ModusToolbox™ソフトウェアを使用する EZ-PD™ PMG1 MCU 入門

本書について

適用範囲と目的

本アプリケーション ノートでは、EZ-PD™ PMG1 (第 1 世代 パワー デリバリー マイクロコントローラー) ファミリの USB-C パワー デリバリー (PD) を備えた高電圧マイクロコントローラー (MCU) の機能をご紹介します。また、ModusToolbox™ソフトウェアの Eclipse IDE を使用して初めての PMG1 MCU プロジェクトを作成する方法を示します。

対象者

本書は、ModusToolbox™ソフトウェアで EZ-PD™ PMG1 MCU ファミリを使用する開発者を主たる対象とします。

関連製品ファミリ

すべての EZ-PD™ PMG1 MCU デバイス

目次

本書について 1
目次 2
1 はじめに 3
1.1 PMG1 MCU ファミリの概要および比較 3
1.2 PMG1-S0 MCU 5
1.3 EZ-PD™ PMG1-S1 MCU 6
1.4 EZ-PD™ PMG1-S2 MCU 7
1.5 EZ-PD™ PMG1-S3 MCU 8
1.6 ソフトウェア環境: ModusToolbox™ソフトウェア 9
2 開発エコシステム 10
2.1 PMG1 MCU リソース 10
2.2 ModusToolbox™ソフトウェア 10
2.3 PMG1 MCU ソフトウェア リソース 11
2.3.1 コンフィギュレータ 11
2.3.2 PMG1 MCU アプリケーション開発 11
2.4 ModusToolbox™ヘルプ 12
2.5 他の IDE のサポート 13
2.6 Eclipse IDE を使用したプログラミングおよびデバッグ 13
2.7 mtb プログラマを使用したプログラミング 13
2.8 PMG1 MCU キット 15
3 ModusToolbox™ソフトウェア用 Eclipse IDE を使用した PMG1 MCU アプリケーションの作成 16
3.1 事前準備 16
3.2 新しいアプリケーションの作成 16
3.3 設計設定の閲覧と変更 18
3.4 デバイス コンフィギュレータの開始 19
3.5 ファームウェアの記述 21
3.5.1 ファームウェア フロー 21
3.6 アプリケーションのビルドとプログラム 22
3.7 設計のテスト 26
3.8 アプリケーションのデバッグ 28
3.8.1 10 ピン SWD ヘッダを介したデバッグ 30
3.9 ミドルウェアの追加 30
4 USB-PD シンクの例 32
5 PMG1 MCU 間でのプロジェクトの移植 33
6 まとめ 36
頭字語と略語 37
用語集 38
改訂履歴 39
免責事項 40

1 はじめに

1.1 PMG1 MCU ファミリの概要および比較

EZ-PD™ PMG1 (第 1 世代パワーデリバリー マイクロコントローラー) は、高電圧の USB パワー デリバリー (PD) マイクロコントローラー (MCU) のファミリです。これらのチップには、Arm® Cortex®-M0/M0+ CPU, USB PD コントローラー, およびアナログとデジタル ペリフェラルが搭載されます。PMG1 MCU は、高電圧 USB PD ポートとの間で電力を供給/消費し、マイクロコントローラーを活用して追加の制御機能を提供する組込みシステムを対象とします。

Table 1 に、PMG1 MCU ファミリのさまざまな MCU の機能の比較を示します。この表を使用して、アプリケーションに適した PMG1 MCU を選択できます。

Table 1 EZ-PD™ PMG1 MCU ファミリのさまざまな MCU の機能の比較

サブシステムまたは範囲	項目	PMG1-S0	PMG1-S1	PMG1-S2	PMG1-S3
CPU およびメモリ サブシステム	コア	Arm® Cortex®-M0	Arm® Cortex®-M0	Arm® Cortex®-M0	Arm® Cortex®-M0+
	最大周波数 (MHz)	48	48	48	48
	フラッシュ (KB)	64	128	128	256
	SRAM (KB)	8	12	8	32
パワーデリバリー	パワー デリバリー ポート	1	1	1	48QFN の場合は 1 ポート 97BGA の場合は 2 ポート
	ロール	DRP	DRP	DRP	DRP
	MOSFET ゲート ドライバ	2x PFET	2x PFET	2x NFET	柔軟な 2x NFET
	フォールト保護	V_{BUS} OVP, UVP および OCP。SCP (ソース設定のみ)	V_{BUS} OVP, UVP, および OCP。SCP および RCP (ソース設定のみ)	V_{BUS} OVP, UVP, および OCP	V_{BUS} OVP, UVP, および OCP。SCP および RCP (ソース設定のみ)
USB	Billboard クラスをサポートする統合されたフルスピード USB 2.0 デバイス	無	無	有	有
電圧範囲	電源電圧 (V)	V_{DDD} (2.7〜5.5) V_{BUS} (4〜21.5)	V_{SYS} (2.75〜5.5) V_{BUS} (4〜21.5)	V_{SYS} (2.7〜5.5) V_{BUS} (4〜21.5)	V_{SYS} (2.8〜5.5) V_{BUS} (4〜28)
	I/O (V)	1.71〜5.5	1.71〜5.5	1.71〜5.5	1.71〜5.5

サブシステムまたは範囲	項目	PMG1-S0	PMG1-S1	PMG1-S2	PMG1-S3
デジタル	SCB (I2C/UART/SPI に設定可能)	2	4	4	48QFN の場合は 7 (そのうち 5 つは SPI および UART に設定可能) 97BGA の場合は 8
	TCPWM ブロック (タイマー, カウンター, パルス幅変調器に設定可能)	4	2	4	48QFN の場合は 7 97BGA の場合は 8
	ハードウェア認証ブロック (暗号)	無	無	有 (AES-128/192/256, SHA1, SHA2-224, SHA2-256, PRNG, CRC)	有 (AES-128, SHA2-256, TRNG, ベクトル ユニット)
アナログ	ADC	2x 8 ビット SAR	1x 8 ビット SAR	2x 8 ビット SAR	2x 8 ビット SAR 1x 12 ビット SAR
	内蔵温度センサー	有	有	有	有
ダイレクトメモリ アクセス (DMA)	DMA	無	無	無	有
GPIO	I/O の最大数	12 (10+2 Fail-Safe)	17 (15+2 Fail-Safe)	20 (18+2 Fail-Safe)	48QFN の場合は 26 (24+2 Fail-Safe) 97BGA の場合は 50 (48+2 Fail-Safe)
充電規格	充電ソース	BC 1.2, AC, AFC, および QC 3.0	BC1.2, AC	BC1.2, AC	BC 1.2, AC, AFC, および QC 3.0
	充電シンク	BC 1.2, AC, および QC 2.0	BC1.2, AC	BC1.2, AC	BC1.2, AC, および QC 2.0
ESD 保護	ESD 保護	有 (人体モデルおよびデバイス帯電モデル)	有 (人体モデルおよびデバイス帯電モデル)	有 (人体モデルおよびデバイス帯電モデル)	有 (人体モデルおよびデバイス帯電モデル)
パッケージ	パッケージ オプション	24QFN (4x4mm, 0.5mm ピッチ)	40QFN (6x6mm, 0.5mm ピッチ)	40QFN (6x6mm, 0.5mm ピッチ) 42CSP (2.63x3.18mm, 0.4mm ピッチ)	48QFN (6x6mm, 0.5mm ピッチ) 97BGA (6x6mm, 0.5mm ピッチと 0.65mm ピッチ)

1.2 PMG1-S0 MCU

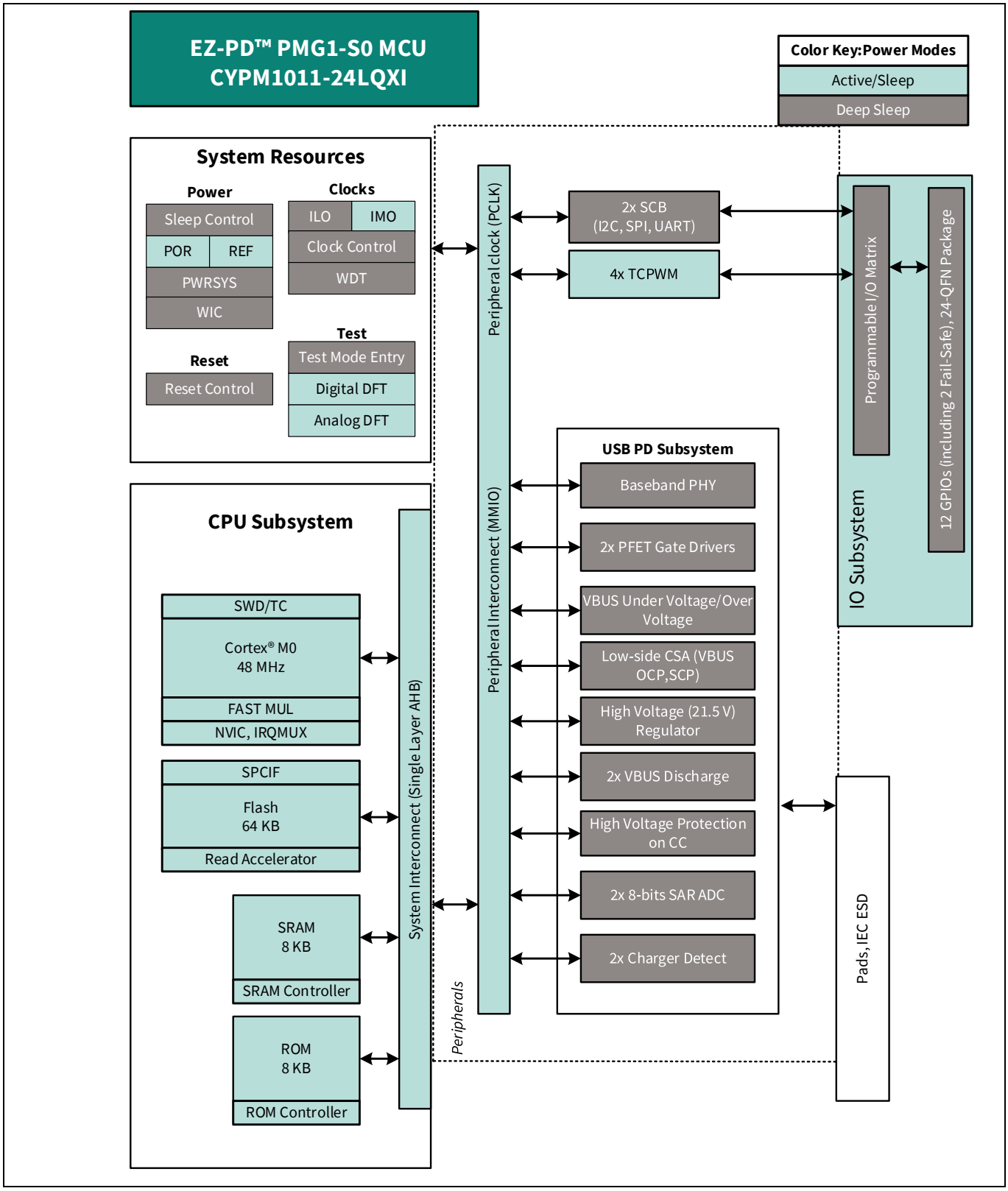


Figure 1 PMG1-S0 MCU ブロック図

1.3 EZ-PD™ PMG1-S1 MCU

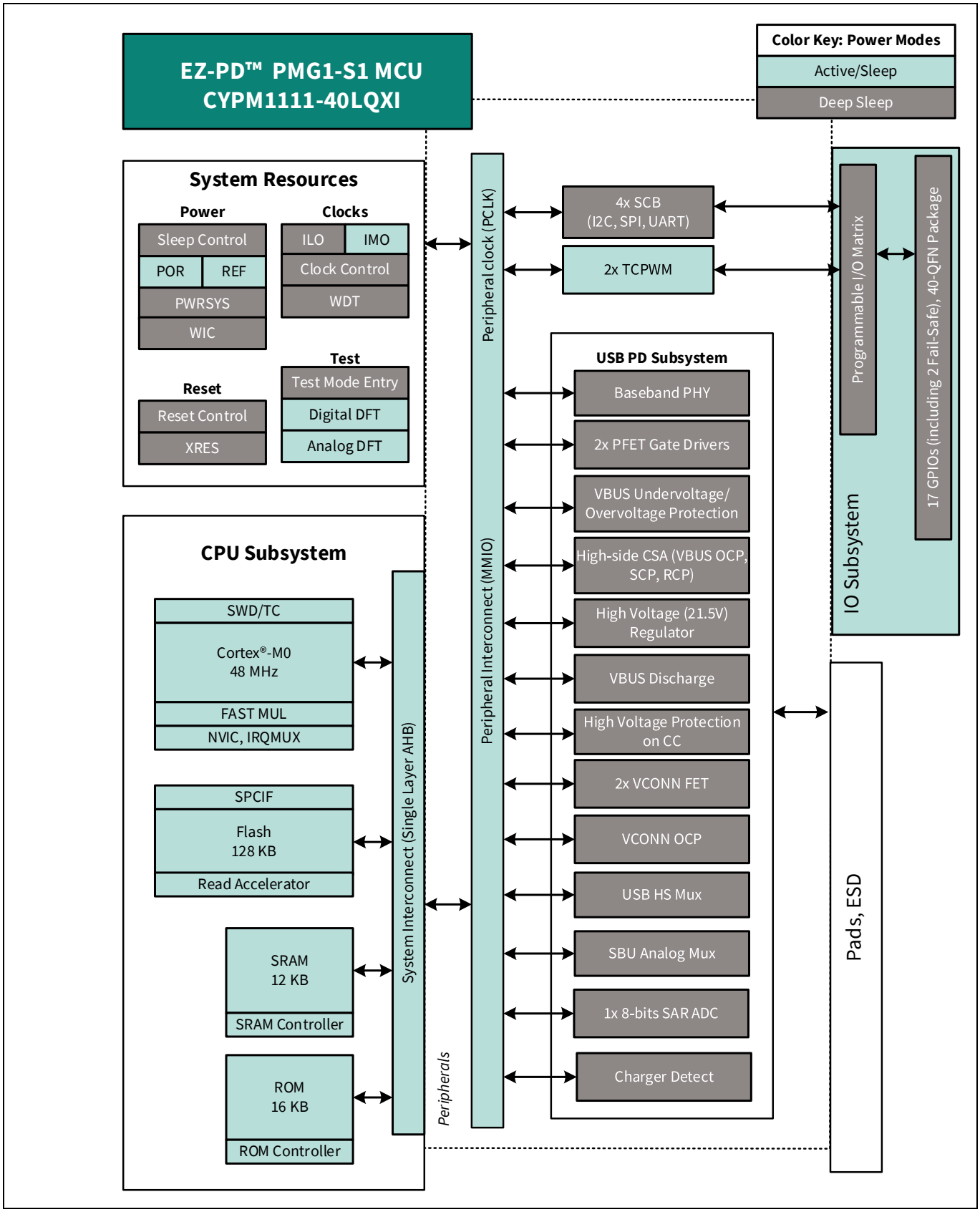


Figure 2 PMG1-S1 MCU ブロック図

1.4 EZ-PD™ PMG1-S2 MCU

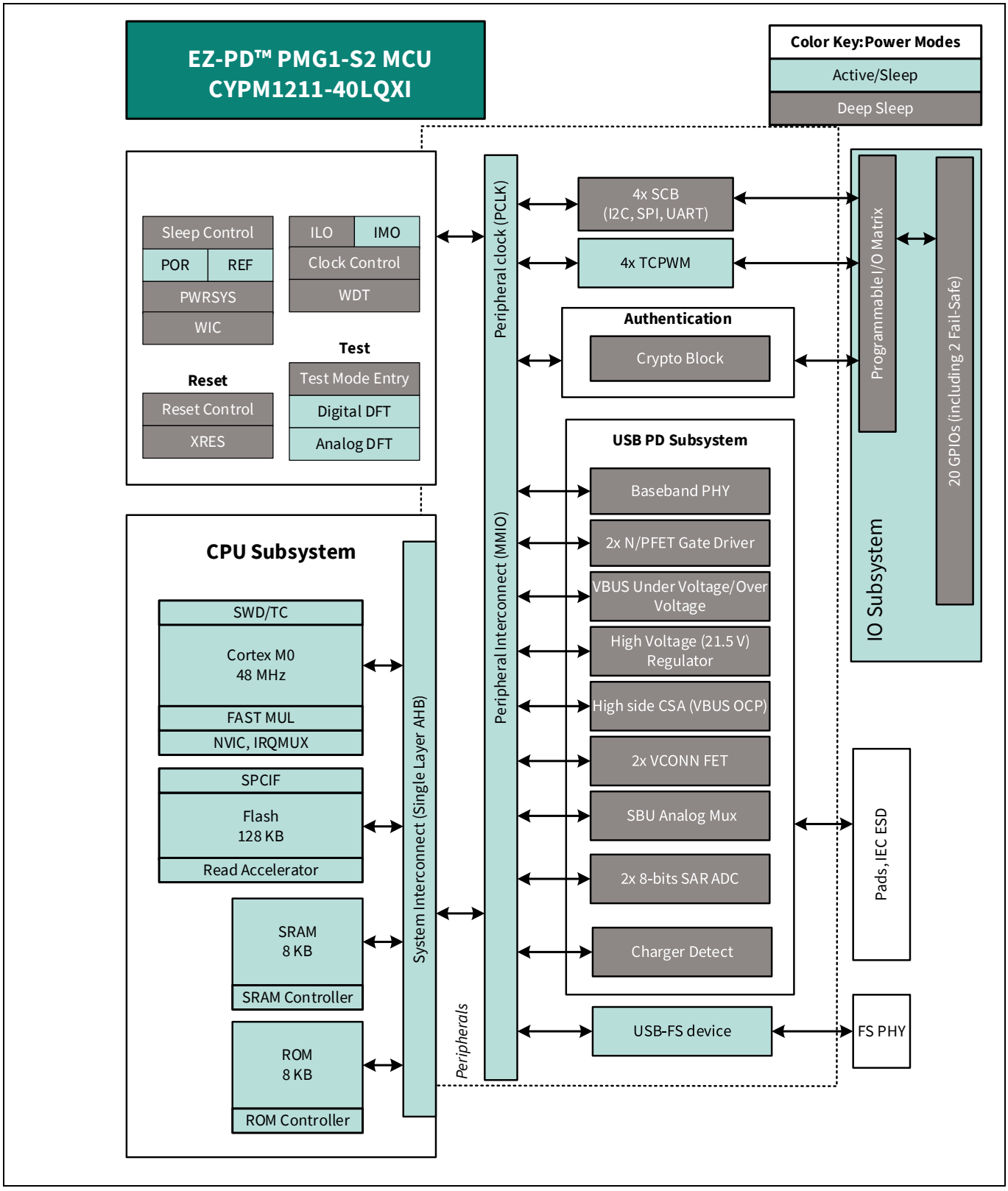


Figure 3 PMG1-S2 MCU ブロック図

1.5 EZ-PD™ PMG1-S3 MCU

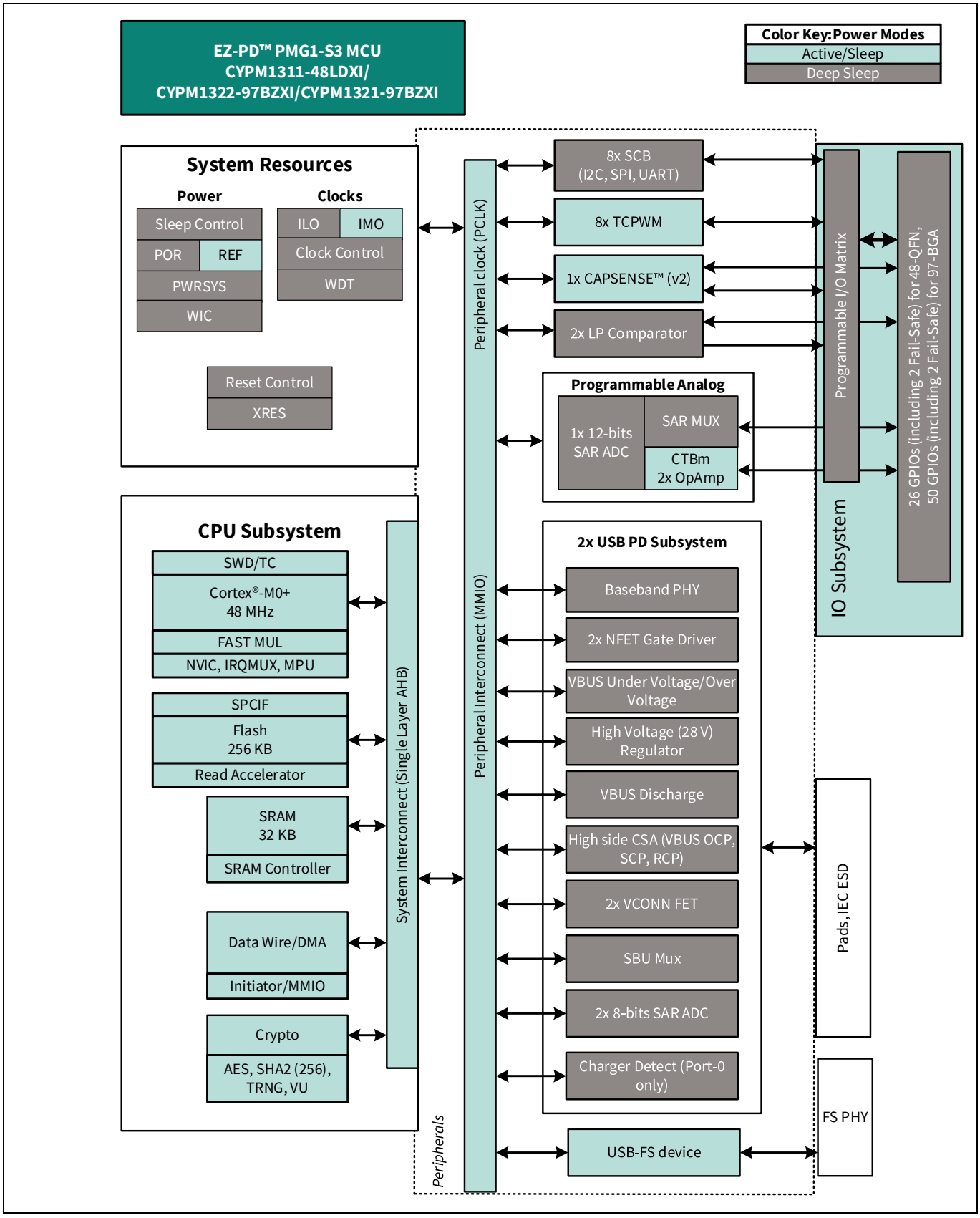


Figure 4 PMG1-S3 MCU ブロック図

1.6 ソフトウェア環境: ModusToolbox™ソフトウェア

ModusToolbox™ソフトウェア環境は、デバイスの設定, 周辺機器のセットアップ, および必要なミドルウェアでプロジェクトを補完するためのツール セットを備えて、PMG1 MCU アプリケーション開発をサポートします。

本アプリケーション ノートでは、ModusToolbox™開発エコシステムの概要を説明し、簡単な「Hello World」アプリケーションをご紹介します。空のアプリケーション テンプレートからアプリケーションを作成するための詳細な手順については、以降の節で説明します。

2 開発エコシステム

2.1 PMG1 MCU リソース

PMG1 MCU ファミリには、開発プロセスを支援する豊富なドキュメント, 開発ツール, およびオンライン リソースが用意されています。詳細は PMG1 MCU Web ページを参照してください。

- データシートには、機能説明や電気的仕様など、選択した MCU を選択して使用するために必要なすべての情報が記載されています。
- アプリケーション ノートおよびサンプル コードは、基本から上級レベルまで、幅広いトピックを提供します。
- リファレンス マニュアルは、各デバイス ファミリのアーキテクチャとレジスタを詳細説明します。
- プロトタイピング キットは、PMG1 MCU を使用したさまざまなアプリケーションの評価, 設計, および開発に利用できます。
- テクニカル サポート: PMG1 MCU コミュニティ フォーラム、ナレッジ ベース記事

2.2 ModusToolbox™ソフトウェア

ModusToolbox™開発プラットフォームは PMG1 MCU を使用したファームウェア/アプリケーション開発に利用されます。この最新世代のツールセットには Eclipse IDE が含まれており、Windows, Linux, および macOS プラットフォームに対応しています。ModusToolbox™ソフトウェアには、コンフィギュレータ, ミドルウェア ライブラリ, PMG1 MCU 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。コンフィギュレータを使用すると、デバイスのさまざまなブロックを設定し、ファームウェア開発に使用するコードを生成できます。

ModusToolbox™のための Eclipse IDE は、Quick Panel でツールおよびデザイン コンフィギュレータ用のクイック ランチャーを統合します。ModusToolbox™は、Microsoft Visual Studio Code, Arm® MDK (µVision), および IAR Embedded Workbench などのサードパーティ IDE にも対応します。

ModusToolbox™ソフトウェアに含まれるツール/リソースの概要を Figure 5 に示します。ModusToolbox™ ソフトウェアの詳細な概要は ModusToolbox™ tools package user guide を参照してください。

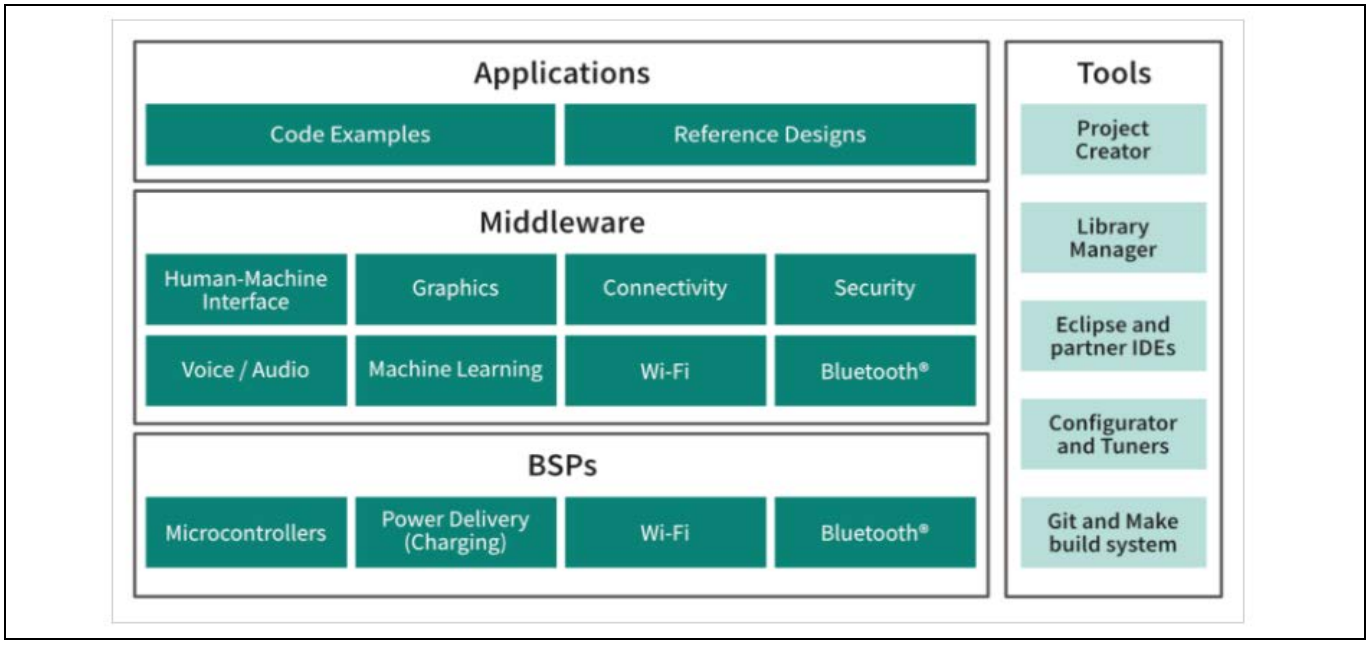


Figure 5 ModusToolbox™ソフトウェア

2.3 PMG1 MCU ソフトウェア リソース

PMG1 MCU ソフトウェア リソースには、PMG1 MCU を使用したファームウェア開発を始めるためのコンフィギュレータ, ドライバ, ライブラリ, ミドルウェアのほか、Makefile およびスクリプトが含まれています。

2.3.1 コンフィギュレータ

ModusToolbox™ソフトウェアには、ハードウェア ブロックまたはミドルウェアの設定を容易にする、コンフィギュレータと呼ばれるツールが含まれます。例えば、シリアル通信ブロックを希望のコンフィギュレーションを持つ UART として設定するために、すべてのドキュメントを検索する代わりに、デバイスコンフィギュレータを起動して、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 コンフィギュレーションを保存すると、ツールはハードウェアを希望のコンフィギュレーションで初期化する「C」コードを生成します。

コンフィギュレータは互いに独立していますが、組み合わせて使用することで柔軟なコンフィギュレーション 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使用することも、他のツールと組み合わせて使用することもできます。コンフィギュレータは以下の目的で使用されます。

- オプションを設定し、コードを生成して、ペリフェラル ドライバを設定する
- ペリフェラルのピンやクロックなどの接続を設定する
- オプションを設定し、コードを生成して、ミドルウェアを設定する

PMG1 MCU 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には次のものがあります。

- **Device Configurator**: クロックやピンなどのデバイス周辺機器や、独自のコンフィギュレータ ツールを必要としない標準 MCU 周辺機器を有効にして設定するために使用されます。このコンフィギュレータは、アプリケーションで使用される初期化コードを生成します。
- **EZ-PD™ Configurator**: PMG1 ファミリのデバイス用の電力供給スタック (PDStack) ミドルウェアの機能を選択し、パラメーターを設定するために使用されます。このツールは、PDStack ミドルウェアで参照できる C 言語の設定コードを生成します。
- **CAPSENSE™ Configurator:** CAPSENSE™コンフィギュレータは ModusToolbox™ソフトウェアに含まれており、CAPSENSE™ウィジェットを作成および設定し、静電容量式タッチボタンとスライダーをインターフェースするためのコードを生成します。
- **USB Configurator**: USB コンフィギュレータは、アプリケーションで USB 2.0 を実装するために必要な USB データ記述子を設定するために使用されます。

これらの各コンフィギュレータは、独自のファイル (例: EZ-PD™コンフィギュレータ用の *design.mtbezpd*) を作成します。コンフィギュレータ ファイル (*design.modus*)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を作成すると、ファイルはアプリケーションにコピーされます。開発者は、アプリケーション用にカスタム 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を上書きすることもできます。

2.3.2 PMG1 MCU アプリケーション開発

ModusToolbox™ ソフトウェア ツールとライブラリには、PMG1 MCU のソフトウェア開発を可能にするソース コードとリファレンス ファームウェアが付属します。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。これにより、開発者は、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構築できます。

Figure 6 に示すように、ModusToolbox™ソフトウェアを使用したアプリケーション開発は次のようになります。

1. **Create New Application** をクリックしてポップアップ ウィンドウを起動し、BSP (プロジェクト クリエーター) を選択してください。
2. キットでフィルターされたテンプレート アプリケーションの一覧に基づいて、新しいアプリケーションを作成してください。これで、選択したアプリケーションのファームウェアが表示されます。
3. **Library Manager** を開き、**Add Library** を選択してミドルウェア/ソフトウェア コンポーネントを追加し、**OK** をクリックしてください。
4. PDL を使用してアプリケーション ファームウェアの開発を開始してください。

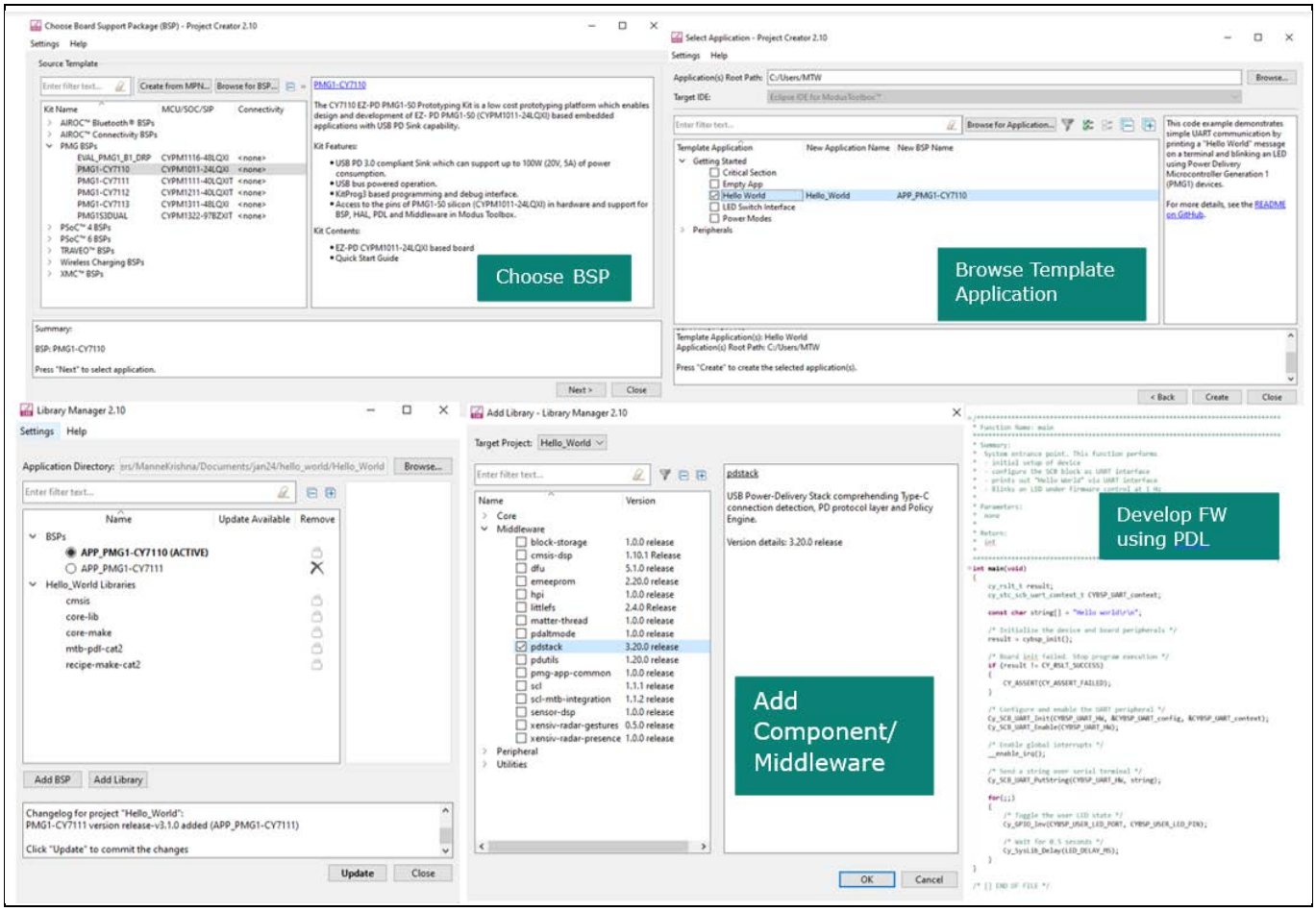


Figure 6 ModusToolbox™ リソースおよびミドルウェア用の Eclipse IDE

2.4 ModusToolbox™ヘルプ

ModusToolbox™ソフトウェアのホーム ページへアクセスして、ModusToolbox™ソフトウェアの最新版をダウンロードおよびインストールしてください。ModusToolbox™の Eclipse IDE を起動して、ModusToolbox™ヘルプの次の項目に移動してください。

Help > Eclipse IDE for ModusToolbox™ General Documentation を選択してください。

- **ModulsToolbox™ documentation:** このページには、ModusToolbox™ソフトウェアの一部として含まれるさまざまな種類のドキュメントの簡単な説明とリンクが記載されています。
- **User Guide:** アプリケーションの作成、および Eclipse IDE を使用したアプリケーションの構築, プログラミング, デバッグについて説明します。
- **Installation guide :** このガイドには、ModusToolbox™ツール パッケージのインストール手順が記載されています。

2.5 他の IDE のサポート

IAR Embedded Workbench や Microsoft Visual Studio Code などのお気に入りの IDE を使用し、PMG1 MCU のファームウェアを開発できます。

詳細については、ModusToolbox™ tools package user guide の「Exporting to IDEs」セクションを参照してください。インフィニオンは、ModusToolbox™ソフトウェアに付属のコンフィギュレーション ツールを使用してリソース コンフィギュレーションを生成することを推奨します。

2.6 Eclipse IDE を使用したプログラミングおよびデバッグ

ModusToolbox™ソフトウェアの Eclipse IDE は、KitProg3 を必要とし、PMG1 MCU アプリケーションのデバッグに Open On-Chip Debugger (OpenOCD) プロトコルを使用します。また、Segger J-Link などの業界標準プローブを使用して GDB (GNU デバッガ) もサポートしています。

Note: すべての PMG1 MCU キットには、KitProg3 オンボード プログラマ/デバッガが搭載されています。Cortex® マイクロコントローラー ソフトウェア インターフェース標準 - デバッグ アクセス ポート (CMSIS-DAP) をサポートします。詳細は、KitProg3 user guide を参照してください。

ModusToolbox™ソフトウェアによる PMG1 デバイスのファームウェアのプログラミング/デバッグの詳細については、Eclipse IDE for ModusToolbox™ user guide の「Program and Debug」セクションを参照してください。

2.7 mtb プログラマを使用したプログラミング

ModusToolbox™プログラマは、対象デバイスのフラッシュをプログラム, 消去, 検証, および読み出しをするためのグラフィカル ユーザー インターフェースを提供する、スタンドアロンのクロスプラットフォーム フラッシュ プログラマ ツールです。ModusToolbox™プログラミング ツール パッケージに同梱されており、HEX, SREC, ELF, および BIN プログラミング ファイル形式をサポートします。ダウンロードするには、mtb-programmer にアクセスしてください。

1. プログラミングの前に、プロトタイピング キットで次のハードウェア接続を行ってください。
   a) キットの J1 (KitProg3 USB Type-C ポート) コネクタからホストに Type-C ケーブルを接続してください。
   b) プログラミング モードの場合は、J5 (電源選択ジャンパー) ピン 2 と 3 を接続してください。

Note: 動作モードの場合は、J5 (電源選択ジャンパー) ピン 1 と 2 を接続し、Type-C ケーブル J1 と J10 の両方をそれぞれホストと電源に接続して、デバイスをプログラミングすることもできます。

2. プローブ/キットドロップダウンでデバイス名 (CY7110) を選択してください。
3. **Open Programming File** ダイアログで、ロードする HEX, SREC, ELF, または BIN ファイルの場所に移動し、それを選択して **Open** をクリックしてください。
4. **Connect** をクリックしてください。ModusToolbox™プログラマはデバイスと通信し、ログにさまざまなメッセージを表示します。次に、Figure 7 に示すように、ステータスバーのメッセージで接続が行われていることが示されます。
5. **Program** をクリックしてください。ModusToolbox™プログラマはプログラムファイルをデバイスにダウンロードし、ログにメッセージを表示します。mtb-programmer を使用したプログラミングの詳細については、ModusToolbox™ Programmer GUI user guide のセクション 3 を参照してください。

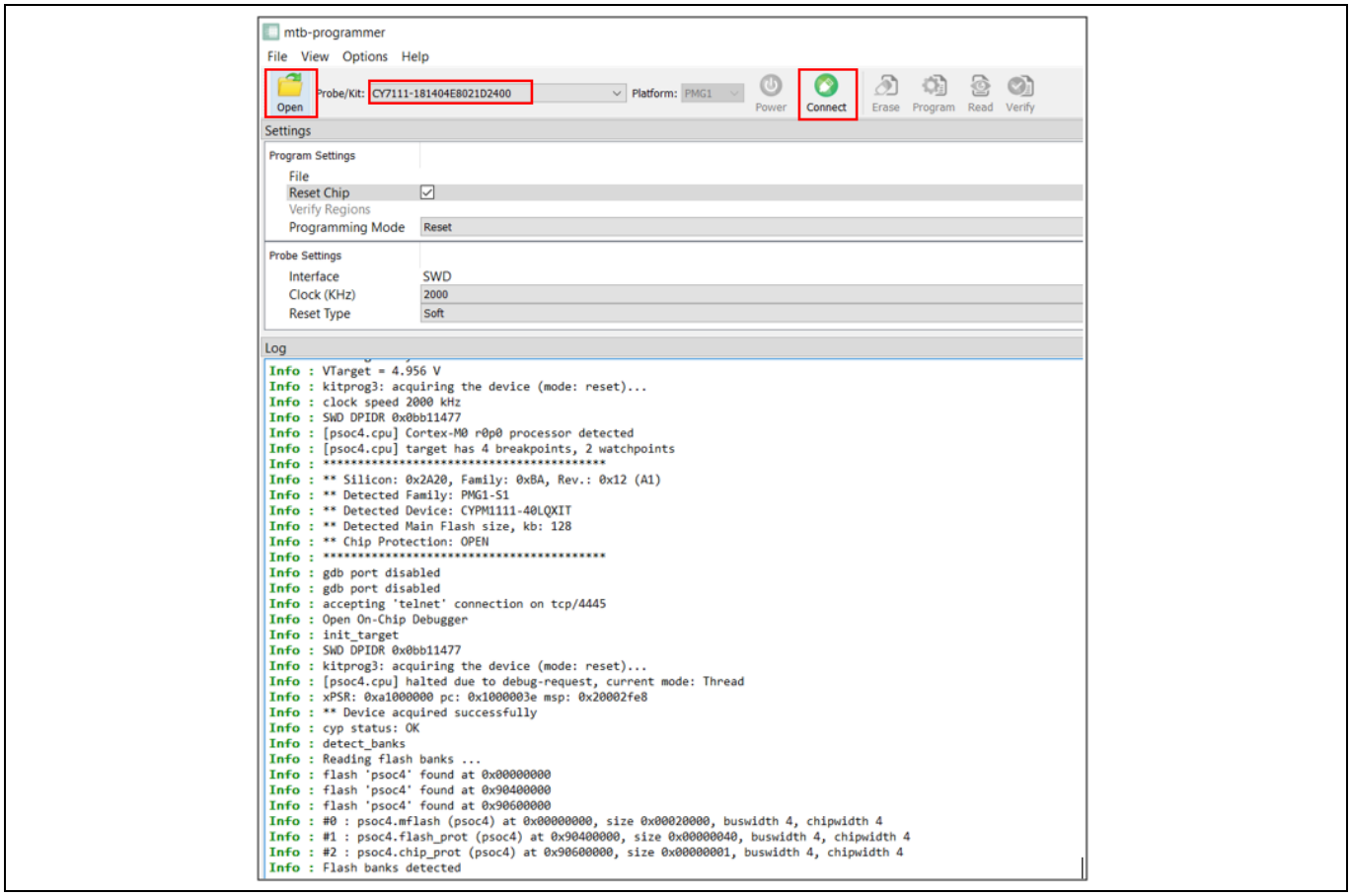


Figure 7　mtb-プログラマ ツール

mtb-プログラマは、Figure 8 に示すように、ダウンロード後に ModusToolbox™ソフトウェアの **Quick Panel** で利用できます。そこから直接開くことができます。

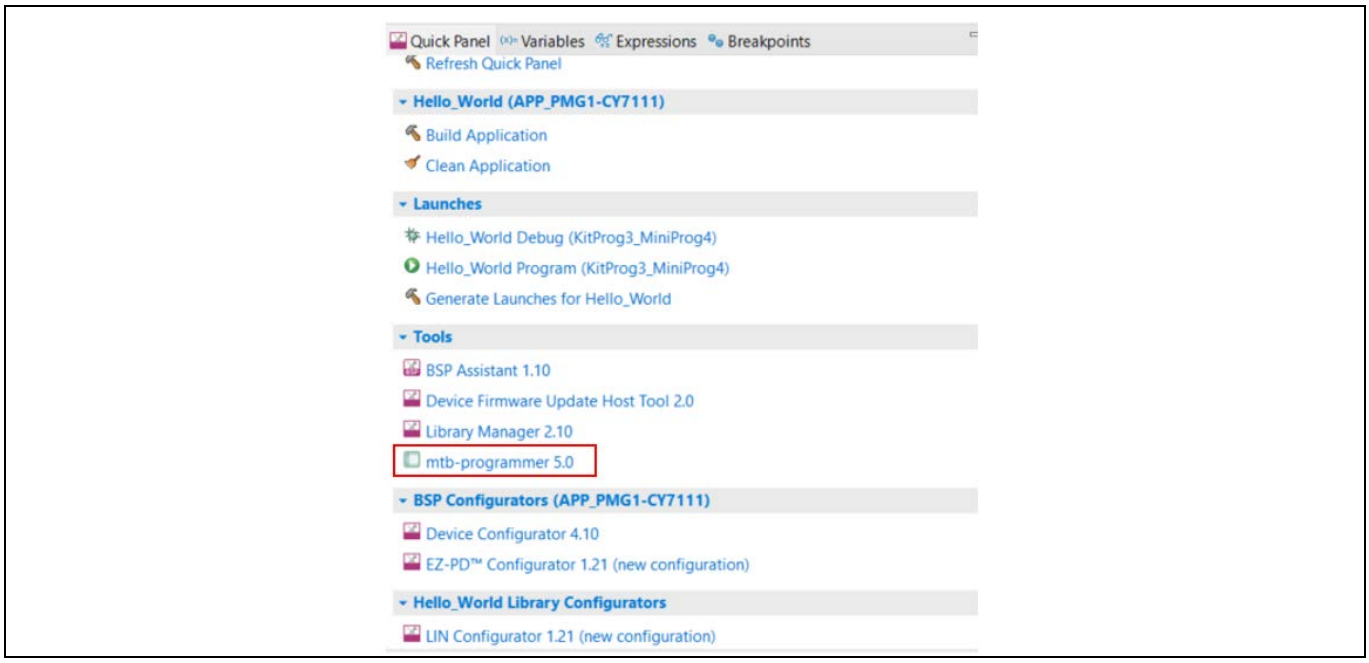


Figure 8　Quick Panel の mtb-プログラマ ツール

2.8 PMG1 MCU キット

Table 2 PMG1 MCU キットの BSP と対応する BSP

Sl no.	PMG1 MCU	PMG1 MCU キット	BSP
1	CYPM1011-24LQXI	CY7110 EZ-PD™ PMG1-S0 Prototyping Kit	PMG1-CY7110
2	CYPM1111-40LQXI	CY7111 EZ-PD™ PMG1-S1 Prototyping Kit	PMG1-CY7111
3		EVAL_PMG1_S1_DRP	EVAL_PMG1_S1_DRP
4	CYPM1211-40LQXI	CY7112 EZ-PD™ PMG1-S2 Prototyping Kit	PMG1-CY7112
5	CYPM1211-42-CSP	該当なし	PMG1-CY7112
6	CYPM1311-48LQXI	CY7113 EZ-PD™ PMG1-S3 Prototyping Kit	PMG1-CY7113
7	CYPM1322-97BZXI	該当なし	PMG1S3DUAL
8	CYPM1321-97BZXI	EVAL_PMG1_S3_DUALDRP	EVAL_PMG1_S3_DUALDRP

3 ModusToolbox™ソフトウェア用 Eclipse IDE を使用した PMG1 MCU アプリケーションの作成

ここでは、ModusToolbox™ソフトウェアを使用して EZ-PD™ PMG1 MCU のプロジェクト作成, プログラミング, およびデバッグを行う方法を紹介します。

3.1 事前準備

- PMG1 MCU 製品ラインから適切なキットを選択してください。
- ModusToolbox™ ウェブページから最新の ModusToolbox™ソフトウェアをダウンロードしてください。

ソフトウェアをインストールしたら、ModusToolbox™ tools package user guide を参照してソフトウェアの概要を確認してください。プロジェクト作成中に GitHub リポジトリを取得するには、インターネットアクセスも必要です。

PMG1 MCU キットのリストについては、Table 2 を参照してください。

3.2 新しいアプリケーションの作成

1. ModusToolbox™ 用の Eclipse IDE を開きます。
2. **Quick Panel** に移動し、**Start** の下にある **New Application** をクリックします。または、Figure 9 に示すように、**File** > **New** > **ModusToolbox™ Application** を選択することもできます。

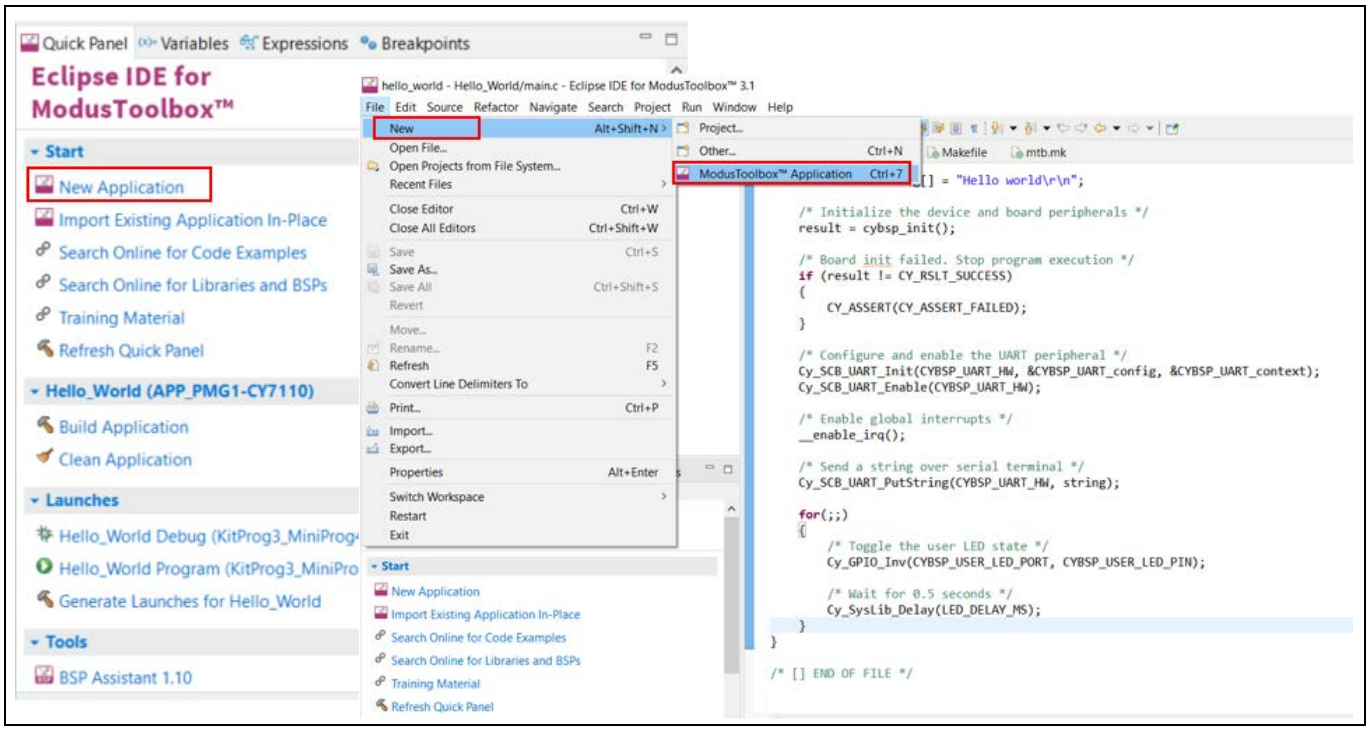


Figure 9 新しい ModusToolbox™アプリケーションの作成

3. Choose Board Support Package (BSP) ダイアログで、キット名を選択します。例えば、PMG1-S0 の場合は、PMG1-CY7110 を選択します。Figure 10 を参照してください。
4. **Next** をクリックします。

3 ModusToolbox™ソフトウェア用 Eclipse IDE を使用した PMG1 MCU アプリケーションの作成

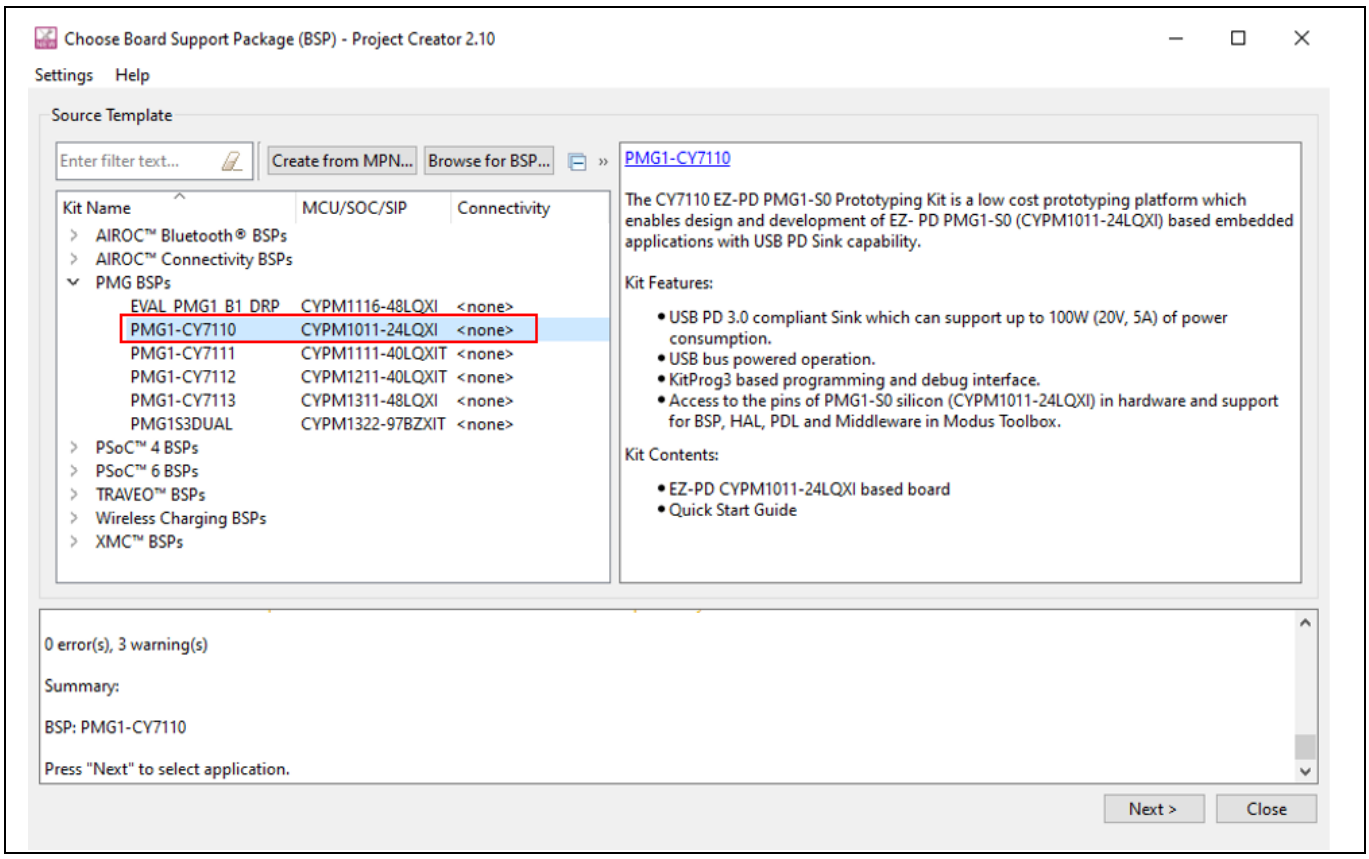


Figure 10　ターゲット ハードウェアの選択

5. **Select Application** ダイアログで、Figure 11 に示すように、Hello World テンプレート アプリケーションを選択します。
6. **Name** フィールドに、PMG1_S0_Hello_World などのアプリケーションの名前を入力します。または、デフォルト名のままにすることもできます。
7. application(s) Root Path に、アプリケーション ファームウェアを保存するワークスペースの場所を入力します。
8. **Create** をクリックしてアプリケーションを作成します。
9. プロジェクトが正常に作成されると、**Project Creator** が自動的に閉じます。

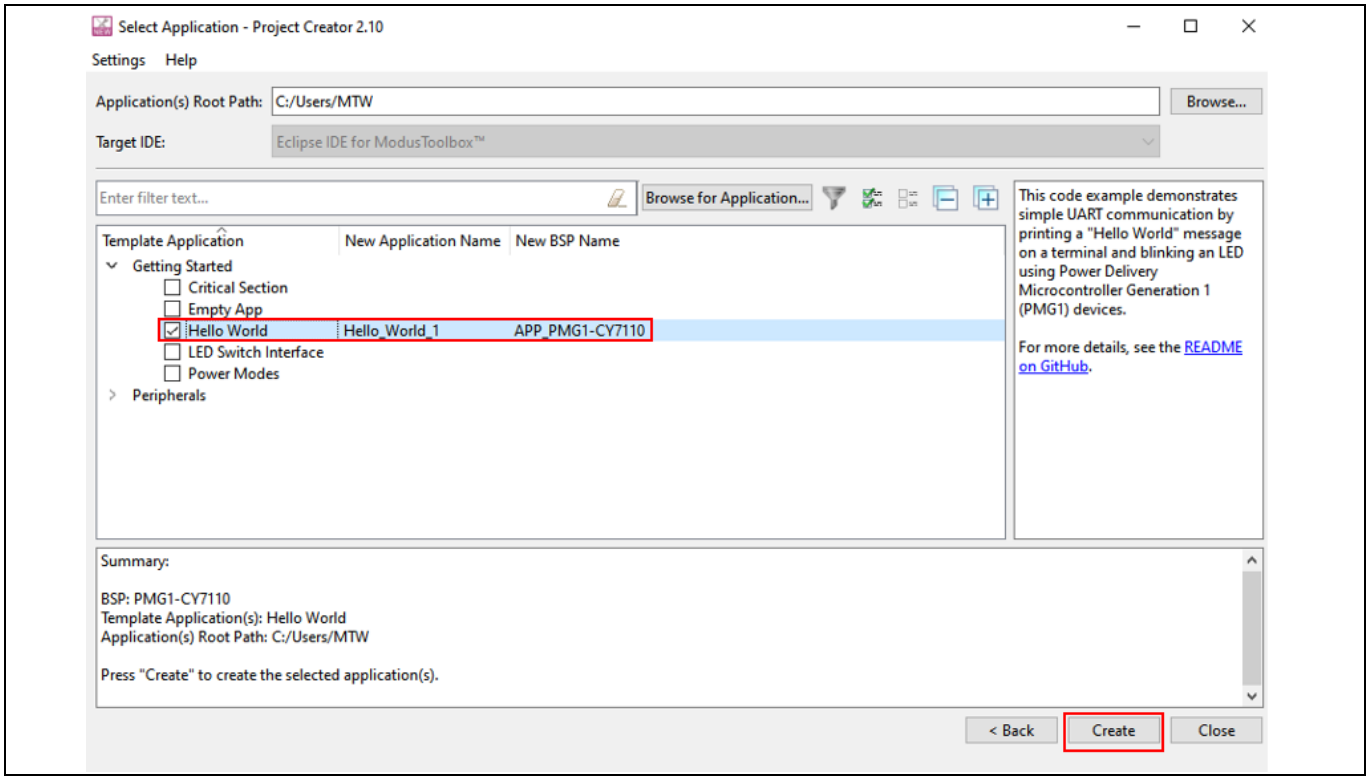


Figure 11　スターターアプリケーションの選択

これで、PMG1-S0 MCU 用の新しい ModusToolbox™ソフトウェア アプリケーションが正常に作成されました。BSP は、CY7110 EZ-PD™ PMG1-S0 prototyping kit (PMG1-CY7110) 上の CYPM1011-24LQXI デバイスを使用します。

アプリケーションが正常に作成されると、テンプレート アプリケーションの実装に関する情報を含む *README.md* ファイルが開きます。

PMG1-S0 MCU または別の PMG1 MCU 製品番号に基づくカスタム ハードウェアを使用する場合は、ModusToolbox™ user guide (**Help** > **ModusToolbox™ General Documentation> ModusToolbox™ user guide**) の「Creating/edit BSP」を参照してください。

3.3　設計設定の閲覧と変更

Figure 12 に、アプリケーション プロジェクトの構造を示す Eclipse IDE Project Explorer インターフェースを示します。

ModusToolbox™ソフトウェアの Eclipse IDE では、PMG1 MCU アプリケーションは、CM0/CM0+ CPU のコードを開発するプロジェクトで構成されています。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、それぞれのサブフォルダがプロジェクトの特定の側面を示します。

- アプリケーション プロジェクトは、通常ルート フォルダにある Makefile を含みます。これには、プロジェクトの再作成方法についての説明があります。アプリケーションには複数のプロジェクトが存在する場合があります。通常、依存する各プロジェクトは、アプリケーション フォルダ内の独自のフォルダ内に存在し、独自の Makefile を含みます。
- build 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。

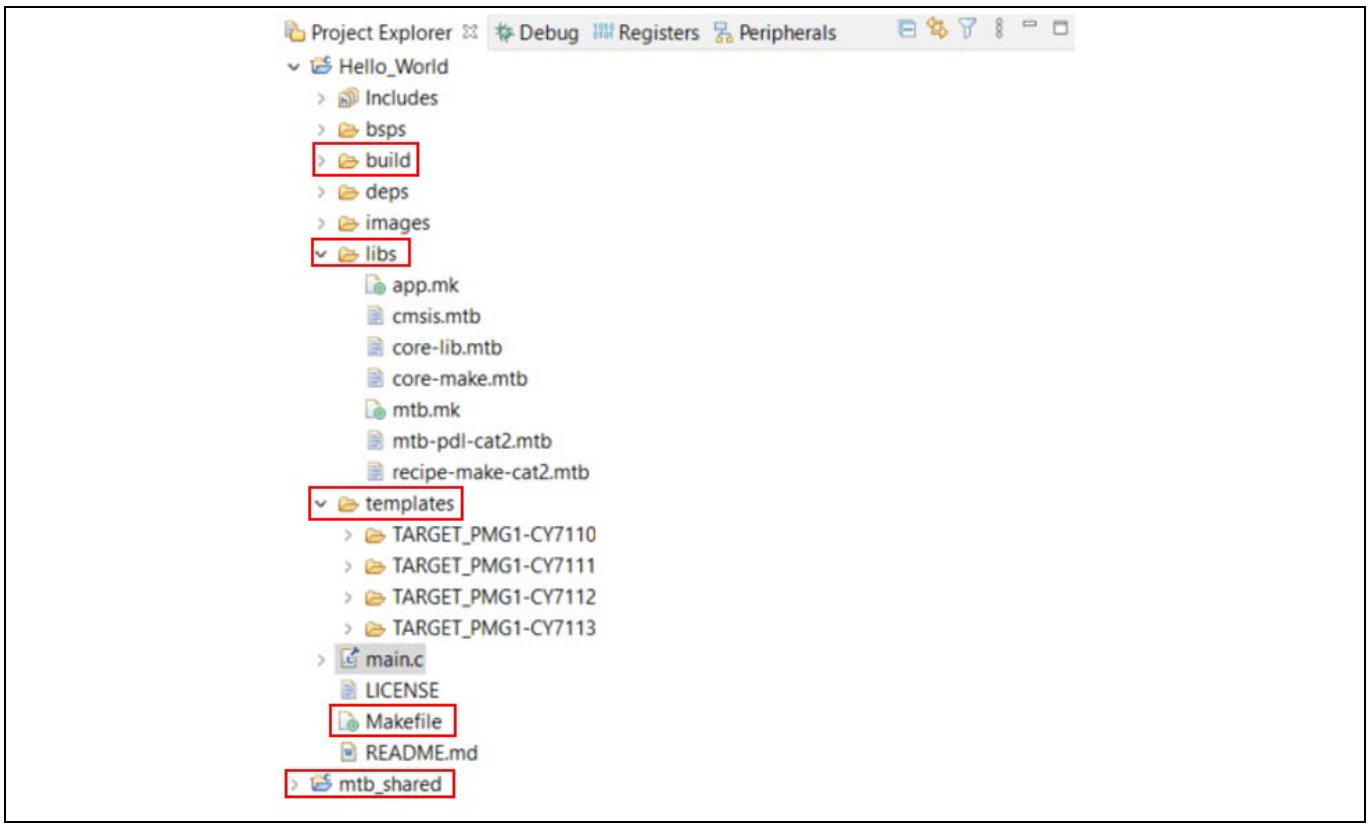


Figure 12　Project Explorer ビュー

- libs フォルダには、アプリケーションに必要なすべてのライブラリの相対パスを格納する *mtb.mk* ファイルが含まれます。ビルド システムはこのファイルを使用して、アプリケーションに必要なすべてのライブラリを検索します。
- デフォルトでは、新しいアプリケーションを作成するか、既存のアプリケーションにライブラリを追加して共有として指定すると、すべてのライブラリはアプリケーション ディレクトリに隣接する *mtb_shared* ディレクトリに配置されます。*mtb_shared* フォルダは、同じバージョンの BSP/library を使用する異なるアプリケーション間で共有されます。

3.4　デバイス コンフィギュレータの開始

テンプレート フォルダにはターゲット ファイルが含まれており、*design.modus* ファイルがあります。

1. *design.modus* ファイルをクリックして、BSP で定義された設計設定を開きます。
2. **Quick Panel** の **Device Configurator** リンクをクリックして、設計設定を表示および変更します。Figure 13 を参照してください。

テンプレート セクションの下にある他の *design.modus* ファイルをダブルクリックして開き、それぞれのコンフィギュレータを開くか、**Quick Panel** の対応するリンクをクリックすることもできます。

3 ModusToolbox™ソフトウェア用 Eclipse IDE を使用した PMG1 MCU アプリケーションの作成

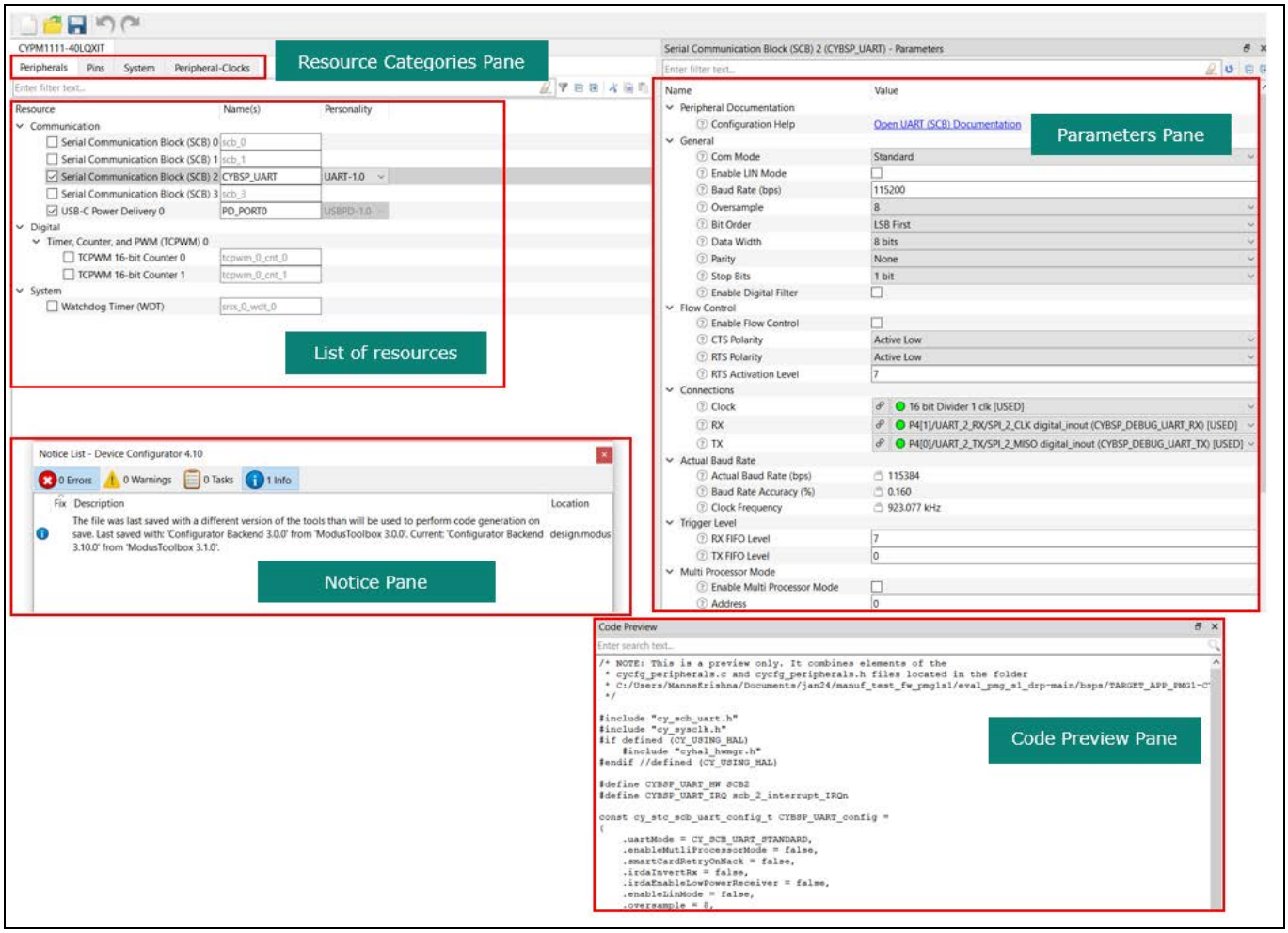


Figure 13 ***design.modus*** **の概要**

3. **Device Configurator** の **Resources Categories** タブの **List of Resources** から、ペリフェラル, ピン, およびクロックなどのデバイスで利用可能なさまざまなリソースを選択します。
   リソースの Personality を選択することにより、リソースの動作を選択できます。例えば、Serial Communication Block (SCB) リソースには、EZI2C, I2C, SPI または UART のパーソナリティを設定できます。Alias 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、カンマで区切って (スペースなし) 指定できます。
4. **Parameters** ペインでは、有効な各リソースと選択したパーソナリティの設定パラメーターを入力します。
   Code Preview ペインには、選択した設定パラメーターごとに生成されたコンフィギュレーション 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。
   Notices ペインには、設定から生じるエラー, 警告, および情報メッセージが表示されます。開発者は、PDL が提供する API 呼び出しを使用して、周辺機器を初期化して使用できます。詳細については、main.c (Hello World サンプルコード) の「UART ペリフェラルの設定と有効化」を参照してください。Figure 14 を参照してください。

3.5 ファームウェアの記述

開発プロセスのこの時点で、アプリケーション テンプレートを使用してアプリケーションを作成しました。

- PMG1 MCU の「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べてアプリケーションにすでに含まれています。このアプリケーションは、必要に応じて変更できます。
- 独自のアプリケーションを最初から作成する場合は、「**Select Application**」で利用可能な「**Empty App**」を選択し、*main.c* ファイルとデバイス コンフィギュレータを適宜変更して、必要な機能を実現します。空のアプリケーションについては、Figure 11 を参照してください。

3.5.1 ファームウェア フロー

- アプリケーションの *main.c* ファイル内のコードを調べます。ファームウェア フローチャートについては、Figure 14 を参照してください。
- このサンプルのリソース初期化は、CM0 CPU によって実行されます。これはシステム クロック, ピン, ペリフェラル接続のクロック, および他のプラットフォーム リソースを設定します。
- CM0 CPU が有効になると、クロックとシステム リソースは BSP 初期化関数により初期化されます。UART ペリフェラルが設定され、有効になります。端末エミュレータに「Hello World」メッセージが出力されます。CPU は、500 ミリ秒の遅延で LED の状態を継続的に切り替えます。

Note: アプリケーション コードは、BSP/ミドルウェア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。

```c
/*******************************************************************************
 * Include header files
 ******************************************************************************/
#include "cy_pdl.h"
#include "cybsp.h"


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


/* Function Name: main
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_FW_LED_PORT, CYBSP_FW_LED_PIN);

        /* Wait for 0.5 seconds */
        Cy_SysLib_Delay(LED_DELAY_MS);
    }
}

/* [] END OF FILE */
```

Figure 14 main.c

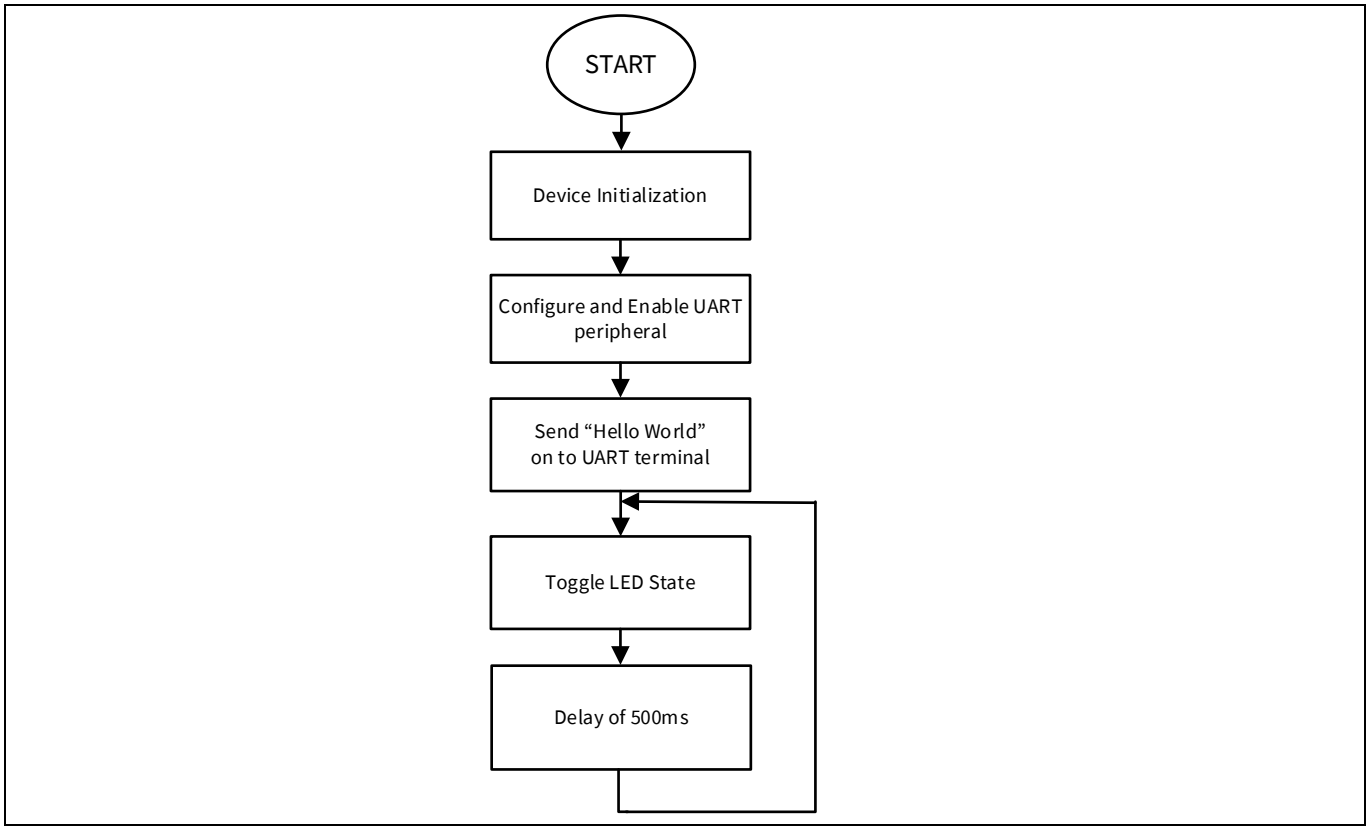


Figure 15　ファームウェア フローチャート

これで、サンプル コードでのファームウェアの動作概要が完了です。

3.6　アプリケーションのビルドとプログラム

プログラムのビルド

1. 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**<application name>**グループの下にある **Build Application** ショートカットをクリックします。これにより、**Debug** ビルド コンフィギュレーションが選択され、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。
2. Figure 16 に示すように、**Console** ビューにはビルド動作の結果が一覧表示されます。

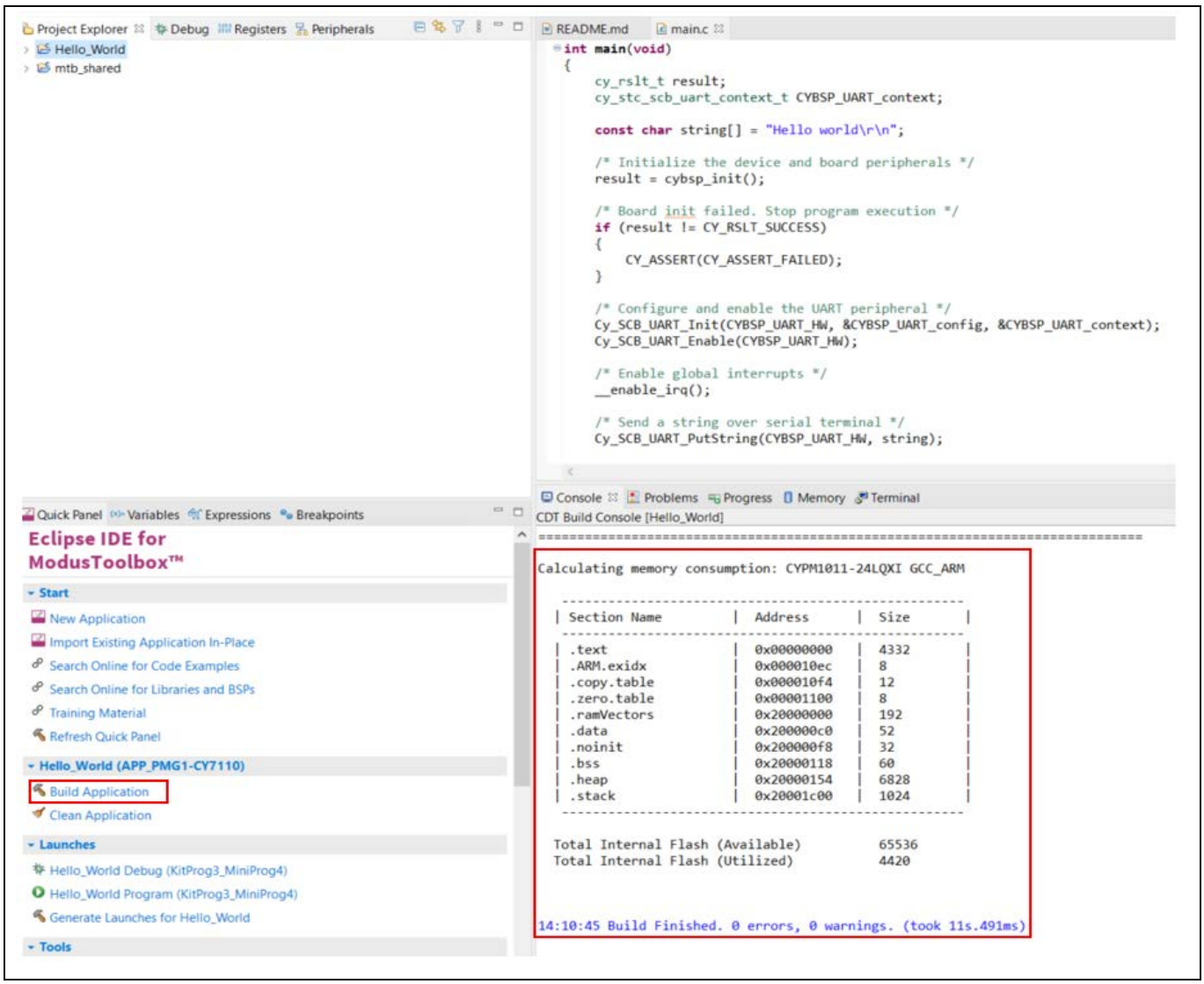


Figure 16　アプリケーションのビルド

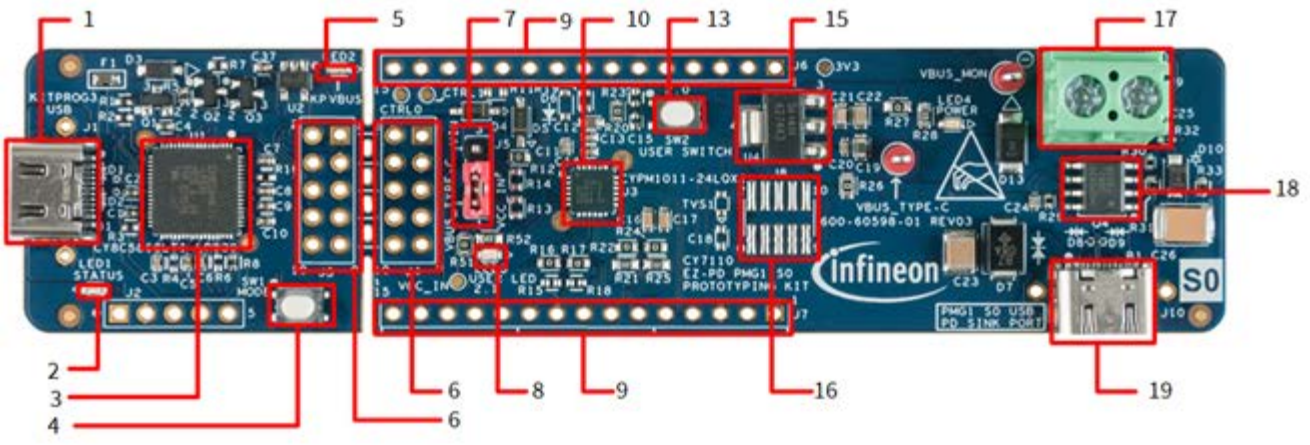
エラーが発生した場合は、次のオプションを試行します。

- 前のステップに戻って必要な作業がすべて完了したことを確認します。
- **Problems** タブをチェックして、エラーの詳細を見つけ、それらを解決します。
- **Quick Panel** で「Clean」をクリックして、再度プロジェクトをビルドします。

*Note:　アプリケーションをビルドするために、コマンド ライン インターフェース(CLI) も使用できます。ModusToolbox™ user guide の ModusToolbox™ **build system** セクションを参照してください。このドキュメントは、ModusToolbox™ソフトウェアの **help menu > ModusToolbox™ General Documentation** からも参照できます。*

デバイスのプログラミング

Figure 17 に CY7110 EZ-PD™ PMG1-S0 プロトタイピング キットを示します。PMG1 MCU web ページの「Kit operation」セクションを参照してください。



1. KitProg3 USB Type-C port (J1)
2. KitProg3 status LED (LED1)
3. PSoC™ 5LP controller (U1)
4. KitProg3 mode switch (SW1)
5. KitProg3 VBUS LED (LED2)
6. KitProg3 header (J3, J4) *
7. Power selection jumper (J5)
8. User LED (LED3)
9. I/O headers (J6, J7) *
10. CYPM1011-24LQXI (PMG1-S0)
13. User switch (SW2)
15. 3.3V on-board LDO
16. 10-pin SWD/JTAG header *
17. DC_OUT terminal block (J9)
18. Load switch
19. PMG1 MCU USB Type-C port (J10)

* Footprint only; not populated on the board.

Figure 17　CY7110 EZ-PD™ PMG1-S0 プロトタイピング キットの選択

3. ModusToolbox™ソフトウェアは、OpenOCD プロトコルを使用して PMG1 MCU 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。

内蔵プログラマ (KitProg3) を備えた PMG1 MCU プロトタイピング キットを使用する場合、ステップ 4 と 2 に従います。

ご自身でハードウェア開発をされる場合、例えばインフィニオン CY8CKIT-005 MiniProg4 などのハードウェア プログラマ/デバッガが必要な場合があります。

4. **プロトタイピング キットで以下のハードウェア接続を行います。**
   c) Type-C ケーブルをキットの J1 (KitProg3 USB Type-C ポート) コネクタからホストに接続します。
   d) プログラミング モード用に J5 (電源選択ジャンパー) のピン 2 とピン 3 を接続します。

Note:　動作モード用に J5 (電源選択ジャンパー) のピン 1 とピン 2 を接続し、Type-C ケーブルを J1 と J10 の両方をそれぞれホストと電源に接続することによって、デバイスをプログラムすることもできます。

5. **ボードをプログラムします。**
   a) **ModusToolbox™ software** で、**Project Explorer** からアプリケーションプロジェクトを選択します。
   b) **Quick Panel** で、Figure 18 に示すように、**<Application Name> Program (KitProg3_MiniProg4)** をクリックします。

IDE は適切な実行設定を選択して実行します。この手順では、前回のビルド以降に変更されたファイルがある場合にもビルドが実行されることに注意してください。

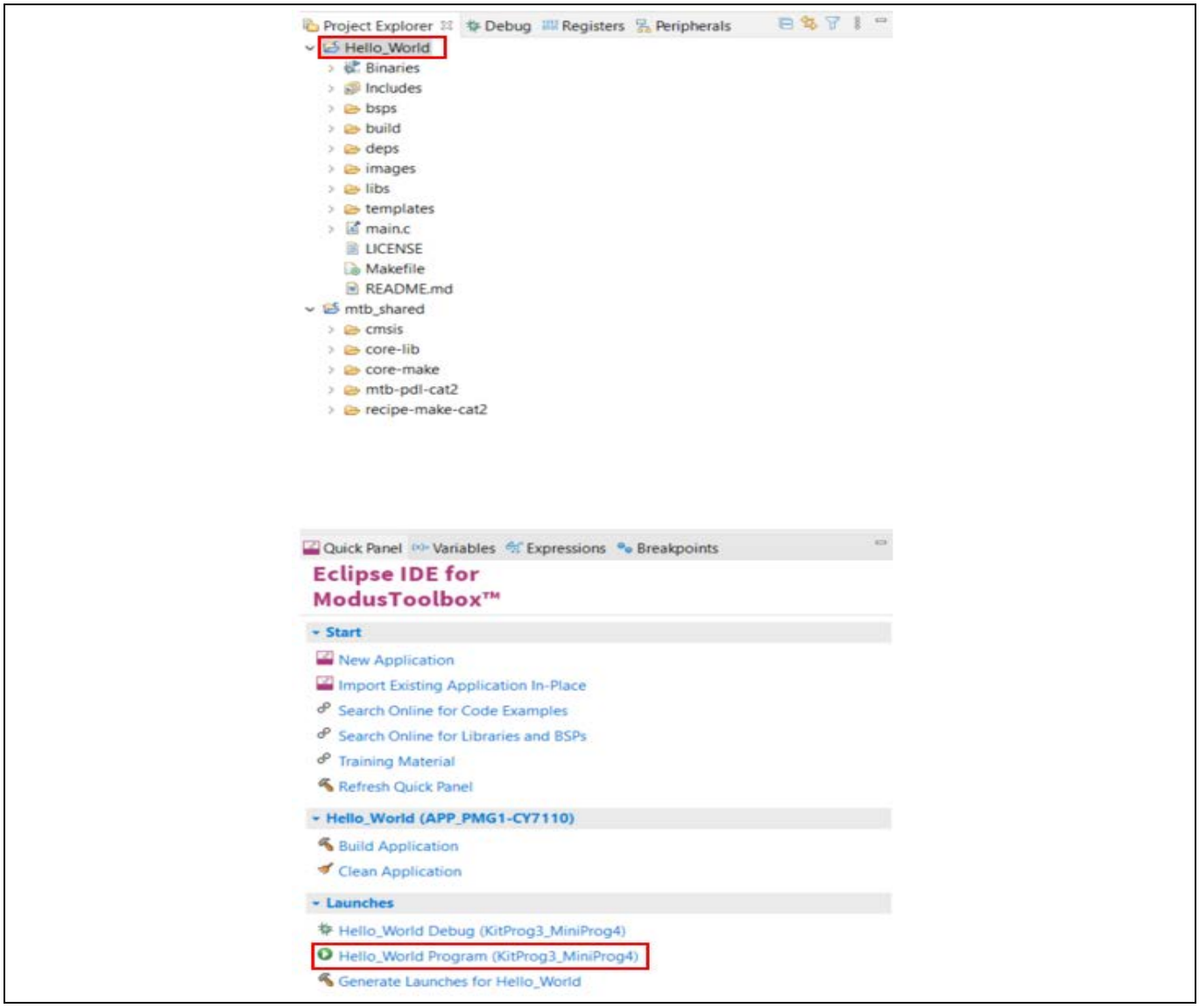


Figure 18　デバイスへのアプリケーション プログラミング

6. Figure 19 に示すように、**Console** ビューにはプログラミング動作の結果が一覧表示されます。

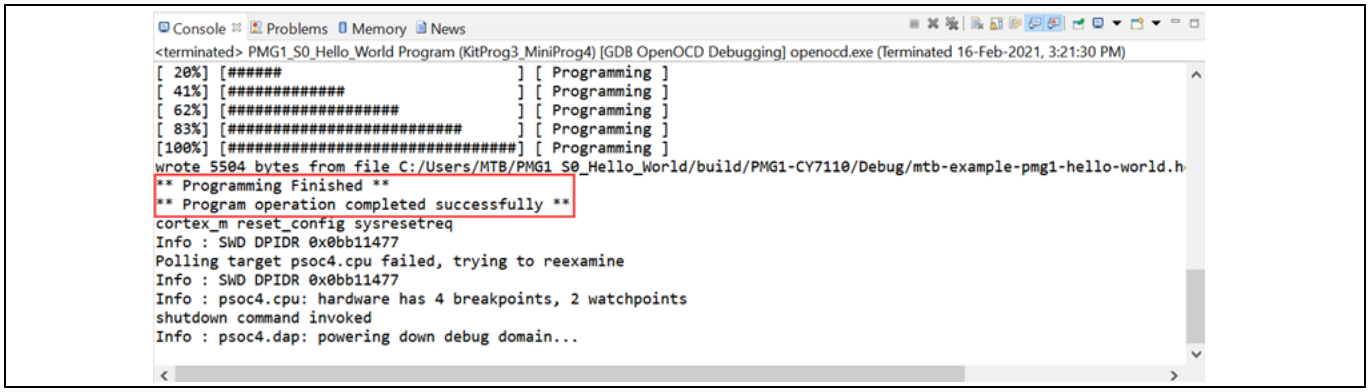


Figure 19　Console – プログラミング結果

3.7 設計のテスト

1. 以下のハードウェア接続を行います。
   a) 動作モード用に J5 (電源選択ジャンパー) のピン 1 とピン 2 を接続します。
   b) J6.10 を J3.8 に、J6.9 を J3.10 に接続して、KitProg3 と PMG1-S0 の間に UART 接続を確立します。他のキットとの UART 接続については readme に従ってください

Note: UART 接続を確立するためのこの手順は、キットボードの次の古いリビジョンに対してのみ実行する必要があります。

- CY7110 board revision 3 またはそれ以下
- CY7111 board revision 2 またはそれ以下
- CY7112 board revision 2 またはそれ以下
- CY7113 board revision 3 またはそれ以下

Figure 20 に示すように、キットのリビジョン番号 (600-60xxx-01 Revxx) は、キットボードのシルクスクリーンにマークされています。



Figure 20 PMG1 MCU キットボードのリビジョン番号の識別

   c) Type-C ケーブルをキットの J1 (KitProg3 USB Type-C ポート) コネクタからホストに接続します。
   d) KitProg3 VBUS LED (LED2) およびステータス LED (LED1) がアンバー色に点灯することを確認します。
2. Figure 21 に示すように、ターミナル プログラムを開き、KitProg3 COM ポートを選択します。本アプリケーション ノートでは、Tera Term を UART 端末エミュレータとして使用し、結果を表示します。任意の端末を使用して出力を表示できます。

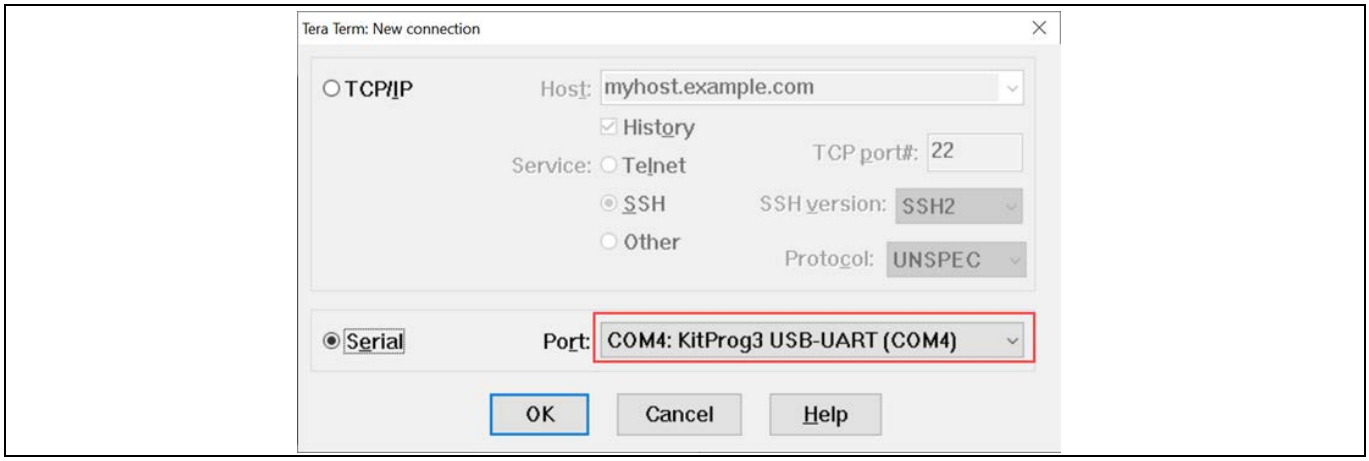


Figure 21　Tera Term での KitProg3 COM ポートの選択

3. シリアル ポート パラメーターを 8N1 および 115200 ボーに設定します。

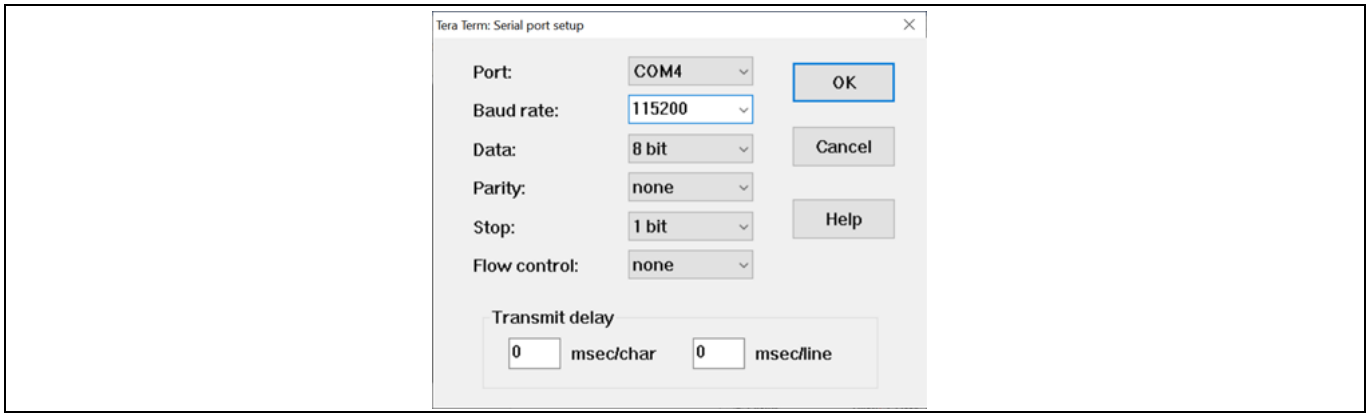


Figure 22　Tera Term でのボーレートの設定

4. Type-C ケーブルを J10 (PMG1 MCU USB Type-C ポート) コネクタからホストに接続します。
5. UART 端末に「Hello World」が表示されていることを確認します。

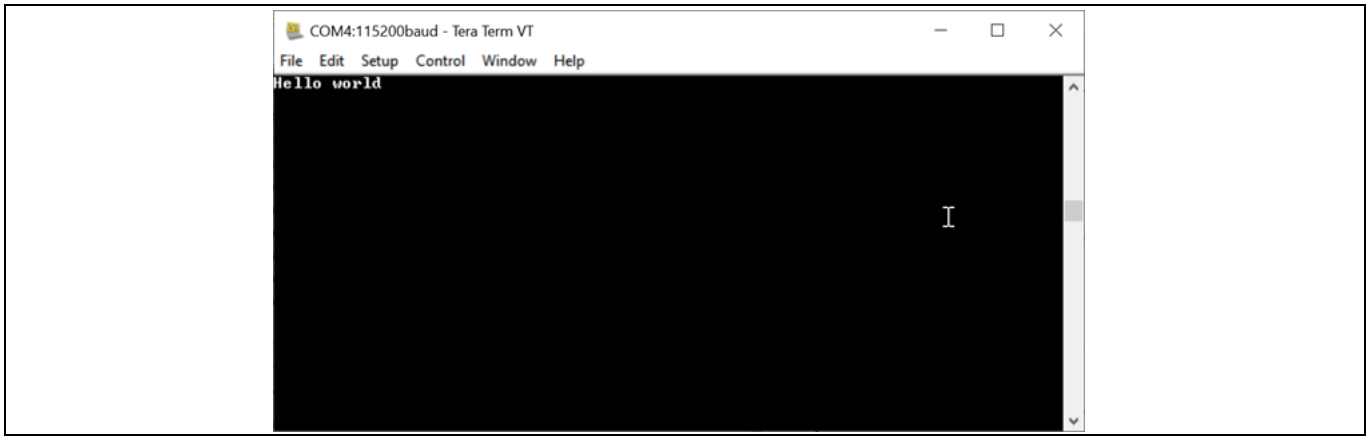


Figure 23　端末出力の「Hello World」

6. 電源 LED (LED4) がアンバー色に点灯し、ユーザーLED (LED3) が点滅することを確認します。

3.8 アプリケーションのデバッグ

ModusToolbox™ソフトウェアの Eclipse IDE を使用して、アプリケーションをデバッグできます。プログラミング モードと同様に、プロトタイピング キットは次の 3 つのインターフェースのいずれかでデバッグ モードに入ります。

- KitProg3 USB Type-C ポート
- 10 ピン SWD ヘッダ
- I/O ヘッダ ピン

デバッグを開始する前に、キットを動作モードに設定する必要があります。

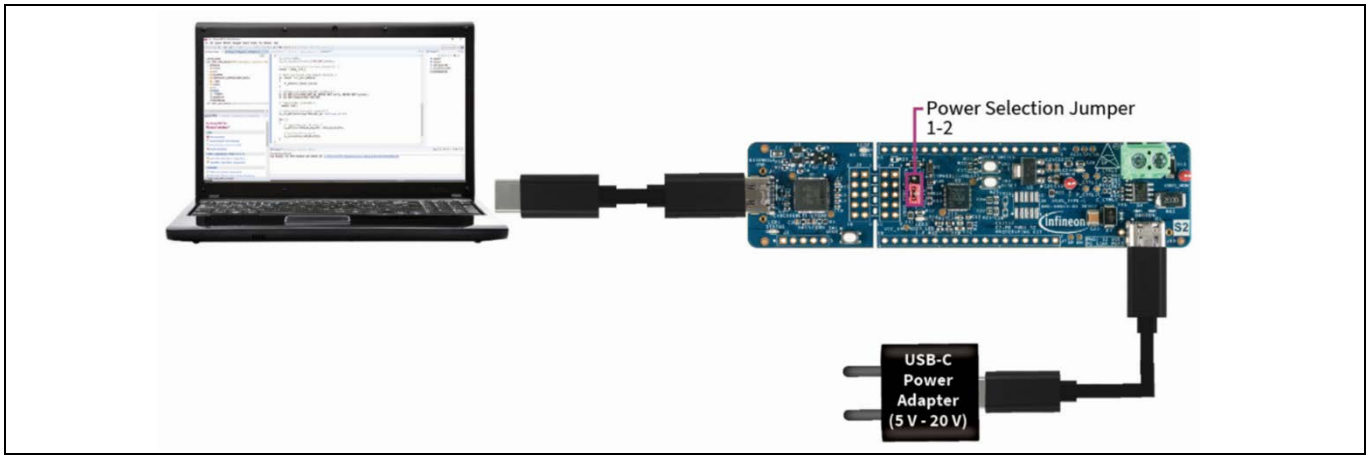


Figure 24 KitProg3 を介したデバッグ

ModusToolbox™ソフトウェアの Eclipse IDE から KitProg3 インターフェースを介して CY7110, CY7111, CY7112, および CY7113 プロトタイピング キット上のアプリケーションをデバッグするには、次の手順を実行します。

1. 電源セクションジャンパー (J5) のピン 1-2 にジャンパー シャントを配置して、キットを動作モードに設定します。
2. USB-PD シンク ポートを USB PD ソースに接続して、オンボード LDO, ロード スイッチ, およびユーザーLED をアクティブにします。電源 LED である LED4 が緑色に点灯することを確認します。
3. KitProg3 USB Type-C ポート (J5) を介してキットをホスト PC に接続します。
4. LED1 と LED2 がアンバー色に点灯していることを確認します。LED2 (KitProg3 電源 LED) は、KitProg3 モジュールの電源が入っていることを示します。LED1 (ステータス LED) は、プログラミング/デバッグ モードとステータスを示し、KitProg3 の電源が入っているときにオンになります。
5. **Quick Panel** タブに移動します。**Launches** セクションから**<Application name> Debug (KitProg3_MiniProg4)** をクリックします。
   IDE はデバッグ モードに切り替わり、main () 関数の最初の行で停止します。これは、アプリケーションがデバッグの準備ができていることを示します。

または、

1. ジャンパー シャント (J5) をピン 2-3 に配置して、ModusToolbox™ソフトウェアがターゲット (PMG1-S0) をパワー サイクル モードで取得できるようにします。
2. **USB PD sink port** を **USB PD source** に接続して、オンボード LDO, ロード スイッチ, およびユーザー LED をアクティブにします。電源 LED である LED4 が緑色に点灯していることを確認します。

3 ModusToolbox™ソフトウェア用 Eclipse IDE を使用した PMG1 MCU アプリケーションの作成

3. プログラミング KitProg3 USB Type-C ポートを介してキットをホスト PC に接続します。
   LED1 と LED2 がアンバー色に点灯していることを確認します。
   LED2 (KitProg3 電源 LED) は、KitProg3 モジュールの電源が入っていることを示します。
   LED1 (ステータス LED) は、プログラミング/デバッグ モードのステータスを示し、KitProg3 の電源が入っているときにオンになります。
4. Figure 25 に示すように、**Run** メニューに移動し、**Debug Configurations** をクリックします。
5. Figure 26 に示すように、**GDB OpenOCD Debugging** で *<Application name>* **Attach (KitProg3_MiniProg4)** を選択し、**Debug** をクリックします。

これにより、プログラミングやリセットを行わずに実行中のターゲットにアタッチするデバッグ セッションが開始されます。デバッグ設定の詳細については、ModusToolbox™ user guide を参照してください。

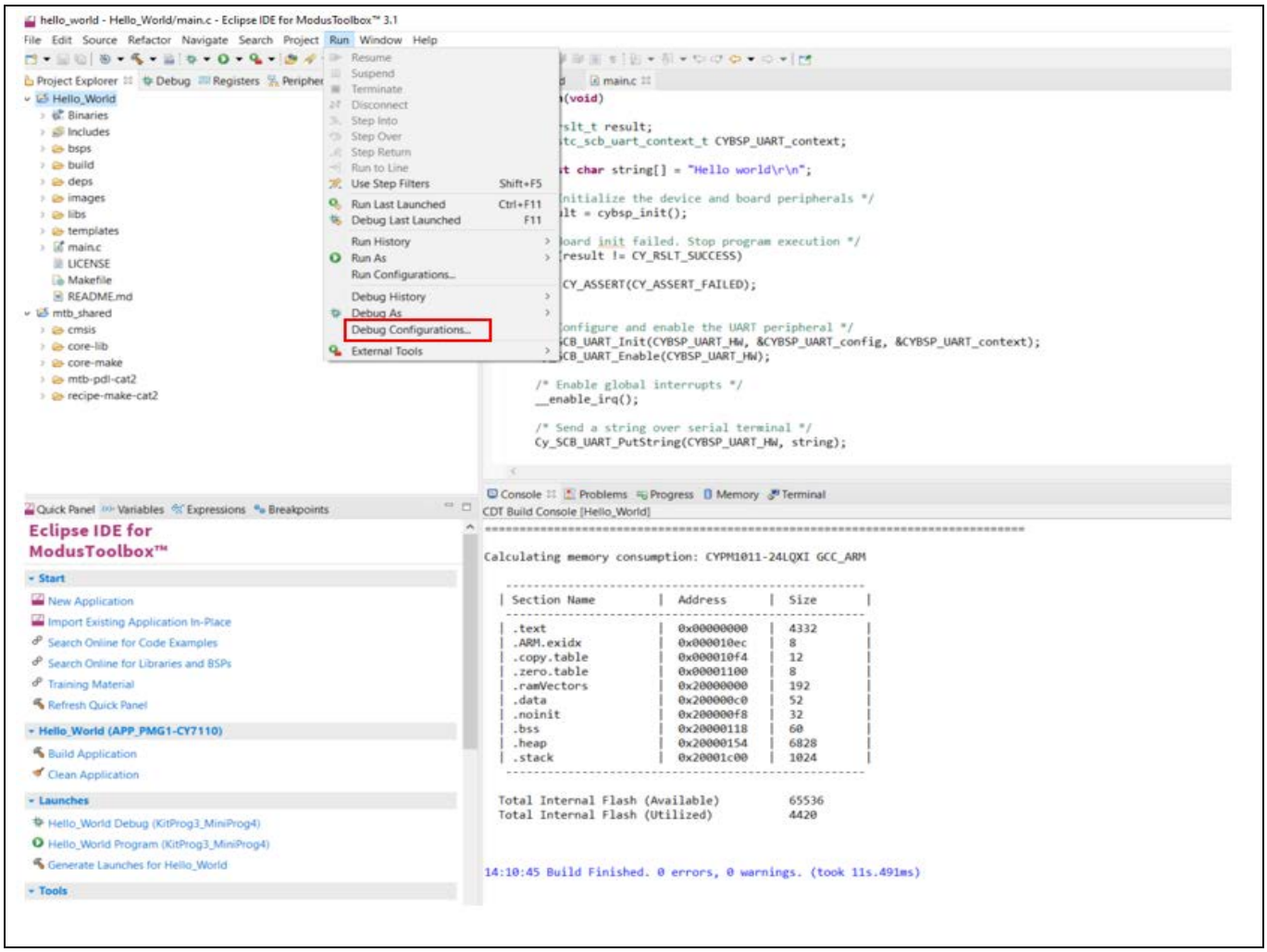


Figure 25　Run メニューから Debug Configurations の選択

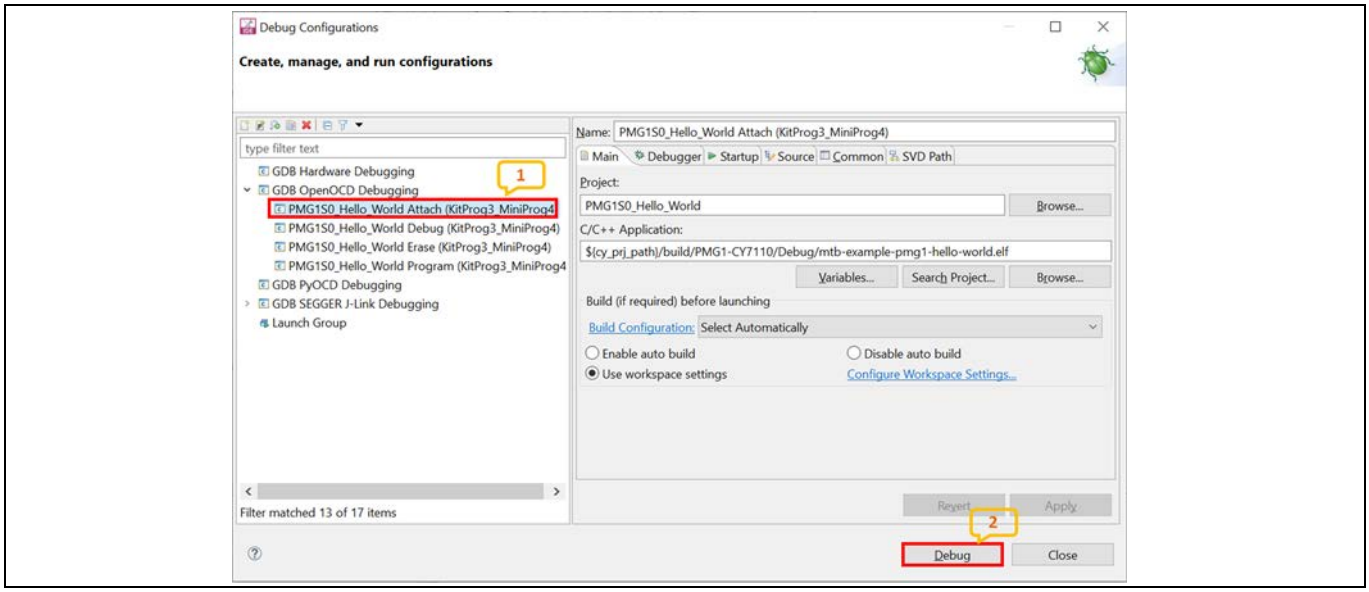


Figure 26　デバッグ アタッチの選択およびデバッグ セッションの開始

コンパイル, ビルド, およびデバッグの起動が正常に完了すると、ModusToolbox™ソフトウェアはキットをデバッグ モードに入れます。

3.8.1　10 ピン SWD ヘッダを介したデバッグ

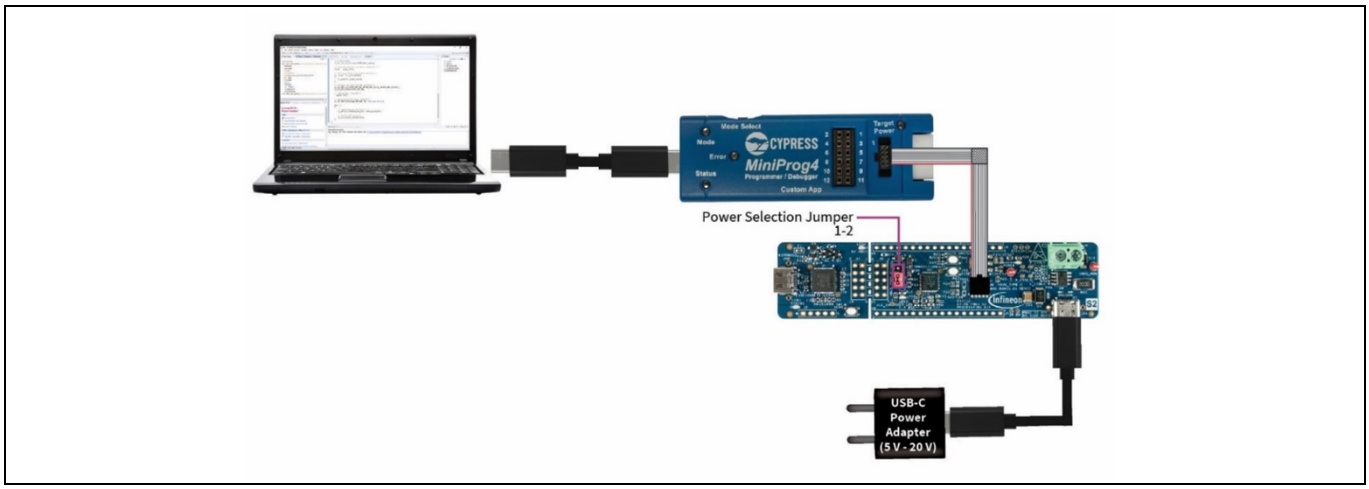


Figure 27　10 ピン SWD ヘッダを介したデバッグ

Figure 27 に、10 ピン SWD コネクタを介してプロトタイピング キットをデバッグするためのセットアップを示します。ModusToolbox™ソフトウェアの Eclipse IDE の使用手順は、KitProg3 インターフェースの手順と同じです。Debugging through the KitProg3 interface でリストされたステップ 4 - 5 に従います。

3.9　ミドルウェアの追加

ミドルウェアの初期化は *main.c* コードで実行されます。

1. **Quick Panel** で、**Library Manager** リンクをクリックします。
2. 続くダイアログで、**Add Library** タブを選択します。
3. **Core**, **Middleware**, **Peripheral**, および **Utilities**,で、必要なミドルウェアを選択して有効にします。

4. Figure 28 に示すように、**OK** と **Update** をクリックします。

ミドルウェアを使用するために必要なファイルは、*mtb_shared* フォルダに追加されます。

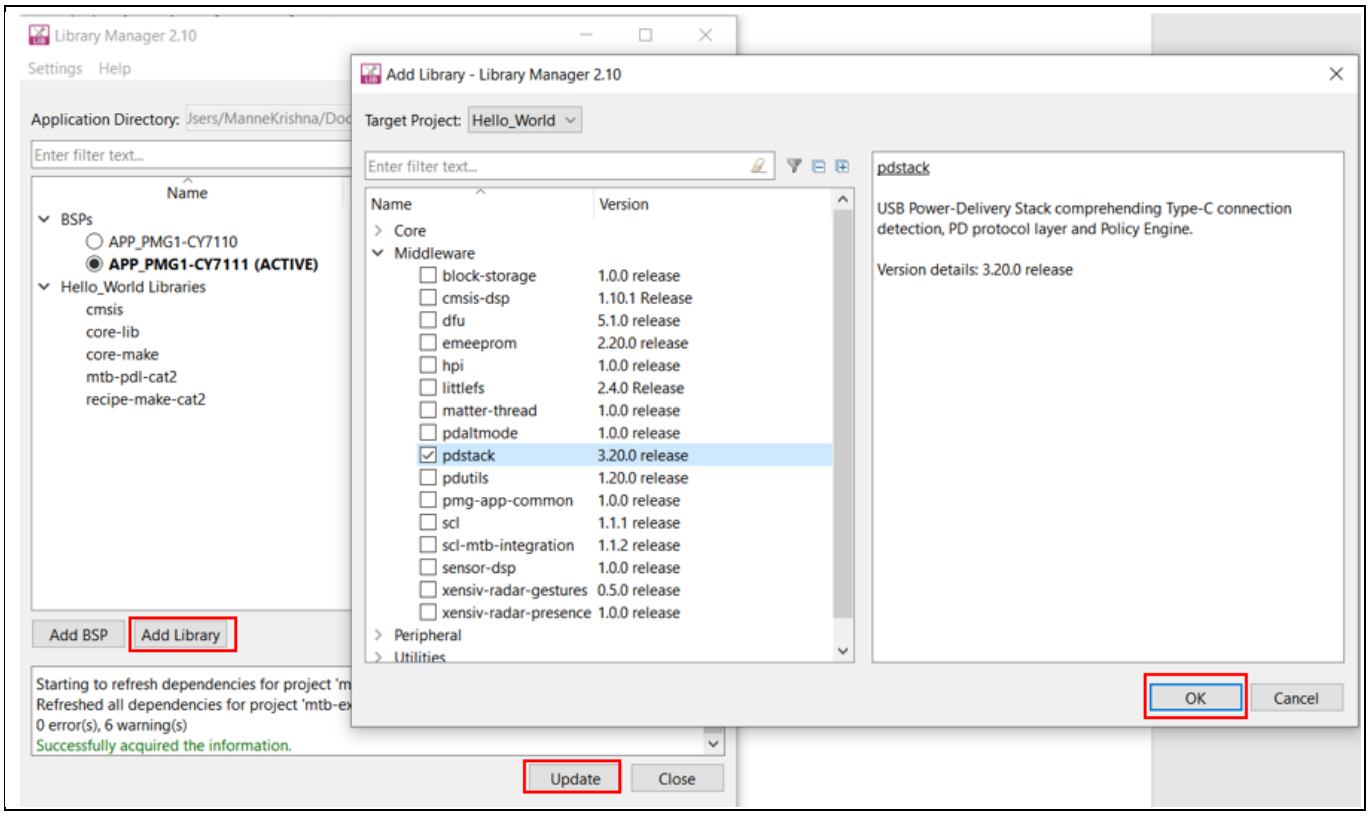


Figure 28　ミドルウェアの追加

4 USB-PD シンクの例

リストから必要な BSP と「USBPD_Sink」アプリケーションを選択すると、USB-PD シンクのサンプル コードが **Select New Application** オプションで利用できるようになります。Figure 29 を参照してください。

PMG1 MCU デバイスは、パートナー デバイスの接続を検出し、それらとパワー コントラクトをネゴシエートするために必要な Type-C 終端, コンパレータ, および電力供給トランシーバを統合する USBPD ブロックをサポートします。

このアプリケーションは、UFP (アップストリーム ポート) - シンク コンフィギュレーションで PDStack を使用します。PMG1 MCU デバイスにはデッド バッテリ Rd 終端があり、PMG1 MCU デバイスに電力が供給されていない場合でも、接続されている USB-C ソース/充電器がシンクの存在を検出できるようにします。

アプリケーションは、PMG1 MCU デバイスをディープ スリープ状態に維持しようとします。この状態では、すべてのクロックが無効になり、限られたハードウェア ブロックのみが有効になります。USBPD ブロックの割込みは、デバイスがスリープ状態のときに発生する変更を検出し、さらなる処理のためにデバイスをウェイクアップするように設定されています。

USBPD ブロック内の過電圧 (OV) コンパレータは、電源が誤った電圧レベルを供給している場合を検出し、ボード上の残りのコンポーネントを保護するために電源スイッチを自動的にシャットダウンするために使用されます。

README.md ファイルは、アプリケーションが正常に作成された後に開かれます。このファイルには、テンプレート アプリケーションの実装に関する情報が含まれます。

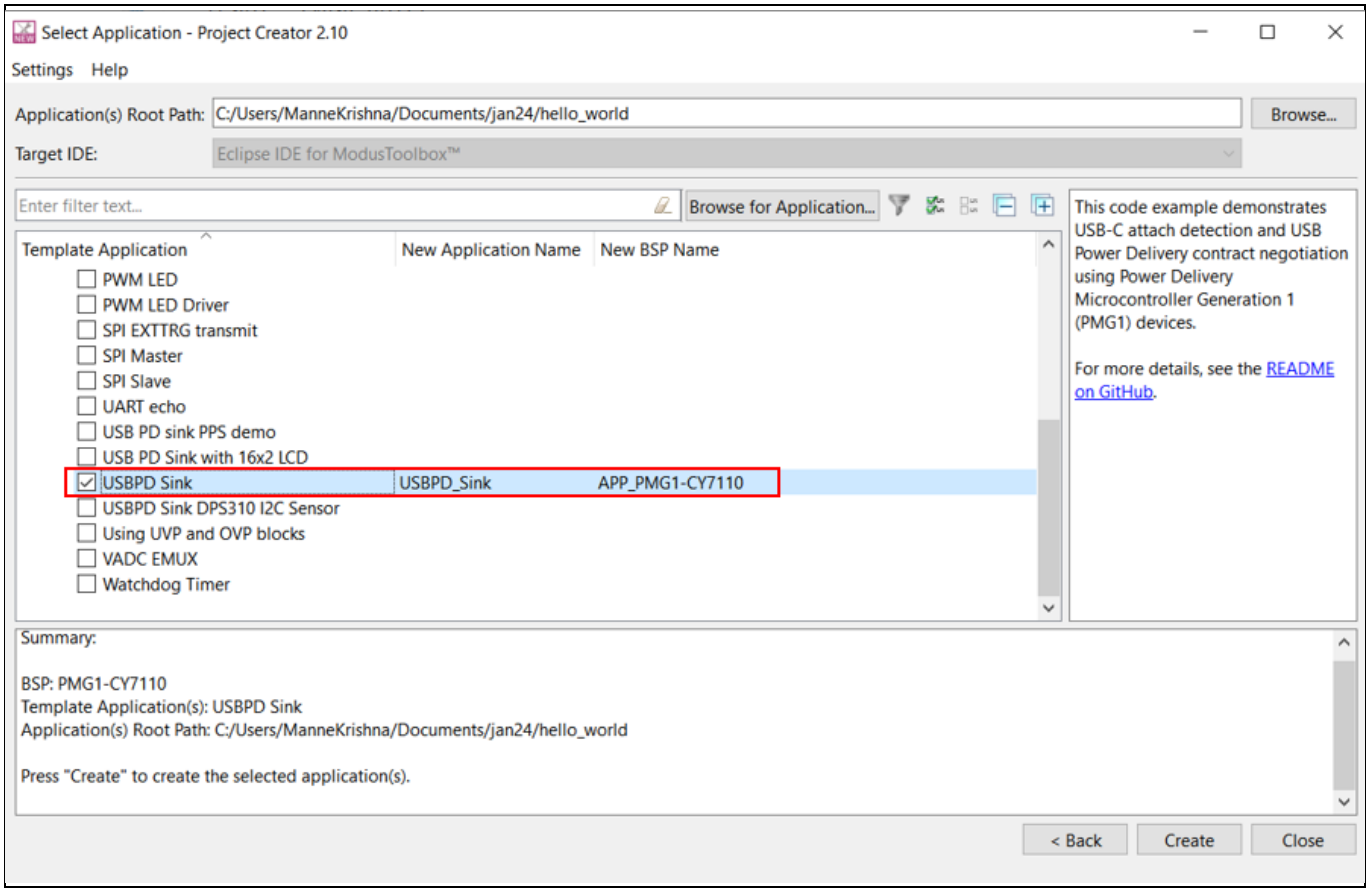


Figure 29 USBPD シンク CE の選択

5 PMG1 MCU 間でのプロジェクトの移植

1 つの MCU 用に作成されたプロジェクトは、PMG1 MCU ファミリの他の MCU で動作するように移植できます。移植する特長/機能/リソースが新しい MCU で利用可能であることを考慮してください。PMG1 MCU 間でプロジェクトを移植するには、以下の手順を参照してください。

1. Quick Link から Library Manager を開きます。
2. Library Manager で次の手順を実行します。

   a) **Add BSP** をクリックして新しい BSP を追加し、現在の BSP の選択を解除して削除し、アクティブな BSP を選択します。Figure 30 では、PMG1-CY7110 が選択解除され、**PMG1-CY7111** が選択されています。これは、PMG1 MCU の「Hello World」サンプル コードを PMG1-S1 MCU に移植していることを意味します。

Note: 「Hello World」サンプル コードをカスタム ハードウェアに移植するには、作成されたカスタム BSP を選択します。

   b) **Update** をクリックします。

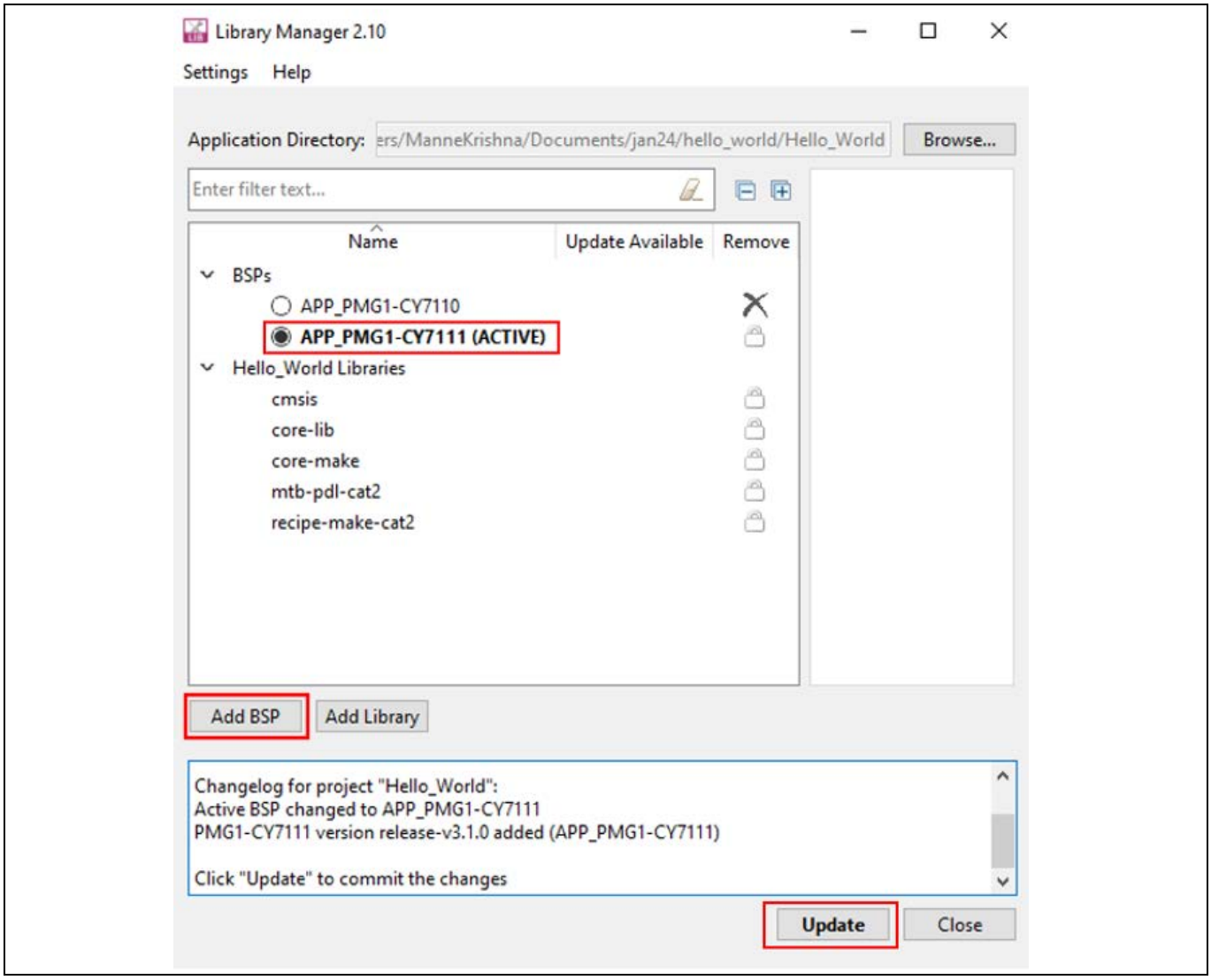


Figure 30 別の BSP へのプロジェクトの移植

3. Makefile で、ターゲット名が前の手順で選択した新しい BSP に変更されます。Figure 31 を参照してください。

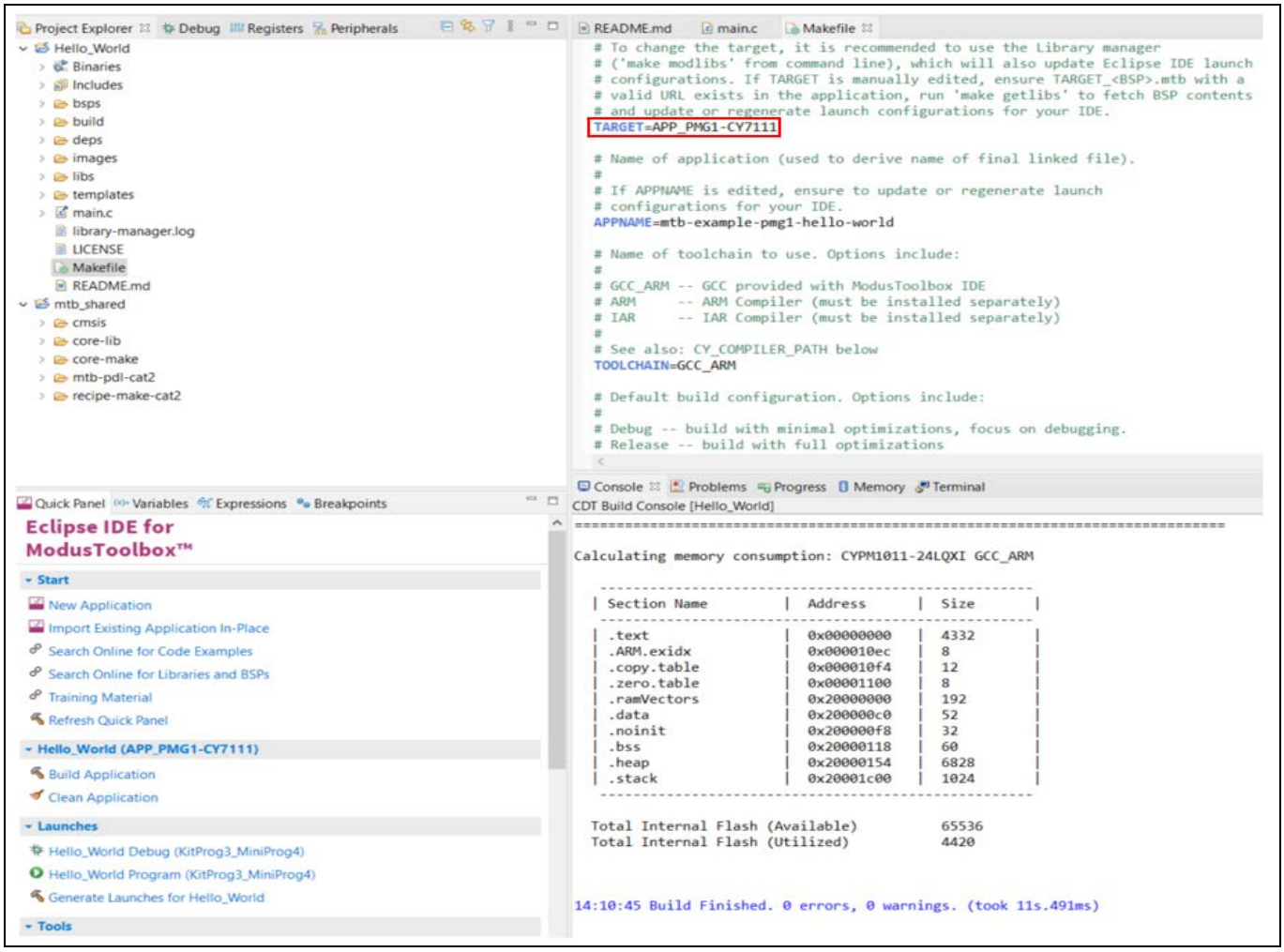


Figure 31　ターゲット名の編集

4. **Quick Panel** で次の手順を実行します。
   a) <project name> の **Generate Launches** をクリックします。Figure 32 を参照してください。
   b) **Refresh Quick Panel** をクリックします。

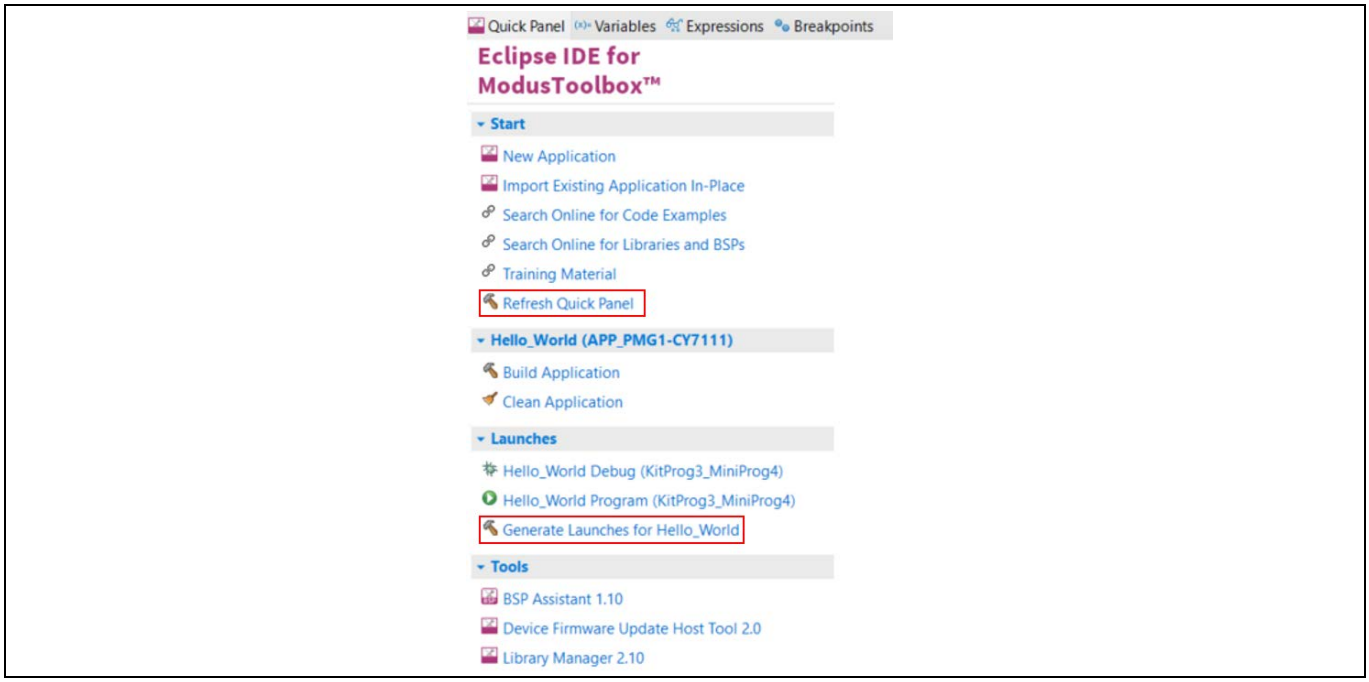


Figure 32　起動コンフィギュレーションの再生成

これで、Figure 33 に示すように、**Device Configurator** が新しい BSP に更新されます。

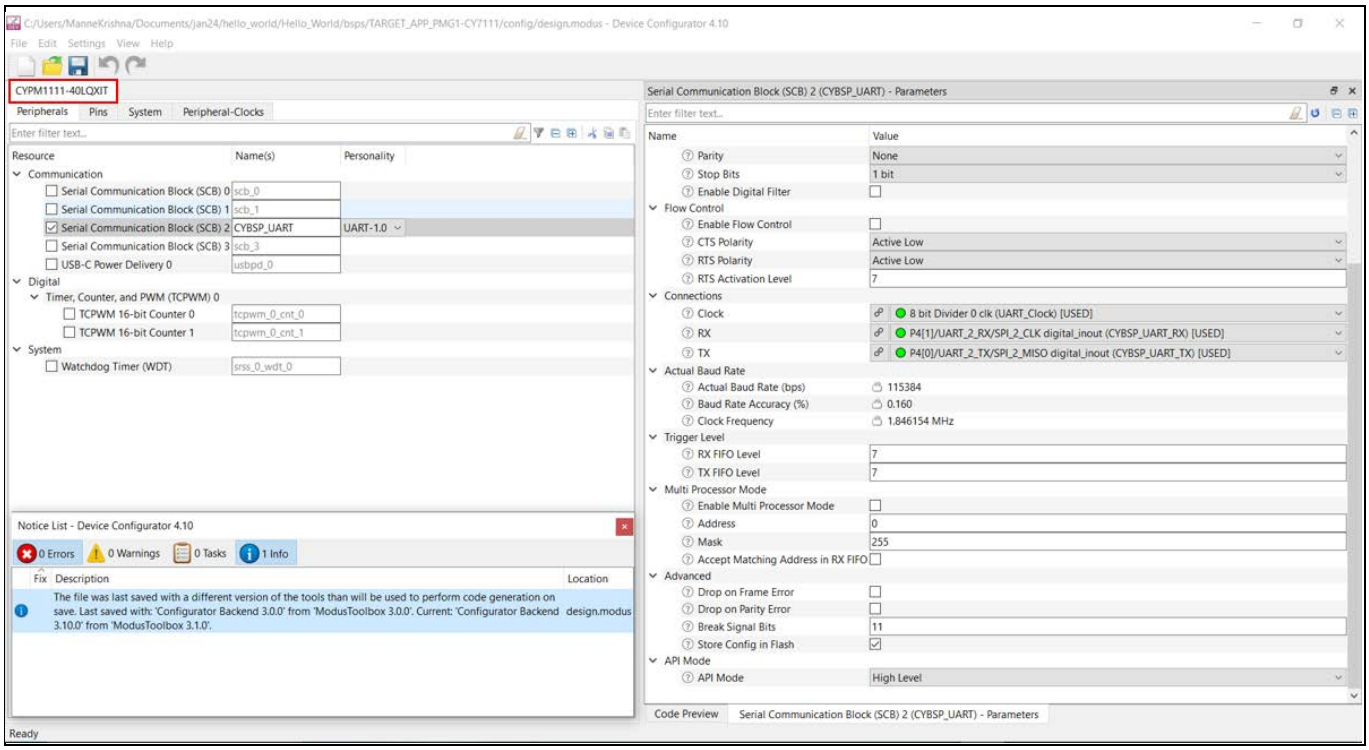


Figure 33　新しい BSP を備えた Device Configurator

5. ビルド, プログラム, およびプロジェクトのテストまで、残りの開発手順を続行します。

6 まとめ

このアプリケーション ノートでは、USB-C PD を備えた高電圧 MCU の EZ-PD™ PMG1 (Power Delivery Microcontroller Gen1) ファミリの機能について説明し、ModusToolbox™ソフトウェア用の Eclipse IDE で PMG1 MCU を使用した最初のプロジェクトを開始するのに役立ちます。

頭字語と略語

略語	説明
AC	Apple Charging
AES	advanced encryption standard (高度暗号化標準)
AFC	adaptive fast charging (アダプティブ急速充電)
BC	battery charging (バッテリ充電)
CDM	charged device model (デバイス帯電モデル)
CRC	cyclic redundancy check (巡回冗長検査)
HBM	human body model (人体モデル)
OCP	overcurrent protection (過電流保護)
OVP	overvoltage protection (過電圧保護)
OVT	overvoltage tolerant (過電圧耐性)
PRNG	pseudo random number generation (擬似乱数生成器)
RCP	reverse current protection (逆電流保護)。ソース コンフィギュレーションでのみサポート
SCP	short-circuit protection (短絡保護)。ソース コンフィギュレーションでのみサポート
SHA	secure hash algorithm (セキュア ハッシュ アルゴリズム)
TRNG	true random number generation (真の乱数生成)
UVP	undervoltage protection (低電圧保護)

用語集

ここでは、ModusToolbox™ソフトウェアで PMG1 MCU ファミリを操作する際によく出てくる用語について説明します。

ボード サポート パッケージ (BSP): BSP は、ボード固有のドライバおよびその他の機能を含むファームウェアのレイヤです。ボード サポート パッケージは、ボードを初期化し、ボード レベルのペリフェラルにアクセスできるファームウェア API のライブラリ セットです。

KitProg3: KitProg3 は、USB-I2C および USB-UART ブリッジ機能を備えたオンボードのプログラマ/デバッガです。KitProg3 は、ほとんどの PMG1 MCU 開発キットに統合されています。

MiniProg3/**MiniProg4**: プログラムおよびデバッグ キット。独自のハードウェアを開発している場合や KitProg3 が動作していない場合に、MCU のプログラミングやデバッグに役立ちます。

Personality (パーソナリティ): これは、機能に対してリソースが設定可能かを表します。例えば、SCB リソースは UART, SPI, または I2C パーソナリティに設定できます。

ミドルウェア: これは、アプリケーションに特定の機能を提供するファームウェア モジュールのセットです。デバイス機能 (PDStack など) への高レベルのソフトウェア インターフェースを提供します。

ModusToolbox™ソフトウェア: 開発者向けの Eclipse ベースの組み込み設計プラットフォームで、ドライバ ライブラリ, ミドルウェア, 最小の消費電力を組み合わせた、単一の一貫性のある使い慣れた設計エクスペリエンスを提供し、クラス最高のセンシングを備えた最も柔軟な MCU をサポートします。

ペリフェラル ドライバ ライブラリ: ペリフェラル ドライバ ライブラリ (PDL) は、PMG1 MCU アーキテクチャのソフトウェア開発を簡素化します。PDL により、レジスタの使用法とビット構造を理解する必要性が減るため、利用できる広範なペリフェラル セットのソフトウェア開発が容易になります。

改訂履歴

版数	発行日	変更内容
**	2021-12-09	これは英語版 002-32553 Rev. *A を翻訳した日本語版 Rev. **です。
*A	2022-08-01	これは英語版 002-32553 Rev. *B を翻訳した日本語版 Rev. *A です。
*B	2025-03-18	これは英語版 002-32553 Rev. *C を翻訳した日本語版 Rev. *B です。

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

Edition 2025-03-18
Published by
Infineon Technologies AG
81726 Munich, Germany

© 2025 Infineon Technologies AG. All Rights Reserved.

Do you have a question about this document?
Email: erratum@infineon.com

Document reference
002-34377 Rev. *B

重要事項
本手引書に記載された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。