4　　PSoC™ 4100 ファミリの特長

特長	PSoC™ 4100	PSoC™ 4100S	PSoC™ 4100S Plus	PSoC™ 4100S Plus 256K	PSoC™ 4100 PS	PSoC™ 4100 M	PSoC™ 4100 BL[4]	PSoC™ 4100 S Max
CPU	24 MHz Cortex®-M0	48 MHz Cortex®-M0+	48 MHz Cortex®-M0+	48 MHz Cortex®-M0+	48 MHz Cortex®-M0+	24 MHz Cortex®-M0	24-MHz Cortex®-M0	48 MHz Cortex®-M0+
DMA	該当なし	該当なし	8 チャネル	8 チャネル	8 チャネル	8 チャネル	8 チャネル	16 チャネル
フラッシュメモリ	32 KB	64 KB	128 KB	256 KB	32 KB	128 KB	256 KB	384 KB
SRAM	4 KB	8 KB	16 KB	32 KB	4 KB	16 KB	32 kB	32 KB
GPIO	36	36	54	54	38	55	36	84
CAPSENSE™	1 チャネル, 35 センサー	1 チャネル, 35 センサー	1 チャネル, 53 センサー	1 チャネル, 53 センサー	1 チャネル, 33 センサー	2 チャネル, 54 センサー	1 チャネル, 35 センサー	2 チャネル, 80 センサー (32 control mux)
シーケンサ付き 12 ビット SAR ADC	806 KSPS	1 MSPS	1 MSPS	1 MSPS	1 MSPS	806 KSPS	806 KSPS	1 MSPS
オペアンプ (プログラム可能)	2	2	2	2	4/PGA	4	2	2
プログラム可能電圧リファレンス (PVref)	なし	なし	なし	なし	4 チャネル	なし	なし	なし
電圧 DAC (VDAC)	なし	なし	なし	なし	2 つの 13 ビット VDAC	なし	なし	なし
コンパレータ (ウェイクアップ機能付き低電力)	2	2	2	2	2	2	2	2
IDAC[5]	1 つの 7 ビットと 1 つの 8 ビット	2 つの 7 ビット	2 つの 7 ビット	2 つの 7 ビット	2 つの 7 ビット	2 つの 7 ビットと 2 つの 8 ビット	1 つの 7 ビットと 1 つの 8 ビット	なし

4 PSoC™ 4 Bluetooth® LE ファミリデバイスの使用を開始するためには、**AN91267** を参照してください。

5 IDAC は、CAPSENSE™を使用しない場合にのみ利用可能です。詳細は、各 PSoC™ 4 アーキテクチャ TRM を参照してください。

特長		PSoC™ 4100	PSoC™ 4100S	PSoC™ 4100S Plus	PSoC™ 4100S Plus 256K	PSoC™ 4100 PS	PSoC™ 4100 M	PSoC™ 4100 BL[4]	PSoC™ 4100 S Max
スマート I/O ポート		なし	2	3	2	1	なし	なし	3
電源電圧範囲		1.71 V〜5.5 V	1.71 V〜5.5 V	1.71 V〜5.5 V	1.71 V〜5.5 V	1.71 V〜5.5 V	1.71 V〜5.5 V	1.71 V〜5.5 V	1.71 V〜5.5 V
低消費電力モード	ディープスリープ	1.3 µA	2.5 µA	2.5 µA	2.5 µA	2.5 µA	1.35 µA	2.5 µA	2.5 µA
	ハイバネート	150 nA	該当なし	該当なし	該当なし	該当なし	150 nA	該当なし	該当なし
	ストップ	20 nA	該当なし	該当なし	該当なし	該当なし	35 nA	該当なし	該当なし
セグメント LCD 駆動		4 COM	4 COM	4 COM	4 COM	4 COM	4 COM	4 COM	4 COM
プログラム可能な I^2C, SPI, または UART を備えた SCB		2	3	5	5	3	4	2	5
TCPWM		4	5	8	8	8	8	4	8
CAN		なし	なし	1	なし	なし	なし	なし	1
Bluetooth® LE		なし	なし	なし	なし	なし	なし	4.1/4.2	なし
クロック	IMO	3 MHz〜24 MHz	24 MHz〜48 MHz	24 MHz〜48 MHz	24 MHz〜48 MHz	24 MHz〜48 MHz	3 MHz〜48 MHz	24 MHz〜48 MHz	24 MHz〜48 MHz
	ILO	32 kHz	40 kHz	40 kHz	40 kHz	40 kHz	32 kHz	40 kHz	40 kHz
	WCO	Nil	32 kHZ	32 kHZ	32 kHZ	32 kHZ	32 kHZ	32 kHZ	32 kHZ
電源監視		POR, BOD, 低電圧検知 (LVD)	POR, BOD	POR, BOD	POR, BOD	POR, BOD	POR, BOD, LVD	POR, BOD, LVD	POR, BOD
対応するキット		**CY8CKIT-049 prototyping kit**	**CY8CKIT-041 pioneer kit**	**CY8CKIT-149 prototyping kit**	-	**CY8CKIT-147 prototyping kit**	**CY8CKIT-044 pioneer kit**	**CY8CKIT-042 Bluetooth® LE pioneer kit**	**CY8CKIT-041S-MAX pioneer kit**

特長	PSoC™ 4100	PSoC™ 4100S	PSoC™ 4100S Plus	PSoC™ 4100S Plus 256K	PSoC™ 4100 PS	PSoC™ 4100 M	PSoC™ 4100 BL[4]	PSoC™ 4100 S Max
対応する IDE	PSoC™ Creator	PSoC™ Creator, ModusToolbox™	PSoC™ Creator, ModusToolbox™	PSoC™ Creator, ModusToolbox™	PSoC™ Creator	PSoC™ Creator	PSoC™ Creator	ModusToolbox™

Table 5 PSoC™ 4200 ファミリの特長

特長		PSoC™ 4200	PSoC™ 4200DS	PSoC™ 4200M	PSoC™ 4200L	PSoC™ 4200 BL[6]
CM0 CPU		48 MHz Cortex®-M0	48 MHz Cortex®-M0	48 MHz Cortex®-M0	48 MHz Cortex®-M0	48 MHz Cortex®-M0
DMA		なし	8 チャネル	8 チャネル	32 チャネル	なし
フラッシュメモリ		32 KB	64 KB	128 KB	256 KB	256 KB
SRAM		4 KB	8 KB	16 KB	32 KB	32 KB
GPIO		36	21	55	96	36
CAPSENSE™		1 チャネル, 35 センサー	なし	2 チャネル, 54 センサー	2 チャネル, 94 センサー	1 チャネル, 35 センサー
ADC (12 ビット, シーケンサ付き 1 MSPS SAR ADC)		1	なし	1	1	1
オペアンプ (プログラム可能)		2	なし	2	4	2
コンパレータ (ウェイクアップ機能付き低電力)		2	2	2	2	2
IDAC[7]		1 つの 7 ビットと 1 つの 8 ビット	なし	2 つの 7 ビットと 2 つの 8 ビット	2 つの 7 ビットと 2 つの 8 ビット	1 つの 7 ビットと 1 つの 8 ビット
プログラム可能論理ブロック (UDB)		4	4	4	8	4
スマート I/O ポート		なし	1	なし	なし	なし
電源電圧範囲		1.71 V～5.5 V	1.71 V～5.5 V	1.71 V～5.5 V	1.71 V～5.5 V	1.71 V～5.5 V
低消費電力モード	ディープスリープ	1.3 µA	2 µA	1.3 µA	1.3 µA	1.5 µA
	ハイバネート	150 nA	該当なし	150 nA	150 nA	150 nA
	ストップ	20 nA	該当なし	20 nA	20 nA	20 nA
セグメント LCD 駆動		4 COM	なし	4 COM	8 COM	4 COM
プログラム可能 I^2C, SPI, または UART 付き SCB		2	3	4	4	2
TCPWM		4	4	8	8	4
CAN		なし	なし	2	2	なし
Bluetooth® LE		なし	なし	なし	なし	4.1/4.2

6 PSoC™ 4 Bluetooth® LE ファミリデバイスの使用を開始するためには、**AN91267** を参照してください。

7 IDAC は、CAPSENSE™を使用しない場合にのみ利用可能です。詳細は、各 PSoC™ 4 アーキテクチャ TRM を参照してください。

PSoC™ 4 の機能セット

特長		PSoC™ 4200	PSoC™ 4200DS	PSoC™ 4200M	PSoC™ 4200L	PSoC™ 4200 BL[6]
USB フルスピードデバイスコントローラ (USB)		なし	なし	なし	あり	なし
クロック	IMO	3 MHz～48 MHz	3 MHz～48 MHz	3 MHz～48 MHz	3 MHz～48 MHz	3 MHz～48 MHz
	ILO	32 kHz	40 kHz	32 kHz	32 kHz	32 kHz
	WCO	なし	なし	32 kHz	32 kHz	32 kHz
	外部水晶発振器 (ECO)	なし	なし	4 MHz～33 MHz	なし	なし
電源監視		POR, BOD, LVD	POR, BOD	POR, BOD, LVD	POR, BOD, LVD	POR, BOD, LVD
対応するキット		**CY8CKIT-042 pioneer kit**	**CY8CKIT-146 prototyping kit**	**CY8CKIT-044 pioneer kit**	**CY8CKIT-046 pioneer kit**	**CY8CKIT-042 Bluetooth® LE pioneer kit**
対応する IDE		**PSoC™ Creator**	**PSoC™ Creator**	**PSoC™ Creator**	**PSoC™ Creator**	**PSoC™ Creator**

Table 6　PSoC™ 4500 および PSoC™ 4700 ファミリの特長

特長		PSoC™ 4500S	PSoC™ 4700S
CM0+ CPU		48 MHz Cortex®-M0+	48 MHz Cortex®-M0+
DMA		8 チャネル	なし
フラッシュメモリ		256 KB	32 KB
SRAM		32 KB	4 KB
GPIO		53	36
CAPSENSE™		1 チャネル, 52 センサー	1 チャネル, 35 センサー
MagSense		なし	1 チャネル
ADC		2 つの 12 ビット, シーケンサ付き 1 MSPS SAR ADC	10 ビット, 16.8 ksps シングルスロープ ADC
オペアンプ (プログラム可能)		4	なし
コンパレータ (ウェイクアップ機能付き)		2	2
IDAC[8]		2 つの 7 ビット	2 つの 7 ビット
スマート I/O ポート		2	2
電源電圧範囲		1.71 V～5.5 V	1.71 V～5.5 V
低消費電力モード	ディープスリープ	1.3 µA	2.5 µA
	ハイバネート	150 nA	NA
	ストップ	20 nA	NA
セグメント LCD 駆動		4 COM	8 COM

8 IDAC は、CAPSENSE™が使用されていない場合にのみ使用できます。詳細については、それぞれの PSoC™4 アーキテクチャ TRM を参照してください。

PSoC™ 4 の機能セット

特長		PSoC™ 4500S	PSoC™ 4700S
プログラム可能 I^2C, SPI, または UART 付き SCB		5	2
TCPWM		8	5
モーター制御アクセラレーション (MCA)		2	なし
クロック	IMO	24 MHz～48 MHz	24 MHz～48 MHz
	ILO	40 kHz	40 kHz
	WCO	32 kHz	32 kHz
	ECO	4 MHz～33 MHz	なし
電源監視		POR, BOD	POR, BOD
対応するキット		**CY8CKIT-045S pioneer kit**	**CY8CKIT-148 evaluation kit**
対応する IDE		**PSoC™ Creator**	**PSoC™ Creator**

4 PSoC™が MCU より優れている点

Figure 10 は標準的な MCU を示し、CPU (8051 または Arm® Cortex®など) および ADC, DAC, UART, SPI 等のペリフェラル機能および汎用 I/O を含みます。これらはすべて CPU のレジスタ インターフェースにリンクします。MCU 内で、CPU はデバイスの 「心臓」です。CPU はセットアップ, データ移動, タイミングなどをすべての処理を管理します。CPU なしでは、MCU は機能しません。

Figure 11 に示すように、PSoC™はまったく違います。PSoC™では、CPU, アナログ, デジタル, および I/O は、プログラマブルなシステムで同等に重要なリソースです。*PSoC™の心臓はシステムの相互接続とプログラマビリティであり、CPU ではありません。*ペリフェラル アナログとデジタルは高度に設定可能な信号マトリックスおよびデータ バス メッシュを介して相互接続します。これにより、ユーザーがアプリケーション要件を満たすカスタム設計を作成可能にします。*PSoC™をプログラムして MCU をエミュレートできますが、MCU をプログラムして PSoC™をエミュレートできません。*

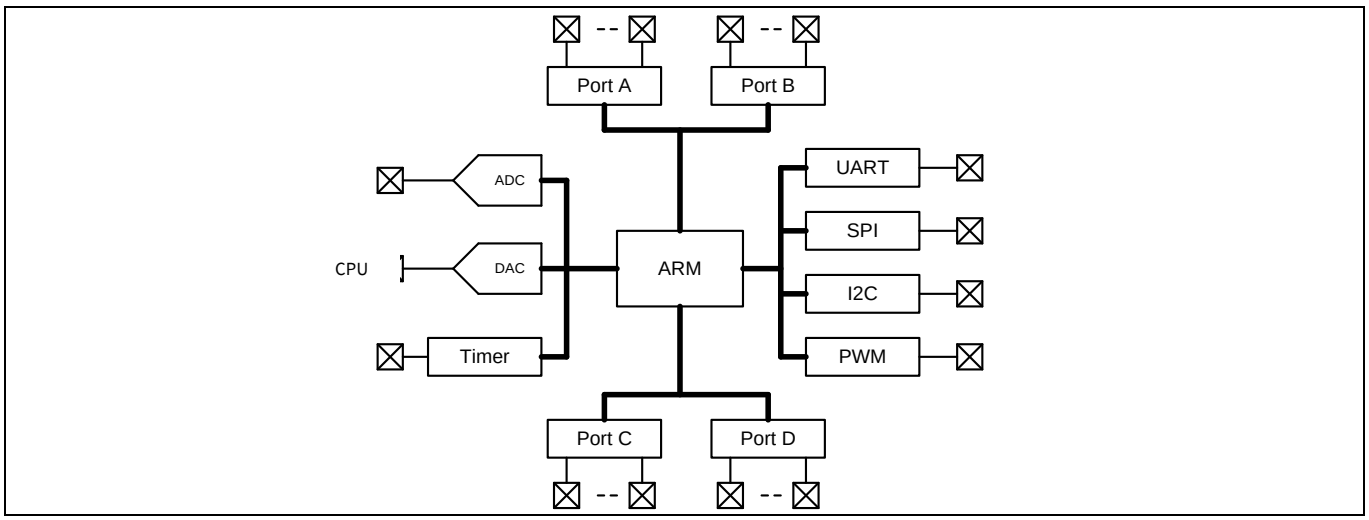

Figure 10 標準的な MCU のブロックダイヤグラム

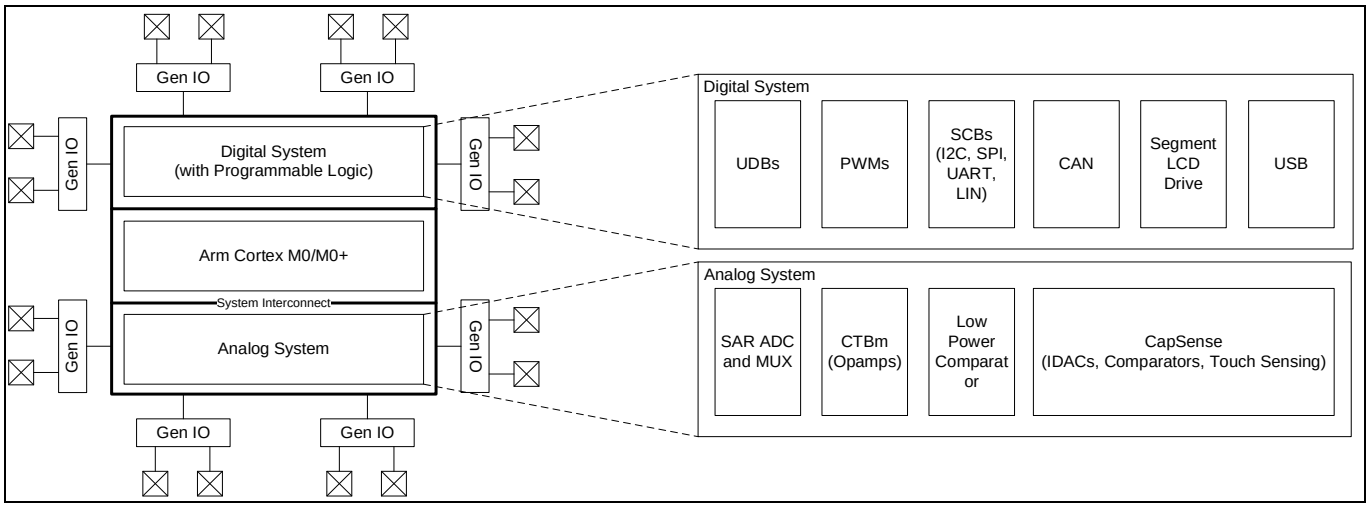

Figure 11 PSoC™のブロックダイヤグラム

標準的な MCU は、ステート マシンを処理し、タイマーをタイミングに使用し、出力ピンを駆動するために、CPU ファームウェアを必要とします。したがって、機能的なパスはほとんどの場合が CPU を介します。しかし、PSoC™では非同期の平行処理が可能です。PSoC™を設定し、CPU から独立して動作する

要素を取得できます。このアプリケーション ノートに組み込まれるプロジェクトはこの概念を示します。PSoC™は CPU にコードを書かずに LED を点滅させるように設定されます。

4.1 PSoC™ Creator コンポーネントのコンセプト

PSoC™のその他の重要な点は PSoC™ Creator IDE の使用可能性です。PSoC™ Creator では、異なる PSoC™ リソースは設計を迅速にビルドするために、回路図にドラッグ アンド ドロップできるコンポーネントと呼ばれているグラフィック要素として構成されます。すべての PSoC™のペリフェラルは、検証済み PSoC™ Creator コンポーネント (PWM コンポーネント、ADC コンポーネント、DAC コンポーネント、CAPSENSE™コンポーネント、UART コンポーネントなど) として使用できます。PSoC™ Creator の検証済みコンポーネントの使用可能性は開発時間を著しく短縮します。それにより、グラフィック オプションを用いて、設計も迅速に変更できます。

例えば、標準的なマイクロコントローラーで PWM を設定し、LED を点滅させることは以下のことを含みます。

1. PWM ブロックに対応したレジスタの位置付け
2. 要求される PWM 周期およびデューティ比に基づいて、PWM レジスタに書き込む値を計算
3. PWM レジスタを設定するために数多くのコード ラインを書き、ピン駆動モードを設定し、PWM 出力をピンに接続。数多くの MCU は内部ブロックに接続するために、代わりのピンは提供しない

PSoC™での同じ機能の実装は、本アプリケーション ノートの後半で説明する簡単な操作です。これから、タイマーに同じ PWM ブロックを再設定する場合は、PSoC™ Creator ではわずかなマウス クリックだけで、それ以上に手間がかかりません。

PSoC™には、ユニバーサル デジタル ブロック (UDB) として知られるプログラマブル デジタル ブロックもあります。PSoC™ Creator はまた、UART、SPI、I2S、タイマー、PWM、カウンター、デジタル ゲート (AND、OR、NOT、XOR) など、UDB からなる様々なコンポーネントを提供します。PSoC™ Creator で UDB を使用し、自分のカスタム ステート マシンおよびデジタル ロジックを作り上げられます。カスタム PSoC™ Creator コンポーネントの作成方法は **PSoC™ Creator™ Component author guide** を参照してください。

5 ModusToolbox™を使用するはじめての PSoC™ 4 設計

ここでは、

- 従来の MCU 以上のことを行うように PSoC™をプログラムする方法を示します。
- 単純な PSoC™アプリケーションを構築し、それを開発キットにプログラムする方法を示します。
- **ModusToolbox™**に Eclipse IDE を使用するための詳細な手順を提供します。

ただし、ModusToolbox™ソフトウェアは IDE に依存しません。ここの手順では、ModusToolbox™用の Eclipse IDE を使用して、Project Creator を起動します。結果のプロジェクトは自動的に IDE にインポートされます。

Linux, macOS, または Windows では、Project Creator, Library Manager, および任意の Configurator をスタンドアロンツールとして使用できることに注意してください。これらのツールは、アプリケーションフォルダー内のファイルを作成または変更します。コマンドラインから、そのアプリケーションを、VS Code, IAR Embedded Workbench, Keil µVision などのサポートされているサードパーティの IDE にエクスポートできます。Eclipse IDE の使用は必須ではありません。詳細については、**ModusToolbox™ user guide** の「IDE へのエクスポート」を参照してください。

5.1 インストールの前に

5.1.1 ModusToolbox™をインストールしましたか？

ModusToolbox™ホームページから ModusToolbox™をダウンロードしてインストールしてください。ソフトウェアをインストールした後、ModusToolbox™ IDE のクイックスタートガイドとユーザーガイドを参照して、ソフトウェアの概要を確認してください。

5.1.2 開発キットまたはプロトタイピングキットはありますか？

この設計をテストするためには、プログラマが統合された、**Table 7** にリストされているキットの 1 つが必要です。

Table 7 PSoC™ 4 パイオニアキット、プロトタイピングキット、およびサポートされているデバイスのリスト

キット名	キット形式	対応するデバイスファミリ	製品番号
CY8CKIT-145	プロトタイピングキット	PSoC™ 4000S	CY8C4045AZI-S413
CY8CKIT-041-41XX	パイオニアキット	PSoC™ 4100S	CY8C4146AZI-S433
CY8CKIT-149	プロトタイピングキット	PSoC™ 4100S Plus	CY8C4147AZI-S475
CY8CKIT-041S-Max	パイオニアキット	PSoC™ 4100S Max	CY8C4149AZI-S598

5.2 これらの手順の使用

これらの手順は、いくつかのセクションにグループ化されています。各セクションでは、アプリケーション開発ワークフローのフェーズについて説明します。主なセクションは次のとおりです。

- **パート 1: 新しいアプリケーションの作成**
- **パート 2: 設計の参照と変更**
- **パート 3: ファームウェアの書き込み**
- **パート 4: アプリケーションの作成**
- **パート 5: デバイスのプログラム**

- パート 6: 設計のテスト

この設計は、**Table 7** にリストされているキット用に開発されています。アプリケーションの作成中に適切なキットを選択することにより、この例をテストできます。

5.3 設計について

この設計では、PSoC™ 4 の CM0+ CPU を使用して、UART 通信と LED 制御の 2 つのタスクを実行します。CM0+ CPU は UART を使用して「Hello World」メッセージをシリアルポートストリームに出力し、キットのユーザーLED の点滅を開始します。

5.4 パート 1: 新しいアプリケーションの作成

ここでは、新しいアプリケーションを作成するためのステップバイステップのプロセスについて説明します。「**Empty PSoC™4 App**」スターターアプリケーションは、設計開発段階とプログラミングをガイドするために使用されます。

ModusToolbox™を使用したプロジェクトの開発に精通している場合は、「**Hello World**」スターターアプリケーションを直接使用できます。詳細については、サンプルコードを参照してください。このスターターアプリケーションは完全な設計であり、サポートされているキット用にファームウェアが作成されています。手順を確認し、サンプルコードで手順がどのように実装されているかを確認できます。

最初から始めてこのアプリケーションノートの指示に従う場合でも、サンプルコードを参照として使用できます。

ModusToolbox™を起動して開始します。ModusToolbox™は、スターターアプリケーションをマシンに正常に複製するために、インターネットにアクセスする必要があることに注意してください。

5.4.1 新しい workspace (ワークスペース) の選択

ModusToolbox™を起動すると、ワークスペースのディレクトリを選択できるダイアログが表示されます。ワークスペースディレクトリは、ワークスペースの設定と開発成果物を保存するために使用されます。**Browse** をクリックして、既存の空のディレクトリを選択してください。または、完全なパスとともにディレクトリ名を入力すると、ModusToolbox™によってディレクトリが作成されます。

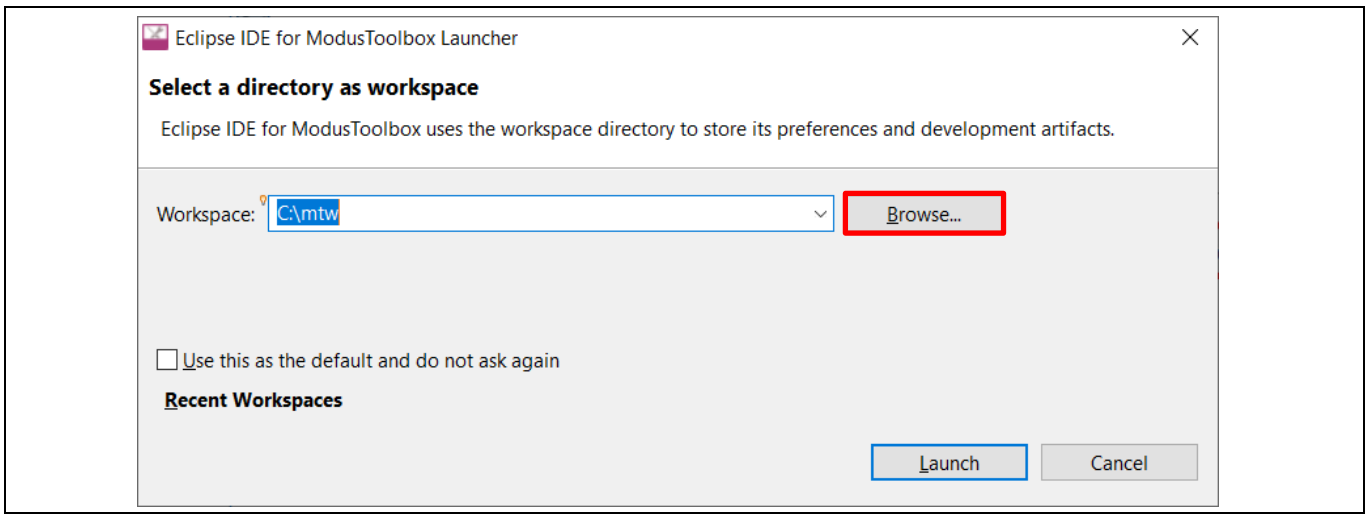

Figure 12 ワークスペースのディレクトリの選択

5.4.2 新しい ModusToolbox™アプリケーションの作成

a) **Quick Panel** の **Start** グループで **New Application** をクリックしてください。

b) または、**File > New > ModusToolbox Application** を選択してください。

ModusToolbox™ IDE application ウィンドウが表示されます。

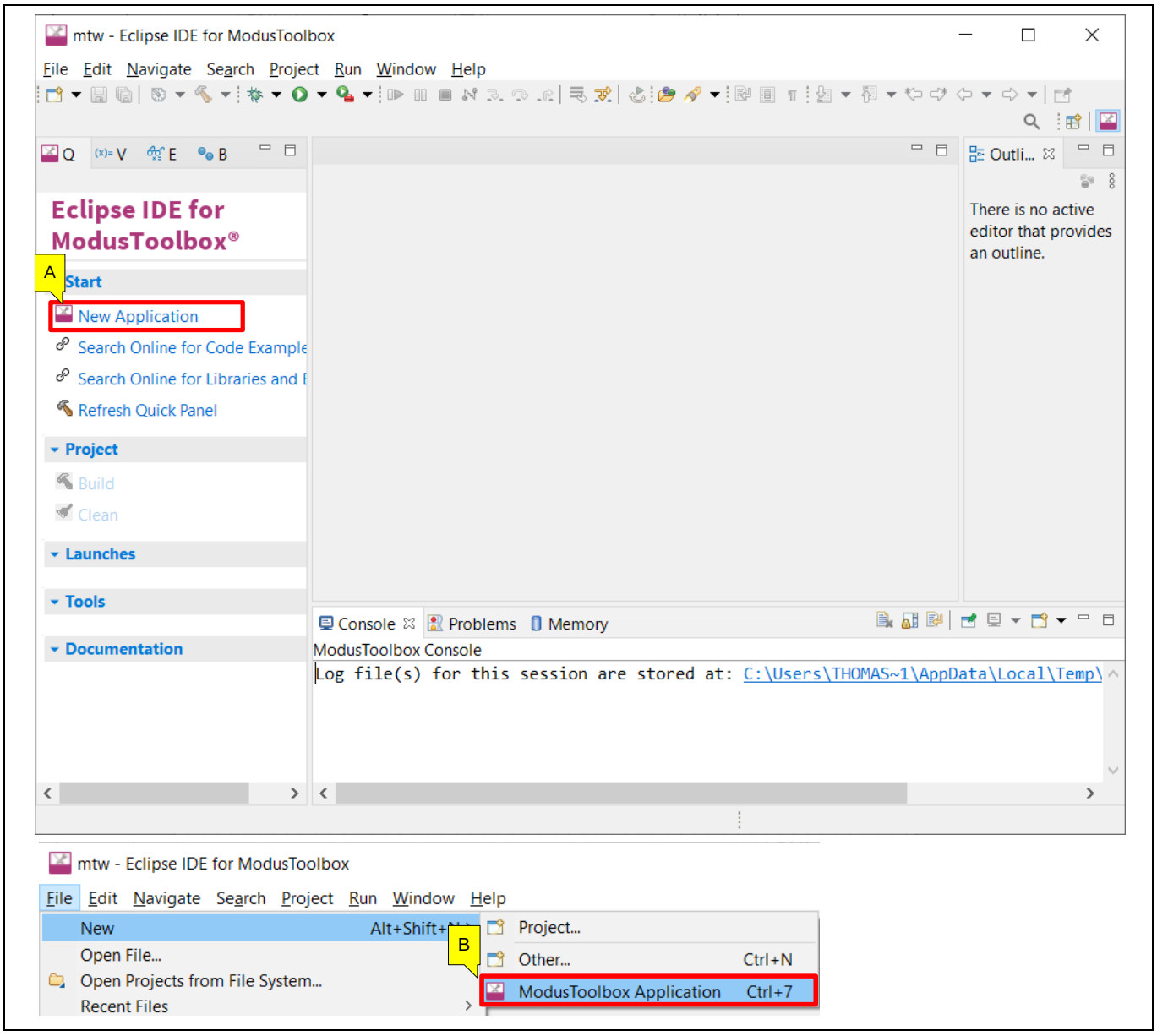

Figure 13 新しい ModusToolbox™ IDE アプリケーションの作成

5.4.3 ターゲット PSoC™ 4 開発キットの選択

ModusToolbox™は、新しいアプリケーションダイアログで指定された開発キットのさまざまなワークスペース/プロジェクトオプションを設定する BSP を提供することにより、開発プロセスをスピードアップします。

a) Choose Board Support Package (BSP)ダイアログで、Kit Name を選択してください (例えば CY8CKIT-041S-MAX) 。

b) **Next** をクリックしてください。

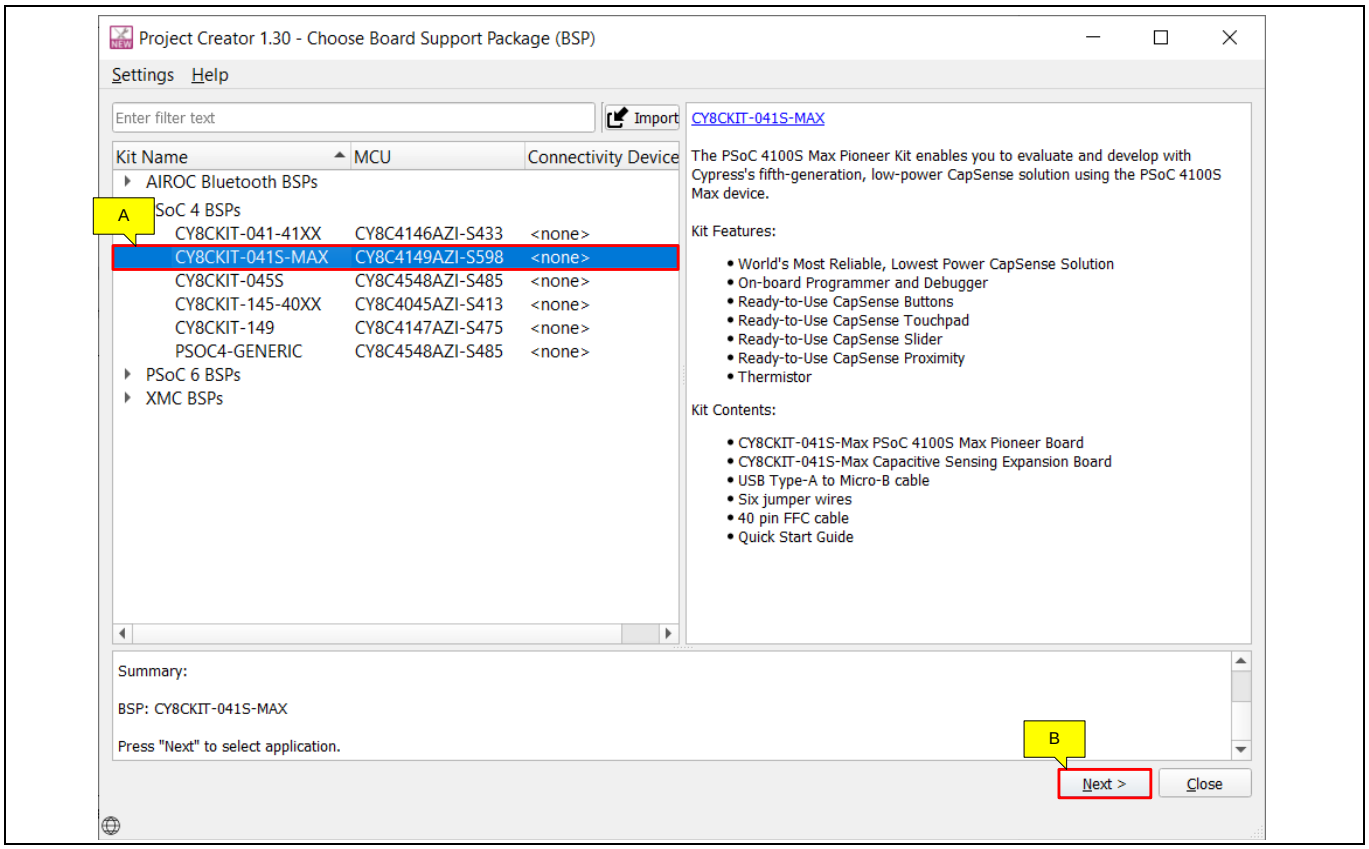

Figure 14　　ターゲットハードウェアの選択

c) Select Application ダイアログで、Empty PSoC™4 App テンプレートアプリケーションを選択してください。

d) **New Application Name** フィールドに、**Hello_World** などのアプリケーションの名前を入力してください。デフォルトの名前を保持することもできます。

e) **Create** をクリックしてください。ModusToolbox™はアプリケーションプロジェクトを作成します。

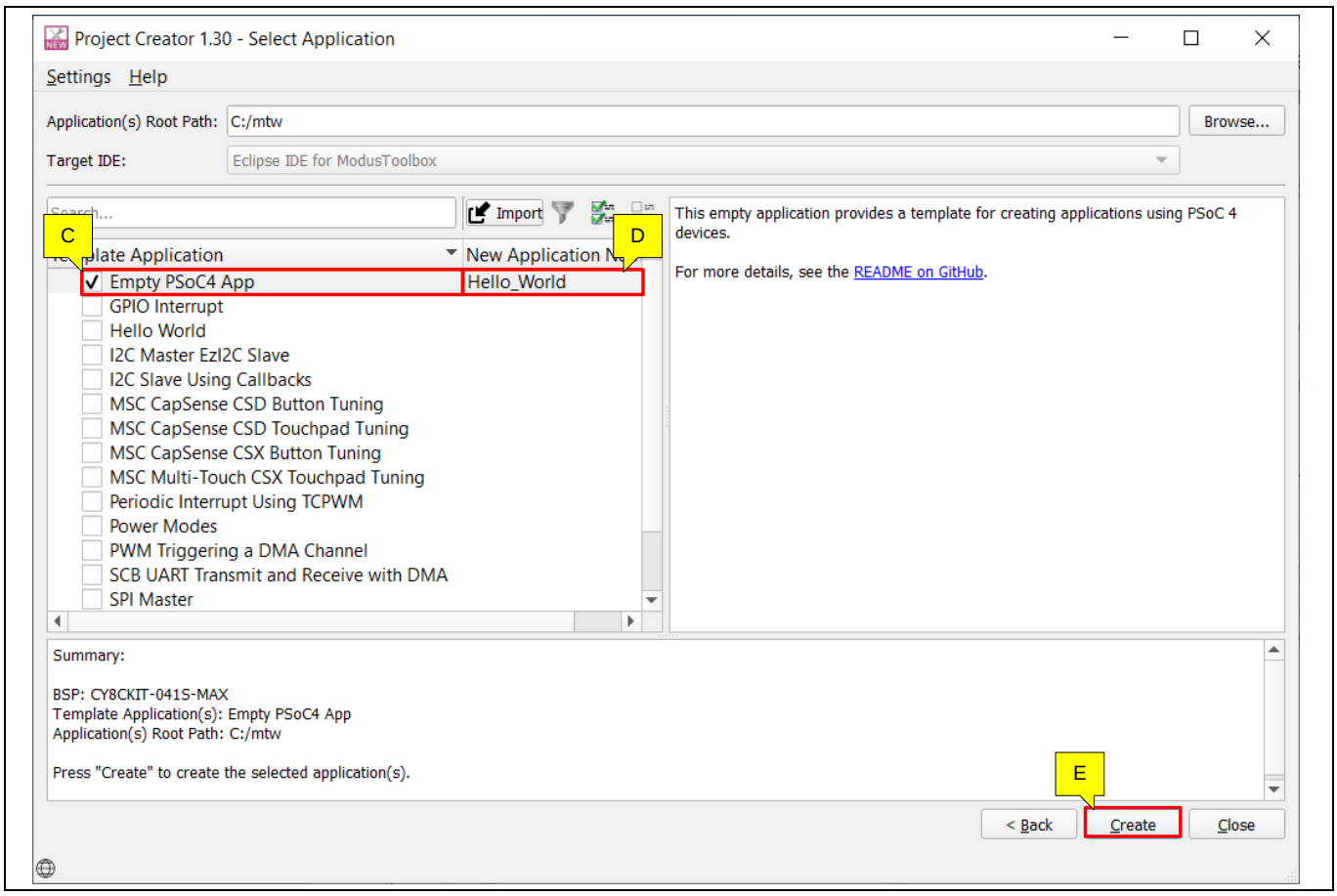

Figure 15 スターターアプリケーションの選択

これで、PSoC™ 4 用の新しい ModusToolbox™アプリケーションが正常に作成されました。

BSP は、選択したキットに基づいて、**Table 7** にリストされているデバイスを使用します。

PSoC™ 4 に基づくカスタムハードウェア、または別の PSoC™ 4 製品番号を使用している場合は、**ModusToolbox™ user guide** を参照してください。このガイドは、ModusToolbox™インストールディレクトリの *ide_2.2>docs* フォルダにもあります。

5.5 パート 2: 設計の参照と変更

5.5.1 プロジェクト構成

Figure 16 に、アプリケーションプロジェクトの構造を表示する ModusToolbox™プロジェクトエクスプローラーを示します。

ModusToolbox™ IDE では、PSoC™ 4 アプリケーションは CM0+ CPU のコードを開発するプロジェクトで構成されます。

a) プロジェクトフォルダはさまざまなサブ フォルダで構成され、それぞれがプロジェクトの特定の側面を示します。

b) *ビルド*フォルダには、プロジェクトの make ビルドから生じるすべてのアーティファクトが含まれます。出力ファイルは、ターゲット BSP ごとに編成されます。

c) *libs* フォルダには、アプリケーションに対してローカルなライブラリがあります。デフォルトでは、作成されたアプリケーションの BSP はローカルであり、このフォルダで使用できます。これにより、変更が他のアプリケーションに伝播することを心配せずに、特定のアプリケーションの BSP を変更できます。

d) *mtb_shared* フォルダには、異なるミドルウェアに属するサブフォルダ (PSoC™ 4 HAL, PSoC™ 4 PDL, CAPSENSE™など) があります。これらは、プロジェクト内で提供される*.mtb* ファイルに基づいてダウンロードされる個々のライブラリです。これらのライブラリは、デフォルトで、ワークスペース内のプロジェクト間で共有されます。

e) *mtb* ファイルは、ModusToolbox™がコンテンツをプルする場所を提供します。これらのファイルには通常、ライブラリ全体の GitHub の場所が含まれます。*.mtb* ファイルは、別の*.mtb* ファイルを含むコンテンツを指すことができます。ModusToolbox™は、このネストされた*.mtb* ファイルを再帰的に処理し、すべてのライブラリをダウンロードします。

例えば、BSP lib ファイル *TARGET_CY8CKIT-041S-MAX.mtb* は、**https://github.com/infineon/TARGET_CY8CKIT-041S-MAX#latest-v1.X**。リンク内の latest-v1.X タグは、BSP の特定のリリースを示します。

別の例では、capsense.mtb は **https://github.com/Infineon/capsense#latest-v3.X** でホストされるライブラリを指します。

f) アプリケーションプロジェクトには Makefile が含まれていることに注意してください。これには、プロジェクトを再作成する方法の説明があります。このファイルには、make ツールがアプリケーションプロジェクトをコンパイルおよびリンクするために使用する一連のディレクティブも含まれます。

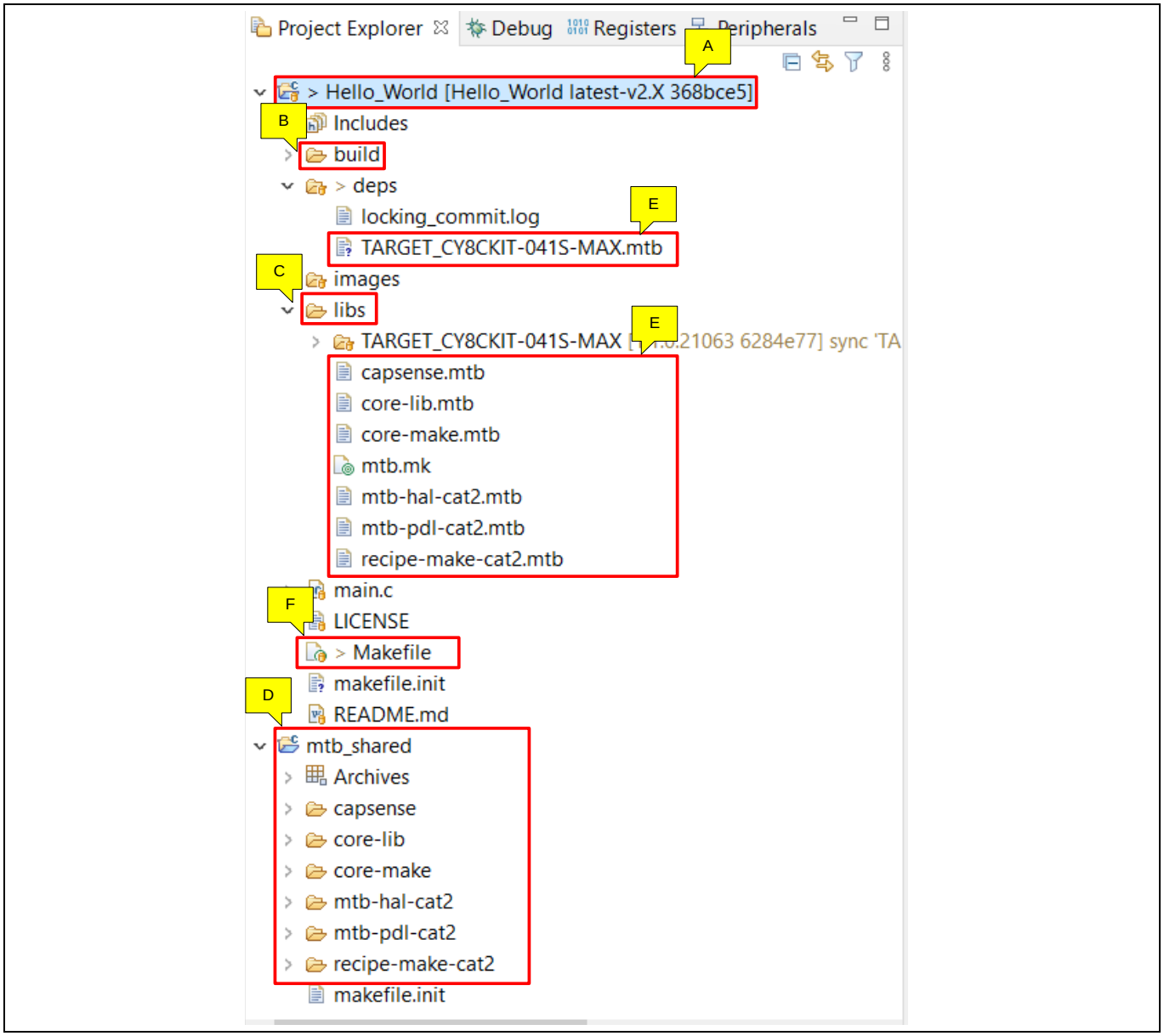

Figure 16　　プロジェクトエクスプローラー外観

g) BSP によって提供されるファイルは、*libs* フォルダの *TARGET_x* フォルダの下にリストされています。デバイスおよび周辺コンフィギュレーターによって生成されたすべての設定ファイルは、BSP の *COMPONENT_BSP_DESIGN_MODUS / GeneratedSource* フォルダに含まれ、接頭辞 *cycfg_* が付けられます。これらのファイルには、BSP によって定義された設計設定が含まれます。 *design.modus* ファイルをダブルクリックして、デザイン設定を表示してください。

BSP フォルダには、プロジェクトで使用されるミドルウェアとその他のライブラリを指定する *deps* フォルダ内の.*lib* ファイル、およびボードで使用される PSoC™ 4 デバイスのリンカースクリプトと起動コードも含まれます。

Figure 17 に、このワークスペースの BSP ファイルを含む *libs* フォルダの構造を表示する ModusToolbox™プロジェクトエクスプローラーを示します。

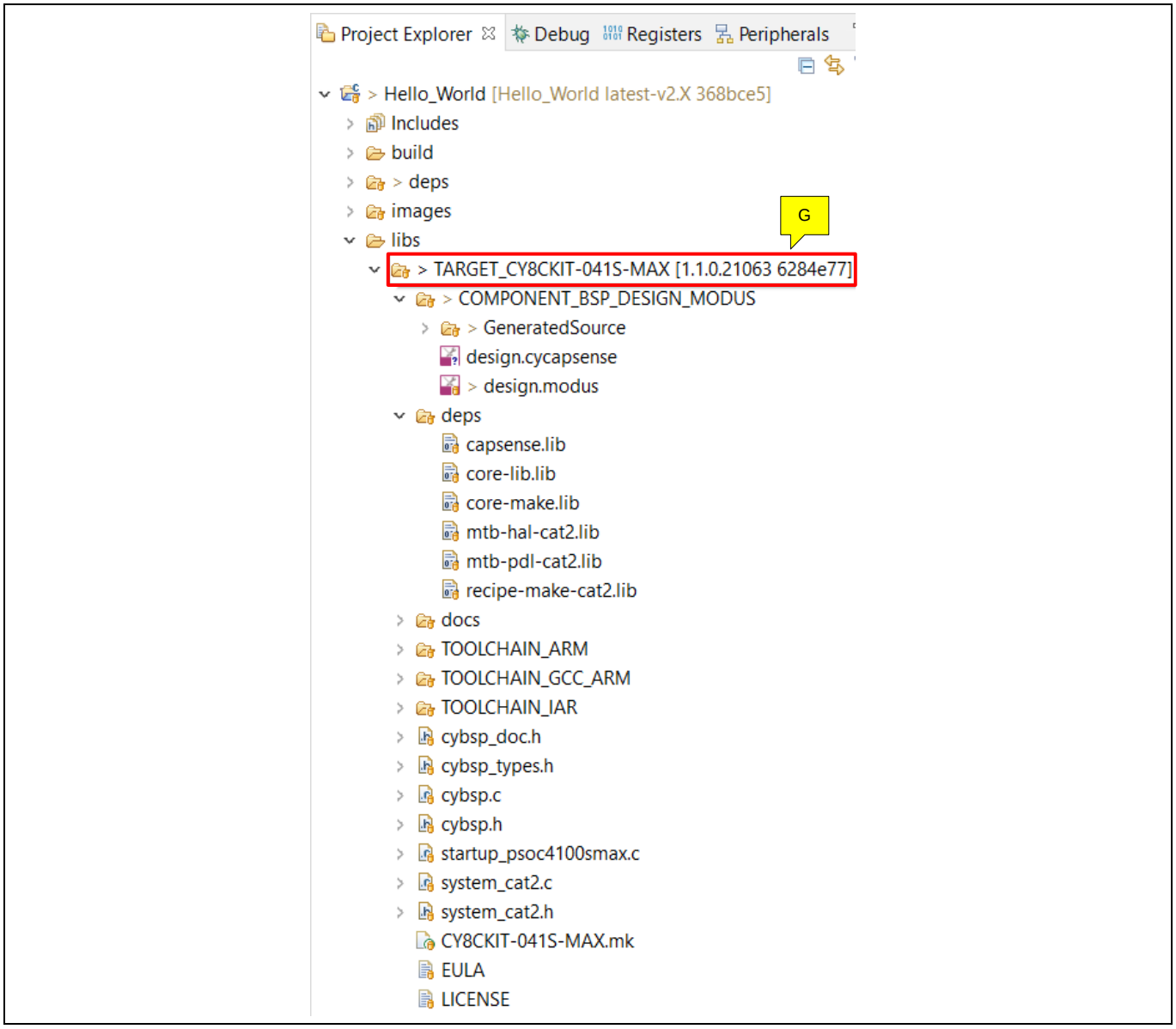

Figure 17　　プロジェクトエクスプローラー外観 - 展開された *libs/TARGET_x* フォルダ

5.5.2　　設計の変更

1. *libs>TARGET_X>COMPONENT_BSP_x* フォルダから design.modus ファイルをダブルクリックしてください。

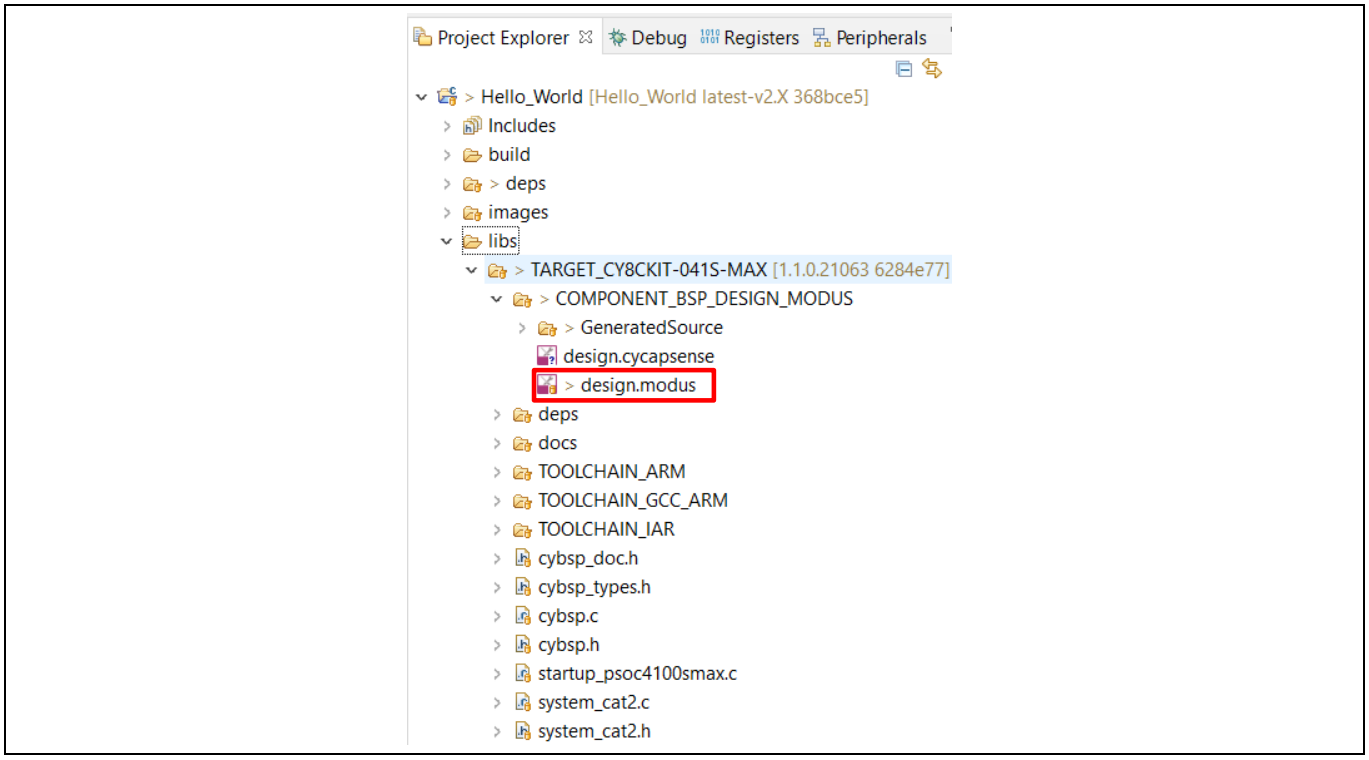

Figure 18 ***design.modus* ファイルを開く**

これにより、**Device Configurator** ダイアログが開きます。他の *design.x* ファイルをダブルクリックしてそれぞれのコンフィギュレーターで開くか、**Quick Panel** の対応するリンクをクリックして必要に応じて設定することもできます。

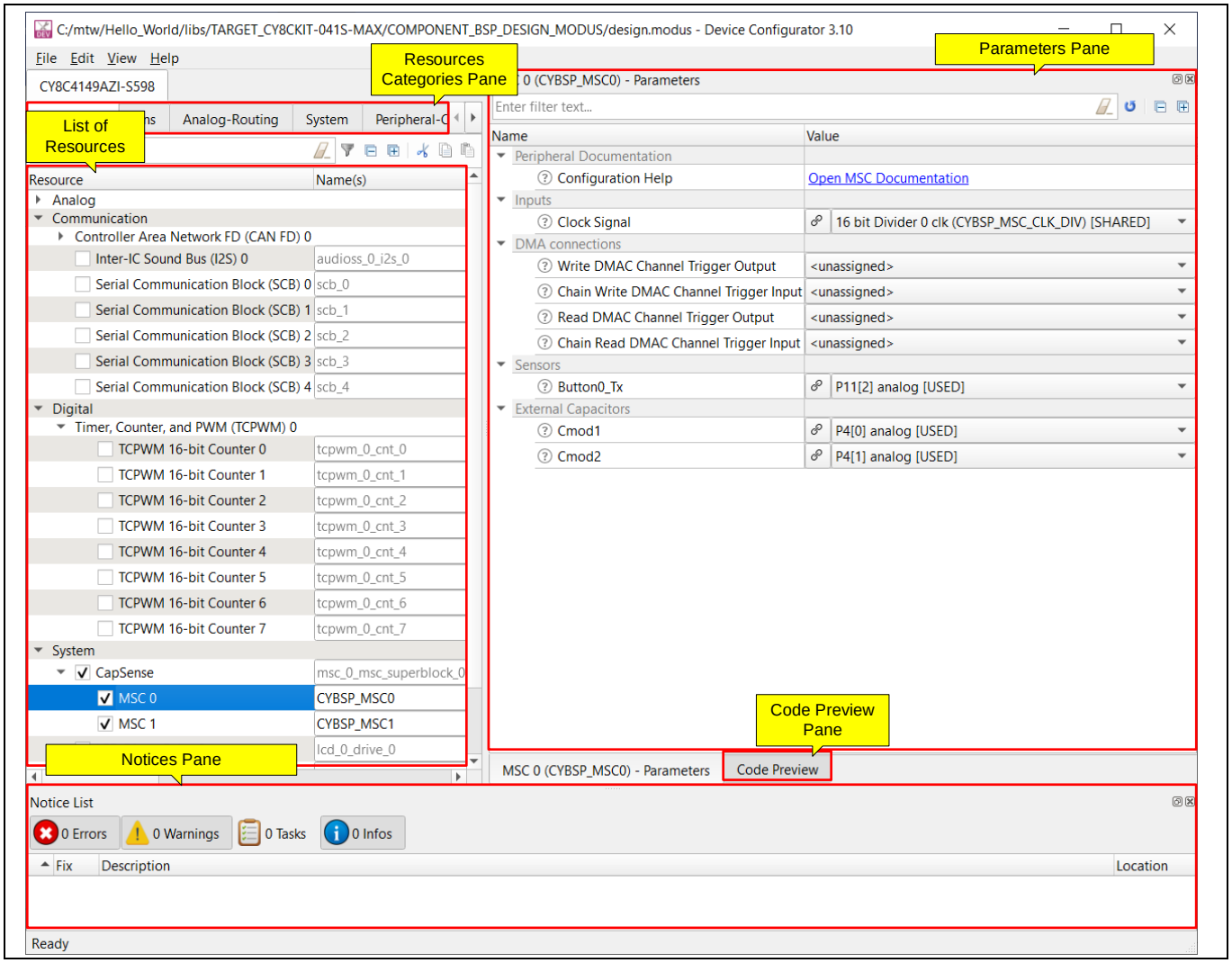

Figure 19 ***design.modus* の概要**

Device Configurator ダイアログの **Resources Category** ペインから、デバイスで使用可能な周辺機器、ピン、クロックなどのさまざまなリソースから選択できます。**Peripheral** タブには、デバイスで使用可能な **List of Resource** のリストが表示されます。

Personality は、リソースの動作を定義します。たとえば、**Serial Communication Block (SCB)** リソースには、**EZI2C**, **I2C**, **SPI**, または **UART** パーソナリティを含めることができます。**Name(s)**フィールドは、ファームウェア開発で使用されるリソースの名前です。1 つ以上の名前をコンマで区切って指定できます (スペースなし)。

Parameter ペインで、有効な各リソースおよび選択したパーソナリティの構成パラメーターを入力できます。**Code Preview** ペインには、選択した設定パラメーターに対して生成された設定コードが表示されます。このコードは、*GeneratedSource* フォルダの *cycfg_*ファイルに入力されます。設定に起因するエラー, 警告, および情報メッセージは、**Notices** ペインに表示されます。

アプリケーションプロジェクトには、CM0+ CPU (*main.c*) 用のアプリケーションの作成に役立つ関連ファイルが含まれます。この C ファイルは、通常のビルドプロセスの一部としてコンパイルされ、CM0+イメージにリンクされます。

2. ResourceCategories ペインの **Peripheral-Clocks** タブに移動して、UART コンポーネントのクロックを設定してください。

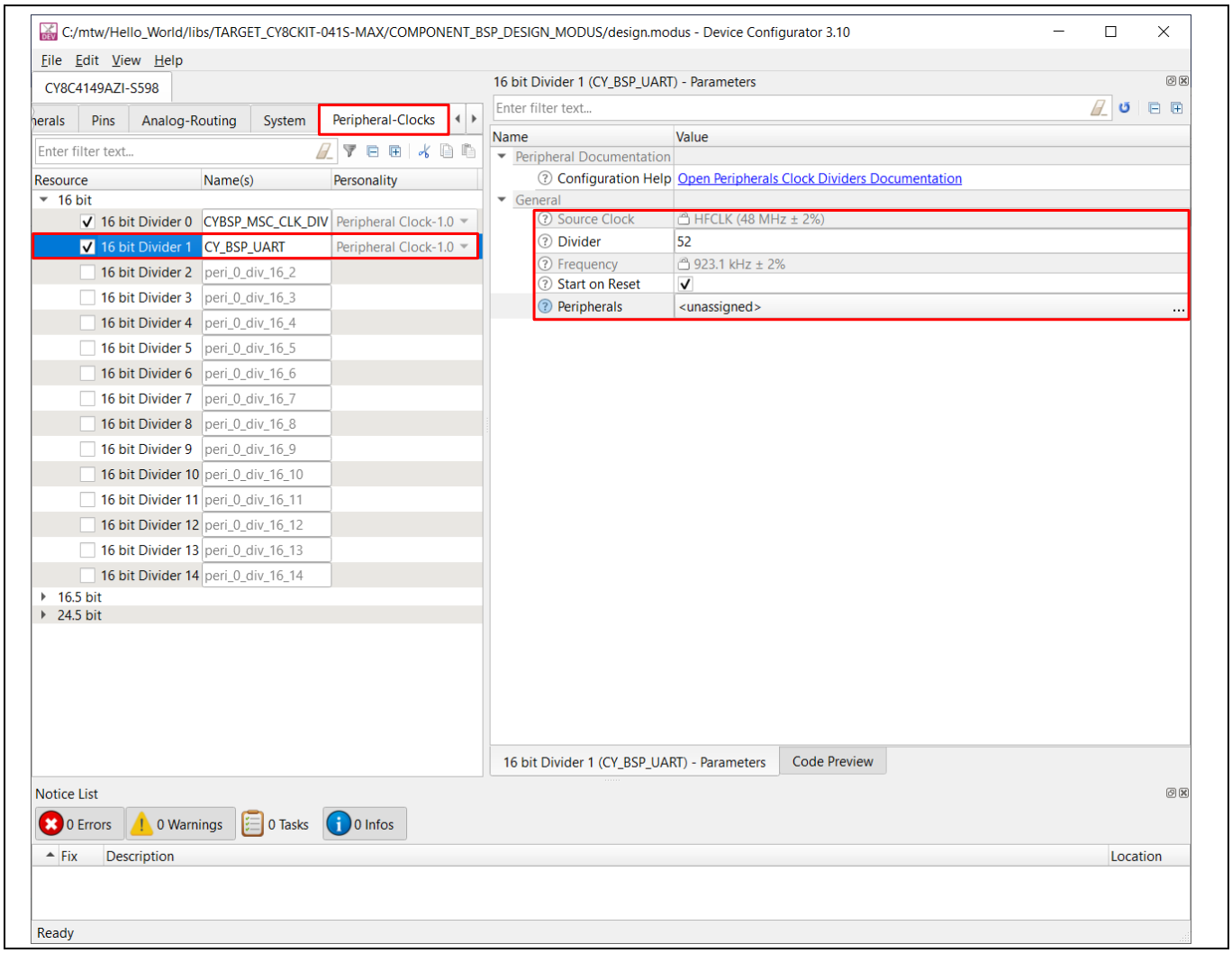

Figure 20　　クロックの設定

3. Resource Categories ペインの Peripherals タブに移動し、Serial Communication Block (SCB) 1 を有効にして、パーソナリティを UART-1.0 に設定し、名前を CYBSP_UART にしてください。**Figure 21** では、キットに必要な Rx/Tx ピンに基づいて SCB インスタンス 1 が選択されています。各インスタンスで使用可能なピンについては、デバイスデータシートを参照してください。

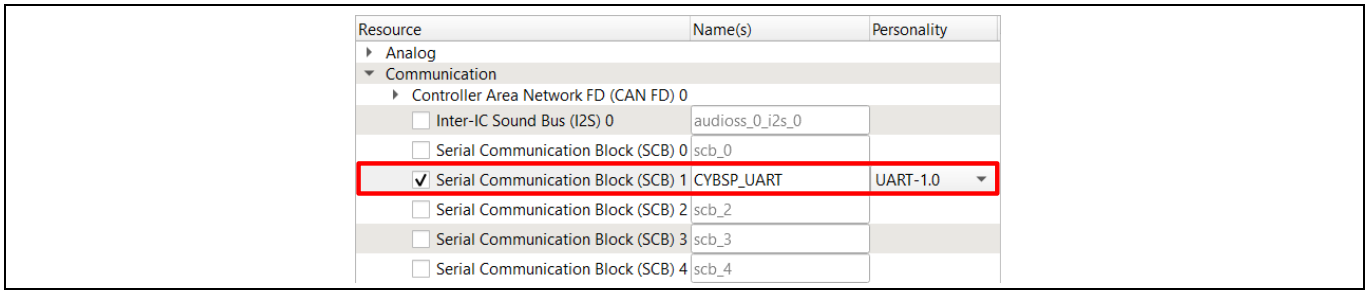

Figure 21　　UART として SCB の有効化

4. **Parameter** ペインで、**Clock**、**Rx**、および **Tx** などの必要な設定パラメーターを設定してください。他のパラメーターのデフォルト値を保持してください。

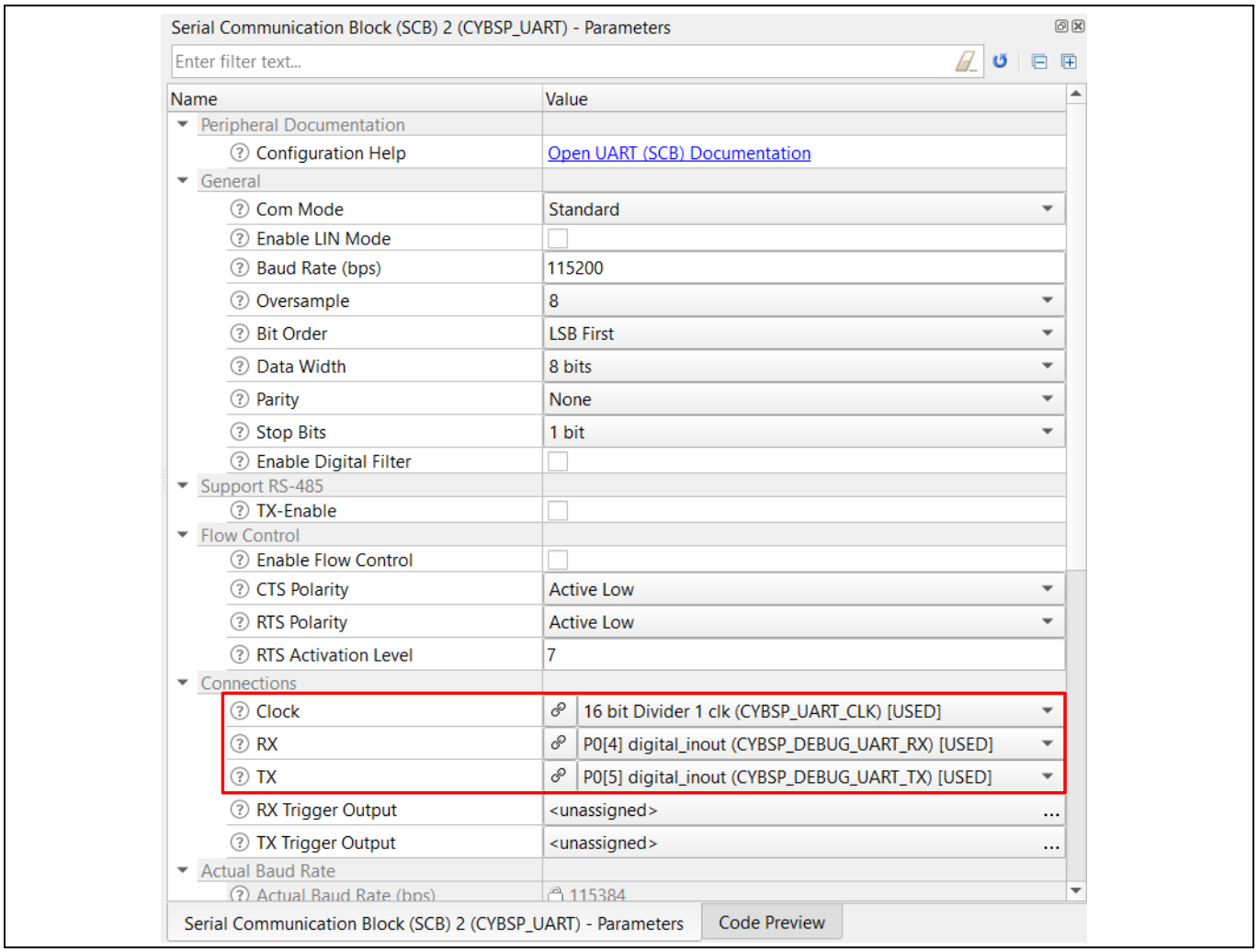

Figure 22　　UART 設定パラメーターの設定

5. **Code Preview** ペインに移動して、生成されたコードをプレビューしてください。

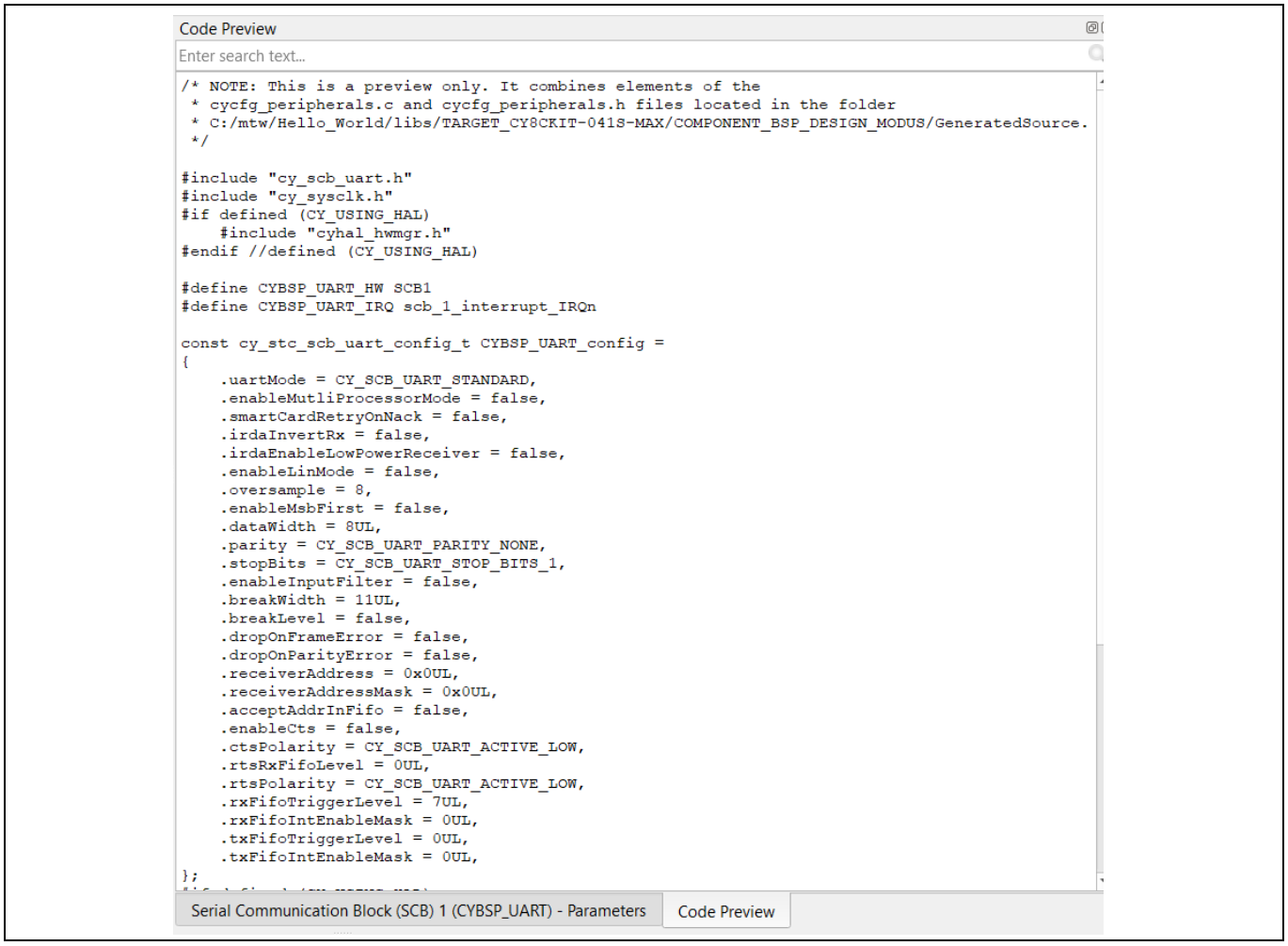

Figure 23　　コードプレビューペイン

6. **Resource Categories** ペインの **Pins** タブに移動して、キットのユーザーLED が点滅するように GPIO を構成してください。別の PSoC™ 4 キットを使用している場合は、**Table 8** を参照してください。

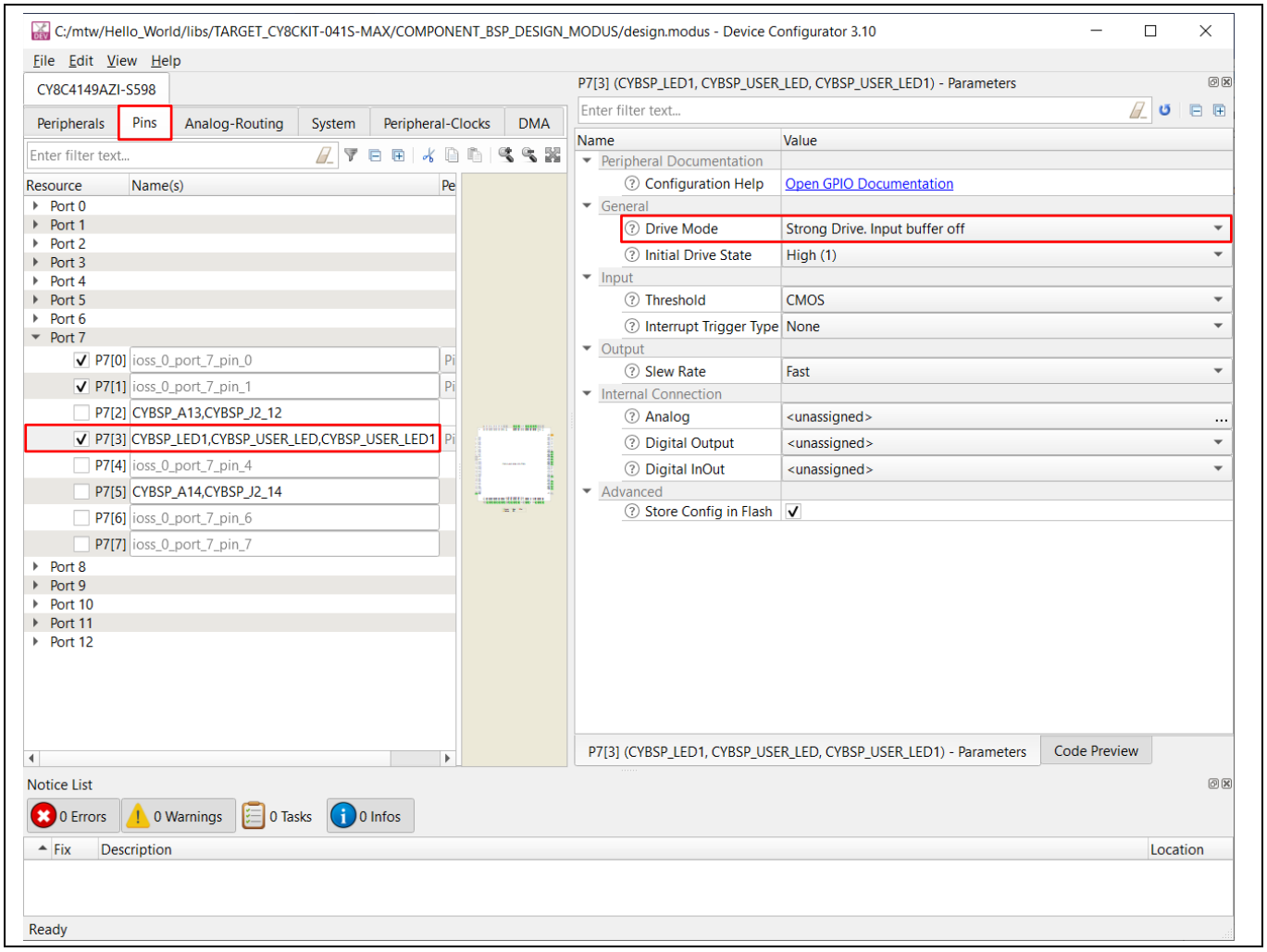

Figure 24 GPIO の設定

Table 8 PSoC™ 4 キット全体のピンマッピング表

機能	CY8CKIT-145 (PSoC™ 4000S)	CY8CKIT-041-41XX (PSoC™ 4100S)	CY8CKIT-149 (PSoC™ 4100S Plus)	CY8CKIT-041S-MAX (PSoC™ 4100S Max)
ユーザーLED	P2[5]	P3[4]	P3[4]	P7[3]

7. **Resource Categories** ペインのさまざまなタブに移動して、他のリソースを表示および設定してください。この設計では、これ以上の変更は必要ありません。
8. パス **File → Save** に従うか、**[Ctrl]+[S]**を押して構成を保存し、*GeneratedSource* フォルダにソースコードを生成してください。

Note: ほとんどの場合、アプリケーションで実際の設計作業を行う前に、デバイスまたはボードの BSP を作成する必要があります。これにより、独自のカスタムハードウェアまたはさまざまなリンカーオプションの設定を構成できます。また、将来のアプリケーションで使用するために BSP を保存することもできます。詳細については、 ***ModusToolbox™ user guide****Creating a BSP for Your Board を参照してください。*

5.6 パート 3: ファームウェアの書き込み

この設計では、PSoC™ 4 の CM0+ CPU を使用して、UART 通信と LED 制御の 2 つのタスクを実行します。CM0+ CPU は、UART を使用して「Hello World」メッセージをシリアルポートストリームに出力し、キットのユーザーLED を点滅させます。

Empty PSoC™ 4 スターターアプリケーションを使用している場合は、ここで提供されているプログラムコードからアプリケーションプロジェクトの *main.c* ファイルにそれぞれのソースコードをコピーできます。**Hello World** サンプルコードを使用している場合、必要なファイルはすでにアプリケーションにあります。

5.6.1 ファームウェアフロー

ここでは、アプリケーションの *main.c* ファイルのコードについて説明します。

この例では、CM0+ CPU がリセットから抜け出し、リソースの初期化を実行します。システムクロック, ピン, クロックから周辺機器への接続, およびその他のプラットフォームリソースを設定します。

クロックとシステムリソースは、BSP 初期化関数によって初期化されます。PDL 関数は、UART コンポーネントを構成して有効にするために使用されます。UART は、ターミナルエミュレータに「Hello World」メッセージを出力します。オンボードの KitProg3 は、仮想 COM ポートを作成するための USB-UART ブリッジとして機能します。ソフトウェア遅延を伴う無限 FOR ループは、ユーザーLED を定期的に切り替えるために使用されます。

アプリケーションコードは、BSP/PDL 関数を使用して目的の機能を実行することに注意してください。

- `cybsp_init()` - この BSP 関数は、システムクロックと電力レギュレーターを含みますが、これらに限定されないデバイスのシステムリソースを初期化します。
- `Cy_SCB_UART_Init()` - この PDL 関数は、UART 操作用に SCB ブロックを初期化します。設定構造 `CYBSP_UART_config` は、UART を構成するためのこの関数のパラメーターとして使用されます。この構造は、適用された構成に基づいて *design.modus* によって自動生成されます。
- `Cy_SCB_UART_Enable()` - この PDL 関数は、UART 操作の SCB ブロックを有効にします。
- `Cy_SCB_UART_PutString()` - この PDL 関数は、NULL で終了する文字列を UART TX FIFO に配置します。
- `Cy_GPIO_Inv()` - この PDL 関数は、ピン出力ロジック状態を現在の出力ロジック状態の逆に設定します。
- `Cy_SysLib_Delay()` - この PDL 関数は、指定されたミリ秒数だけ遅延します。

Code Listing 1 をアプリケーションプロジェクトの *main.c* にコピーしてください。

Code Listing 1 プログラムコード *main.c*

```
#include "cy_pdl.h"
#include "cybsp.h"


/*******************************************************************************
* Macros
********************************************************************************/
#define LED_DELAY_MS               (500u)
#define CY_ASSERT_FAILED           (0u)
```

```
/*******************************************************************************
*****
* Function Name: main
********************************************************************************
*****
* Summary:
*  System entrance point. This function performs
*  - initial setup of device
*  - configure the SCB block as UART interface
*  - prints out "Hello World" via UART interface
*  - Blinks an LED under firmware control at 1 Hz
*
* Parameters:
*  none
*
* Return:
*  int
*
*******************************************************************************
****/
int main(void)
{
    cy_rslt_t result;
    cy_stc_scb_uart_context_t CYBSP_UART_context;

    /* Initialize the device and board peripherals */
    result = cybsp_init();

    /* Board init failed. Stop program execution */
    if (result != CY_RSLT_SUCCESS)
    {
        CY_ASSERT(CY_ASSERT_FAILED);
    }

    /* Configure and enable the UART peripheral */
    Cy_SCB_UART_Init(CYBSP_UART_HW, &CYBSP_UART_config,
&CYBSP_UART_context);
    Cy_SCB_UART_Enable(CYBSP_UART_HW);

    /* Enable global interrupts */
    __enable_irq();

    /* Send a string over serial terminal */
    Cy_SCB_UART_PutString(CYBSP_UART_HW, "Hello world\r\n");

    for(;;)
    {
        /* Toggle the user LED state */
        Cy_GPIO_Inv(CYBSP_USER_LED1_PORT, CYBSP_USER_LED1_PIN);

        /* Wait for 0.5 seconds */
```

```
        Cy_SysLib_Delay(LED_DELAY_MS);
    }
}
```

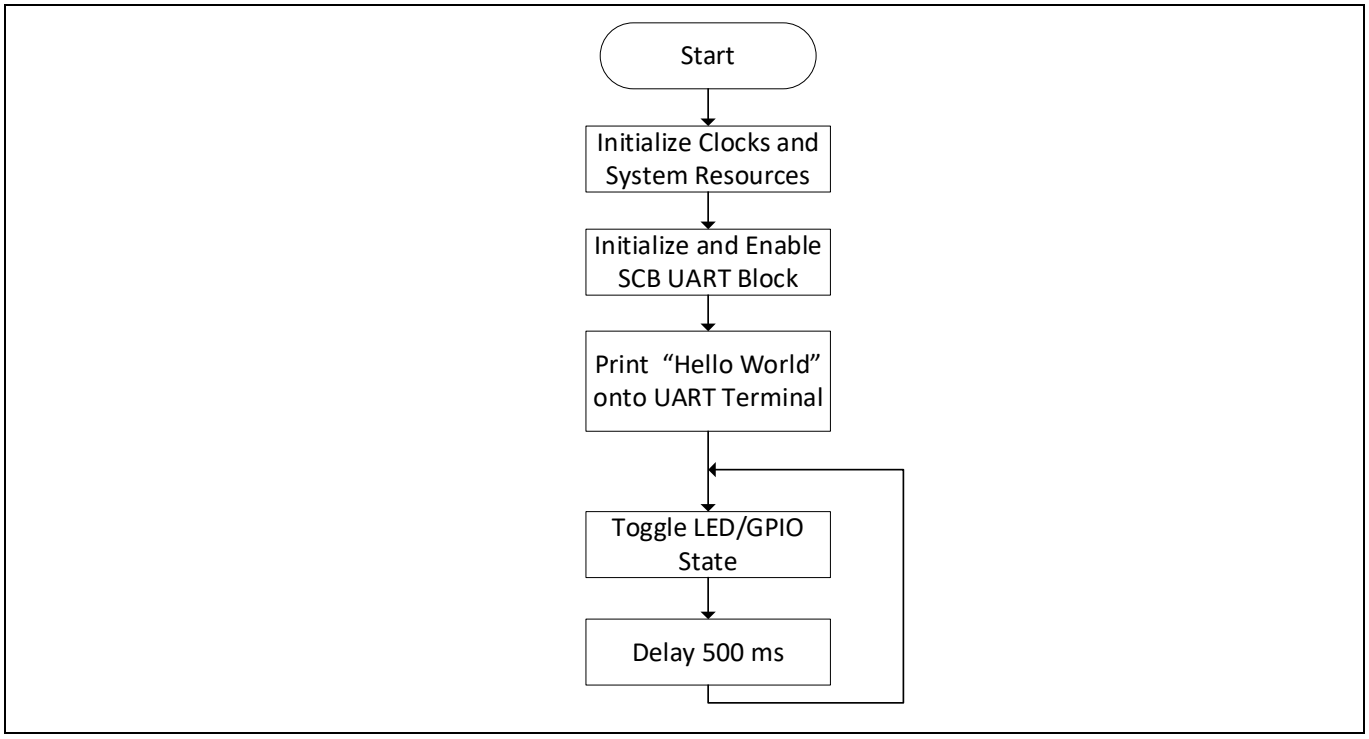

Figure 25　　ファームウェアフローチャート

これは、サンプルコードでファームウェアがどのように機能するかをまとめたものです。理解を深めるために、ソースファイルを調べてください。

5.7　　パート 4: アプリケーションの作成

ここでは、アプリケーションを構築する方法を示します。

5.7.1　　アプリケーションのビルド

a) Project Explorer ウィンドウでアプリケーションプロジェクトを選択し、Quick Panel の**<name>**グループの下にある **Build <name> Application** ショートカットをクリックしてください。これによって **Debug** build configuration が選択され、アプリケーションを構成するすべてのプロジェクトがコンパイル/リンクされます。

b) **Console** ビューには、ビルド操作の結果が一覧表示されます。

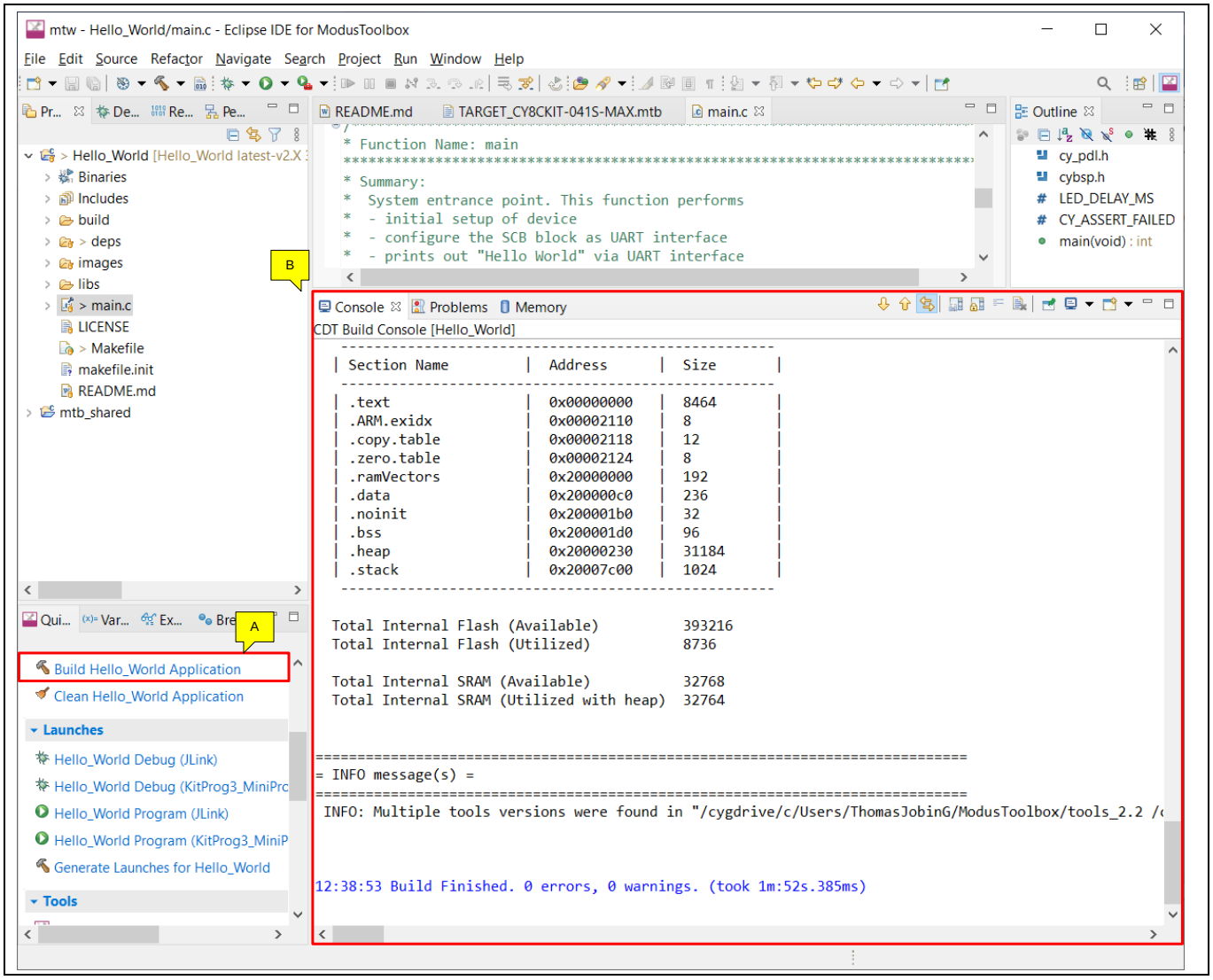

Figure 26 アプリケーションのビルド

エラーがある場合は、手順をチェックして、必要なタスクが完了していることを確認してください。

Note: *CLI を使用してもアプリケーションをビルドできます。**ModusToolbox™ user guide** の **Running ModusToolbox from the Command Line** の実行を参照してください。このドキュメントは、ModusToolbox™インストールディレクトリの/ide_2.2/docs/フォルダにあります。*

5.8 パート 5: デバイスのプログラム

ここでは、PSoC™ 4 デバイスをプログラムする方法を示します。

ModusToolbox™は、SWD プロトコルを使用して、PSoC™ 4 デバイス上のアプリケーションをプログラムおよびデバッグします。ModusToolbox™がキット上のデバイスを識別するためには、キットで KitProg3 が実行されている必要があります。一部のキットには、KitProg3 ではなく KitProg2 ファームウェアが付属します。ModusToolbox™には、KitProg ファームウェアを KitProg2 から KitProg3 に切り替えるための `fw-loader` コマンドラインツールが含まれます。詳細は、**ModusToolbox™ IDE user guide** の KitProg Firmware Loader セクションを参照してください。

自分でハードウェアを開発している場合は、ハードウェアプログラマー/デバッガが必要になる場合があります。例えば、**CY8CKIT-005 MiniProg4** です。

5.8.1 アプリケーションのプログラム

a) キットを PC に接続してください。

b) アプリケーションプロジェクトを選択し、クイックパネルの **Launches** グループの下にある **<aaplication name> Program (KitProg3_MiniProg4)** ショートカットをクリックしてください。IDE は、適切な実行構成を選択して実行します。最後のビルド以降にファイルが変更されている場合、このステップでもビルドが実行されることに注意してください。

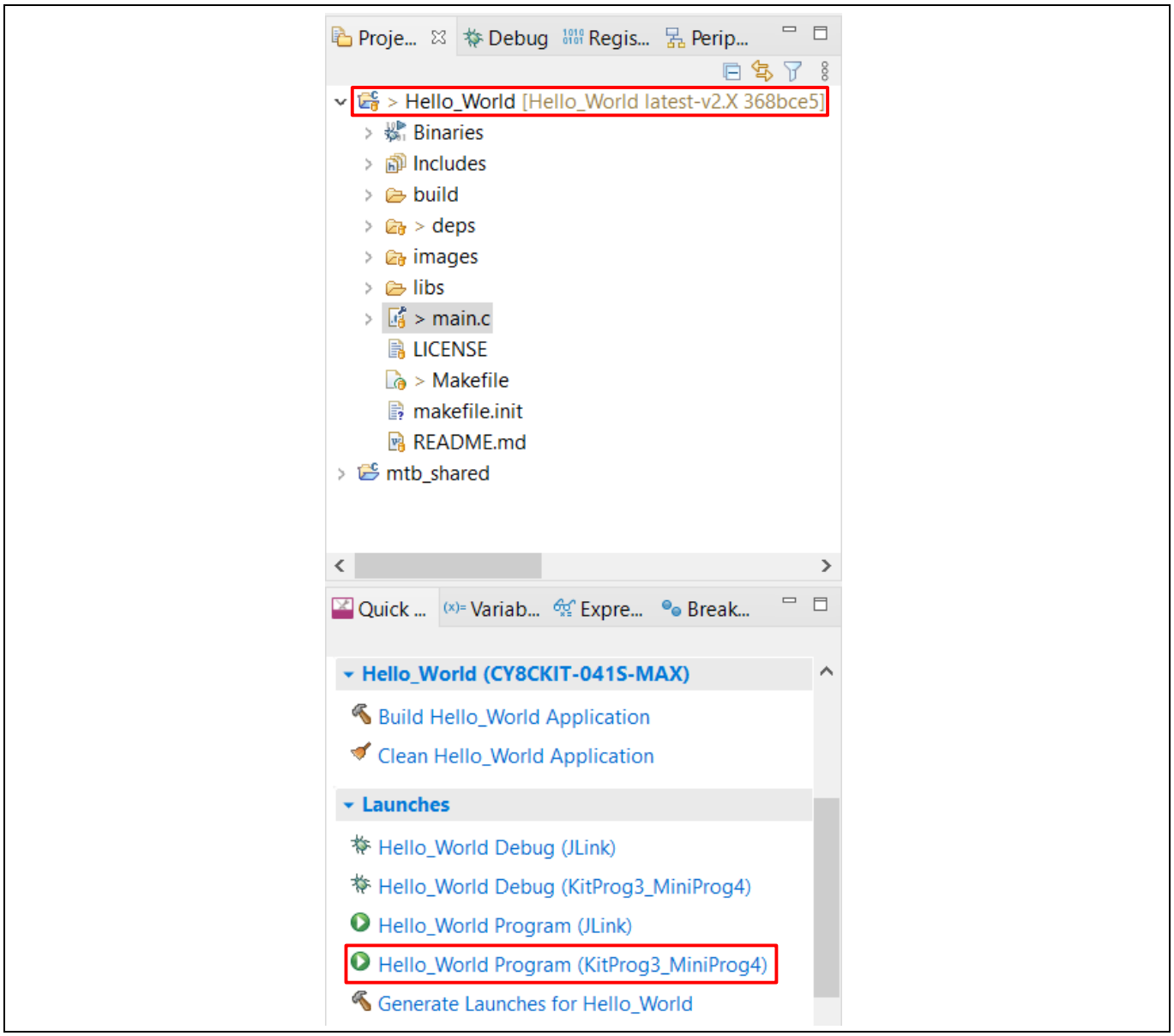

Figure 27 デバイスへのアプリケーションのプログラム

コンソールビューには、プログラミング操作の結果が一覧表示されます。

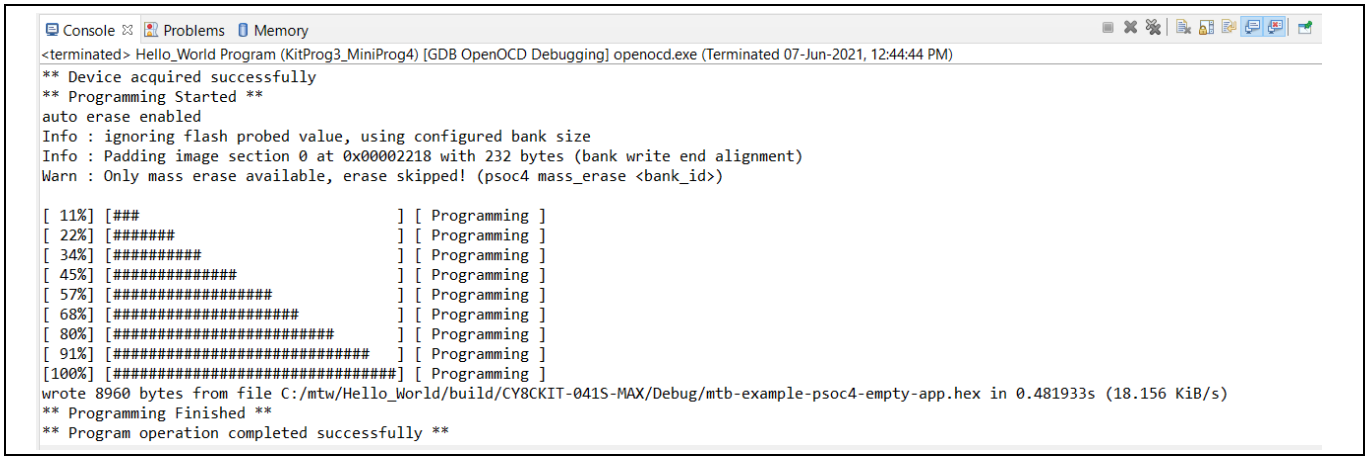

Figure 28　　コンソール - プログラミング結果

5.9　　パート 6: 設計のテスト

ここでは、設計をテストする方法について説明します。

次の手順に従って、設計の出力を観察してください。このアプリケーションノートでは、結果を表示するための UART ターミナルエミュレータとして Tera Term を使用します。任意の端末を使用して出力を表示できます。

5.9.1　　シリアルポートの選択

Figure 29 に示すように、Tera Term を起動し、USB-UART COM ポートを選択してください。COM ポート番号は異なる場合があることに注意してください。

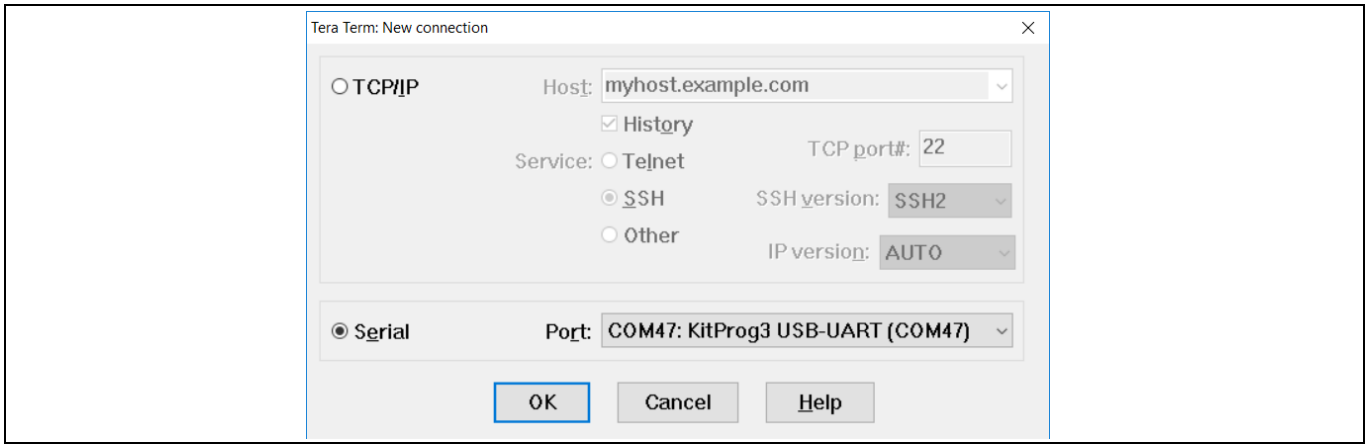

Figure 29　　Tera Term での KitProg3 COM ポートの選択

5.9.2 ボーレートの設定

Setup > Serial port へ移動してください。ボーレートを **115200** に設定してください。

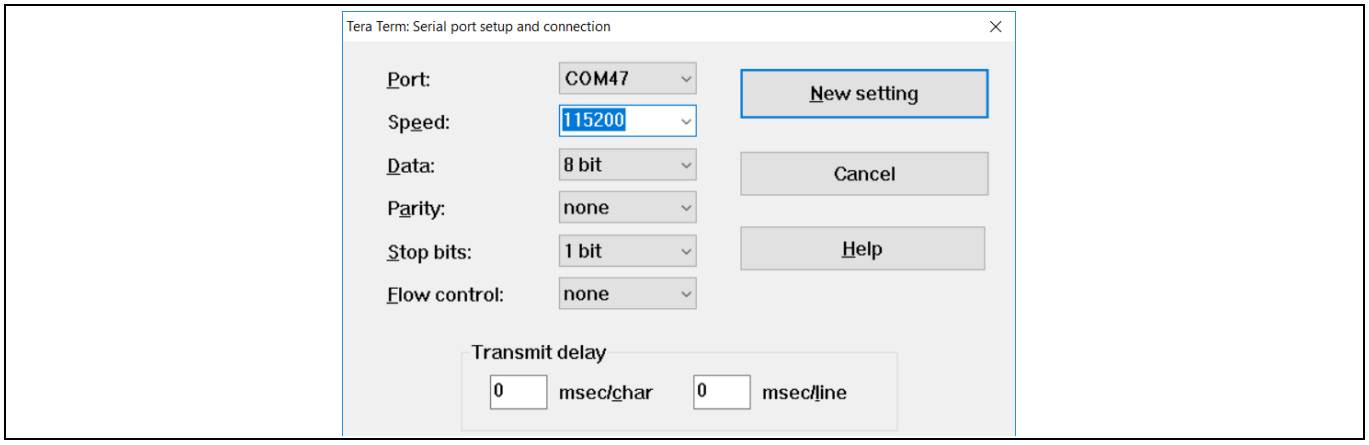

Figure 30 Tera Term でのボーレートの設定

5.9.3 デバイスのリセット

キットのリセットスイッチ (**SW1**) を押してください。「Hello World」メッセージが端末に表示されます。キットのユーザーLED が点滅し始めます。

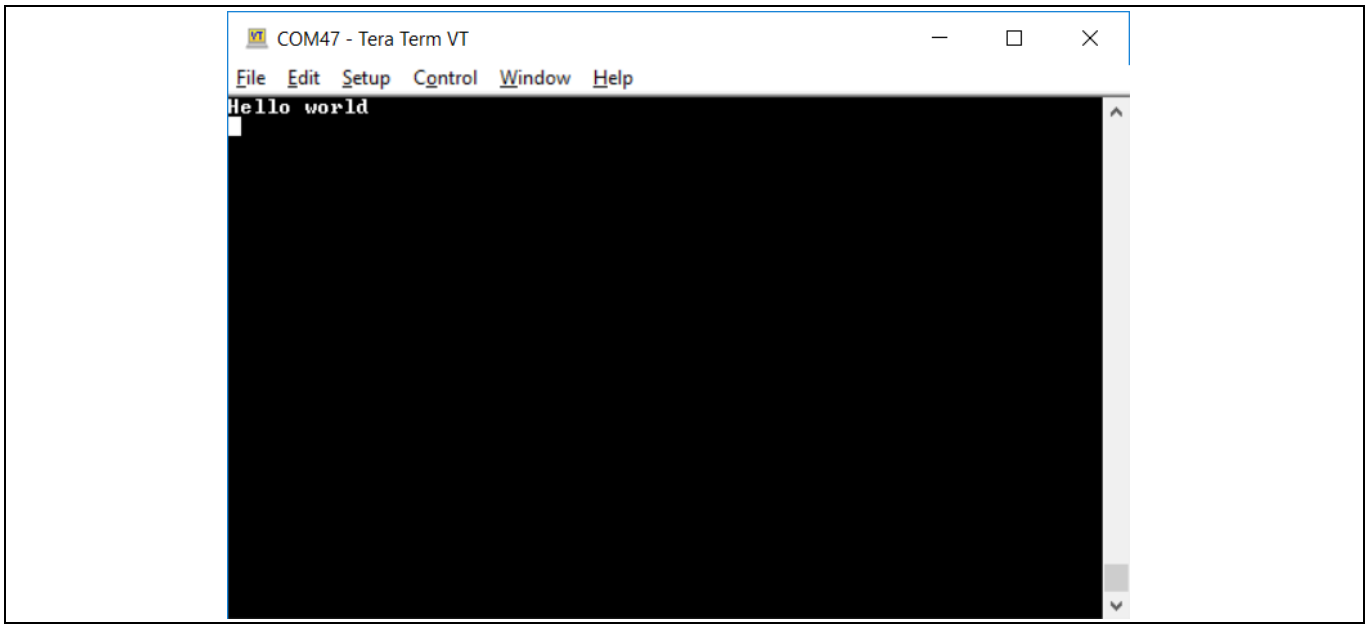

Figure 31 PSoC™ 4 からプリントされた UART メッセージ

6 PSoC™ Creator を使用するはじめての PSoC™ 4 設計

ここでは以下を説明します。

- 従来の MCU 以上のことができるように PSoC™をプログラムするデモンストレーション
- 単純な PSoC™の設計をビルドし、それを開発キットにインストールする方法の提示
- PSoC™設計技術の学習を容易にする詳細な手順と、**PSoC™ Creator** の使用方法の提示

6.1 インストールの前に

6.1.1 PSoC™ Creator をインストールしましたか?

PSoC™ Creator ホームページから PSoC™ Creator をダウンロードし、インストールします。ツールセットのインストールは時間がかかることに注意してください。詳細については PSoC™ Creator リリース ノートを参照してください。

6.1.2 開発キットまたは Prototyping Kit をお持ちですか?

この設計のテストにはプログラマを搭載する **Table 9** に示すキットの 1 つが必要です。

Table 9 PSoC™ 4 Pioneer Kit, Prototyping Kit および対応するデバイスの一覧

キット名	キットタイプ	対応するデバイスファミリ	製品番号
CY8CKIT-040	Pioneer kit	PSoC™ 4000	CY8C4014LQI-422
CY8CKIT-041	Pioneer kit	PSoC™ 4100S	CY8C4146AZI-S433
CY8CKIT-042	Pioneer kit	PSoC™ 4200	CY8C4245AXI-483
CY8CKIT-044	Pioneer kit	PSoC™ 4200M	CY8C4247AZI-M485
CY8CKIT-046	Pioneer kit	PSoC™ 4200L	CY8C4248BZI-L489
CY8CKIT-042-BLE	Pioneer kit	PSoC™ 4200 Bluetooth® LE	CY8C4247LQI-BL483
CY8CKIT-045S	Pioneer kit	PSoC™ 4500S	CY8C4548AZI-S485
CY8CKIT-043	Prototyping kit	PSoC™ 4200M	CY8C4247AZI-M485
CY8CKIT-145	Prototyping kit	PSoC™ 4000S	CY8C4045AZI-S413
CY8CKIT-147	Prototyping kit	PSoC™ 4100PS	CY8C4145LQI-PS433
CY8CKIT-149	Prototyping kit	PSoC™ 4100S Plus	CY8C4147AZI-S475

6.1.3 実行中のプロジェクトをご覧になりたいですか?

設計プロセスを実行したくない場合は、PSoC™ Creator で **Find Code Example (File > Code Example… > CE230991_My_First_Project)** を使用して、完成した PSoC™ Creator プロジェクトを取得できます。詳細については、**サンプルコード**を参照してください。その後、**ビルド**と**プログラム**のステップにジャンプできます。

6.2 設計について

Figure 32 に示すように、この設計は TCPWM コンポーネントを使用し、単純に 2 個の LED を点滅させます。TCPWM は PWM モードとして設定されます。PWM の 2 個の相補出力は LED を制御します。PWM は LED のトグルが見えるように、非常に低い周波数と 50 パーセントのデューティ比で動作します。2 個の個別の LED の代わりにデュアル カラーLED を使用すると、このプロジェクトはデュアル カラーLED の色をトグルできます。

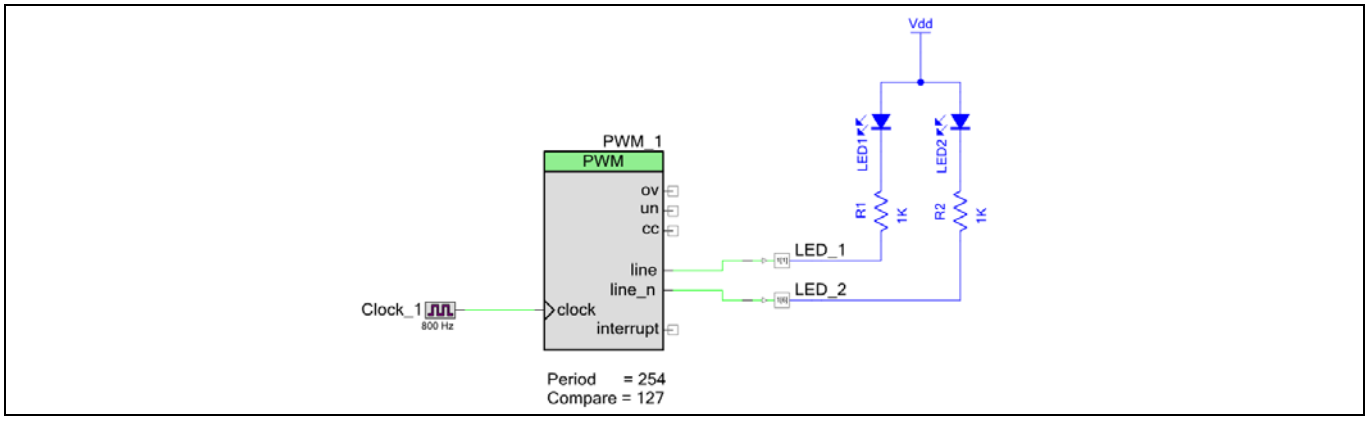

Figure 32　　はじめての PSoC™ 4 設計

6.3　　パート 1: 設計の作成

ここでは設計プロセスの概要について順に説明します。空のプロジェクトを作成し、ハードウェアとファームウェア設計入力について解説します。

1. **Figure 33** に示すように、PSoC™ Creator を起動し、**File** メニューから **New > Project** を選択してください。

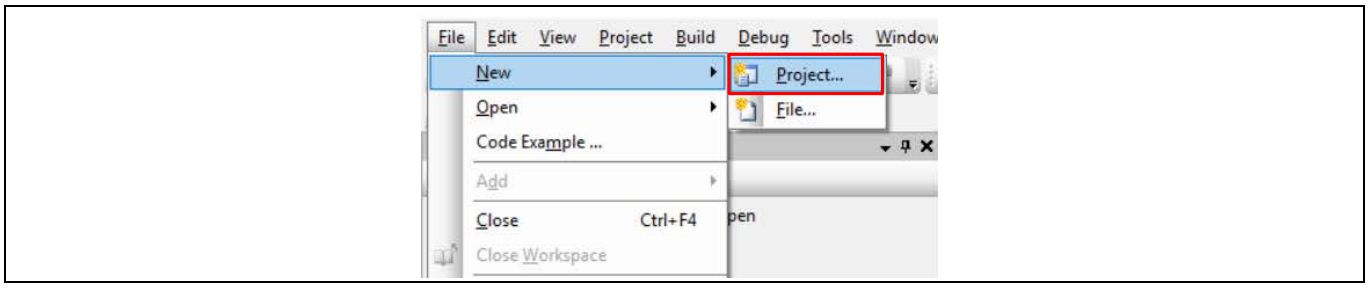

Figure 33　　新規プロジェクトの作成

2. ポップアップ ウィンドウで開発キットを選択してください。例えば、CY8CKIT-149 をお持ちの場合、**Kit: CY8CKIT-149 (PSoC™ 4100S Plus)** を選択し、**Next** をクリックしてください。メニューにお持ちの PSoC™ 4 開発キットがない場合、ウェブサイトからキット セットアップをダウンロードし、インストールしてください。

   または、ターゲット ハードウェアの代わりに、ターゲット デバイス ラジオ ボタンを選択して適切なデバイスを選んで、**Next** をクリックすることもできます。

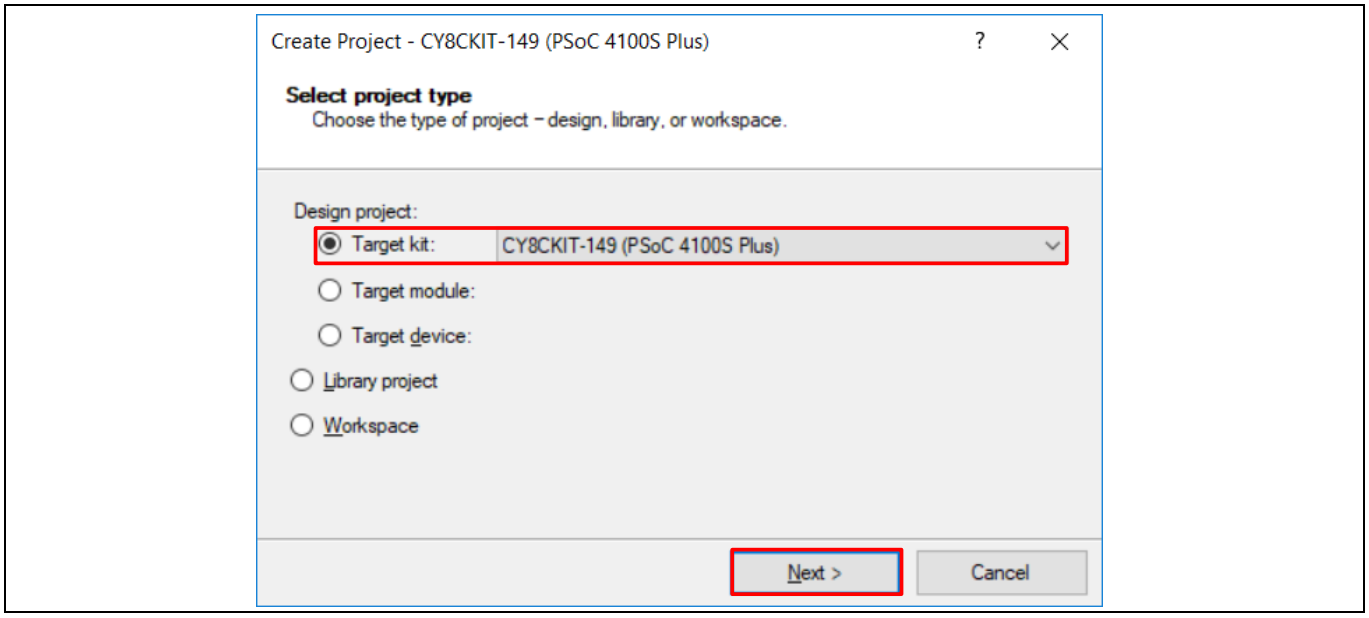

Figure 34　　空の PSoC™ 4 新規プロジェクトの作成

3. 次のウィンドウから **Empty schematic** オプションを選択し、**Next** をクリックしてください。

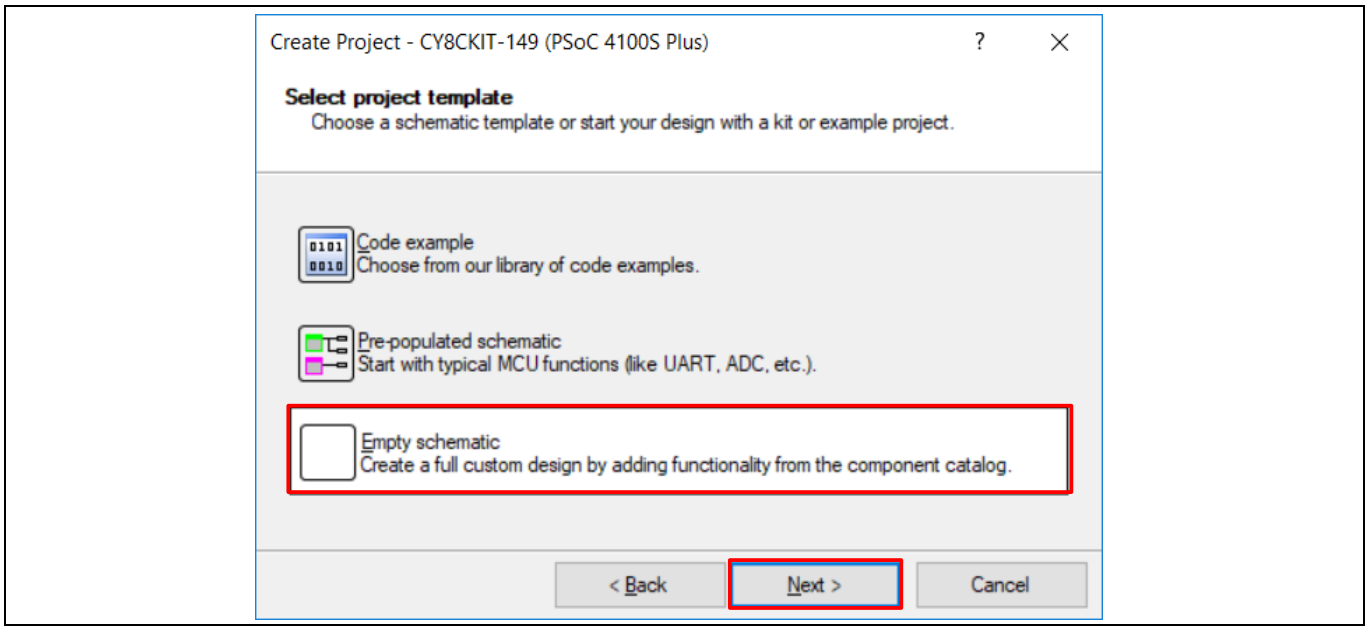

Figure 35　　Empty schematic の選択

4. **Figure 36** に示すように、プロジェクト名 (例えば「My_First_Project」) およびワークスペース名をつけてください。新規プロジェクトに適切な場所を選択して、**Finish** をクリックしてください。

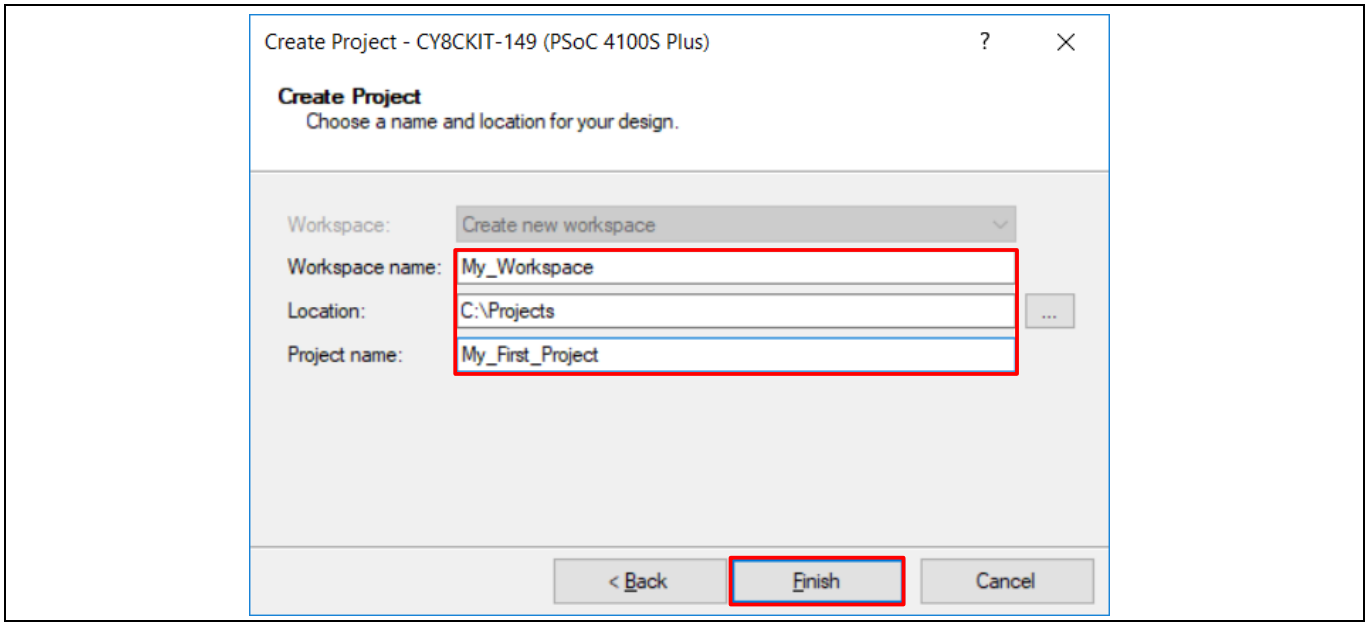

Figure 36　　プロジェクト名と場所の選択

5. **Workspace Explorer** (Figure 37 を参照) に示すように、新規プロジェクトを作成すると、ファイルのベースライン セットを含むプロジェクト フォルダが作成されます。*TopDesign.cysch* をダブルクリックし、プロジェクトの回路図ファイルを開いてください。

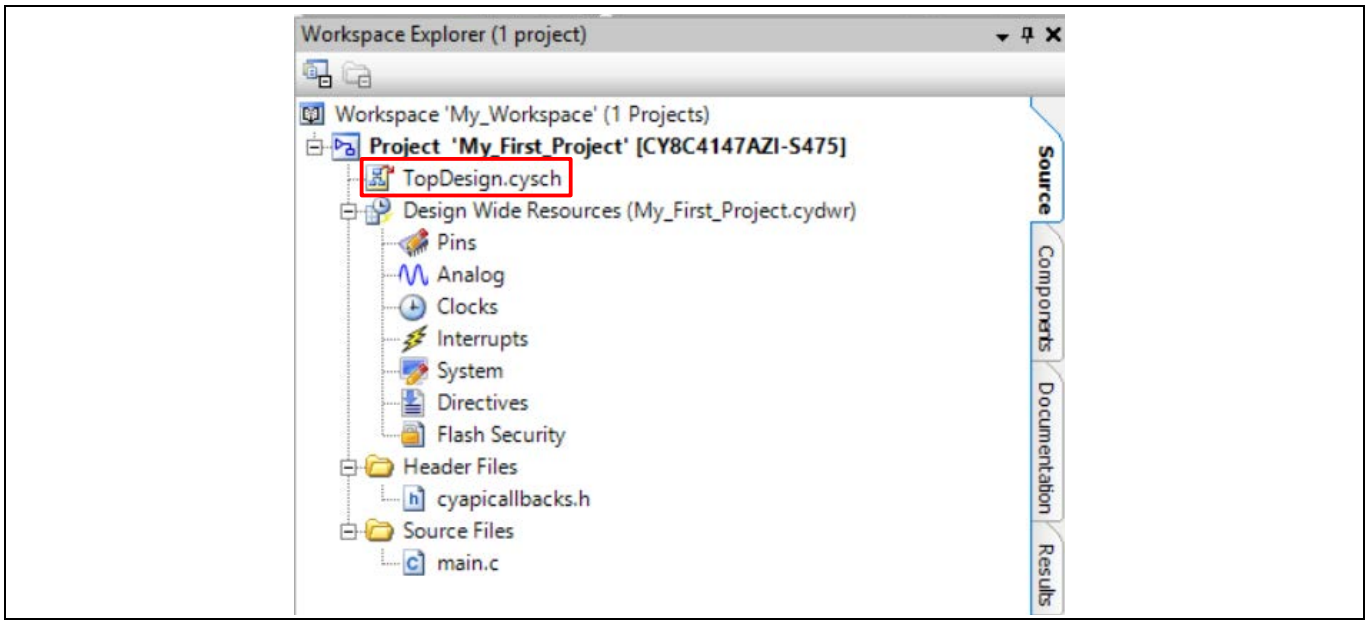

Figure 37　　TopDesign 回路図を開く

6. Figure 38 に示すように、**Component Catalog** から回路図に 1 つの **PWM (TCPWM mode)** コンポーネントをドラッグしてください。

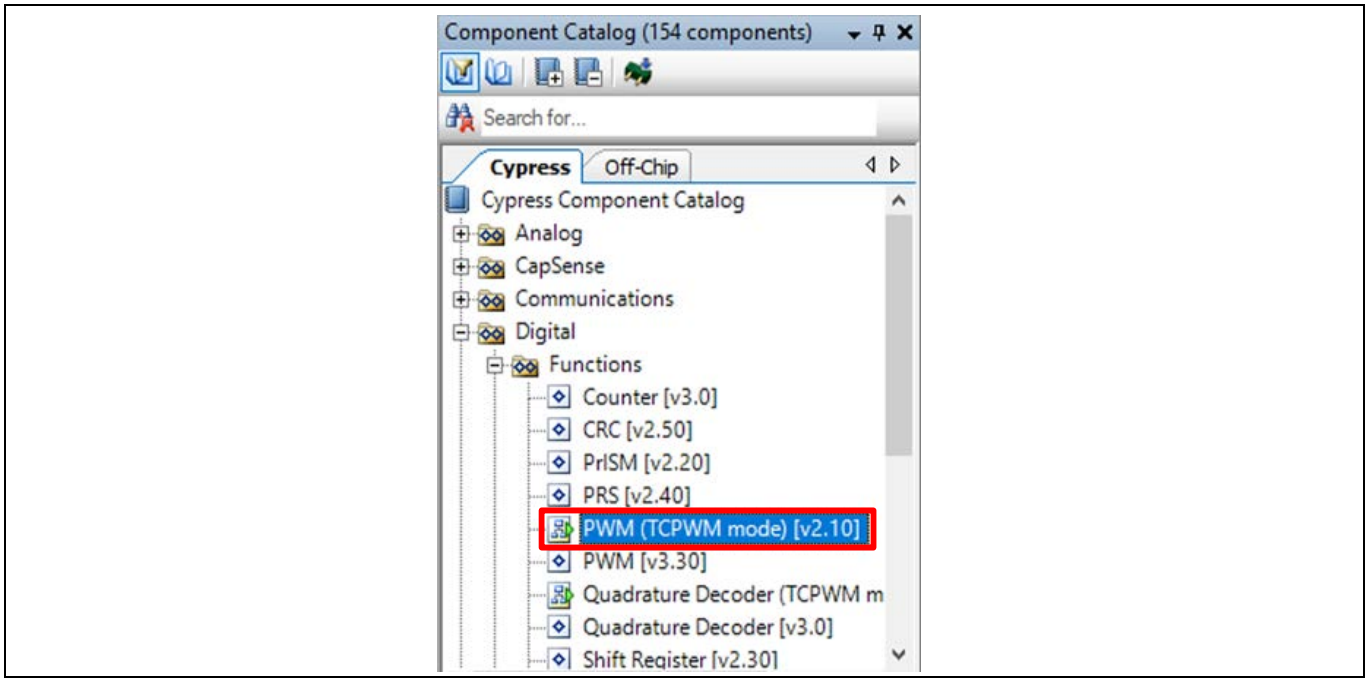

Figure 38　　PWM コンポーネントの配置

7. **Figure 39** に示すように、回路図の PWM コンポーネントをダブルクリックし、コンポーネントのプロパティを設定してください。**PWM** タブをクリックして、デューティ比が 50 パーセントの PWM 信号を生成するために、周期値を 254 に比較値を 127 にセットしてください。

   入力クロック周波数を 8 で分周するために、**Prescaler** を 8x にセットしてください。

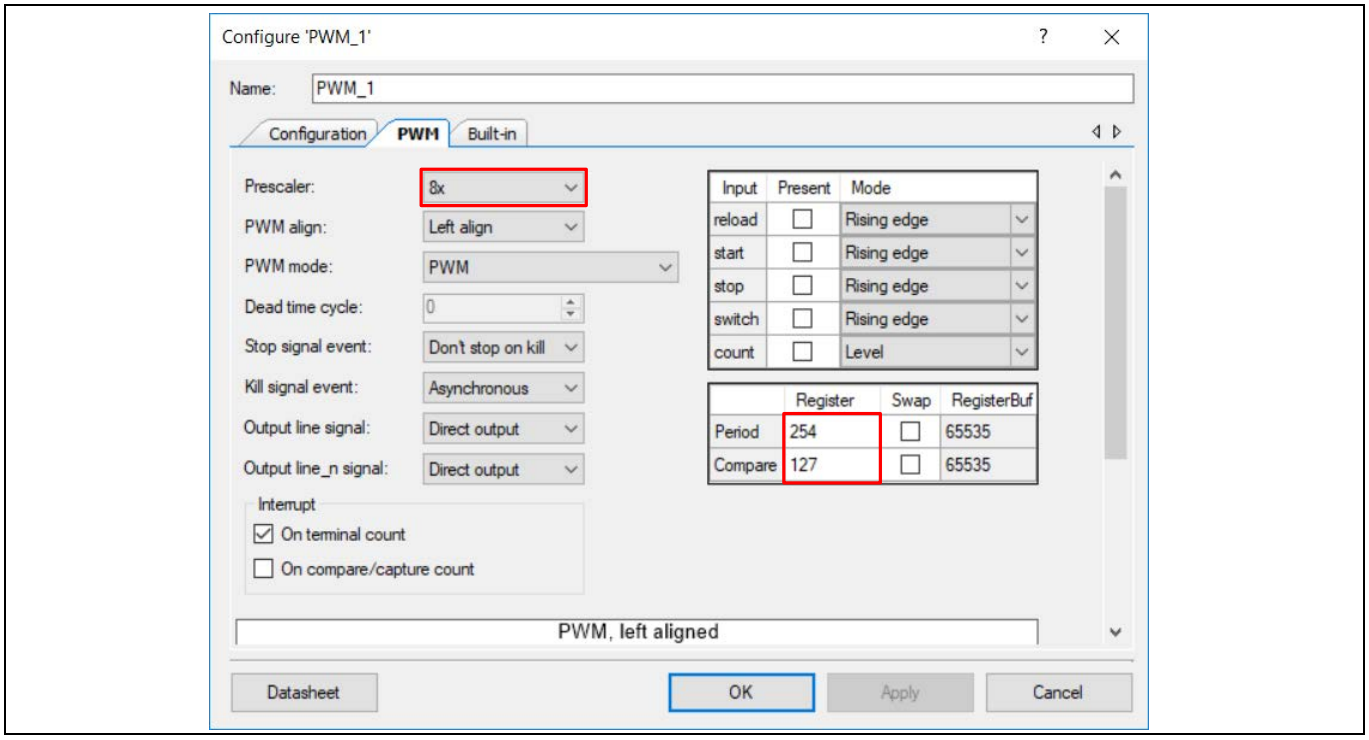

Figure 39　　PWM コンポーネントの設定

8. PWM コンポーネントが動作するために 1 つの入力クロックを必要とします。**Figure 40** と **Figure 41** に示すように、回路図に **Clock** コンポーネントをドラッグ アンド ドロップし、そのコンポーネントをダブルクリックして、**Frequency** を 800 Hz に設定してください。PWM コンポーネントの Prescaler 値設定が 8 なので、PWM の実効入力クロックはわずか 100 Hz です。したがって、PWM 周期が 254 の場合、PWM 出力周期時間は 2.54 秒になります。

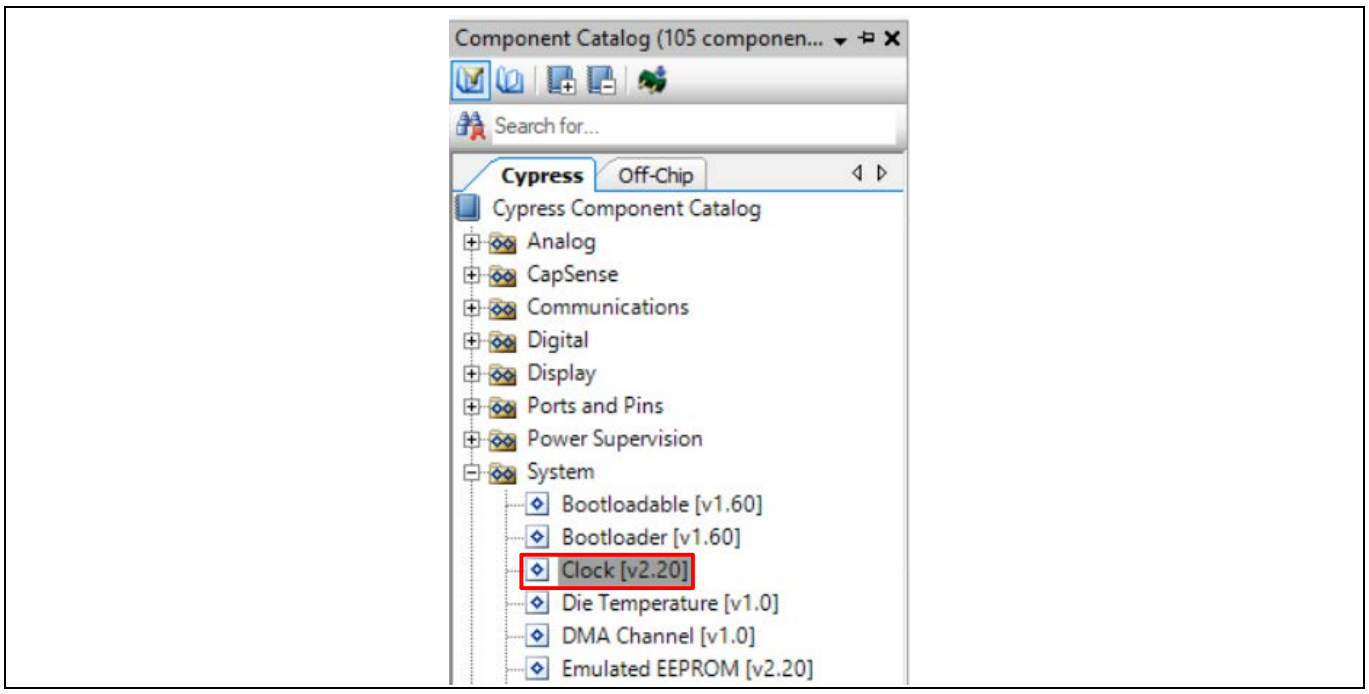

Figure 40　　クロック コンポーネントの場所

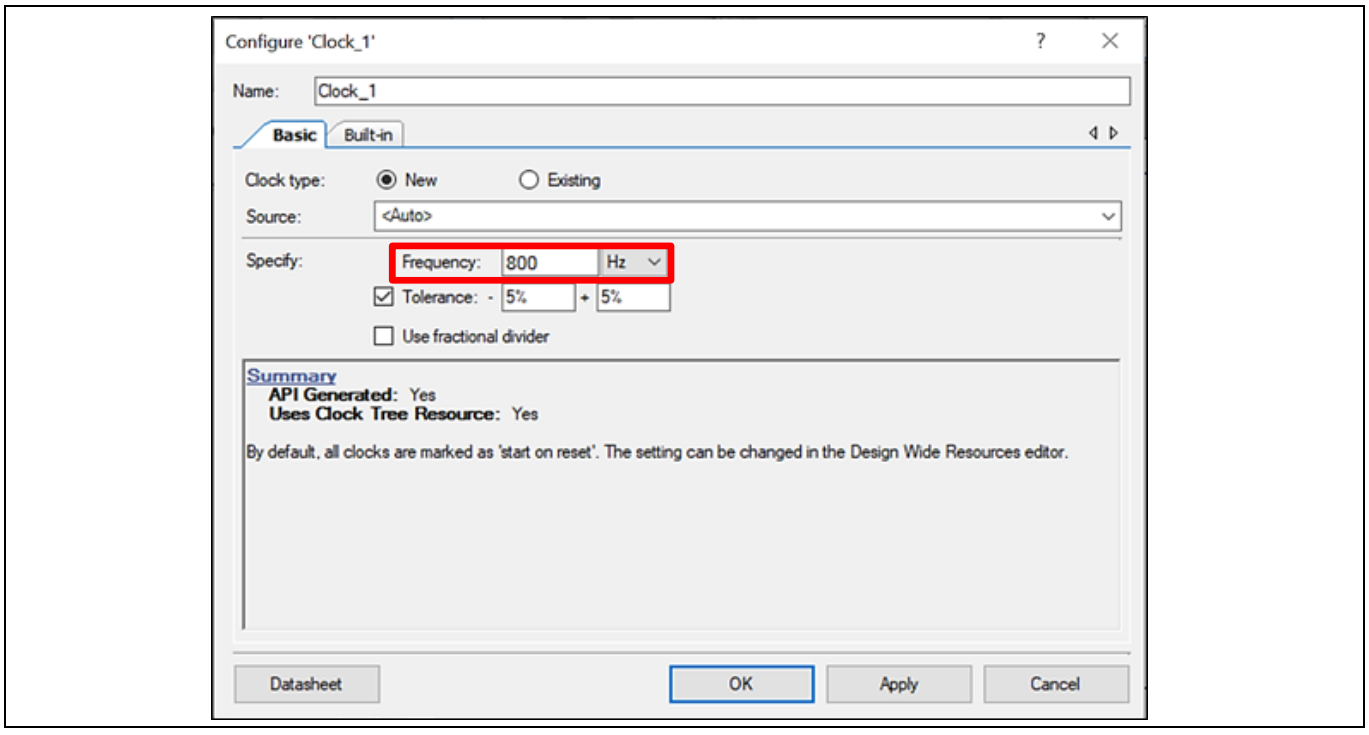

Figure 41　　クロック コンポーネントの設定

9. **Digital Output Pin** コンポーネントをドラッグ アンド ドロップしてください。**Figure 42** と **Figure 43** に示すように、その名前を LED_1 に変更してください。他のデジタル出力ピン コンポーネントを追加して、LED_2 と名付けてください。

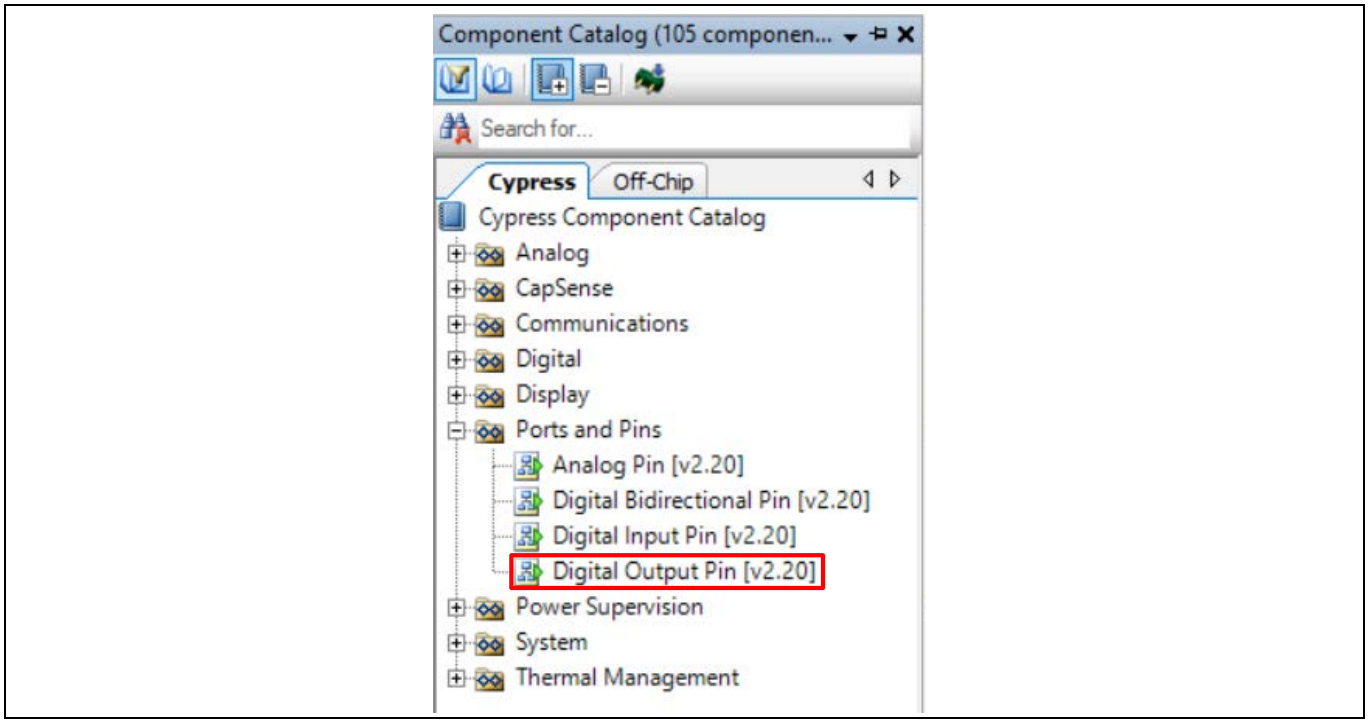

Figure 42　　デジタル出力ピン コンポーネントの場所

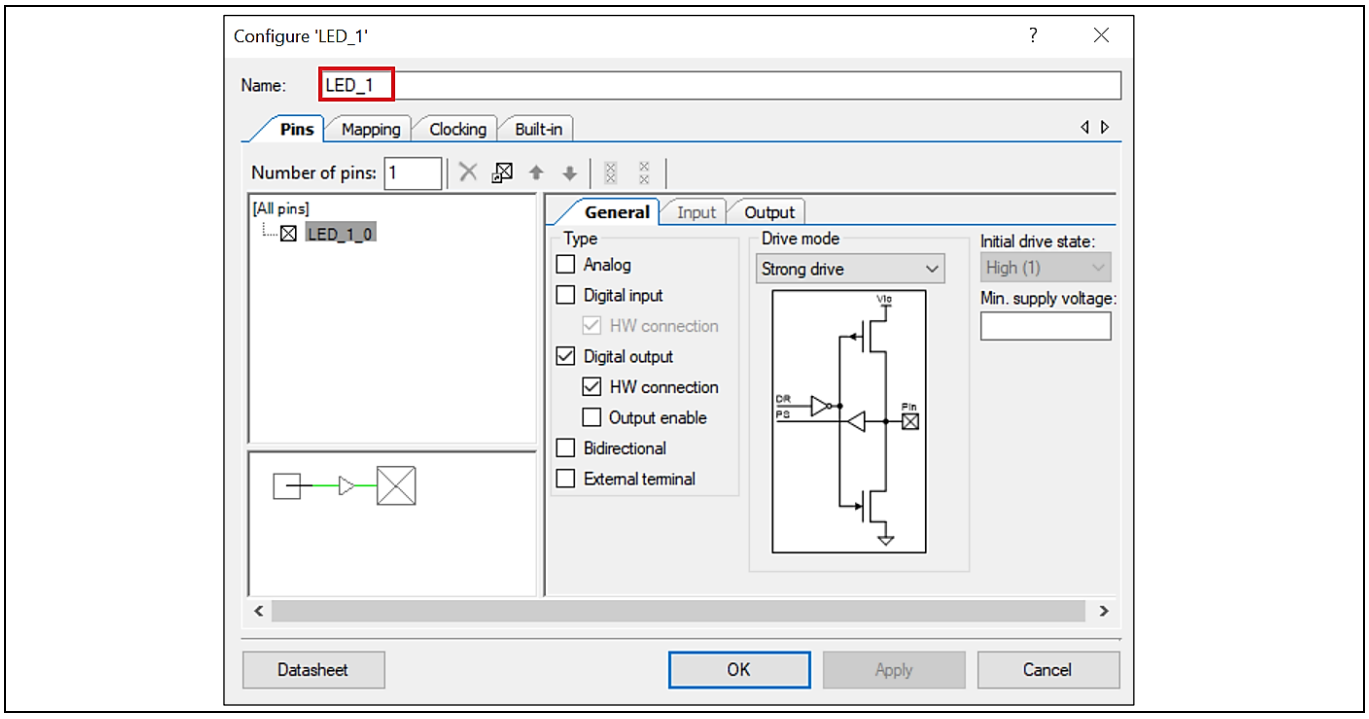

Figure 43　　ピン コンポーネントの再名付け

10. 回路図ウィンドウで、**Figure 44** に示すように、ワイヤー ツールを選択、または **W** を押してください。

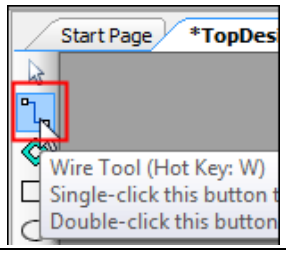

Figure 44　　ワイヤー ツールの選択

11. **Figure 45** に示すように、コンポーネントを一緒に配線してください。

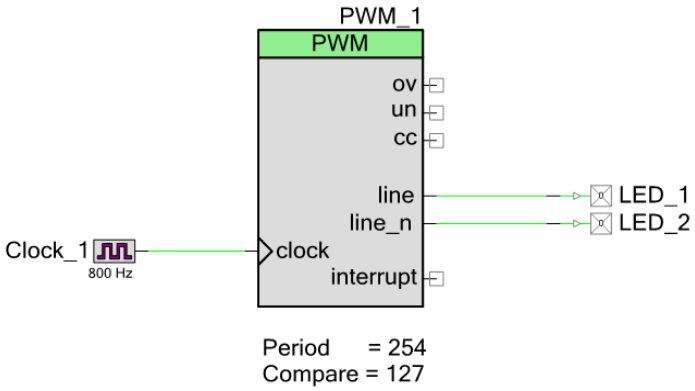

Figure 45　　回路図の配線

12. ほとんどのコンポーネントはデバイス リセットで無効になります。クロック コンポーネントは重要な例外で、デフォルトで自動的に起動されます。これらのコンポーネントを有効にするように、プロジェクトにコードを追加する必要があります。**Workspace Explorer** から *main.c* を開き、**Code Listing 2** のように、コードを `main()`関数に追加してください。

Code Listing 2　　PWM コンポーネントの有効化

```
int main(void)
 {
   /* Enable and start the PWM */
   PWM_1_Start();

   for(;;)
   {

   }
}
```

13. ビルド メニューから **Build My_First_Project** を選択してください。**Figure 46** に示すように、**Workspace Explorer** ウィンドウでは、PSoC™ Creator が PWM, クロック, およびデジタル出力ピン コンポーネントに対しソース コード ファイルを自動的に生成することに注意してください。

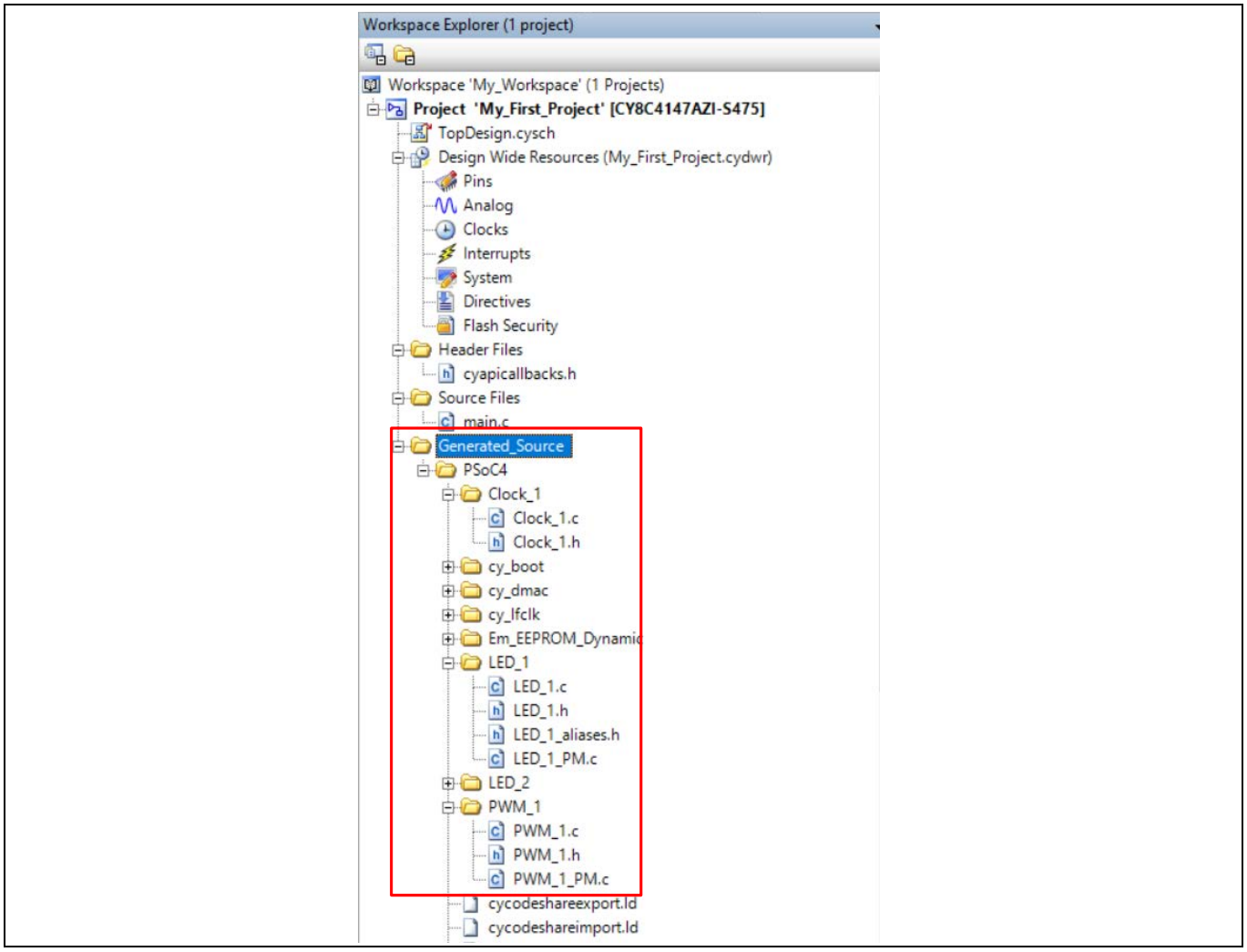

Figure 46　　生成されたソース ファイル

14. **Workspace Explorer** から *My_First_Project.cydwr* (Design-Wide Resource ファイル) ファイルを開き、**Pins** タブをクリックしてください。このタブで LED_1 と LED_2 出力用のデバイス ピンを選択できます。

Figure 47 に、**CY8CKIT-149 PSoC™ 4 prototyping kit** で LED_1 と LED_2 ピンを LED に接続するピン コンフィギュレーションを示します。別の PSoC™ 4 Pioneer Kit をお使いの場合は **Table 10** を、PSoC™ 4 prototyping kit をお使いの場合は **Table 11** を参照してください。

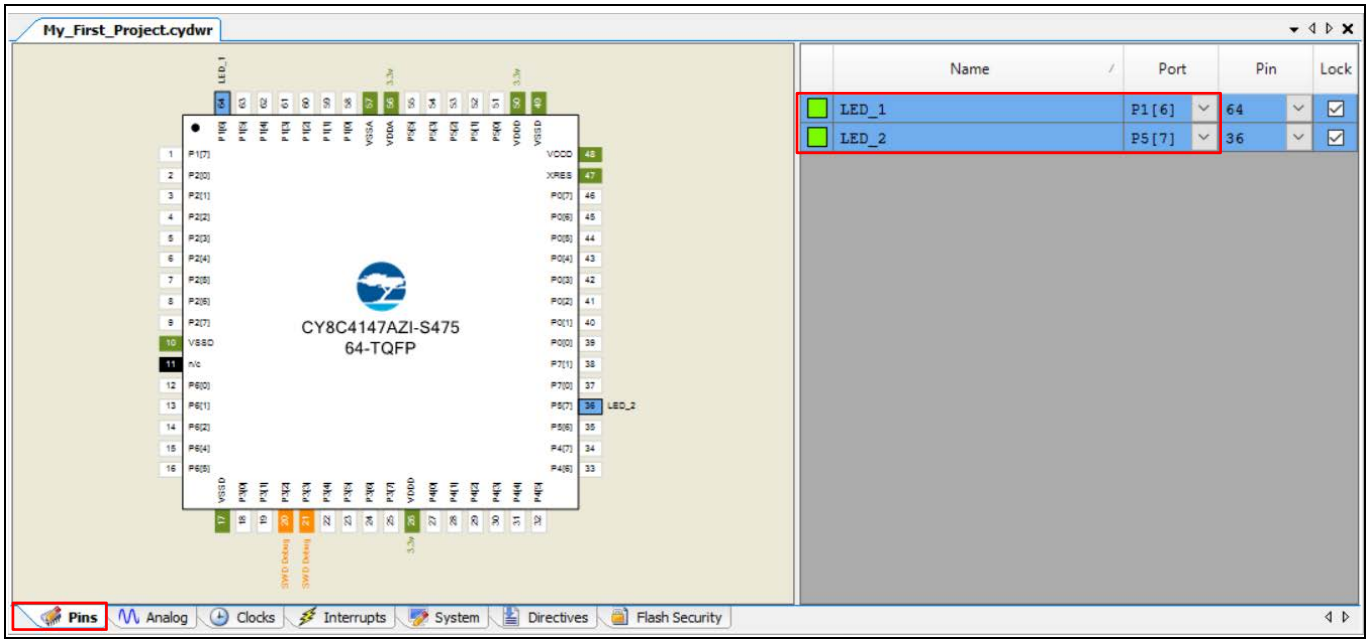

Figure 47　　ピン選択

Table 10　　Pioneer キットのピン マッピング表

機能	CY8CKIT-040 (PSoC™ 4000)	CY8CKIT-041 (PSoC™ 4100S)	CY8CKIT-042 (PSoC™ 4200)	CY8CKIT-042-BLE (PSoC™ 4200 Bluetooth® LE)	CY8CKIT-044 (PSoC™ 4200M)	CY8CKIT-046 (PSoC™ 4200L)	CY8CKIT-045S (PSoC™ 4500S)
緑色 LED (アクティブ LOW)	P1[1]	P2[6]	P0[2]	P3[6]	P2[6]	P5[3]	P0[0]
赤色 LED (アクティブ LOW)	P3[2][9]	P3[4][10]	P1[6]	P2[6]	P0[6]	P5[2]	P1[6]

Table 11　　Prototyping キットのピン マッピング表

機能	CY8CKIT-145 (PSoC™ 4000S)	CY8CKIT-149 (PSoC™ 4100S Plus)
緑色 LED - LED 1 (アクティブ LOW)	P3[4]	P1[6]
緑色 LED - LED 2 (アクティブ LOW)	P3[5]	P5[7]

Note:　　CY8CKIT-043 および CY8CKIT-147 には、それぞれ P1[6] および P0[2] に接続する LED が 1 つだけあります。外部 LED は、CY8CKIT-043 の場合はピン P0[2] に、CY8CKIT-147 の場合はピン P0[3] に接続できます。

9 PSoC™ 4000 製品には、補完的な PWM 出力用の固定ピン P1[1]および P1[6]があります。PWM 出力に他のピンは使用できません。詳細については、デバイスのデータシートを参照してください。**CY8CKIT-040** を使用している場合は、P1[1]に接続する緑色の LED を LED1 として使用できます。赤色 LED を LED2 として使用するためには、ヘッダーJ4 の P3[2]をヘッダーJ3 の P1 [6]にワイヤーで接続します。または、外部 LED も LED2 として P1[6]に接続できます。

10 上記の注記と同様に、**CY8CKIT-041** を使用している場合は、P2[6]に接続する緑色の LED を LED1 として使用し、LED2 の補完的な PWM 出力 P2[7]を使用できます。赤色 LED を LED2 として使用するためには、ヘッダーJ2 の P3[4]をヘッダーJ3 の P2[7]にワイヤーで接続します。または、外部 LED も LED2 として P2[7]に接続できます。

15. 最後に、ステップ 13 で説明したようにプロジェクトを再ビルドします。

6.4 パート 2: デバイスのプログラム

ここでは、デバイスのプログラム方法を説明します。USB ケーブルを使ってキット ボードをコンピューターに接続してください。

1. Figure 48 に示すように、PSoC™ Creator メニューから、**Debug** > **Select Debug Target** を選択してください。

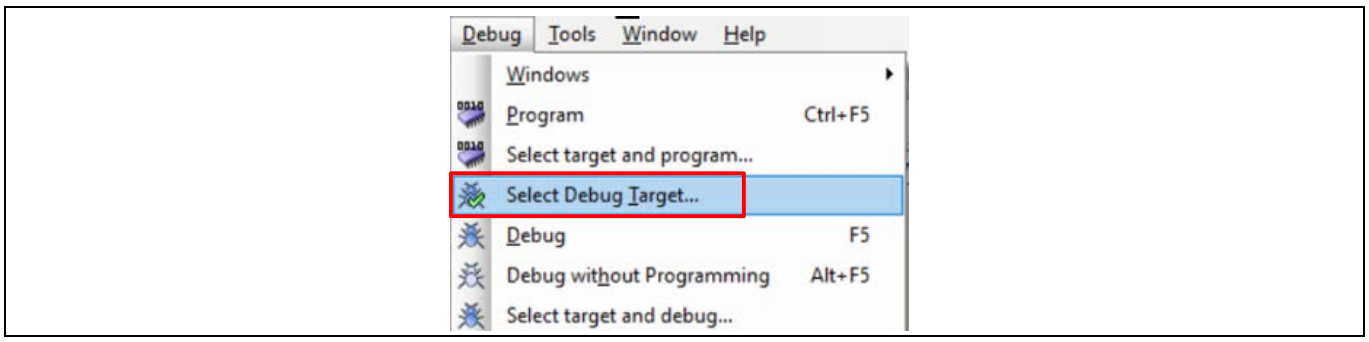

Figure 48 デバッグ ターゲットの選択

2. Figure 49 に示すように、**Select Debug Target** ダイアログボックスで、**Port Acquire** をクリックしてから、**Connect** をクリックしてください。**OK** をクリックし、ダイアログボックスを閉じてください。

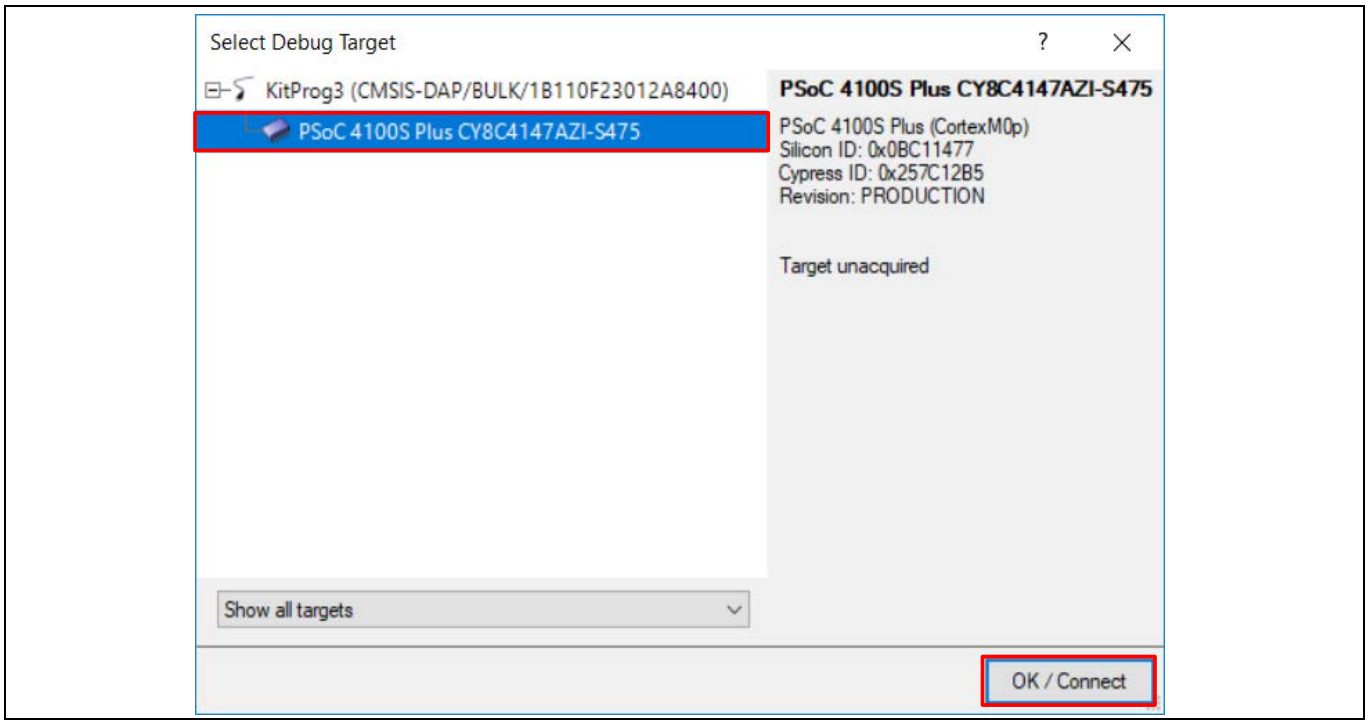

Figure 49 デバイスの接続

3. Figure 50 に示すように、**Debug** > **Program** を選択し、プロジェクトでデバイスをプログラムしてください。

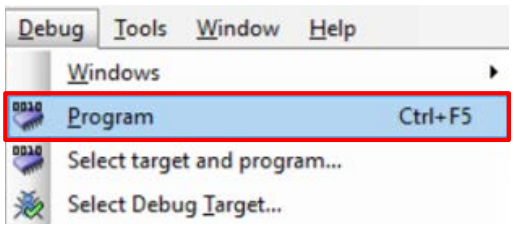

Figure 50　　デバイスのプログラミング

4. **Figure 51** に示すように、プログラミングの状態はステータスバー (ウィンドウの左下隅) に表示されます。

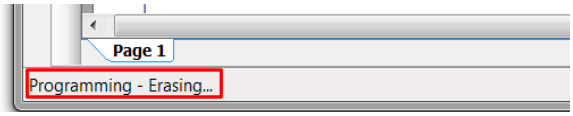

Figure 51　　プログラミングの状態

5. デバイスがプログラムされた後、LED のトグルをチェックし、プロジェクトの動作を検証してください。

7 まとめ

本アプリケーション ノートは PSoC™ 4 アーキテクチャおよび開発ツールを説明しました。PSoC™ 4 は単一チップ上に設定可能なアナログとデジタルのペリフェラル機能, メモリ, および Arm® Cortex®-M0/M0+ マイクロコントローラーを集積した真のプログラマブル組込みシステムオンチップです。内蔵された機能と低リーク電力モードにより、PSoC™ 4 は低消費電力とコスト効率が高い組込みシステムに理想的です。

また、本アプリケーション ノートは PSoC™ 4 を迅速に深く理解できるように包括的なリソースをご紹介しました。

参考資料

[1] **AN54181**: Getting started with PSoC™ 3

[2] **AN77759**: Getting started with PSoC™ 5LP

改訂履歴

Document version	Date of release	Description of changes
**	2013-09-20	これは英語版 001-79953 Rev. *B を翻訳した日本語版 001-89249 Rev. ** です。
*A	2015-04-17	これは英語版 001-79953 Rev. *I を翻訳した日本語版 001-89249 Rev. *A です。
*B	2015-11-16	これは英語版 001-79953 Rev. *K を翻訳した日本語版 001-89249 Rev. *B です。
*C	2016-02-23	これは英語版 001-79953 Rev. *M を翻訳した日本語版 001-89249 Rev. *C です。
*D	2017-04-19	更新されたロゴと著作権。
*E	2017-06-09	No change. 英語版 001-79953 Rev. *M を翻訳した日本語版です。
*F	2017-11-24	これは英語版 001-79953 Rev. *O を翻訳した日本語版 001-89249 Rev. *F です。
*G	2019-01-24	これは英語版 001-79953 Rev. *R を翻訳した日本語版 001-89249 Rev. *G です。
*H	2019-10-21	これは英語版 001-79953 Rev. *S を翻訳した日本語版 001-89249 Rev. *H です。
*I	2021-01-25	これは英語版 001-79953 Rev. *T を翻訳した日本語版 001-89249 Rev. *I です。
*J	2021-12-23	これは英語版 001-79953 Rev. *U を翻訳した日本語版 001-89249 Rev. *J です。

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

Edition 2021-12-23
Published by
Infineon Technologies AG
81726 Munich, Germany

Do you have a question about this document?
Go to www.cypress.com/support

Document reference
001-89249 Rev. *J

重要事項
本文書に記載された情報は、いかなる場合も、条件または特性の保証とみなされるものではありません（「品質の保証」）。本文に記された一切の事例、手引き、もしくは一般的価値、および／または本製品の用途に関する一切の情報に関し、インフィニオンテクノロジーズ（以下、「インフィニオン」）はここに、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。

さらに、本文書に記載された一切の情報は、お客様の用途におけるお客様の製品およびインフィニオン製品の一切の使用に関し、本文書に記載された義務ならびに一切の関連する法的要件、規範、および基準をお客様が遵守することを条件としています。

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。

本製品、技術、納品条件、および価格についての詳しい情報は、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（**www.infineon.com**）。

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。