PSoC™ Creator での PSoC™ 6 Bluetooth® LE 入門

本書について

適応範囲と目的

本アプリケーションノート(AN210781) では、Bluetooth® Low Energy (LE)接続機能を備えた PSoC™ 6 MCU をご紹介します。本 MCU は、Bluetooth® LE 5.0 システム、最新の CAPSENSE™技術および多数のセキュリティ機能を統合する、デュアル CPU Arm® Cortex®-M4 および Cortex®-M0+ベースのプログラマブルシステムオンチップです。本アプリケーションノートをとおして、開発者のお客様が Bluetooth® LE 接続搭載の PSoC™ 6 MCU のアーキテクチャおよび各種の開発ツールを探究できるほか、初めて PSoC™ Creator を使用してプロジェクトをビルドし、サードパーティ製統合開発環境(IDE) へエクスポートしてから、ファームウェア開発を継続する方法も示します。また、Bluetooth® LE 接続機能を備えた PSoC™ 6 MCU については、オンラインで利用可能なリソースもあります。PSoC™ 6 MCU デバイスファミリの使用については、AN221774 – PSoC Creator での PSoC™ 6 MCU 入門を参照してください。

PSoC™サンプル コードのリストにアクセスするには、サンプル コードのウェブ ページをご覧ください。PSoC™ビデオライブラリについてはここからご覧ください。

目次

本書について 1

目次 1

1 はじめに 4

1.1 事前準備 6

1.1.1 ハードウェア 6

1.1.2 ソフトウェア 6

2 開発エコシステム 7

2.1 PSoC™リソース 7

2.2 ファームウェア/アプリケーション開発 7

2.2.1 PSoC™ Creator 7

2.2.1.1 PSoC™ Creator ヘルプ 8

2.2.2 ペリフェラル ドライバー ライブラリ (PDL) 8

2.3 他の IDE のサポート 9

2.3.1 他の IDE を対象にした PSoC™ Creator の使用 9

2.4 RTOS サポート 12

2.5 デバッギング 12

2.6 CY8CKIT-062-BLE PSoC™ 6-BLE Pioneer Kit 13

2.7 CySmart ホスト エミュレーション ツールとモバイル アプリケーション 13

3 デバイスの特長 15

4 開発セットアップ 17

5 初めての Bluetooth® LE 搭載の PSoC™ 6 MCU 設計 19

5.1 インストラクションについて 19

5.2 設計にあたって 19

5.3 設計について 20

目次

5.4 パート 1: 最初からプロジェクトを新規に作成 20
5.5 パート 2: 設計の実装 26
5.6 パート 3: ソース コードの生成 43
5.7 パート 4: ファームウェアの記述 45
5.8 パート 5: プロジェクトのビルド、デバイスのプログラミング 57
5.9 パート 6: デザインをテストする 60

6 まとめ 66

7 関連アプリケーションノートとサンプル コード 67

A Appendix A. 用語集 69

B Appendix B. Bluetooth® LE プロトコル 70
B.1 概要 70
B.2 物理層 (Physical Layer; PHY) 70
B.3 リンク層 (Link Layer; LL) 70
B.4 ホスト制御インターフェース (Host Control Interface; HCI) 71
B.5 論理リンク制御および適応プロトコル (Logical Link Control and Adaptation Protocol ; L2CAP) . . 71
B.6 セキュリティ管理 (Security Manager; SM) 72
B.7 アトリビュート プロトコル (Attribute Protocol; ATT) 72
B.7.1 属性階層 73
B.7.2 属性の操作 75
B.8 汎用属性プロファイル (Generic Attribute Profile; GATT) 75
B.9 汎用アクセス プロファイル (Generic Access Profile; GAP) 76

C Appendix C. デバイスの特長 78
C.1 システム ワイド リソース 78
C.1.1 CPU サブシステム: CM4 および CM0 78
C.1.2 IPC 78
C.1.3 メモリ システム 78
C.1.4 DMA 78
C.1.5 クロック システム 78
C.1.6 システム割込み 79
C.1.7 電源と電圧監視 79
C.1.8 電力モード 80
C.2 セキュア ブート 81
C.3 プログラマブル デジタル ペリフェラル 81
C.3.1 UDB 81
C.3.2 プログラマブル TCPWM 82
C.3.3 SCB 82
C.3.4 BLESS 82
C.3.4.1 ディープ スリープ モード: 83
C.3.4.2 スリープ モード 83
C.3.4.3 アイドル モード 83

目次

C.3.4.4 送信モード . . . 83
C.3.4.5 受信モード . . . 83
C.3.5 オーディオ サブシステム . . . 84
C.3.6 シリアル メモリ インターフェース . . . 84
C.3.7 eFUSE . . . 84
C.3.8 セグメント LCD . . . 84
C.4 プログラマブル アナログ ペリフェラル . . . 84
C.4.1 連続時間ブロック オペアンプ . . . 84
C.4.2 低消費電力コンパレータ . . . 84
C.4.3 SAR ADC . . . 85
C.4.4 DAC . . . 85
C.4.5 CAPSENSE™ . . . 85
C.5 プログラマブル GPIO . . . 86

D Appendix D. IoT 開発ツール . . . 87
D.1 CY8CKIT-062-BLE PSoC™ 6-BLE Pioneer Kit . . . 87
D.2 CySmart ホスト エミュレーション ツール . . . 87
D.3 CySmart モバイル アプリケーション . . . 89

改訂履歴 . . . 91

免責事項 . . . 92

1 はじめに

Bluetooth® LE 接続機能を備えた PSoC™ 6 MCU (以下、「PSoC™ 6-BLE」) は、ウェアラブルおよび IoT 製品専用に設計されたサイプレスの超低消費電力 PSoC™デバイスです。これは、今日の「常時オン」アプリケーションのための新しい低電力規格を確立します。PSoC™ 6 BLE デバイスは、以下をシングルチップに統合するプログラム可能な組込みシステムオンチップです。

- デュアル CPU マイクロコントローラー: CM4 および CM0+
- Bluetooth® LE 5.0 サブシステム
- プログラム可能なアナログおよびデジタル周辺機器
- 最大 1 MB のフラッシュと 288 KB の SRAM
- 第 4 世代の CAPSENSE™技術

PSoC™ 6-BLE は、次のような電源に敏感なさまざまな接続アプリケーションに適しています。

- スマート ウォッチとフィットネス トラッカー
- コネクテッド医療機器
- スマートホームのセンサー/コントローラー
- スマート家電製品
- ゲーム コントローラー
- スポーツ、スマートフォン、バーチャル リアリティ (VR) アクセサリ
- 産業用センサー ノード
- 産業用ロジックコントローラー
- 高級リモートコントローラー

PSoC™ 6-BLE は、MCU と Bluetooth® LE 無線ブロックの組合せの、費用対効果が高く、フットプリントが小さい代替品です。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、PSoC™ 6-BLE アプリケーションを開発するための回路図ベースの設計ツールである PSoC™ Creator を使用して、設計の柔軟性および動的な微調整を実現できます。BLE アプリケーションを開発する際に、複雑な BLE プロトコル スタックの実用的な知識は必要ありません。

PSoC™ 6-BLE は、PSoC™ Creator で、プロトコルの複雑さを簡素化する、設定しやすい GUI ベースの Bluetooth® LE コンポーネントです。

Bluetooth® LE は、Bluetooth® Special Interest Group (SIG) によって策定された短距離通信向けの超低消費電力無線規格です。PSoC™ 6-BLE は、BLE 5.0 仕様準拠のセキュリティ、プライバシー、およびスループットを強化した Bluetooth® LE 4.2 無線ブロックとロイヤリティフリーのプロトコルスタックを統合します。

PSoC™ 6-BLE の CAPSENSE™という第 4 世代静電容量タッチ センシング機能は、比類のない信号対ノイズ比、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 タイプを提供しています。CAPSENSE™ユーザーインターフェイスは、活動モニターやヘルスケアおよびフィットネス機器などのウェアラブル電子機器に幅広く採用されています。CAPSENSE™ソリューションは、ノイズのある環境や液体が存在している状態でも正常に機能します。

PSoC™ 6-BLE は、統合されたソリューションで超低消費電力の接続アプリケーションを実現します。

図 1 に、PSoC™ 6-BLE に基づくフィットネス トラッカーのアプリケーション レベルのブロックダイヤグラムを示します。

1 はじめに

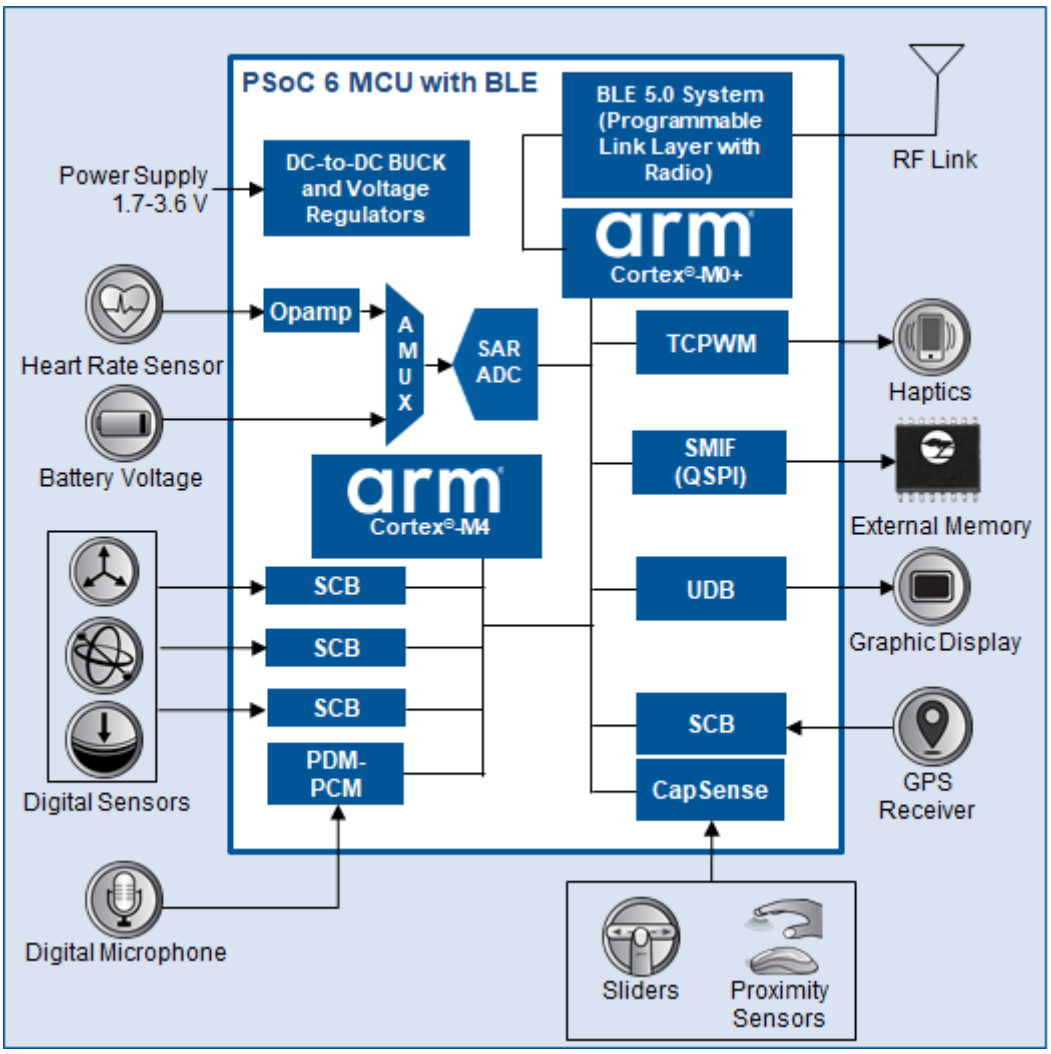


図 1　　フィットネス トラッカー アプリケーション ブロックダイヤグラム

このデバイスはワンチップソリューションを提供し、以下が含まれます。

- 最大 4 つの同時接続を維持できる低消費電力 Bluetooth® LE 5.0 システム
- 超低消費電力動作用の降圧変換器
- 心拍センサー出力を調整、測定し、バッテリ電圧を監視するための内蔵アナログ フロント エンド (AFE)
- グローバル ポジショニング システム (GPS) モジュールを含む複数のデジタル センサーとインターフェースするためのシリアル通信ブロック (SCB)
- パルス密度変調 (PDM)/パルスコード変調 (PCM) ハードウェア エンジンおよび音声用デジタルマイク インターフェース
- 信頼性のあるタッチおよび近接センシング機能を提供する CAPSENSE™技術
- クアッド シリアル ペリフェラル インターフェース (QSPI) 対応の外部メモリへのインターフェースをサポートするシリアル メモリ インターフェース (SMIF)
- ディスプレイとハプティクスデバイスを駆動するためのデジタルロジック (UDB) および周辺機器 (TCPWM)

デバイスの機能の詳細については、デバイスの特長と付録 C を参照してください。

本アプリケーションノートでは、PSoC™ 6-BLE の機能を紹介し、開発エコシステムの概要を示し、即時アラートサービス (Immediate Alert Service; IAS) に対応した Bluetooth® SIG 標準のファインド ミー プロファイル (Find Me Profile; FMP)の簡単な設計を始める手助けをします。この設計を、CY8CKIT-062-BLE 向けのサンプル コード CE212736 としても利用できます。本アプリケーションノートでは、このサンプルコードを幅広く使用しています。

ハードウェア設計上の考慮事項は、「AN218241 – PSoC™ 6 MCU Hardware Design Considerations」を参照してください。
高度なアプリケーション開発については、「アプリケーションノート AN215796 – Designing PSoC™ 6-BLE Applications」を参照してください。

1.1 事前準備

始める前に、開発キットを入手し、サンプル コードを含む必要なソフトウェアをインストールしてください。

1.1.1 ハードウェア

- CY8CKIT-062-BLE PSoC™ 6-BLE pioneer kit
- Windows 7 以降の PC (CySmart™ホスト エミュレーション ツール アプリケーションを使用する場合)
- Android 5/iOS 8 以降の携帯電話 (CySmart™ iOS/Android アプリを使用する場合)

1.1.2 ソフトウェア

- PSoC™ Creator 4.4 と PSoC™ Programmer 3.29
- CE212736 – CY8CKIT-062-BLE PSoC™ 6 BLE Pioneer Kit 向けの Bluetooth® LE ファインド ミー サンプルコード
- CySmart™ホストエミュレーションツール PC アプリケーションまたはアプリ CySmart™ iOS/Android

CySmart アプリをダウンロードするために、携帯電話で以下の QR コードをスキャンしてください。

iOS

Android

2 開発エコシステム

2.1 PSoC™リソース

www.cypress.com に大量のデータを掲載しており、ユーザーが適切な PSoC™デバイスを選択し、設計に迅速かつ効率的に統合する手助けをしています。以下は PSoC™ 6 MCU の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™ 6 MCU
- データシート: 各デバイス ファミリの電気的仕様を説明します。
- アプリケーションノートおよびサンプルコード:基本レベルから上級レベルまでの幅広いトピックを提供します。アプリケーションノートの多くにはサンプルコードを含んでいます。PSoC™ Creator 内部から直接サンプルコードのコレクションも閲覧できます。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します。
- PSoC™ 6 プログラミング仕様: PSoC™ 6 MCU デバイスの不揮発性メモリをプログラムするために必要な情報を提供します。
- CAPSENSE™設計ガイド: PSoC™デバイスを使用して静電容量タッチセンシングアプリケーションを設計する方法を学びます。
- **開発ツール**
- CY8CKIT-062-WiFi-BT PSoC™ 6 WiFi-BT Pioneer Kit は、WiFi および Bluetooth®接続とともに PSoC™ 62 シリーズ MCU をサポートする開発キットです。
- CY8CKIT-062-BLE PSoC™ 6 BLE Pioneer Kit は、PSoC™ 6-BLE 向けの使いやすく安価な開発プラットフォームです。
- **トレーニングビデオ:** PSoC™ 6 MCU の専用シリーズを含む、当社の製品およびツールに関するビデオトレーニングを提供します。

PSoC™ 6 MCU リソースの包括的なリストについては、サイプレスコミュニティの KBA223067 を参照してください。

2.2 ファームウェア/アプリケーション開発

PSoC™ Creator とペリフェラル ドライバー ライブラリ (PDL) は開発プロセスの中心です。

PSoC™ Creator はアプリケーションを構築するために、いくつかのデジタル/アナログ/システムのコンポーネントとファームウェアを組み合わせます。PSoC™ Creator を使用することによって、回路図上のコンポーネントの選択、配置、構成や C/アセンブリ ソース コードの記述、デバイスのプログラムとデバッグができます。

PDL は PSoC™ 6 MCU ファミリ用のソフトウェア開発キットです。ソース コードとして提供される PDL は、サポートされるデバイス用のファームウェア開発を容易にします。これにより、レジスタ セットを理解する必要がなくなり、迅速にファームウェアをカスタマイズして構築できます。PDL は PSoC™ Creator とサードパーティ製 IDE の両方をサポートします。

2.2.1 PSoC™ Creator

PSoC™ Creator は無料の Windows ベースの統合設計環境 (IDE) です。このツールにより、お客様は PSoC™ 6 MCU のハードウェアとファームウェア システムを同時に設計できます。図 2 に示すように、PSoC™ Creator を使用すれば、以下のことを実現できます。

手順

1. **File** > **Code Example...**メニューからサンプルコードのコレクションを閲覧
   a. デバイスファミリに基づいてサンプルをフィルタリングします。

b. **Filter by** オプションに基づいて提供されたサンプルのメニューから選択します。
c. ダウンロードボタンを使用してサンプルコードをダウンロードします。
d. 選択に基づいて新しいプロジェクトを作成します。

2. 100 以上のコンポーネントを含むライブラリを探索
3. コンポーネントをドラッグ &ドロップして、メイン設計ワークスペースでハードウェア システム設計を構築
4. コンポーネント データシートを確認
5. コンフィギュレーション ツールを用いてコンポーネントを設定
6. アプリケーションのファームウェアと PSoC™ハードウェアを相互設計

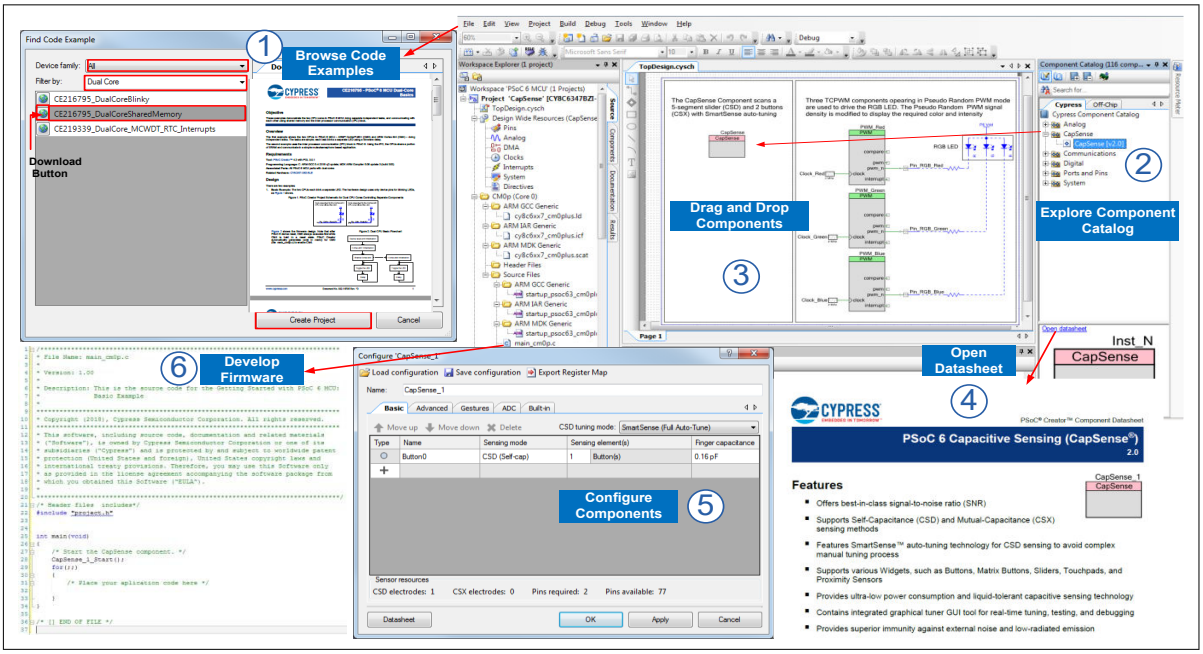


図 2　　PSoC™ Creator の回路図エントリ入力コンポーネント

2.2.1.1 PSoC™ Creator ヘルプ

PSoC™ Creator ホームページへアクセスして PSoC™ Creator の最新版をダウンロードしてインストールしてください。次に、PSoC™ Creator を起動し、以下の項目に移動します:

- **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。
- **サンプルコード**: **File** > **Code Example** を選択するか、**Start Page** タブの **Find Code Example...**リンクをクリックします。これらのサンプルコードは、PSoC™リソースを設定および使用方法を示します。
- **コンポーネントデータシート**: コンポーネントを右クリックし、**Open Datasheet** を選択します。すべてのコンポーネントデータシートの一覧は、PSoC™ 6 MCU コンポーネントデータシートページをご利用ください。

2.2.2 ペリフェラル ドライバー ライブラリ (PDL)

PDL は、PSoC™ 6 MCU デバイス用のソフトウェア開発キットです。PSoC™デバイスでの作業経験はあるものの、PSoC™ 6 MCU を初めて使用する場合、PDL が開発エコシステムへの主要な追加機能であることがわかります。

レジスタ レベルでのファームウェア開発者であっても、PDL をインストールする必要があります。PDL には、プロジェクトに必要なすべてのデバイス固有ヘッダ ファイルとスタートアップ コードが含まれています。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 レベルでどのようにハードウェアにアクセスするのかが分かります。

一部のデバイスは特定のペリフェラル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設計は以下のことを意味します。

- ペリフェラルを初期化、設定、利用するために必要なすべての API があります。
- PDL は使用可能なペリフェラルに関係なく、すべての PSoC™ 6 MCU デバイスで有効です。
- PDL は対象となるペリフェラルが選択されたデバイスに存在することを保証するためのエラーチェックを含んでいます。

これにより、ペリフェラルが利用可能である限り、プラットフォームにわたってコードの互換性が維持されます。デバイス ヘッダ ファイルは、デバイスで利用できるペリフェラルを指定します。サポートされていないペリフェラルを使用するコードを記述すると、コンパイル時にエラーが発生します。ペリフェラルを使用するコードを書く前に、そのペリフェラルのサポートを確認するためにそのデバイスのデータシートを参照してください。

PSoC™ Creator は、PSoC™ 6 MCU デバイスで使用する PDL に基づいたコンポーネントを提供します。これは、コミュニティが開発および事前検証したコンポーネントを利用する PSoC™ Creator の特長を維持します。しかし、PDL は任意の開発環境で使用できるソース コード ライブラリです。

PDL には以下の重要なソフトウェア リソースが含まれています。

- 各ペリフェラル ドライバーのヘッダとソース ファイル
- ミドルウェアライブラリ用のヘッダとソースファイル
- デバイス固有のヘッダ、スタートアップ、およびコンフィギュレーション ファイル
- サポートされるサードパーティ製 IDE 用テンプレート プロジェクト
- ドキュメントについて

  ドキュメントの場所は*<PDL install directory>\doc*です。2 つの重要なドキュメントがあります。

  PDL v3.x User Guide は、以下のような PDL の使用に関する基礎知識について説明します。

  - PDL を使用したカスタム プロジェクトの作成 (サードパーティ製 IDE を含む)
  - ペリフェラルの設定
  - ファームウェアでのピン管理
  - レジスタ ベース プログラミングの学習ツールとして PDL の使用
  - PDL API リファレンス ドキュメントの使用

  次に重要なドキュメントは、PDL 3.x API reference Manual.html です。このリファレンス ドキュメントには、PDL のすべてのドライバーに関する完全な情報が記述されています。これには、概要、構成時に考慮すべき事項と、すべての関数、マクロ、データ構造、および列挙型の詳細などが含まれています。

2.3 他の IDE のサポート

IAR Embedded Workbench などのお気に入りの IDE を使用しても PSoC™ 6 MCU のファームウェアを開発できます。PSoC™ Creator を使用してシステムを設計し、構成コードを生成してターゲット IDE にエクスポートすることにより、別の IDE で PDL を使用できます。

詳細については、AN219434 – PSoC™ 6 MCU Importing Generated Code into an IDE を参照してください。

2.3.1 他の IDE を対象にした PSoC™ Creator の使用

PSoC™ Creator は PSoC™ 6 MCU システムリソースとペリフェラルを設定および構成します。プロジェクトをご使用の IDE にエクスポートし、その IDE でファームウェアの開発を継続します。デバイス コンフィギュレーションに変更がある場合は、PSoC™ Creator で Topdesign 回路図を編集し、対象の IDE 用のコードを再生成します。図 3 にこのワークフローを示します。

2 開発エコシステム

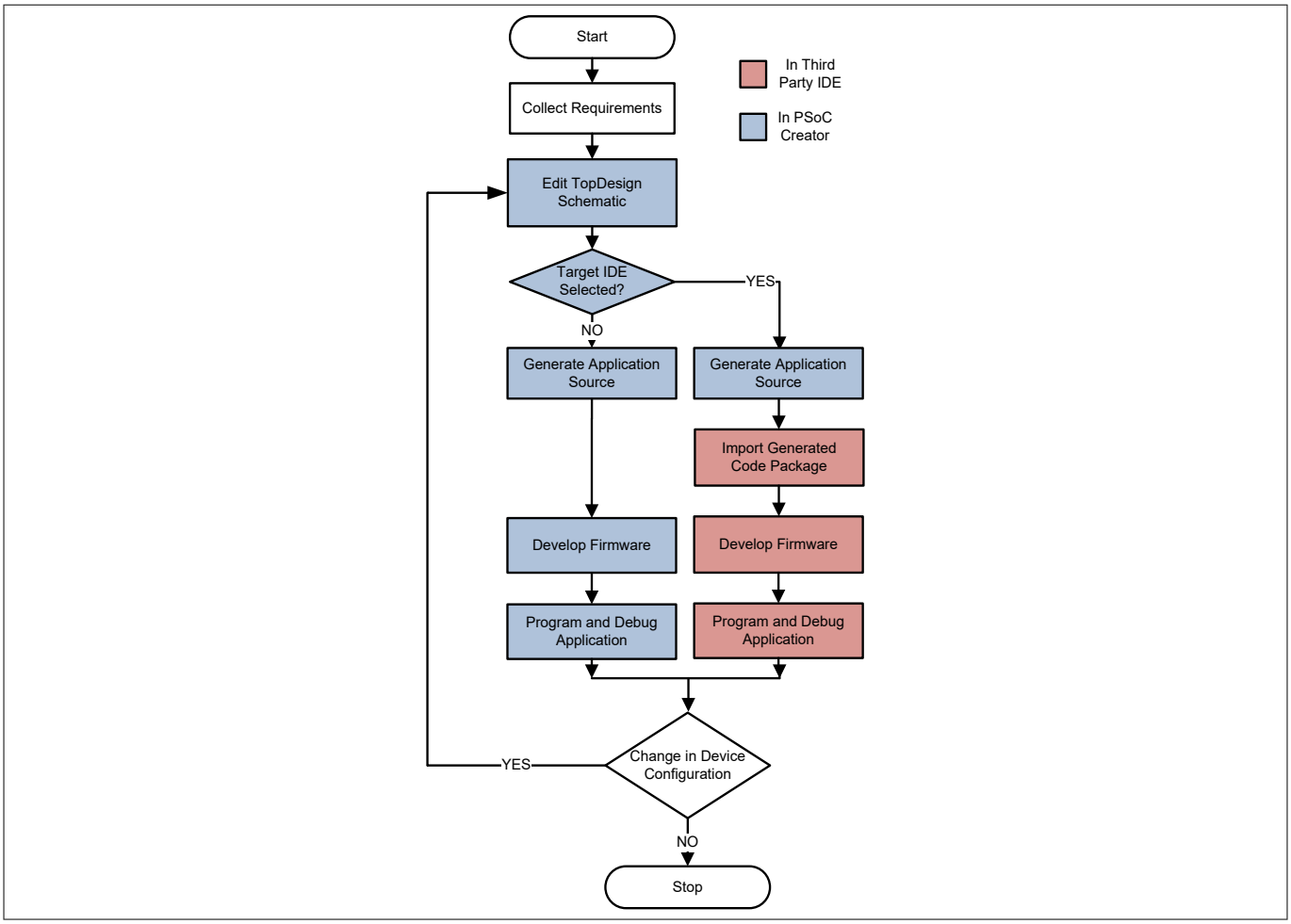


図 3　　PSoC™ Creator を使用した PSoC™ 6 MCU アプリケーション開発フローチャート

PSoC™ Creator から生成されたコードには、設計に基づく必要なヘッダ、スタートアップ、および PDL ファイルのすべてが含まれます。PSoC™ Creator からの生成コードのエクスポートは、Keil µVision、IAR Embedded Workbench、Eclipse ベース IDE、および GNU ベース コマンド ライン ツールでサポートされています。対象の IDE は、**Project** > **Build Settings** > **Target IDEs** のパネルで選択します。図 4 に示すように、対象の IDE のエクスポート パッケージを有効にします。コードを生成すると、PSoC™ Creator は対応するエクスポート パッケージも作成します。

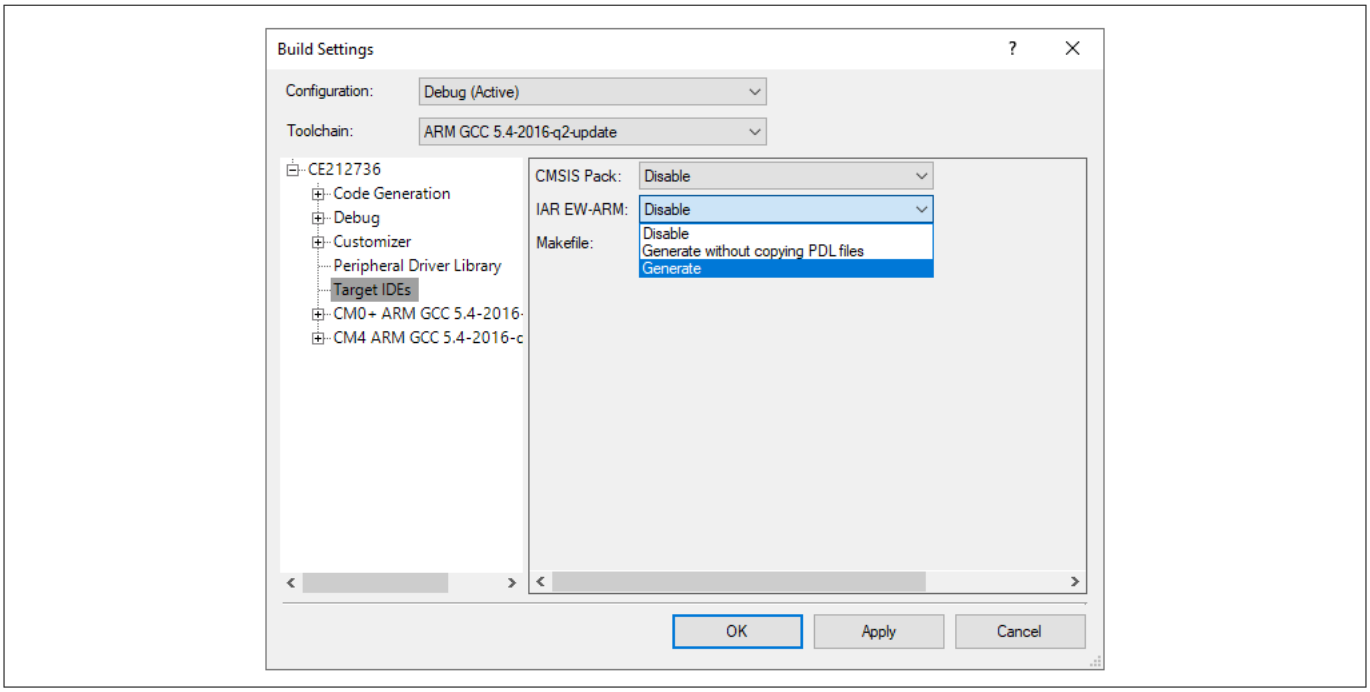


図 4　　Project > Build Settings で対象の IDE を選択

その後、パッケージをご使用の IDE にインポートします。IDE によっては、インポートの詳細は大きく異なります。実行するプロセスを確認するために、**Help** > **PSoC Creator Help Topics** を選択して、PSoC™ Creator ヘルプを参照してください。図 5 を参照してください。「*Integrating into 3rd Party IDEs*」トピックでは、各 PSoC™デバイスファミリおよびサポートされている各 IDE の特定のヘルプファイルを紹介しています。

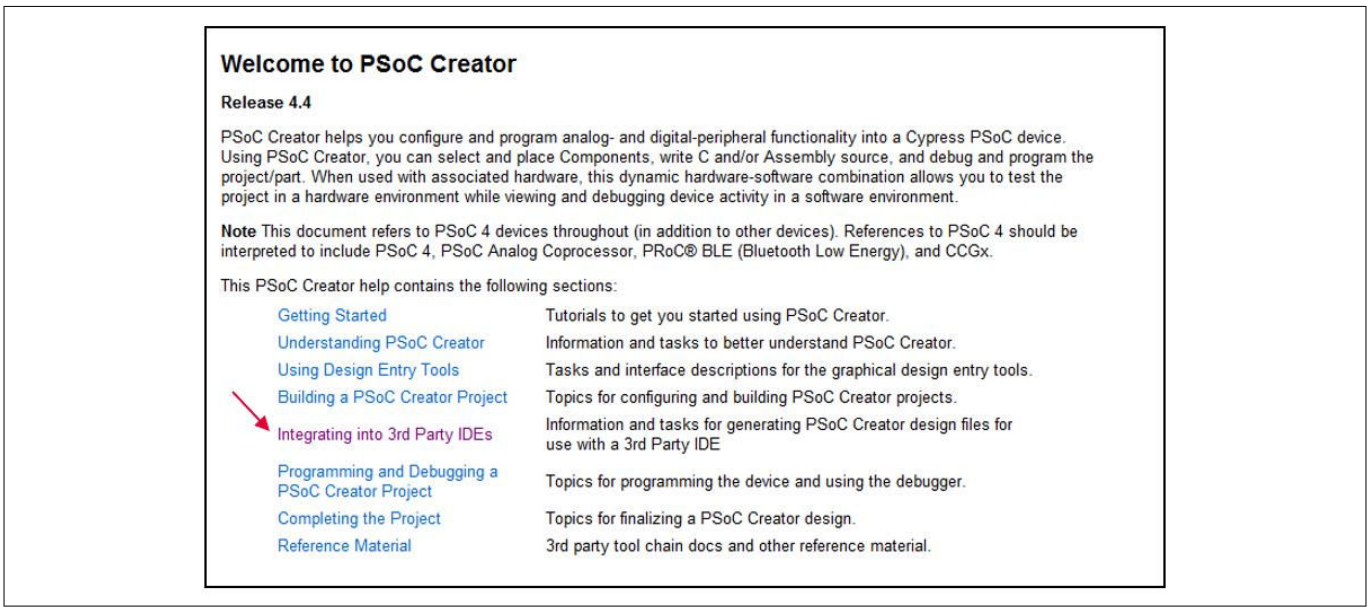


図 5　　PSoC™ Creator ヘルプ

ほとんどの IDE で効果的に作業できます。「Target IDEs」パネルでご使用される IDE がサポートされていない場合でも、PSoC™ Creator は引き続き使用できます。コードの生成後、ご使用の IDE のプロジェクトに必要なファイルを直接追加してください。AN219434 – PSoC™ 6 MCU Importing Generated Code into an IDE は、生成されたコードを別の IDE に手動でインポートするための詳細な手順を示します。

2.4 RTOS サポート

PDL には PSoC™ 6 MCU 開発用の RTOS サポートが含まれています。FreeRTOS ソースコードは PDL に完全に統合され、組み込まれています。PSoC™ Creator RTOS インポートオプションを使用して、FreeRTOS ソフトウェアパッケージをプロジェクトにインポートできます。図 6 に示すように、**Project** > **Build Settings** メニューに移動し、**Peripheral Driver Library** > **FreeRTOS** の Software package imports オプションから FreeRTOS を選択します。

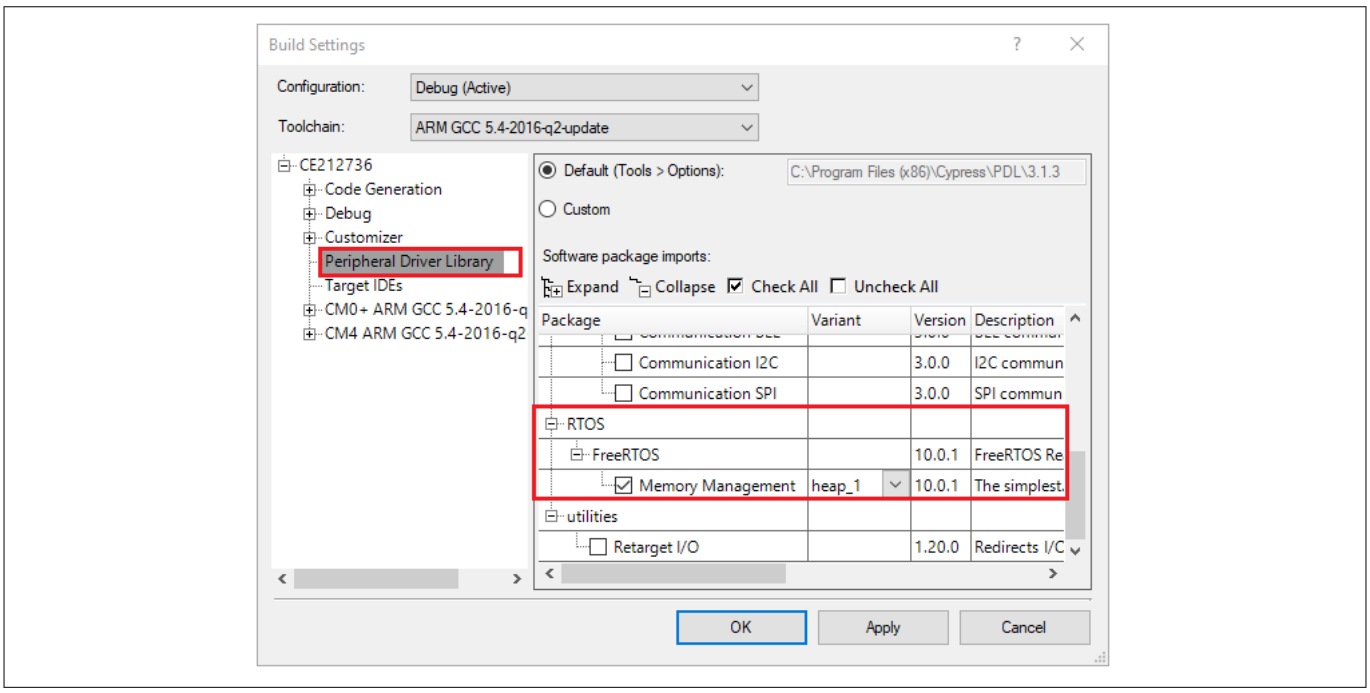


図 6 PSoC™ Creator のプロジェクトへの FreeRTOS のインポート

優先される RTOS がある場合は、そのコードを PDL と統合する方法の例として提供されているリソースを使用してください。

2.5 デバッギング

CY8CKIT-062-BLE PSoC™ 6-BLE Pioneer Kit には KitProg2 オンボード プログラマー/デバッガが含まれています。

注: PSoC™ Programmer 3.29 を使用して、オンボードプログラマ/デバッガを KitProg3 に更新してください。KitProg3 は、業界標準の CMSIS-DAP を転送機構として使用しています。KitProg3 は、USB Bulk エンドポイントを実装し、より高速な通信を実現しています。また、HID エンドポイントが必要なユースケースには対応していますが、通信速度が遅くなります。KitProg3 は、すぐに Bulk エンドポイントを使用します。KitProg3 はブリッジング (USB-UART, USB-I2C および USB-SPI) もサポートしています。これにより、PSoC™ 6 MCU Pioneer Kit のデバッグが非常に柔軟になります。詳細については、KitProg3 ユーザーガイドを参照してください。

PSoC™ Creator は、一度に 1 つの CPU (CM4 または CM0+) のデバッグをサポートします。一部のサードパーティ IDE は、マルチ CPU デバッグをサポートしています。PSoC™ Creator を使用した PSoC™デバイスのデバッグの詳細については、PSoC™ Creator ヘルプを参照してください。オンボードプログラマ/デバッガを Kitprog3 にアップデートするためには、キットをプラグインして開いてください。

PSoC™ Programmer 3.29 ツール。このツールは、キット上の KitProg2 または KitProg3 の旧バージョンを自動的に検出し、図 7 および図 8 に示すように、KitProg ファームウェアを KitProg3 の新バージョンにアップグレードするよう示します。

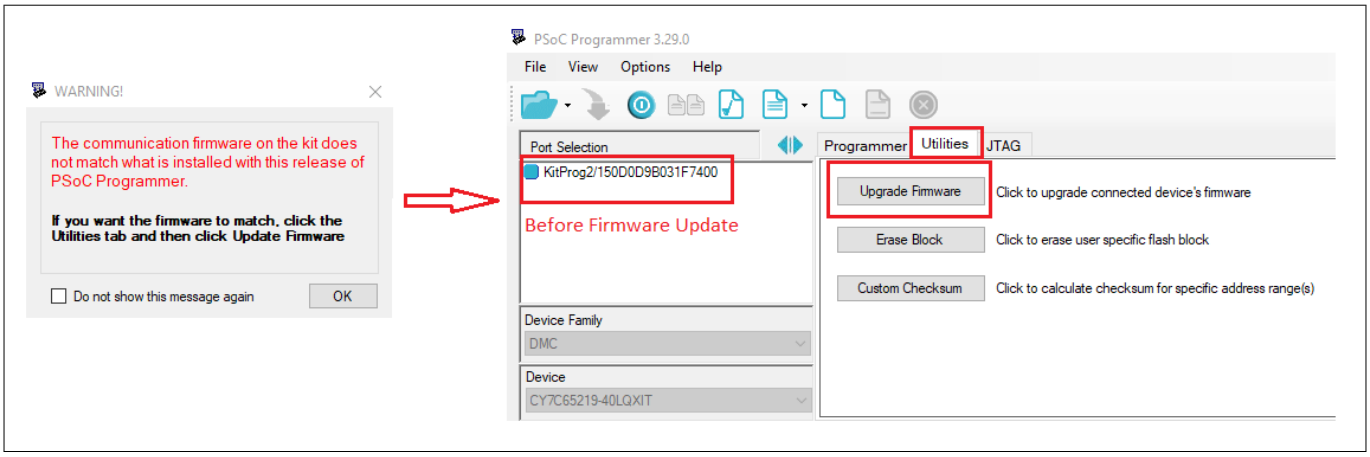


図 7　　KitProg ファームウェアのアップデート

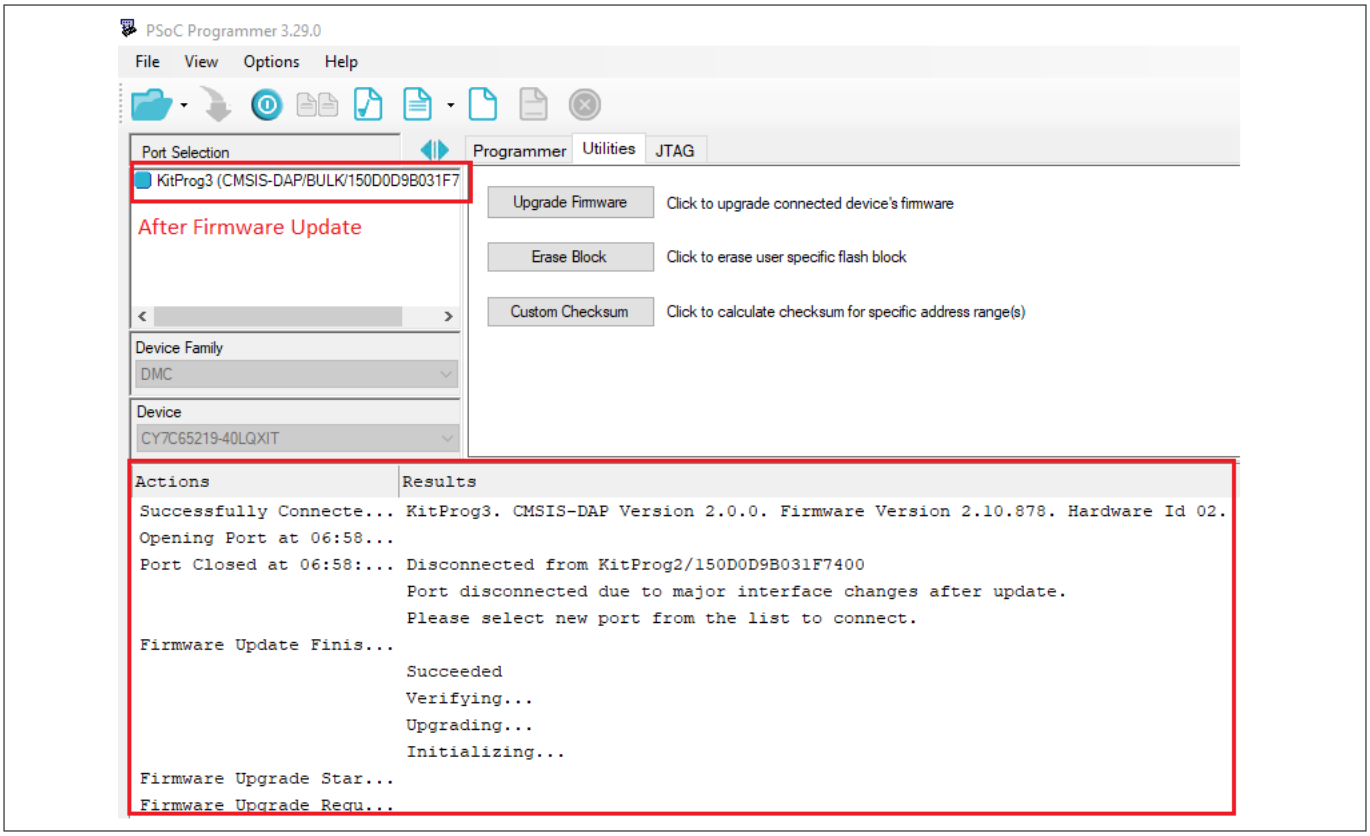


図 8　　ファームウェアのアップデート完了

2.6　CY8CKIT-062-BLE PSoC™ 6-BLE Pioneer Kit

付録 D.1 は、PSoC™ 63BL ファミリのデバイスをサポートするインフィニオンの Bluetooth® LE 開発キットです。詳細については、付録 D の CY8CKIT-062-BLE PSoC™ 6-BLE Pioneer Kit を参照してください。

2.7　CySmart ホスト エミュレーション ツールとモバイル アプリケーション

CySmart ホストエミュレーションツールは、PSoC™ 6-BLE Pioneer Kit の Bluetooth® LE ドングルを使用して Bluetooth® LE セントラル デバイスをエミュレートする Windows アプリケーションです。詳細については、付録 D

の付録 D.2 を参照してください。同様の機能は、iOS および Android デバイスの付録 D.3 で利用できます。このツールは、Bluetooth® LE アプリケーションのテストに非常に役立ちます。

3 デバイスの特長

図 9 に示すように、PSoC™ 6-BLE デバイスには豊富な機能セットがあります。主な機能の一覧は以下のとおりです。詳細はデバイスのデータシート、テクニカル リファレンス マニュアル (TRM)、および関連アプリケーションノートとサンプル コードを参照してください。

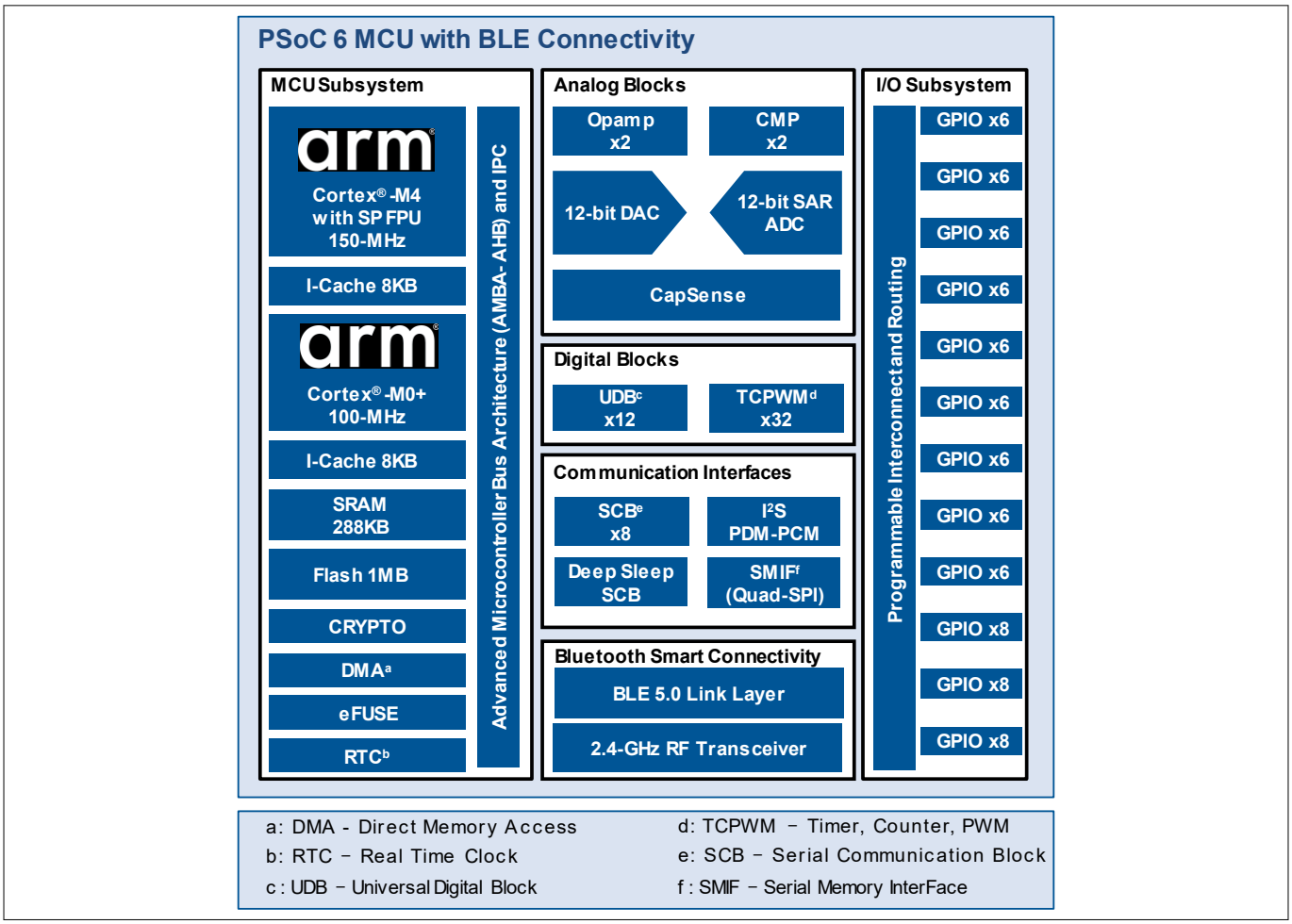


図 9 PSoC™ 6 MCU と Bluetooth® LE 接続のブロックダイヤグラム

- 32 ビットデュアル CPU MCU サブシステム
  - 単精度浮動小数点ユニットを備えた 150 MHz CM4 CPU 100MHz CM0+ CPU
  - 最大 1 MB のユーザーフラッシュ、EEPROM エミュレーション用の追加 32 Kb と監視用 32 KB 付き
  - 最大 288 KB の SRAM、32 KB の保持境界でディープスリープの保持粒度を選択可能
  - ハードウェアでサポートされるプロセッサ間の通信
  - ハードウェアアクセラレーションと真の乱数ジェネレーター機能をサポートする暗号化アクセラレーター
  - 2 つの 32 チャネル DMA コントローラー
  - eFUSE:ワンタイムプログラマブルビット
  - ハードウェア ハッシュ ベース認証を採用する割込み無しのセキュア ブート
- I/O サブシステム
  - アナログ、デジタル、CAPSENSE™、またはセグメント LCD 機能に使用できる最大 78 個の GPIO
  - プログラム可能な駆動モード、駆動強度およびスルー レート
  - ブール演算が実行できるスマート I/O を備えた 2 つのポート

3 デバイスの特長

- プログラマブル アナログ ブロック
  - プログラマブル ゲイン アンプ (PGA)、コンパレータまたはフィルターに設定可能な 6MHz ゲイン帯域幅 (GBW) を持つ 2 個のオペアンプ
  - ディープスリープモードとハイバネートモードで動作する、消費電流が 300nA 未満の 2 つの低消費電力コンパレータ
  - 16 チャネル シーケンサを備えた 1 個の 12 ビット 1Msps SAR ADC
  - 1 個の 12 ビット電圧モード DAC
- SmartSense™自動チューニングを備えた CAPSENSE™
  - CAPSENSE™シグマデルタ (CSD) および CAPSENSE™送信/受信 (CSX) をサポート
  - クラス最高の SNR、耐液性、および近接センシングを提供
- プログラマブル デジタル ブロック、通信インターフェース
  - カスタムデジタルペリフェラル用の 12 個の UDB
  - 16 ビット/32 ビットタイマー、カウンター、PWM、または直交デコーダーとして構成可能な 32 個の TCPWM ブロック
  - I^2C マスターまたはスレーブ、SPI マスターまたはスレーブ、または UART として構成可能な 9 つの SCB
  - 1 つの I^2S インターフェースと 2 つの PDM チャネルを備えたオーディオサブシステム
  - 外部クアッド SPI フラッシュ メモリからの直接実行 (execute-in-place) とオンザフライの暗号化および復号化をサポートする SMIF インターフェース
- Bluetooth® Low Energy 4.2 準拠の Bluetooth®接続機能
  - 集積バランを備えた 2.4G Hz Bluetooth® LE 無線通信
  - Bluetooth® 4.2 準拠のコントローラーとホストを実装
  - リンク層エンジンはマスター/スレーブ モードと最大 4 つの同時接続をサポート
  - 2 Mbps LE データ速度をサポート
- 動作電圧範囲、電源ドメイン、低消費電力モード
  - デバイスの動作電圧範囲: 1.71 V～3.6 V
  - ユーザー選択可能なコア ロジック動作 (1.1 V または 0.9 V)
  - 複数のオンチップ レギュレータ:低ドロップ アウト (アクティブ、ディープ スリープ モード用 LDO)、シングル入力マルチ出力 (SIMO) 降圧変換器
  - 細かい電力管理用のアクティブ、低消費電力アクティブ、スリープ、低消費電力スリープ、ディープ スリープ、およびハイバネートのモード
  - Bluetooth® LE リンクが動作するディープ スリープ モード: 3.3 V かつ 64 KB の SRAM 保持が有効な場合、Typ 電流は 4.5µA
  - 組込み RTC、電源管理集積回路 (PMIC) 制御、および制限つき SRAM バックアップを備えた「常時オン」のバックアップ電源ドメイン

4 開発セットアップ

図 10 に、PSoC™ 6-BLE デバイスを用いて Bluetooth® LE ペリフェラル設計を評価するために必要なハードウェアとソフトウェア ツールを示します。一般な使用事例では、PSoC™ 6-BLE Pioneer Kit (図 10 の青色の基板) は、CySmart iOS/Android アプリまたは CySmart ホスト エミュレーション ツールなどのセントラル デバイスと通信できるペリフェラルとして設定されます。CySmart ホストエミュレーションツールの操作には、Bluetooth® LE ドングル (図 10 の黒色の基板) も必要です。PSoC™ 6-BLE Pioneer Kit はドングルを含みます。

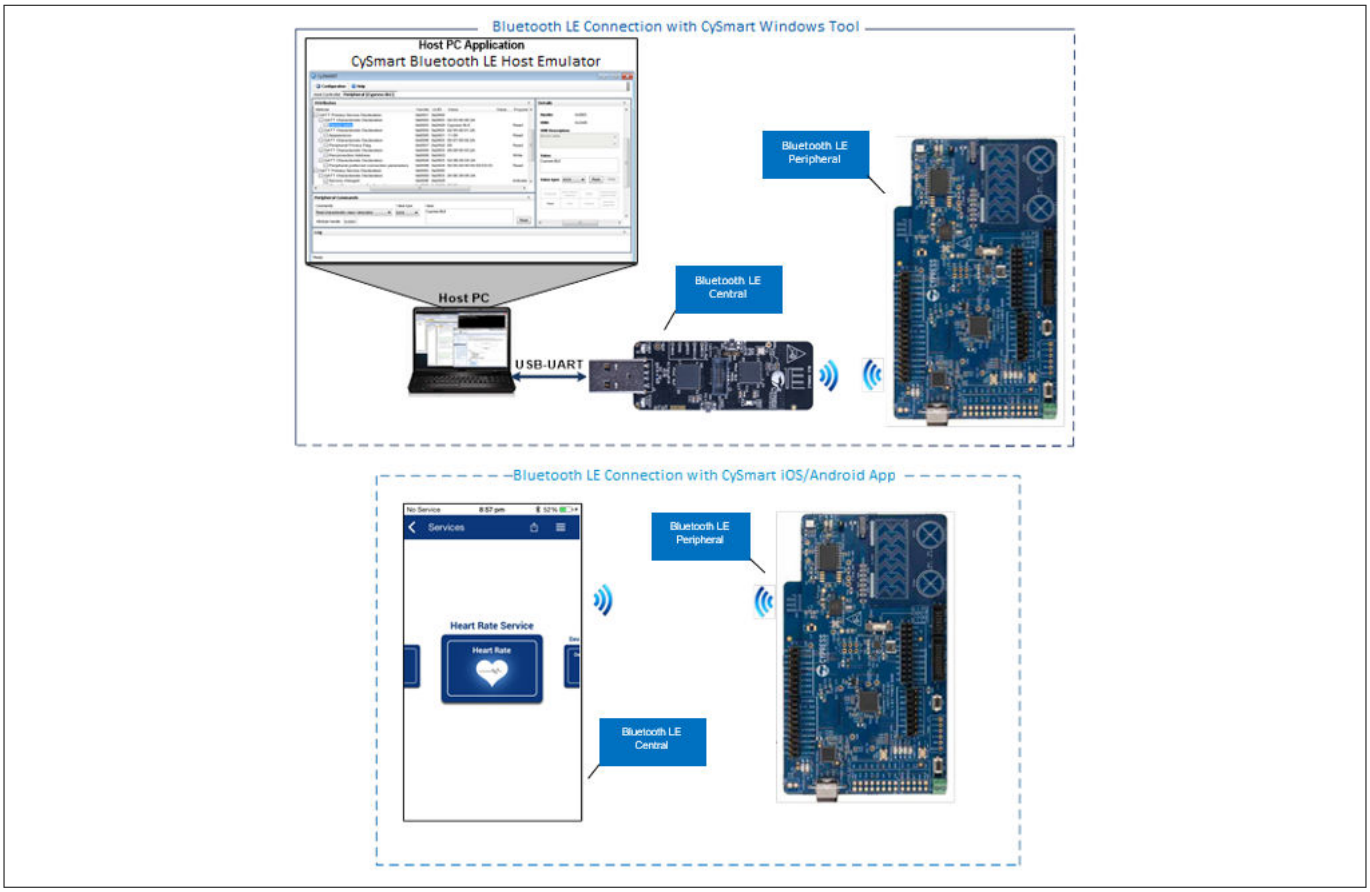


図 10 Bluetooth LE 機能的セットアップ

図 11 に示すように、Bluetooth® LE ドングルは、Windows CySmart ホストエミュレーションツールで動作するように事前にプログラムされています。PSoC™ 6-BLE Pioneer Kit は、Bluetooth® LE 設計をプログラムまたはデバッグするために PSoC™ Creator と併用する基板搭載 USB プログラマを備えています。Bluetooth® LE Pioneer Kit は USB インターフェースで電源供給されます。Bluetooth® LE ドングルと Bluetooth® LE Pioneer Kit の両方を、開発とテストのために共通のホスト PC に同時に接続できます。

4 開発セットアップ

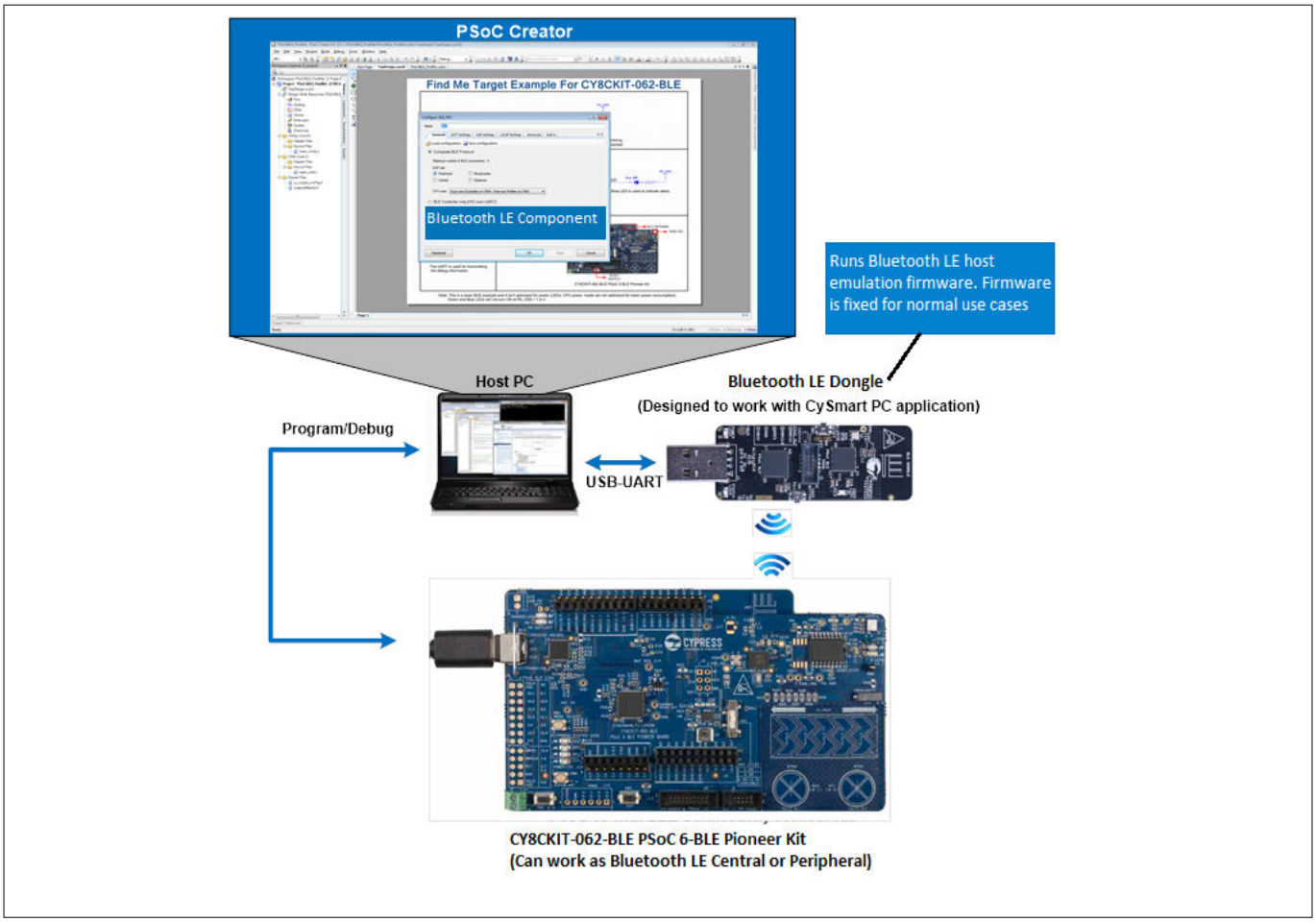


図 11　　Bluetooth® LE 開発セットアップ

5 初めての Bluetooth® LE 搭載の PSoC™ 6 MCU 設計

ここでは、PSoC™ 6-BLE Pioneer Kit のシンプルな設計をするための順を追った手順を提供します。ファインド ミー プロファイル (Find Me Profile; FMP)と呼ばれるシンプルな Bluetooth® SIG 定義標準プロファイルはこの設計で実装されます。高度な標準またはカスタム プロファイル デザインを作成するには、「Designing PSoC™ 6-BLE Applications」および「Creating a BLE Custom Profile」アプリケーション ノートを参照してください。

PSoC™ 6-BLE Pioneer Kit は、Bluetooth® LE ベース アプリケーションの設計を簡素化および合理化するためのものですが、Bluetooth® LE を使用しないアプリケーションを開発する際にも、この設定を使用できます。

5.1 インストラクションについて

これらのインストラクションはいくつかのセクションにグループ化されています。各セクションは、アプリケーション開発ワークフローのフェーズに当てられています。主なセクションは以下のとおりです。

- パート 1: 最初からプロジェクトを新規に作成
- パート 2: 設計の実装
- パート 3: ソース コードの生成
- パート 4: ファームウェアの記述
- パート 5: プロジェクトのビルド、デバイスのプログラミング
- パート 6: デザインをテストする

これらのインストラクションでは、サンプル コードを使用する必要があります。しかし、サンプル コードを使用する範囲は、インストラクションに従うパスによって異なります。

以下に、インストラクションに従った 3 つのパスを定義します。

パス	最初からの作業 サンプル コードを参考とする	サンプル コードの使用 PSoC™ Creator または Bluetooth® LE に詳しくない	サンプル コードの使用 PSoC™ Creator と Bluetooth® LE に精通している
対象	PSoC™ Creator および Bluetooth® LE を理解するために実践体験を学びたい人	ツールや技術を初めて使い、それがどのように動作するかを見たい人	ツールと技術に精通しており、それが PSoC™ 6 プラットフォーム上でどのように動作するかを見たい人

各パスで行う必要があることは、インストラクションの各パートの始めに明確に定義されています。

最初から始め、本アプリケーションノートに記載されているすべて手順に従う場合は、サンプル コードを参考にしてください。最初から作業するパスでは、PSoC™ Creator の設計プロセスと BLE の動作方法について最も多くの知識を取得できますが、一番時間がかかります。これは最も時間がかかります。

サンプル コードを直接使用することもできます。サンプル コードは、すべてのファームウェアが書かれた完成済みの設計内容です。サンプル コードでそのインストラクションがどのように実装されているかを確認できます。これは、PSoC™ Creator に精通しており、Bluetooth® LE コンポーネントがどのように構成されるかを見たい人には特に役立つアプローチです。サンプル コードを直接使用する場合、手順の一部を飛ばせます。

いかなる場合でも、「設計にあたって」および「設計について」をお読みください。

5.2 設計にあたって

ホスト コンピュータには以下のものがインストールされていることをご確認ください。

- PSoC™ Creator 4.4
- PDL v3.1.x
- CySmart ホストエミュレーションツールまたは CySmart iOS/Android アプリ

- CE217236 – このアプリケーションノートで使用されるサンプル コードと
- Kitprog3 と CY8CKIT-062-BLE PSoC™ 6-BLE pioneer kit。Kitprog のファームウェアのアップデート方法については、デバッギングを参照してください。

上記のリンクをクリックしてウェブサイトからサンプル コードをダウンロードできます。あるいは、PSoC™ Creator で **File** > **Code Example** コマンドも使用できます。**Device Family** を PSoC™ 63 に設定します。次に、**Filter by** オプションを **Find Me** に設定します。CE212736_PSoC6BLE_FindMe サンプル コードが表示されます。サンプル コードを選択し、サンプルの横にあるダウンロードアイコンをクリックしてダウンロードします。インストールが完了したら、**Create Project** をクリックし、画面の指示に従います。

この設計は、CY8CKIT-062-BLE PSoC™ 6 BLE Pioneer Kit 用に開発されました。他のハードウェアを使用したい場合、インストラクションを必要に応じて適応させてください。

5.3 設計について

本設計は、即時アラートサービス (Immediate Alert Service; IAS) で構成されるターゲットロールに Bluetooth® LE ファインド ミー プロファイル (FMP) を実装します。FMP および IAS は、Bluetooth® SIG によって定義された Bluetooth® LE 標準プロファイルおよびサービスです。図 12 に示すように、ファインド ミー ロケーターによってトリガされるアラート レベルは Bluetooth® LE Pioneer Kit 上の LED の状態を変化させることで示します。その他、2 つのステータス LED が Bluetooth® LE インターフェースの状態を示します。さらに、ターミナル ウィンドウにデバッグメッセージを出力するオプションの UART コンポーネントもあります。

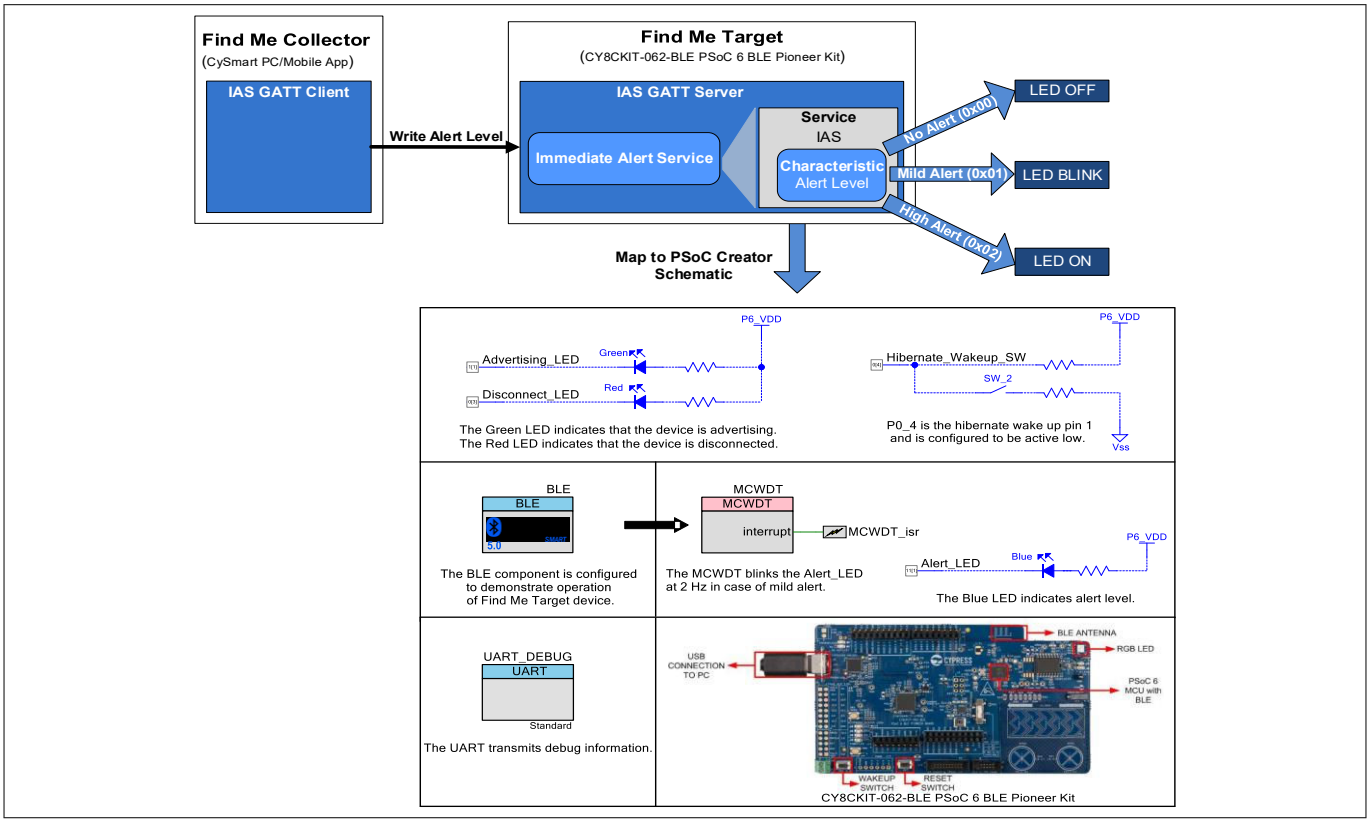


図 12 初めての PSoC™ 6-BLE 設計

5.4 パート 1: 最初からプロジェクトを新規に作成

ここでは、最初からプロジェクト ファイルを作成するプロセスを、順を追って説明します。

パス	最初からの作業 サンプル コードを参考とする	サンプル コードの使用 PSoC™ Creator または Bluetooth® LE に詳しくない	サンプル コードの使用 PSoC™ Creator と Bluetooth® LE に精通している
アクション	すべてのステップを行う	ステップ 1 を行う 残りのステップを読む	ステップ 1 を行ってから、パート 2 へ進む

注: これらの手順では、ユーザーが PSoC™ Creator 4.4 以降を利用していることを前提にします。開発プロセス全体は PSoC™ Creator の後続バージョンでも同じですが、ユーザー インターフェースは変更されることがあります。

PSoC™ Creator を起動して始めましょう。

- **PSoC™ Creator が PDL を利用できるようにする**

  ***重要**: これはインストール時に自動的に正しく設定されますが、誤ってセットアップされると何も実行できません。このステップのヘルプについては、図 13 を参照してください。*

  - メニュー項目の **Tool** > **Options** を選択します。
  - Project Management パネルで、PDL v3 (PSoC™ 6 Devices) location フィールドのパスを確認してください。
  - パスが正しいことを確かめてください。正しくない場合は、**Browse** ボタンをクリックして、PDL がインストールされたディレクトリを探します。デフォルトの場所は *C:\Program Files (x86)\Cypress\PDL\3.0.1* です。

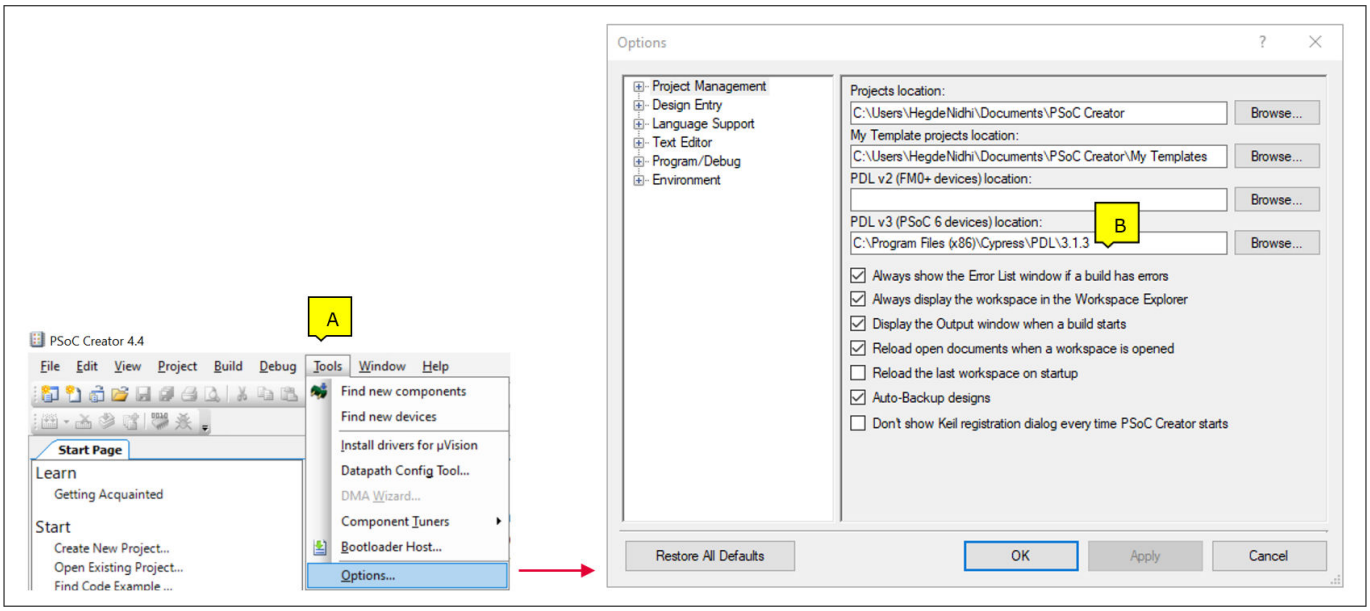


図 13 ペリフェラル ドライバー ライブラリ (PDL) のインストール場所

オプション: パート 2: 設計の実装へ進む

- **PSoC™ Creator プロジェクトを新規作成**
  - 図 14 に示すように **File** > **New** > **Project** を選択します。**Create Project** ウィンドウが表示されます。

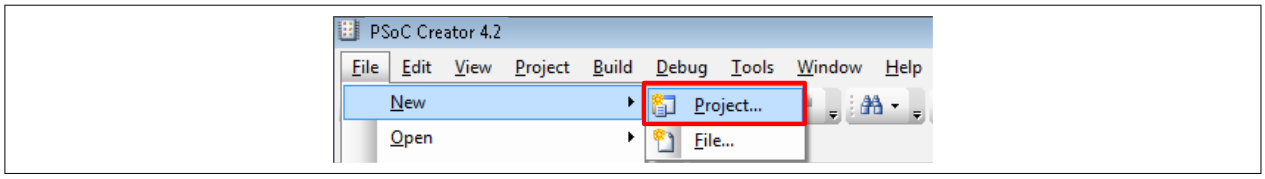


図 14　　PSoC™ Creator プロジェクトを新規作成

*注:　　サンプル コードを使用している場合は、図 15 のように **File** > **Open** > **Project/Workspace** を選択します。**Open** ウィンドウが表われます。サンプル コード ワークスペースの場所を指して、ワークスペースを開きます。*

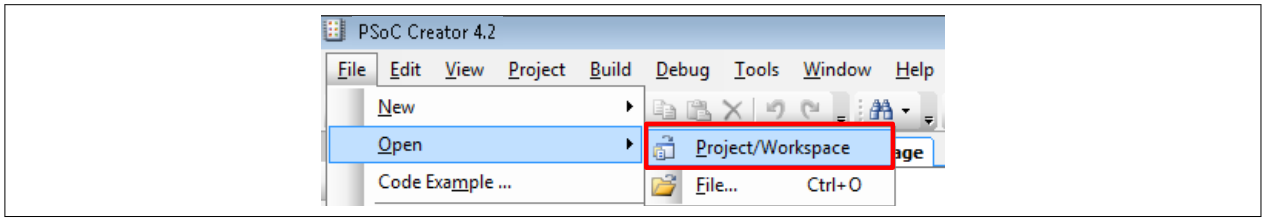
- 

図 15　　既存のサンプル コード ワークスペースを開く

- **PSoC™ 6 Bluetooth® LE をターゲット デバイスに選択**

  ヒント:　　PSoC™ Creator は、自動的にさまざまなプロジェクトオプションを特定の開発キットやターゲットデバイスに設定するため、開発時間を短縮できます。このステップのヘルプについては、図 14 を参照してください。

  - Target Device を選択します。
  - ファミリドロップダウンリストから、PSoC™ 6 を選択します。
  - デバイスのドロップダウンメニューから、PSoC™ 63 をリストします。
  - **Next** をクリックします。プロジェクトテンプレートの選択パネルが表示されます。

PSoC™ Creator は、Bluetooth® LE 搭載の PSoC™ 6 MCU ファミリには CY8C6347BZI-BLD53 をデフォルトデバイスとして使用します。このデバイスは、CY8CKIT-062-BLE PSoC™-BLE Connectivity Pioneer Kit に搭載されています。

PSoC™ 6-BLE に基づくカスタムハードウェアまたは別の PSoC™ 6-BLE 製品番号を使用する場合、これは、**Target device** の中で **Launch device selector** オプションを選んで適切な製品番号を選択する場所です。

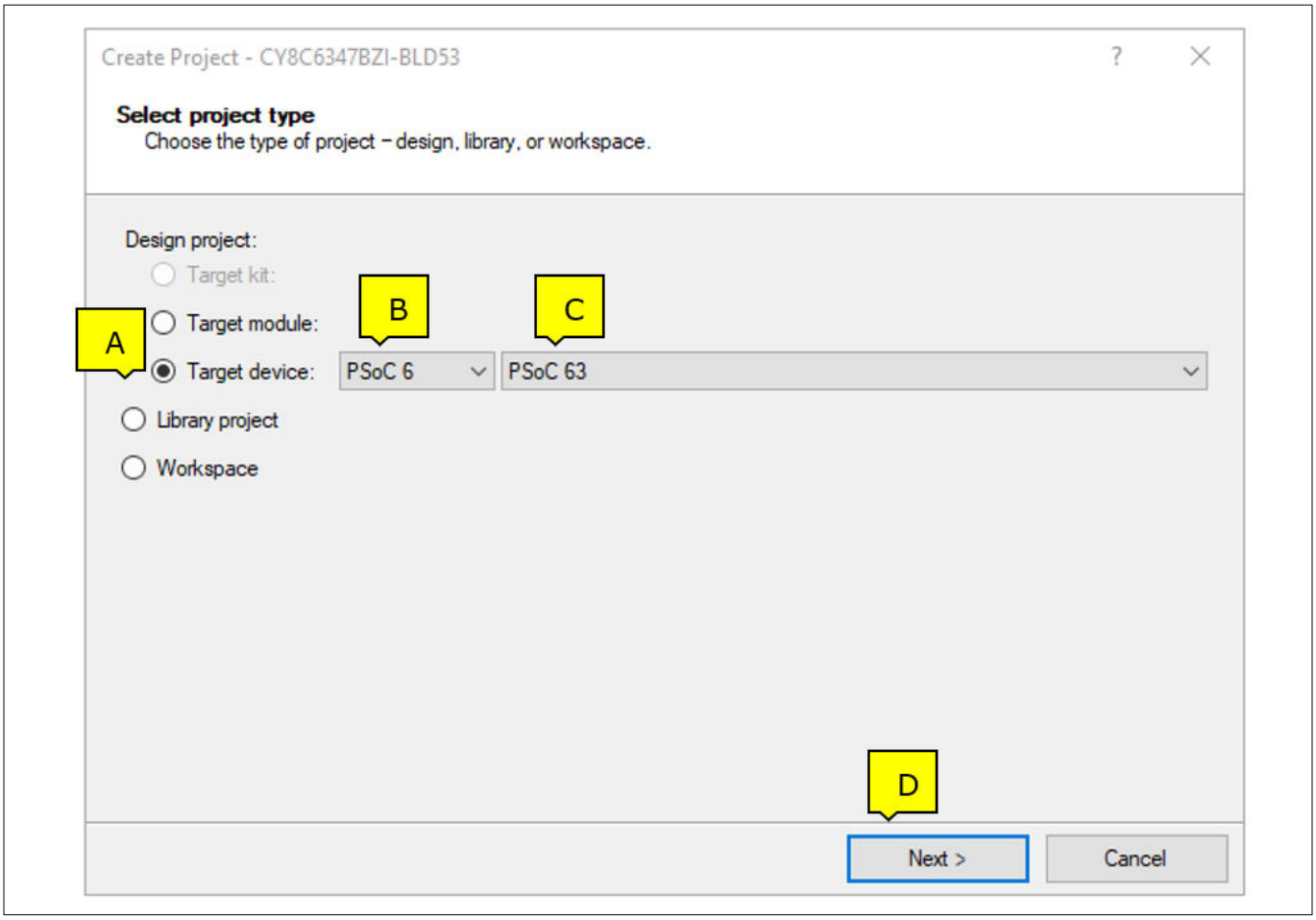


図 16 Target Device を選択します。

- **プロジェクト テンプレートを選択**
  - Empty schematic を選択します。
  - **Next** をクリックします。

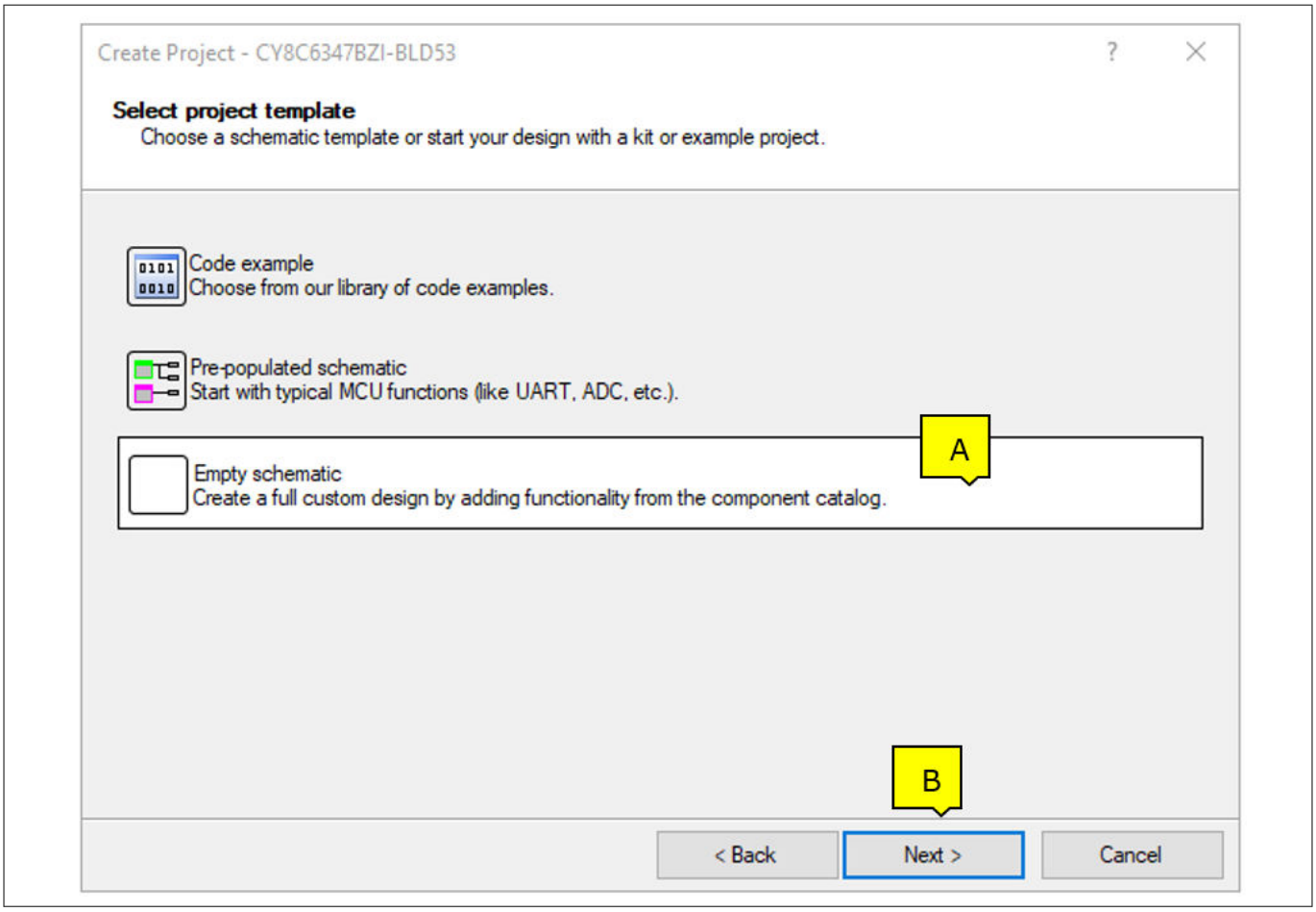


図 17　　プロジェクト テンプレートを選択

- **対象の IDE を選択**

  注:　　プロジェクトからコードをエクスポートする場合、対象の IDE を選択してください。デフォルトでは、すべてのエクスポート オプションは無効になっています。状況が変わる場合、後でこの設定を変更できます。

  - **Next** をクリックして、デフォルト設定を受け入れます。

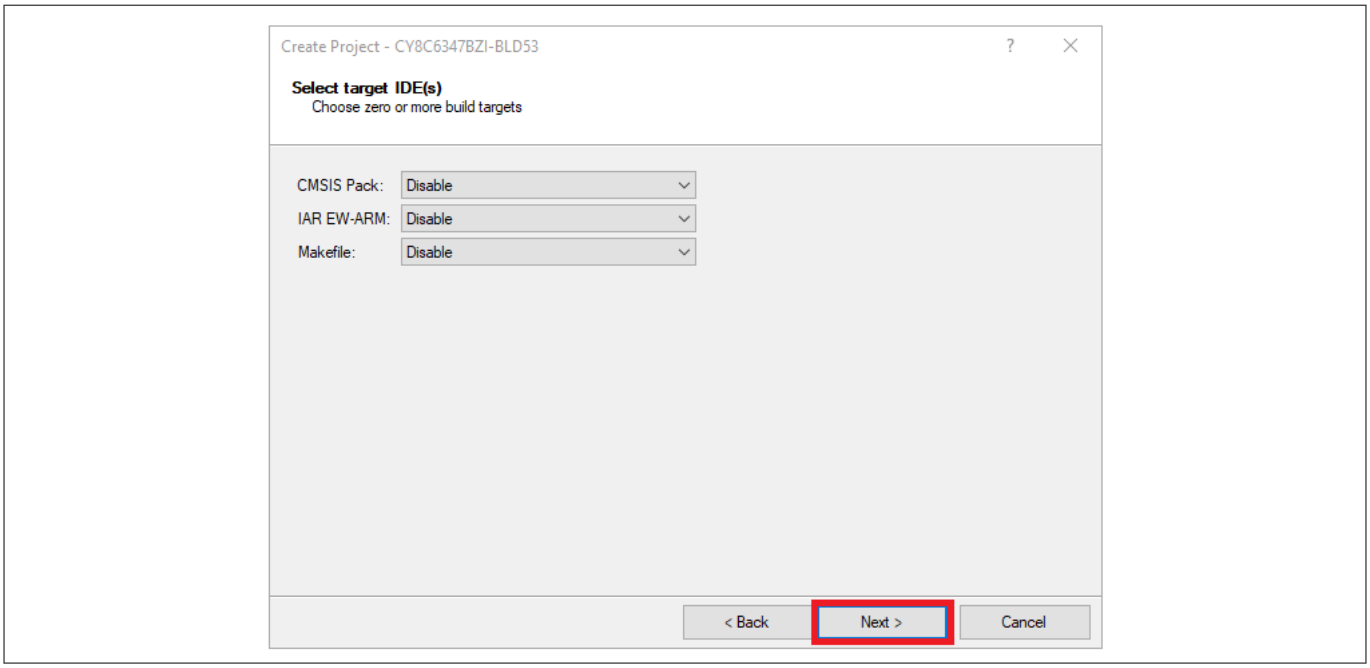


図 18　　対象の IDE を選択 (すべては無効)

- **プロジェクトを作成**

  注:　　このステップでは、ワークプレースの名前と保存先およびプロジェクトの名前をセットします。このステップのヘルプについては、図 19 を参照してください。1 つのワークスペースに 1 つ以上のプロジェクトを格納できます。

  - Workspace name をセットします。
  - ワークスペースの Location を指定します。
  - Project name をセットします。プロジェクト名とワークスペース名は同じでも違っても構いません。
  - **Finish** をクリックします。

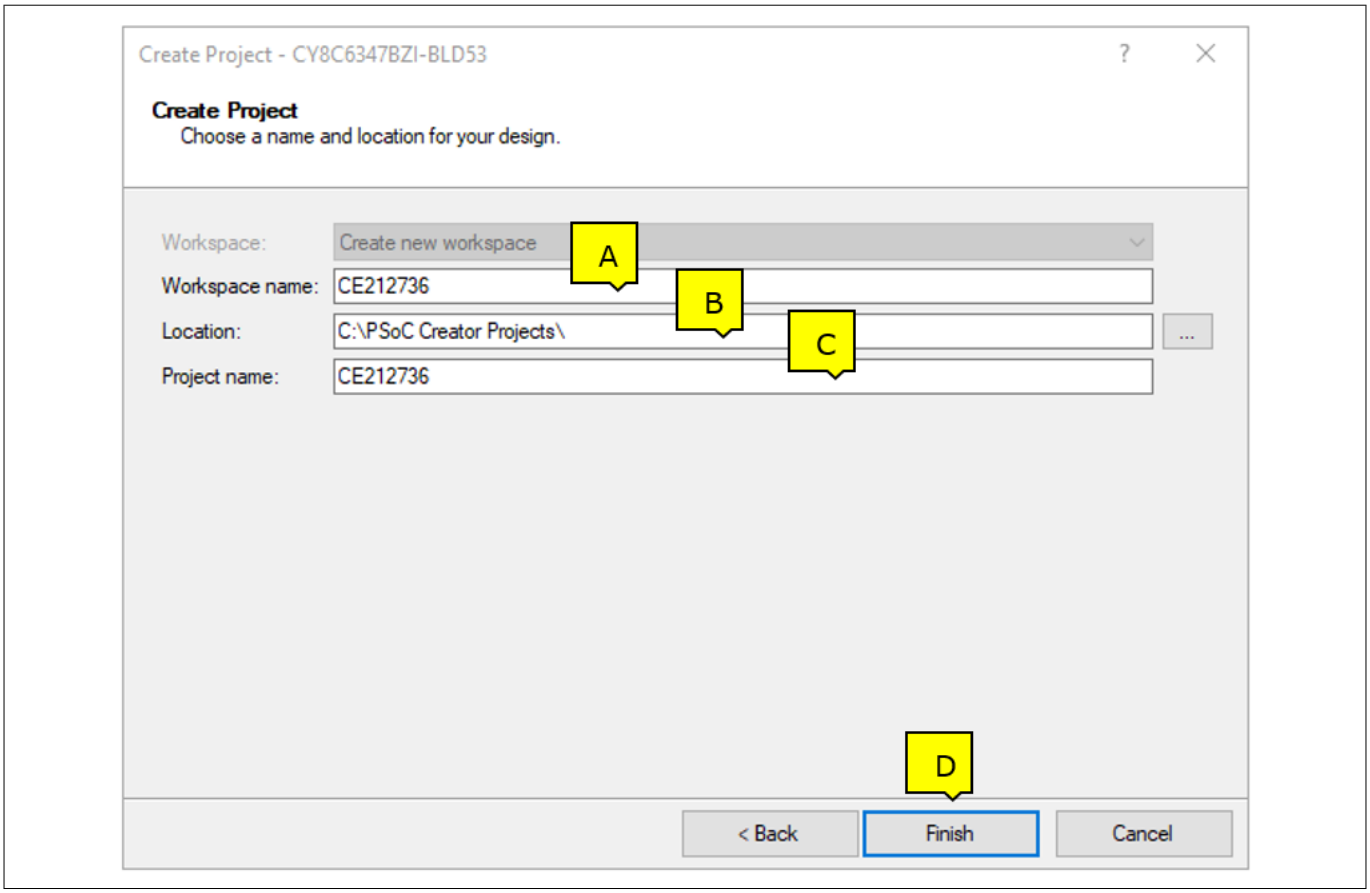


図 19　　プロジェクトの命名および保存先

このように、PSoC™ Creator プロジェクトの新規作成ができました。

5.5　　パート 2: 設計の実装

プロジェクト ファイルを作成したので、このパートでは PSoC™ Creator コンポーネントを使用してハードウェア設計を実装します。サンプル コードを直接使用する場合、それはすでに完成した設計です。この作業で選択したパスに基づいて推奨されるアクションを行います。

パス	最初からの作業 サンプル コードを参考とする	サンプル コードの使用 PSoC™ Creator または Bluetooth® LE に詳しくない	サンプル コードの使用 PSoC™ Creator と Bluetooth® LE に精通している
アクション	すべてのステップを行う	すべてのステップを読んで理解	この部分を飛ばすことができます。パート 3: ソース コードの生成へ進む

デザインを実装する前に、PSoC™ Creator インターフェースのクイックツアーが順を追って行われます。

図 20 に、空の設計回路図を表示する PSoC™ Creator アプリケーションを示します。

このプロジェクトは基本的なファイル セットを格納するプロジェクト フォルダを含みます。これらのファイルは左側の **Workspace Explorer** ペインに表示されます。デフォルトでは、プロジェクト回路図が開きます。これは *TopDesign.cysch* ファイルです。エクスプローラ ペインでファイル名をダブルクリックすると、回路図が開かれます。新規プロジェクトでは、回路図は空です。サンプル コードを使用する場合、これはファインド ミー Bluetooth® LE アプリケーションの回路図となります。

Component Catalog がウィンドウの左側にあります。これは、**View** > **Component Catalog** メニュー アイテムで開けます。**Search for...**テキスト ボックスにコンポーネント名を入力してエンター キーを押すことで特定のコンポーネントを検索できます。図 20 を参照してください。

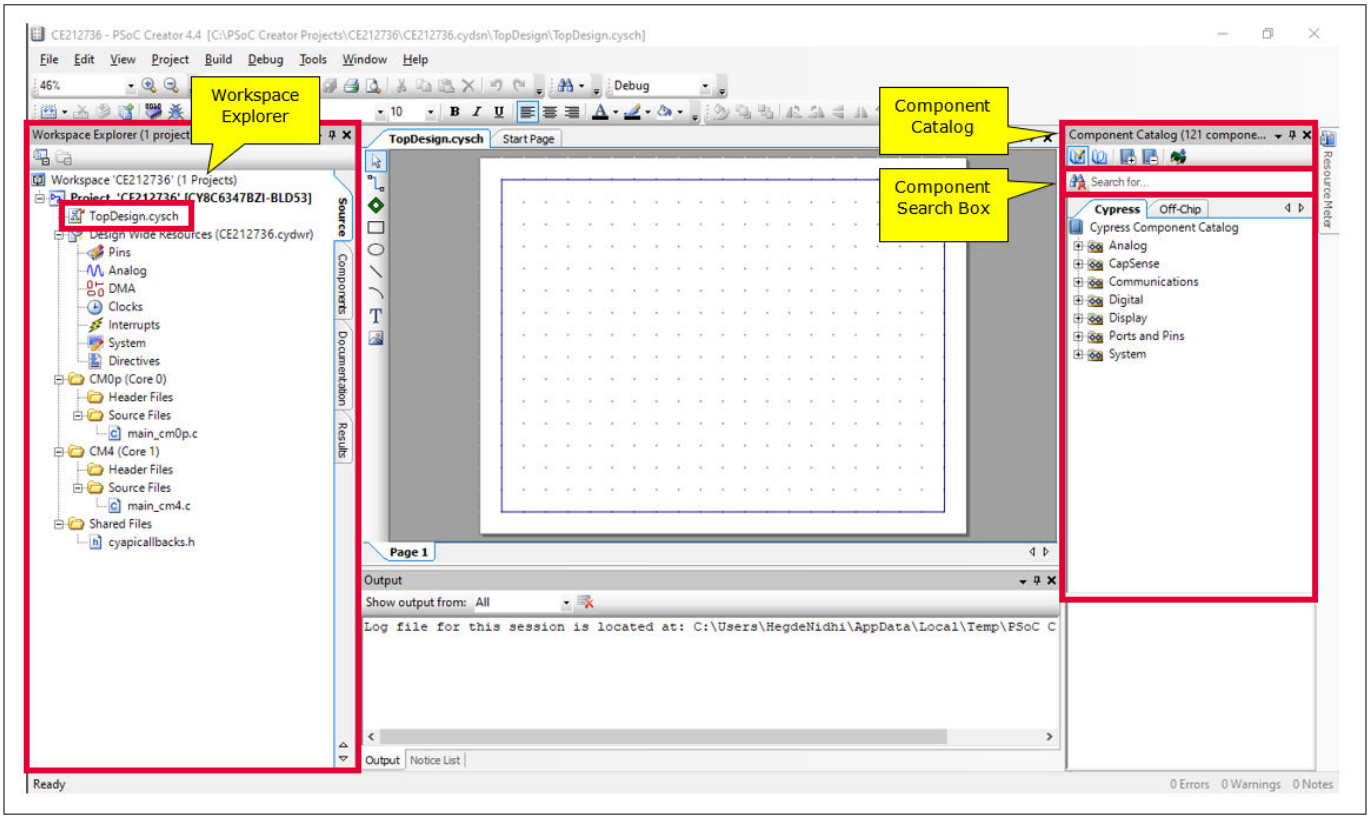


図 20　　回路図およびコンポーネント カタログ

- **デザインにコンポーネントを配置**

このデザインで、Bluetooth® LE コンポーネント、1 つのデジタル入力ピン、3 つのデジタル出力ピン、UART、ウォッチドッグタイマー、および割込みなど、いくつかのコンポーネントを使用します。このステップでは、それらをデザインに追加します。後続ステップで、それらを設定します。図 21 に結果を示します。

- **Component Catalog** で、**Communications** グループを展開し、**Bluetooth® Low Energy** コンポーネントを回路図にドラッグ &ドロップします。コンポーネントは、どこに配置しても構いません。あるいは、Component Catalog 検索ボックスに「BLE」を入力しても Bluetooth® LE コンポーネントを検索できます。
- また **Communications** グループで UART を展開し、**UART (SCB)** コンポーネントをデザインにドラッグします。
- **Ports and Pins** グループを展開し、**Digital Input Pin** をデザインにドラッグします。
- 同じ **Ports and Pins** グループから、**Digital Output Pin** をデザインにドラッグします。これをさらに 2 回繰返し、合計 3 つのピンを作成します。
- **System** グループを展開し、**Interrupt** コンポーネントと **MCWDT** コンポーネントをデザインにドラッグします。

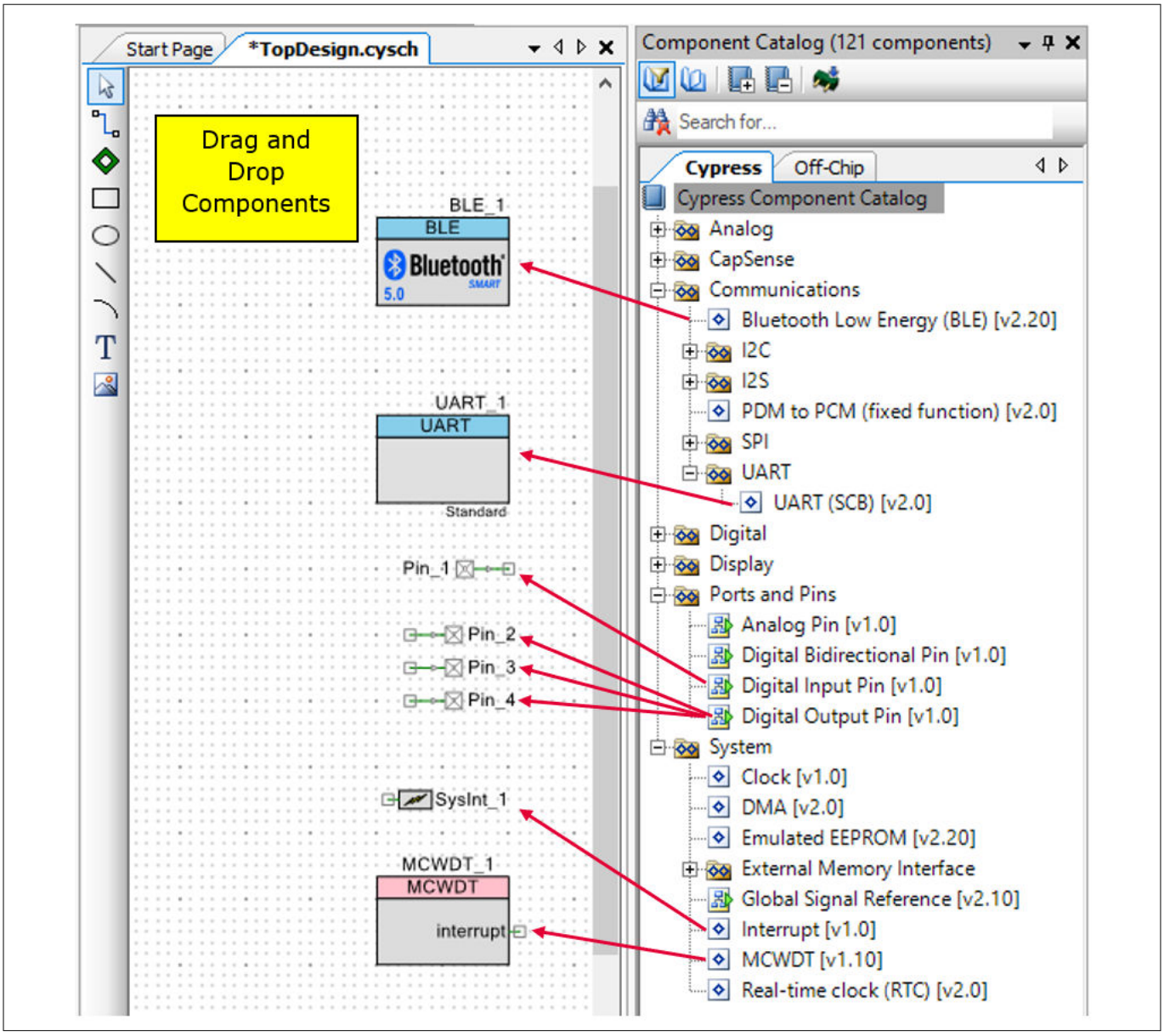


図 21　　デザインにコンポーネントを配置

PSoC™ Creator は各コンポーネントにデフォルトの名前とプロパティを与えます。特定のデザインによっては、デフォルト値はふさわしくない場合があります。後続ステップで、名前およびプロパティの一部を修正します。

- **3 個の LED ピンを設定**

1 つのピンがアラートステータス LED を駆動します。残りの 2 個は、Bluetooth® LE アドバタイズおよび切断状態指示 LED を駆動します。Bluetooth® LE Pioneer Kit では LED はアクティブ LOW です。すなわち、ピンを HIGH に駆動すると LED は消灯し、ピンを LOW に駆動すると LED は点灯します。

3 個のピンは名前を除き、同じように設定されます。これらの指示を 3 回繰り返します (ピンごとに 1 回)。図 22 はコンフィギュレーションを示します。

回路図に配置されたコンポーネントをダブルクリックして、設定ダイアログを開き、以下の手順を実行します。

- 各ピンのコンポーネント インスタンス名を、**Alert_LED**、**Advertising_LED** および **Disconnect_LED** に変更します。
- 各ピンに対し、**HW connection** の選択を解除します。ファームウェアはピンを駆動します。
- 各ピンに対し、**Initial drive state** を **High (1)** に設定します。したがって、LED はデフォルトでオフになります。

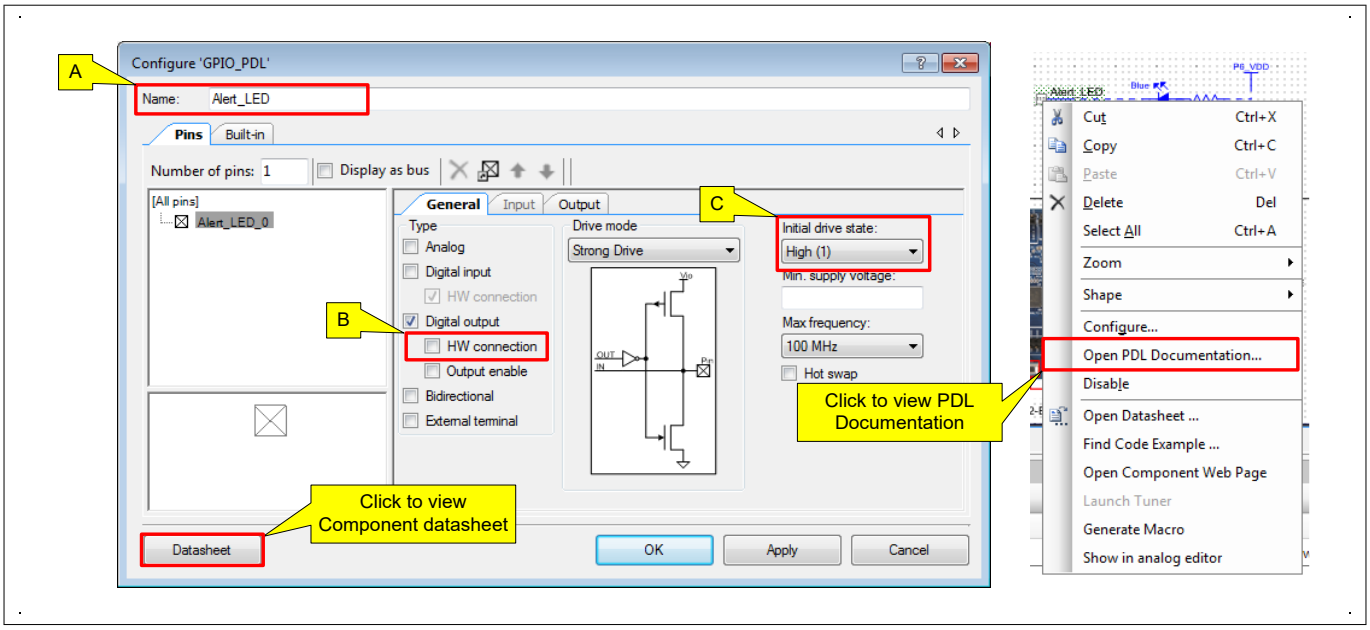


図 22　　出力ピンコンポーネントの設定

必ず 3 個のピンをすべて設定してください。

ヒント:　各コンポーネントには、設定ウィンドウからアクセスできる対応データシートがあります。コンポーネント データシートは、コンポーネントの設定、アプリケーション プログラミング インターフェース (API) および電気的仕様の詳細情報を提供します。

*ヒント:　コンポーネントを右クリックして **Open PDL Documentation...** リンクをクリックすると、コンポーネントに関連 PDL ドライバーの API リファレンスドキュメントを開けます。図 22 を参照してください。*

*ヒント:　ピンに対して **External terminal** を有効にすることで、外部の「オフチップ」コンポーネントをデザインに追加できます。回路図上の外部コンポーネントは、説明のためのみ含まれ、生成されたコードには影響を与えません。オフチップコンポーネントは任意ですが、ハードウェア設計チームがデザインの仕組みを理解する際に役立ちます。また、記述用のテキストボックスを追加することもできます。図 23 に、アラート LED のデザインを拡張する方法を示します。この場合、オフチップコンポーネントは **Instance_Name_Visible** の選択を外して設定します。抵抗は、**Value** フィールドを空白にして設定します。電源端子は、**Supply_Name** が P6_VDD にセットして設定します。*

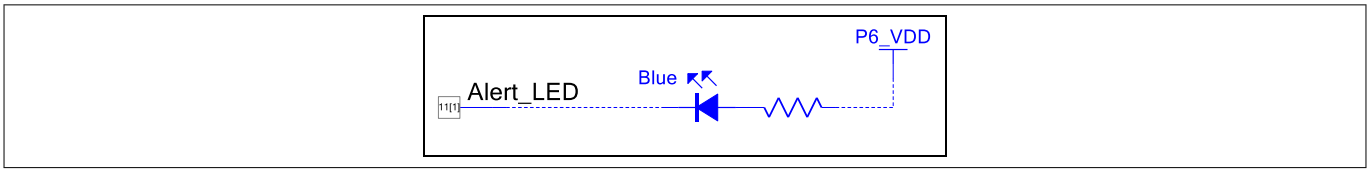


図 23　　外部コンポーネント付きの出力ピン

- **ハイバネート ウェイクアップピンを設定**

Pioneer Kit 上の SW2 スイッチは、ハイバネート ウェイクアップピンの 1 つである P0[4]に接続されており、押されるとポートピンを LOW にします。このピンをハイバネート ウェイクアップ スイッチに設定するには、抵抗プルアップ ピンに設定する必要があります。

図 24 はコンフィギュレーションを示します。回路図に配置されたコンポーネントをダブルクリックして設定ダイアログを開き、以下を行います。

- コンポーネント インスタンス名を **Hibernate_Wakeup_SW** に変更します。

- **HW connection** の選択を外します。アプリケーション ファームウェアは、このピンを駆動しませんが、ピンはハイバネート システムに固定接続されます。ファームウェアでは、ウェイクアップ信号はアクティブ LOW に設定してされます。
- **Drive mode** を **Resistive Pull Up**、**Initial drive state** を **High (1)** にセットします。これにより、HIGH から LOW への遷移を検出してハイバネート モードから復帰するようデバイスを設定します。

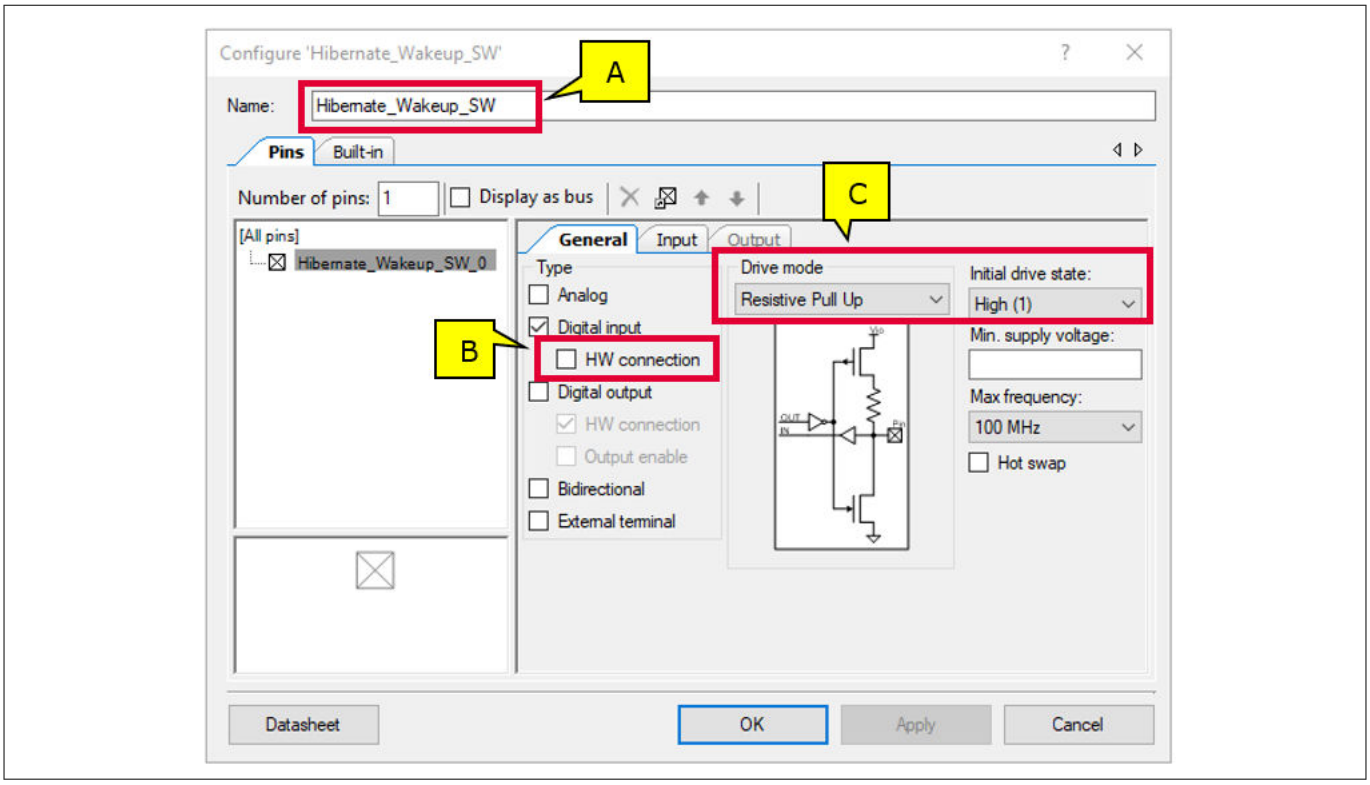


図 24　　入力ピン コンポーネントを設定

- **UART コンポーネントを設定**

回路図に配置されたコンポーネントをダブルクリックして、設定ウィンドウを開きます。このコンポーネントを使用して 115200bps のボーレートでターミナル ウィンドウにデバッグ メッセージを表示します。これは Bluetooth® LE 機能には関係しません。

- コンポーネント インスタンスの **Name** を「**UART_DEBUG**」に変更します。
- **OK** をクリックします。

このデザインで、他のすべての設定にはデフォルト値を使用します。

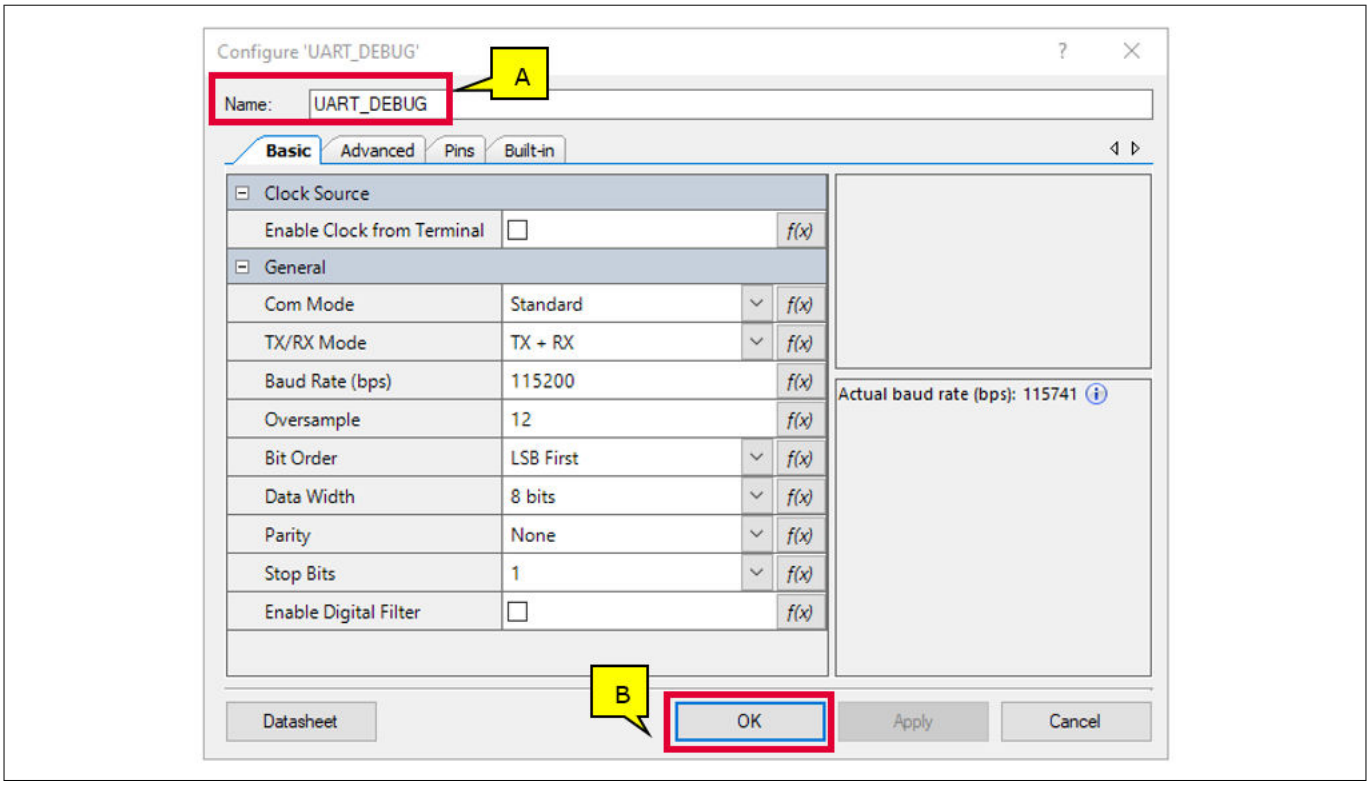


図 25　　SCB ベースの UART コンポーネントを設定

- **一般の Bluetooth® LE オプションを設定**

回路図に配置された Bluetooth® LE コンポーネントをダブルクリックして、設定ウィンドウを開きます。図 26 に示すように **General** プロパティをセットします。このアプリケーションで、コンポーネント名およびスタック操作を除き、デフォルトの一般のプロパティを使用します。

- **Name** を「BLE」に変更します。
- **Complete BLE Protocol** が選択されていることを確認します。
- **Maximum number of BLE connections** を 1 に変更します。このように、BLE スタックを適切に設定します。
- **Peripheral** が **GAP role** に選択されていることを確認します。このように、BLE ペリフェラルデバイスとして機能し、セントラルデバイスの要求に応答するようデバイスを設定します。
- **CPU core** を **Dual core (Controller on CM0+, Host and Profiles on CM4)** オプションに選択します。このように、両方のコアで動作するよう Bluetooth® LE スタックを分割します。CM0+コアはスタックの BLE コントローラー部分を実行し、BLE 接続の維持を担当します。BLE ホストは CM4 コア上で動作し、アプリケーションレベルのタスクを実行します。このデュアル CPU の主な利点は、保留中の Bluetooth® LE 関連のタスクがないときには、CM4 コアがディープスリープ低消費電力モードに移行できることです。

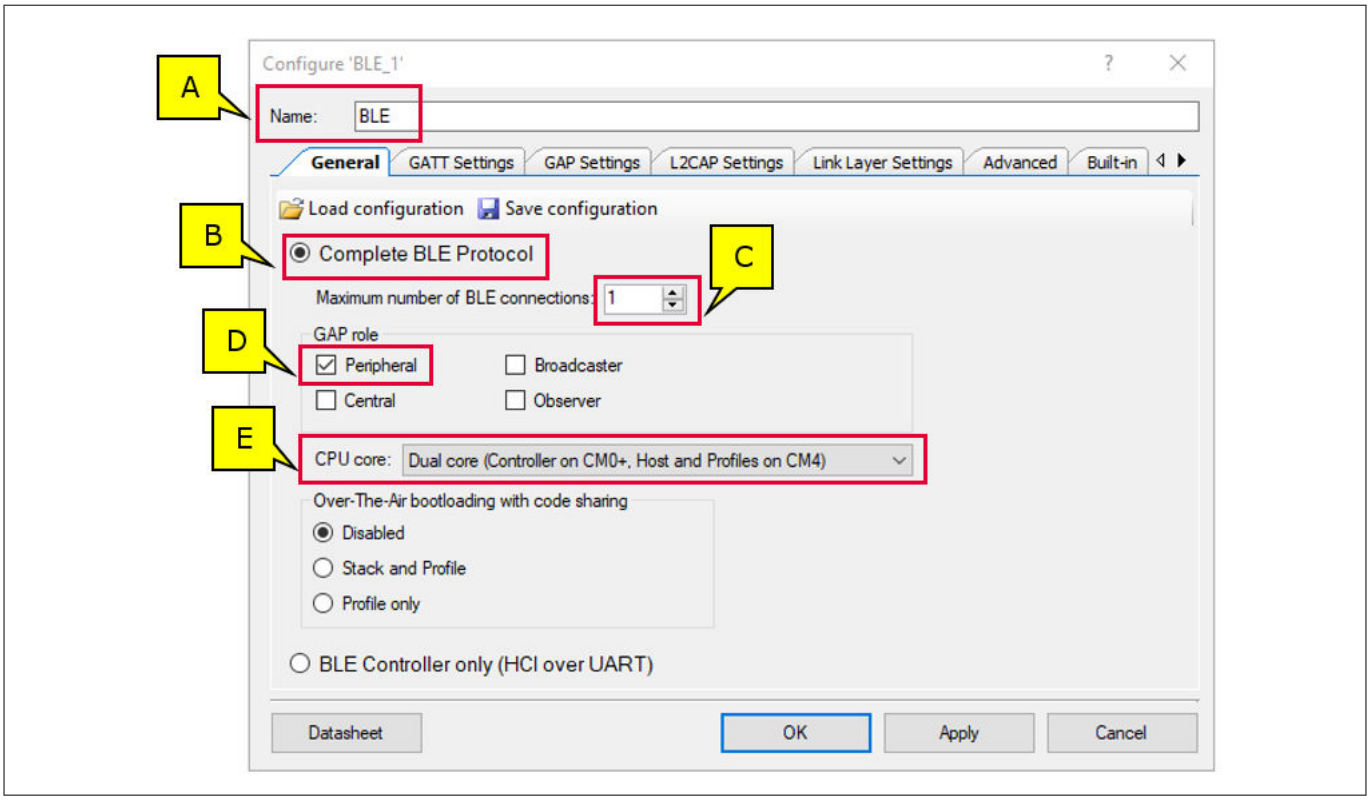


図 26　　Bluetooth® LE コンポーネントの一般設定

- **Generic Attribute (GATT) 設定を指定**

このステップでは、図 27 に示すように Bluetooth® LE プロファイルを設定します。

- **GATT Setting** タブをクリックして、GATT オプションを表示します。
- **Add Profile** ドロップダウンメニューをクリックします。
- **Find Me** > **Find Me Target (GATT Server)** オプションを選択します。このように、GAP ペリフェラル ロール プロファイルをセットします。メニューが閉じると、図 28 に示すように「**Immediate Alert**」という新しいサービスが現れることにご注意ください。

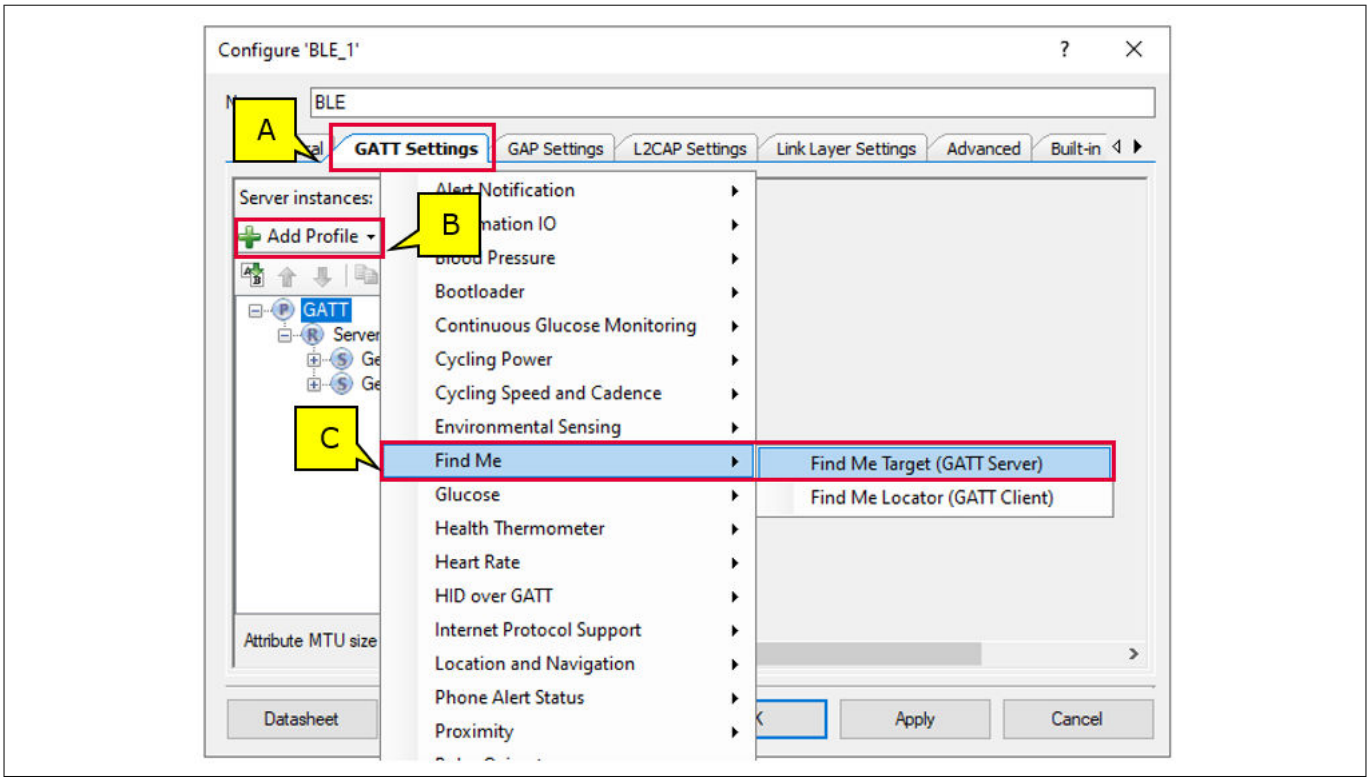


図 27　　Bluetooth® LE コンポーネントの即時アラート サービスを追加

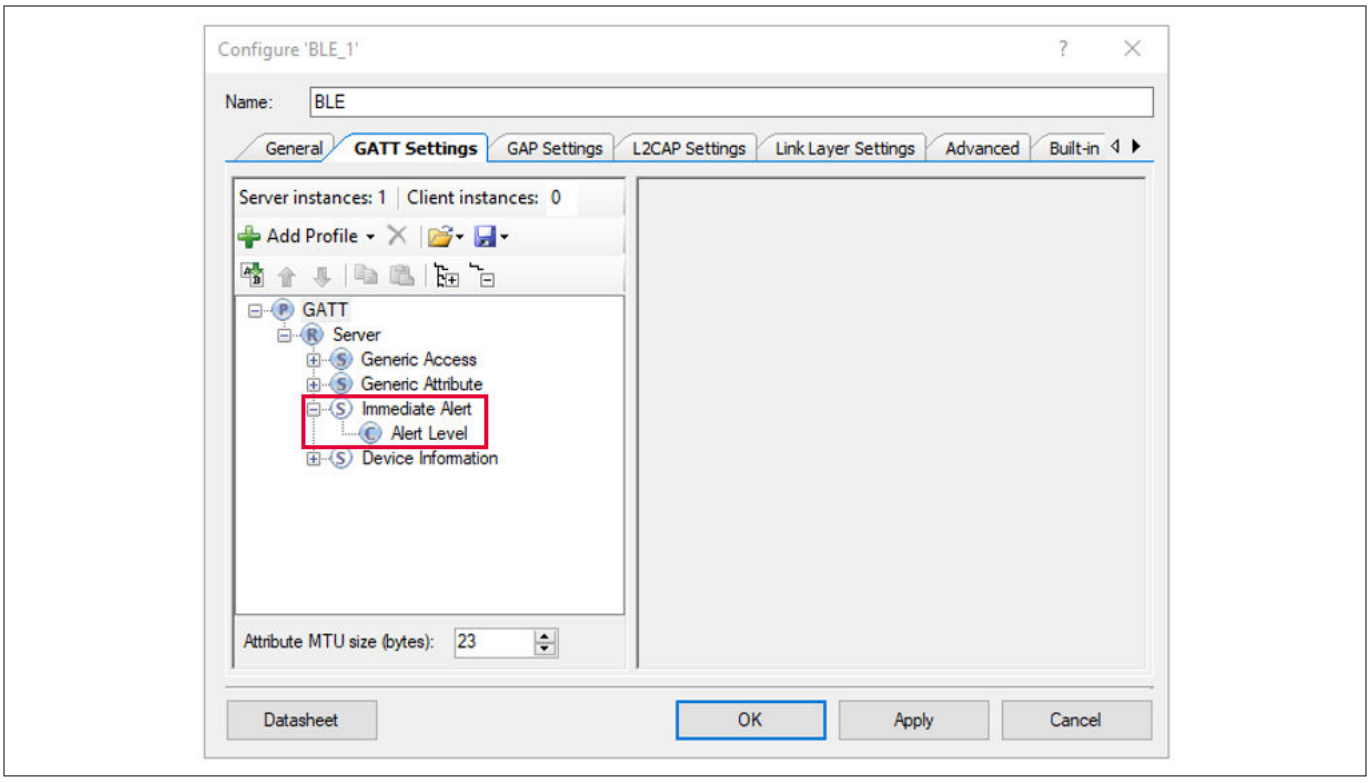


図 28　　Bluetooth® LE コンポーネントの即時アラート サービスを追加

注: *このサンプル コードには、デバイスを識別してファームウェアバージョンを読み出すために、GATT 設定に追加される **Device Information Service** (デバイス情報サービス) があります。このサービスは、**Immediate Alert** が動作する際には必要ありません。*

- **Generic Access Profile (GAP) の一般設定を指定**

GAP 設定のために、一連のパネルがあります。左メニューでは、すべてのパネルへアクセスできます。図 29 を参照してください。

- **GAP Setting** タブをクリックして、GAP オプションを表示します。デフォルトでは **General** パネルが表示されます。
- **Device name** として「**Find Me Target**」を入力します。
- **Appearance** を **Generic Keyring** にセットします。

他のすべての一般設定はデフォルト値を使用します。これには、デバイスは **Silicon generated "Company assigned" part of device address** を使用します。これにより、ホストデバイスがご使用のデバイスを検出しようとし、それに固有の Bluetooth® LE デバイス アドレスを割り当てる時に現れるデバイス名とタイプを設定します。

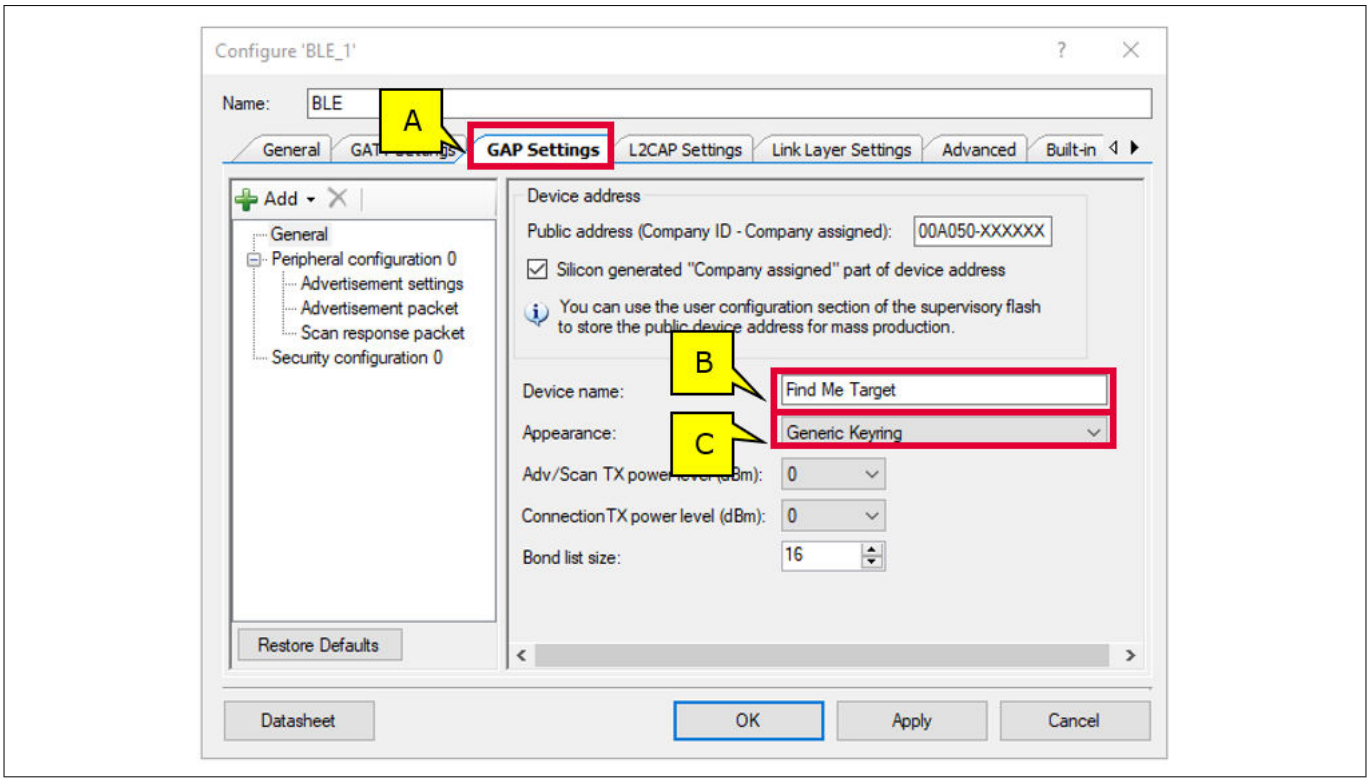


図 29 Bluetooth® LE コンポーネントの GAP 一般設定

- **GAP アドバタイズメント設定を指定**

このステップのヘルプについては、図 30 を参照してください。

- 左メニューで **Advertisement settings** をクリックしてパネルを表示します。パネルが表示されます。
- **Advertising type** を **Connectable undirected advertising** にセットします。
- **Slow advertising interval** チェックボックスのチェックを外します。

それ以外では、このアプリケーションはデフォルト値を使用します。30 s のアドバタイズ タイムアウトおよび 20～30 ms の高速アドバタイズ間隔の有限検出モードを使用します。高速アドバタイズにより、迅速に検出し接続を確立することが可能ですが、RF アドバタイズ パケットが増加するため消費電力が高くなります。

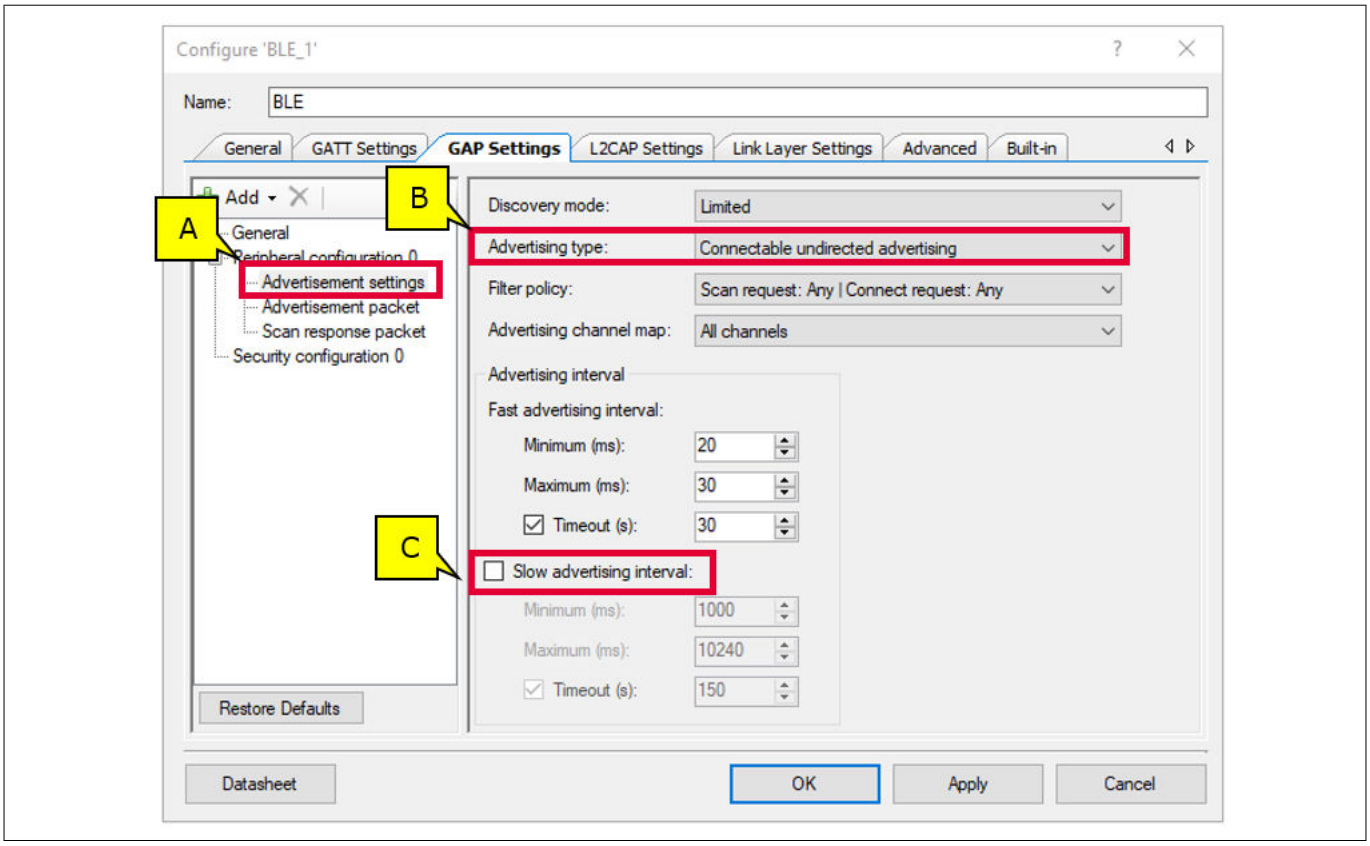


図 30　　Bluetooth® LE コンポーネントの GAP アドバタイズ設定

- **GAP アドバタイズメントパケットの設定を指定**

このステップでは、即時アラートサービス (IAS) に対応するようにデバイスを設定します。図 31 を参照してください。

- 左メニューで **Advertisement packet** をクリックしてパネルを表示します。パネルが表示されます。
- **Service UUID** 項目を展開して **Immediate Alert** を選択します。

このように、IAS が使用可能であることを Bluetooth® LE セントラル デバイスに通知するようデバイスを設定します。項目を追加すると、アドバタイズ パケットの構造および内蔵が設定パネルの右側に現れます。

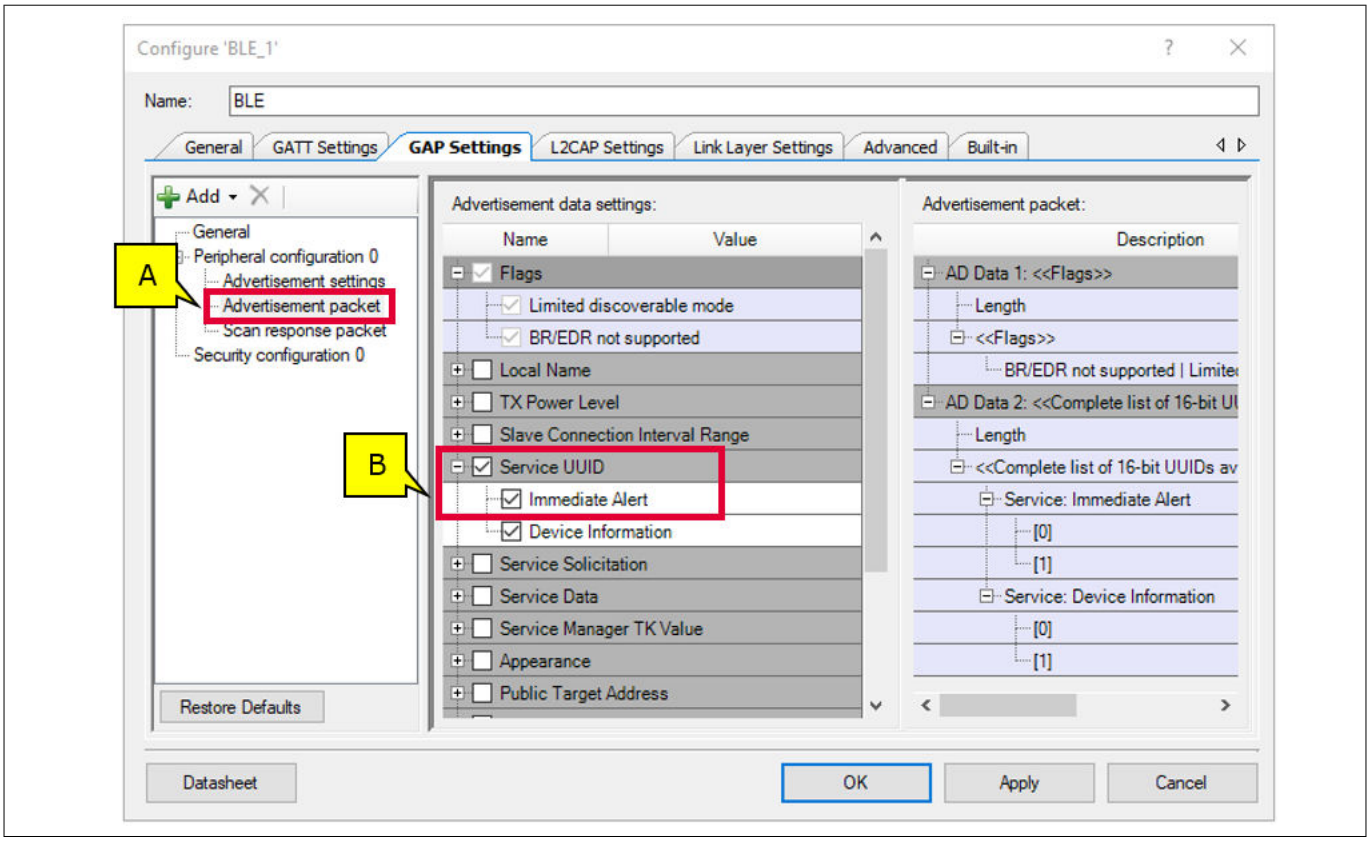


図 31 Bluetooth® LE コンポーネントの GAP アドバタイズ パケット設定

- **スキャン応答パケット設定を指定**

このステップでは、スキャン応答パケットの設定を指定します。図 32 に結果を示します。値を追加するとスキャン応答パケットの構造および内容が設定パネルの右側に現れることにご注意ください。

- 左メニューで **Scan response packet** をクリックしてパネルを表示します。パネルが表示されます。
- **Local Name** を選択してこの項目を応答パケットに含めます。**Complete** であるデフォルト設定を **OK** にします。
- **TX Power Level** を選択してこの項目をパケットに含めます。
- **Appearance** を選択してこの項目をパケットに含めます。

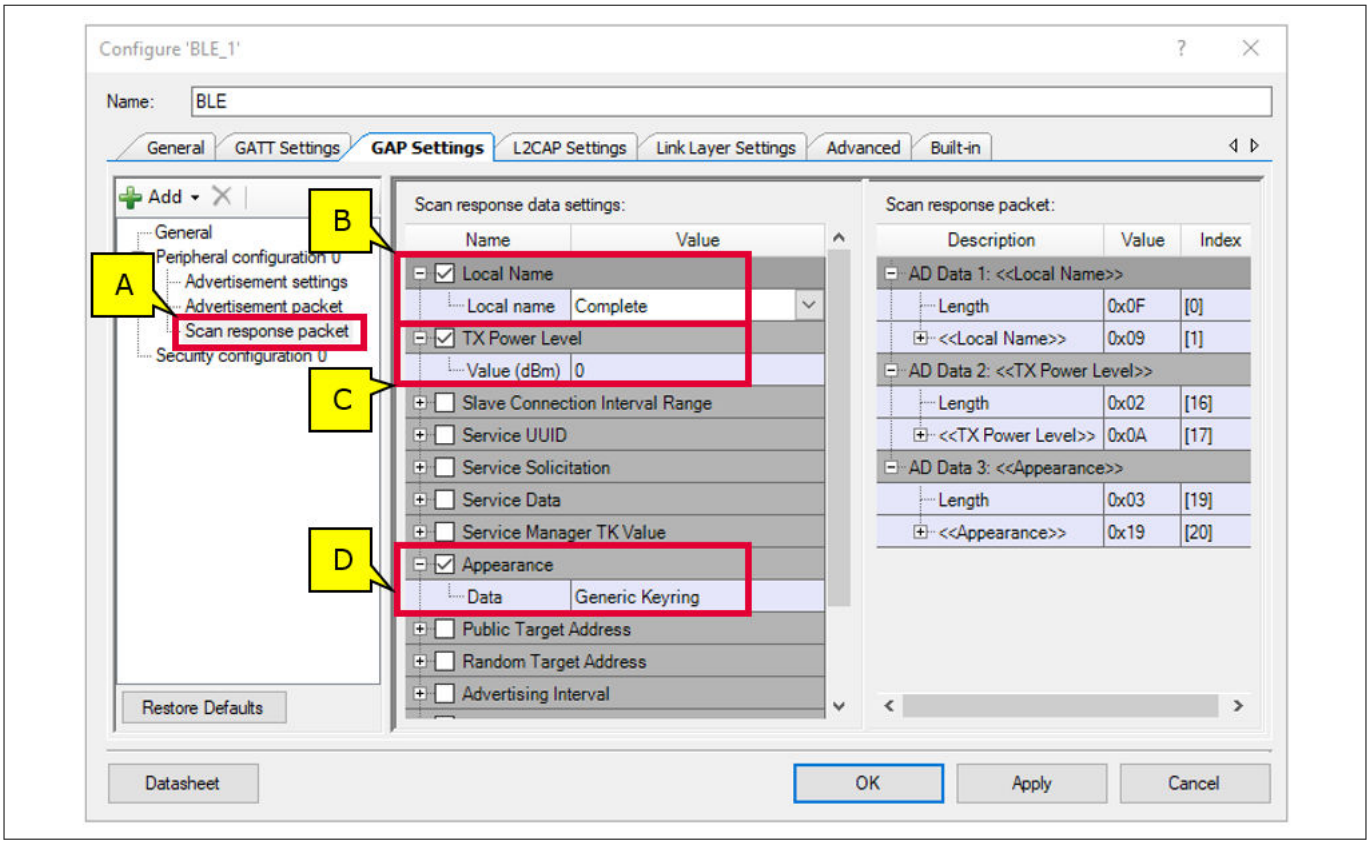


図 32　　Bluetooth® LE コンポーネントの GAP スキャン応答パケット

- **セキュリティコンフィギュレーション設定を指定**

このステップでは、デバイスのセキュリティ設定を行います。デバイスは、データ交換用の認証や暗号化、権限付与が不要なコンフィギュレーションを使用します。図 33 を参照してください。

- 左メニューで **Security configuration 0** をクリックしてパネルを表示します。パネルが表示されます。
- **Security mode** が **Mode 1**、**Security level** が **No Security** であることを確認します。そうでない場合、設定を修正します。
- **IO capabilities** を **No Input No Output** にセットします。
- **Bonding requirement** を **No Bonding** にセットします。
- **OK** をクリックして、Bluetooth® Low Energy (Bluetooth® LE) コンポーネントの設定を完了します。

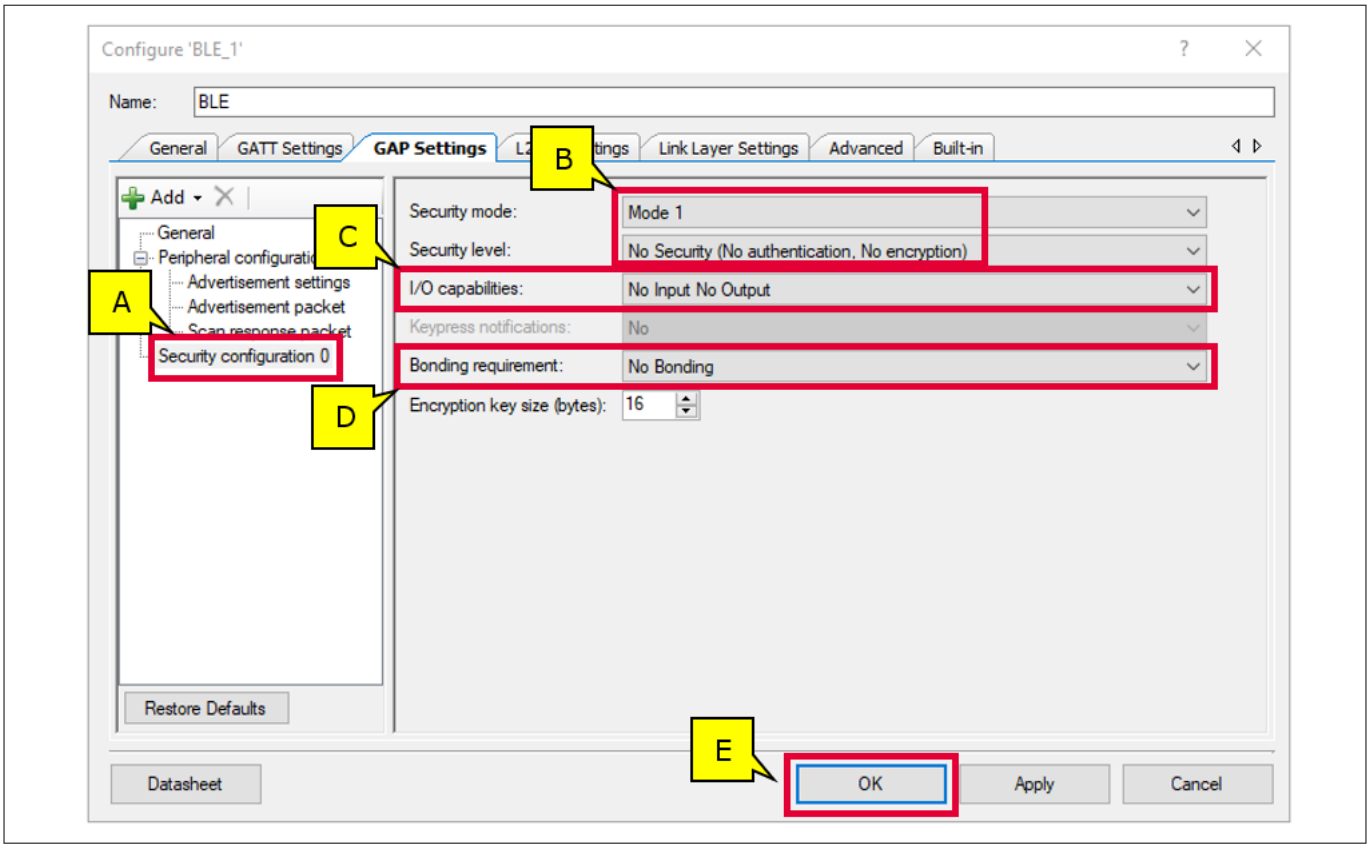


図 33　　Bluetooth® LE コンポーネントの GAP セキュリティ設定

このデザインでは、**LDCAP Settings**、**Link Layer Settings** および **Advanced** タブのすべてのオプションを含む他のすべての設定はデフォルト値を使用します。

各設定の重要性を理解できるように、コンポーネントデータシートを参照してください。

- **割込みをトリガするように MCWDT コンポーネントを設定**

このステップでは、250 ms ごとに (4 Hz で) 割込みをトリガするようマルチカウンター ウォッチドッグ (MCWDT) コンポーネントを設定します。MCWDT のクロックソースは、低周波数クロック (LFCLK) です。LFCLK のソースは、デフォルトでは内部低速発振器 (ILO) です。このデザインはこの割込みを使用して、MILD アラートが受信されるとアラート LED を 2Hz で点滅させます。図 34 を参照してください。回路図に配置されたコンポーネントをダブルクリックし、以下のように設定します。

- **Name** を「**MCWDT**」に変更します。
- Counter0 で、**Match** の値を **7999**、**Mode** を **Interrupt** に変更します。**Clear on Match** フィールドを **Clear on match** に変更します。
- **OK** をクリックして、MCWDT コンポーネントの設定を完了します。

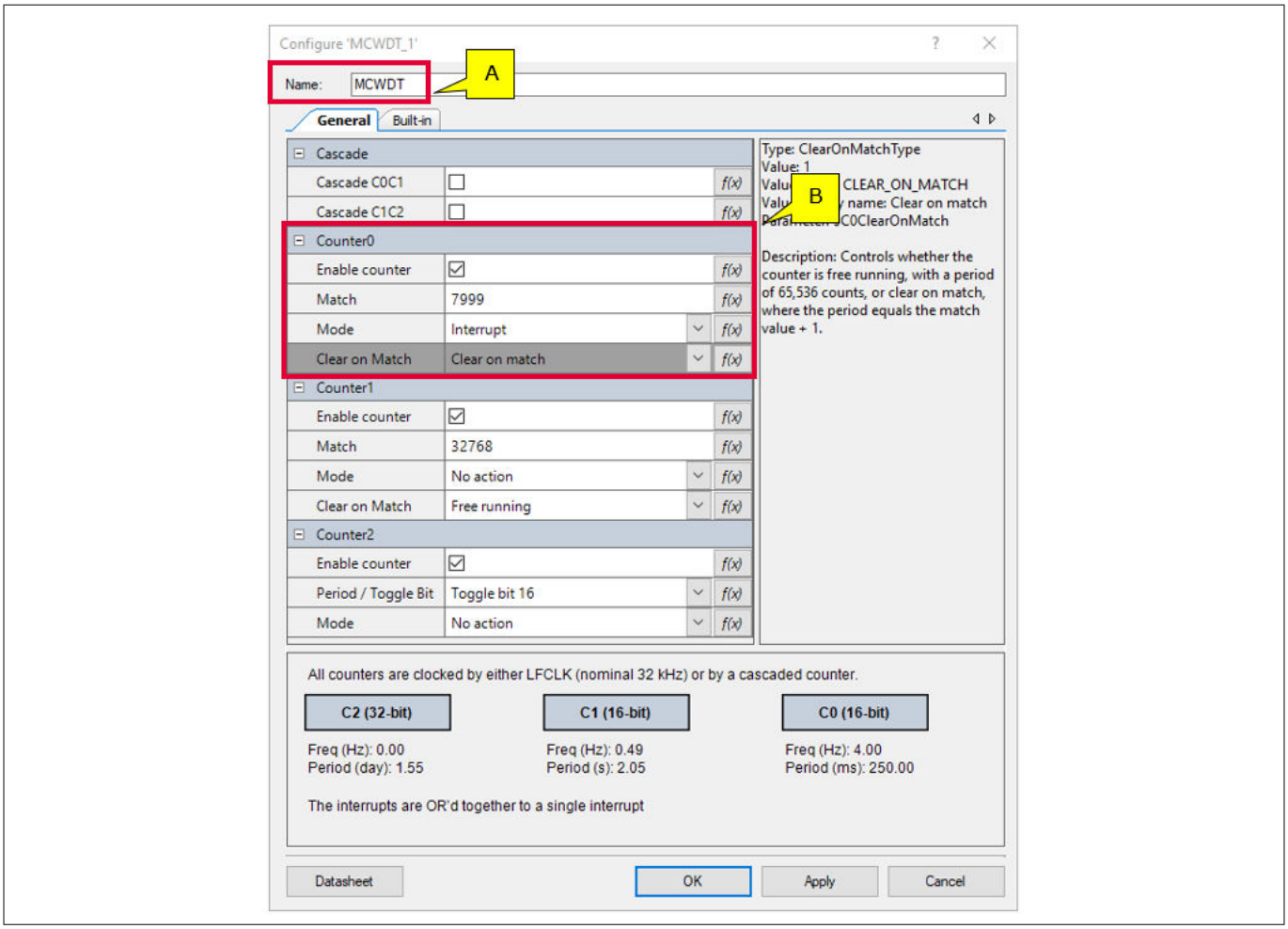


図 34　　MCWDT_PDL の設定

- **ディープスリープモードで CM4 CPU をウェイクアップするよう割込みコンポーネントを設定**

このステップでは、CM4 CPU をウェイクアップするように SysInt コンポーネントを設定します。図 35 を参照してください。

- **Name** を「**MCWDT_isr**」に変更します。
- 割込みの **Deep Sleep Capable** オプションを選択します。
- **OK** をクリックし、SysInt コンポーネントの設定を完了します。

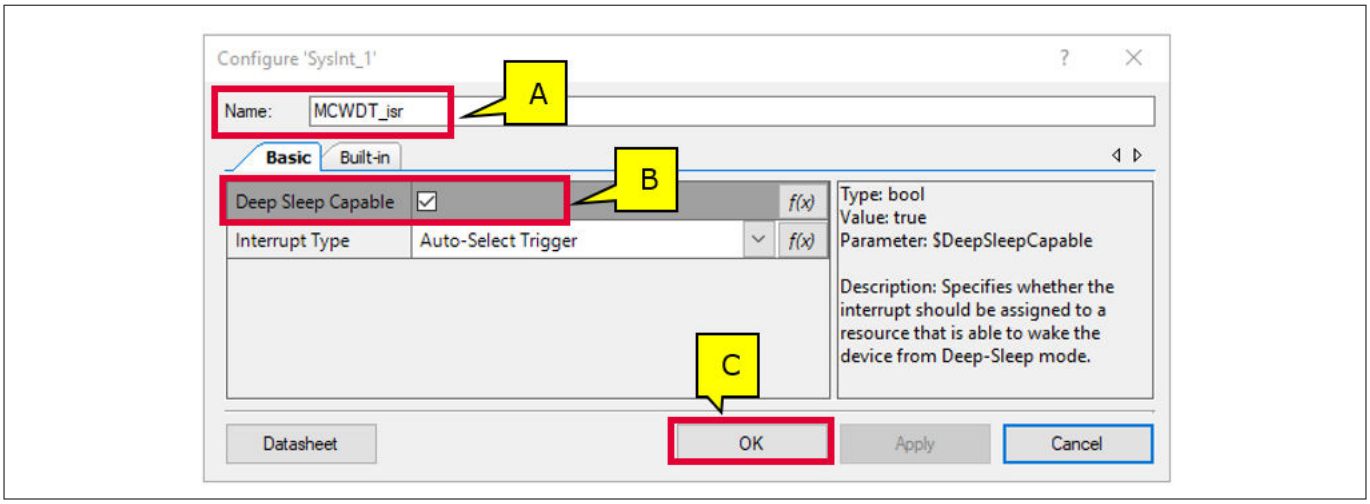


図 35　　SysInt_PDL の設定

最終ステップとして、MCWDT コンポーネントの割込み出力を MCWDT_isr コンポーネントの入力に接続します。このように、ウォッチドッグ割込みを CPU に接続します (この割込みの CM4 CPU での選択は後続ステップでシステム割込み設定でセットされます)。回路図で、ワイヤー ツール ボタンを使用するか、または「**W**」キーを押して、コンポーネントの配線を開始します。

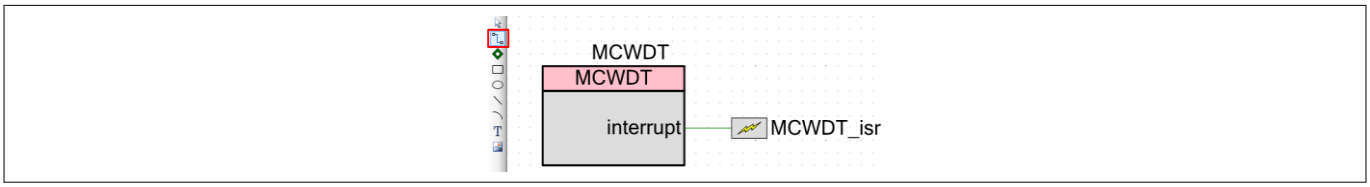


図 36　　MCWDT ペリフェラル割込みを CM4 CPU に接続

- **各ピン コンポーネントの物理ピンを設定**

このステップは、デザインを完了するための最後のタスクです。各コンポーネントをデバイス上の必要な物理ピンに関連付ける必要があります。使用するピンは基板設計によって決まります。この情報はキット回路図からご利用できます。図 37 にこのステップの最終結果を示します。

ピンを設定するために、対応するフィールドにポート番号やピン番号を入力するか、またはドロップダウン メニューでポートやピンを選択します。一般的には、ポート番号は使用する特定のパッケージに依存しないため、ピン番号の代わりに使用されています。

- ピン セレクタを開きます。

Workspace Explorer ペインで、Design Wide Resources の下にある **Pins** 項目をダブルクリックします。デバイスのピン セレクタが表示されます。

- 表 1 に示すように各ピンを設定します。

表 1　　CY8CKIT-062-BLE における物理ピン割当て

ピン コンポーネント名	ポート名
UART_DEBUG:rx	P5[0]
UART_DEBUG:tx	P5[1]
Advertising_LED	P1[1]
Alert_LED	P11[1]
Disconnect_LED	P0[3]
Hibernate_Wakeup_SW	P0[4]

(続く)

表 1 (続き) CY8CKIT-062-BLE における物理ピン割当て

ピン	コンポーネント名	ポート名

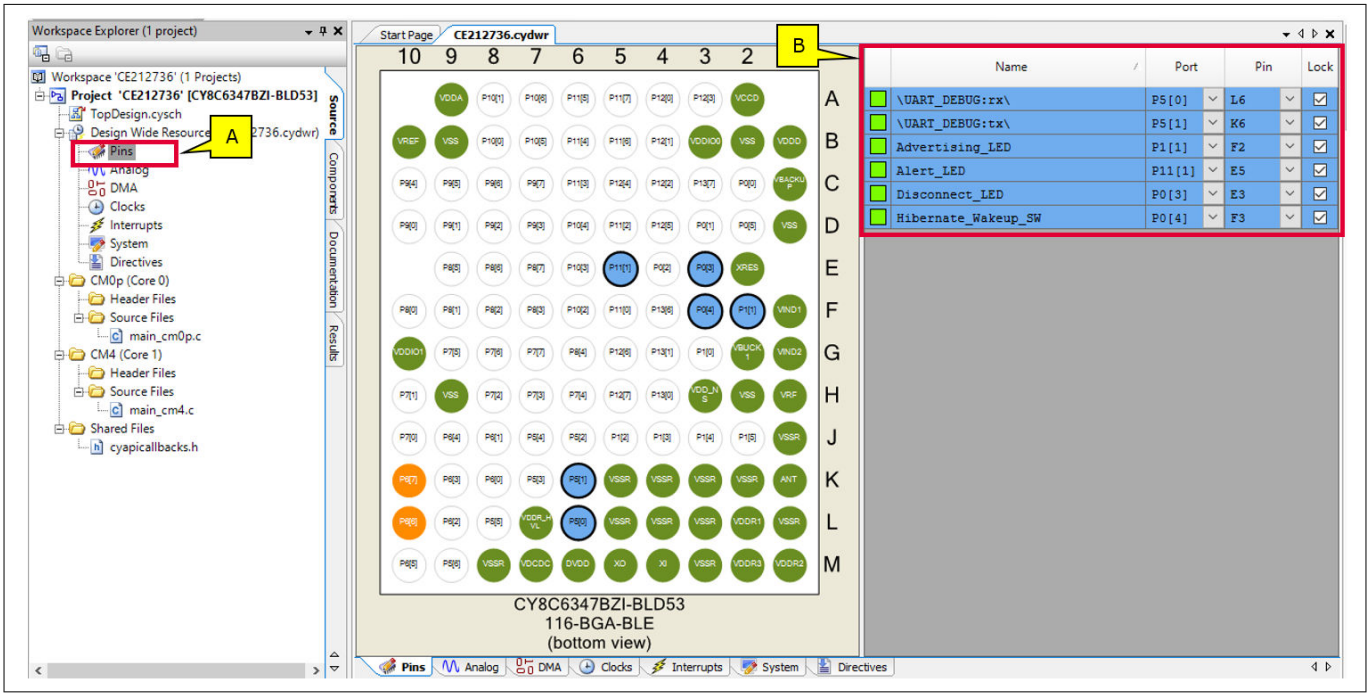


図 37 ピン割当て

- **システム クロックの設定**

このデザインでは、高周波数クロックの設定にデフォルト値を使用します。このデザイン用に高周波数クロックを修正しませんが、PSoC™ Creator がそれらをどのように管理するかを理解することが必要です。自分の基板をご使用の場合は、それらを修正する必要があることがあります。

このステップでは、基板上の正確な時計用水晶発振器 (WCO) になるように低周波数クロックソースを設定します。このクロックは、タイミングの目的に Bluetooth® LE サブシステム (BLESS) によって使用されます。

- **Workspace Explorer** ペインで、**Design Wide Resources (CE212736.cydwr)** の下にある **Clocks** 項目をダブルクリックします。クロック一覧が表示されます。
- **Edit Clock...**をクリックして、**Configure System Clocks** ダイアログが表示します。
- ここにはクロック ツリーが表示され、必要に応じてクロックを修正できます。**Source Clocks**、**FLL/PLL**、**High Frequency Clocks**、および **Miscellaneous Clocks** の異なるクロック タイプ タブがあることにご注意ください。
- **Source Clocks** タブをクリックします。
- **AltHF BLE ECO** ブロックでチェックボックスを選択して BLE ECO を有効にします。パラメーターは基板搭載の水晶と一致する必要があります。CY8CKIT-062-BLE キットでは、**ECO Frequency** が **32 MHz**、**Accuracy** が**± 50 ppm** です。基板上に負荷容量がないため、**9.900** の指定された最小の **Load cap (pF)**を使用します。ECO 水晶発振器の迅速な起動のために、**Startup Time (µs)**が **785 µs** であることを確認します。
- **WCO (32.768 kHz)** ブロックでチェックボックスを選択して WCO クロックを有効にします。
- **Miscellaneous Clocks** タブをクリックします。
- **WCO** を **LFClk** のソースに選択します。
- **WCO** を **BakClk** のソースに選択します。

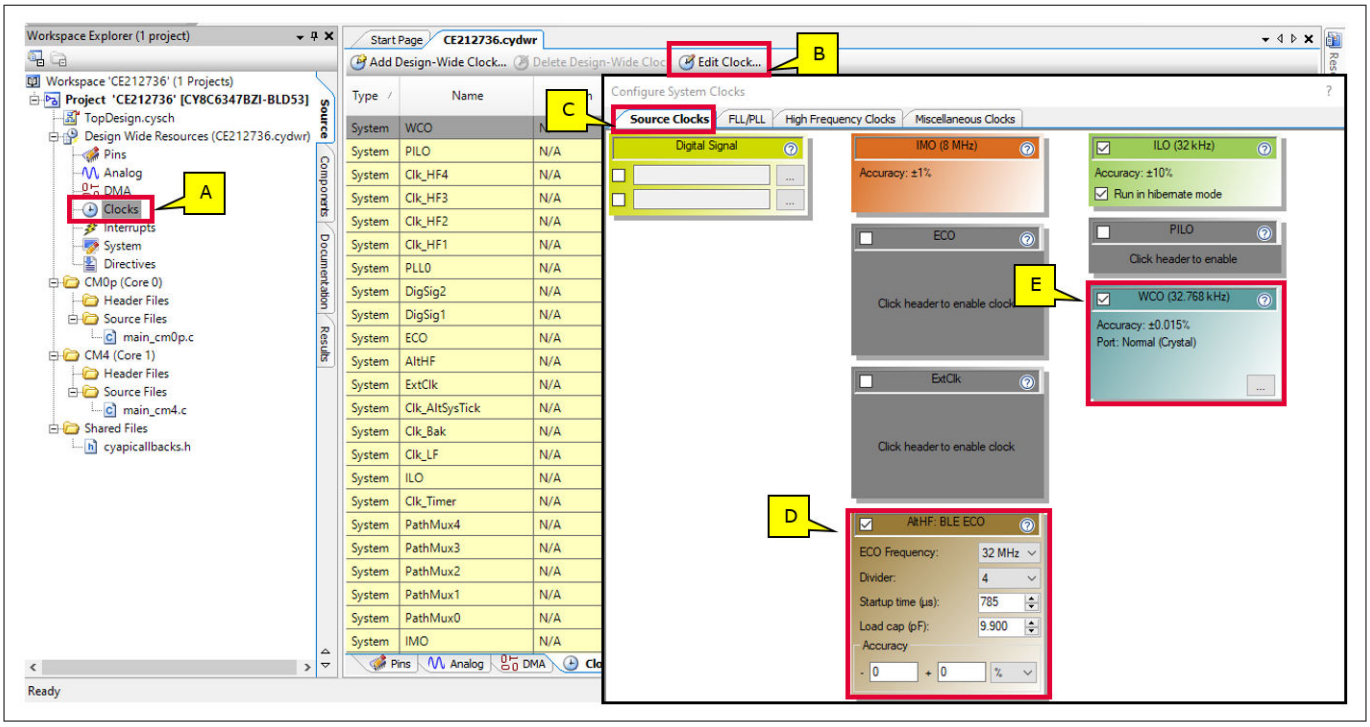


図 38　　クロックの設定 - Source clock

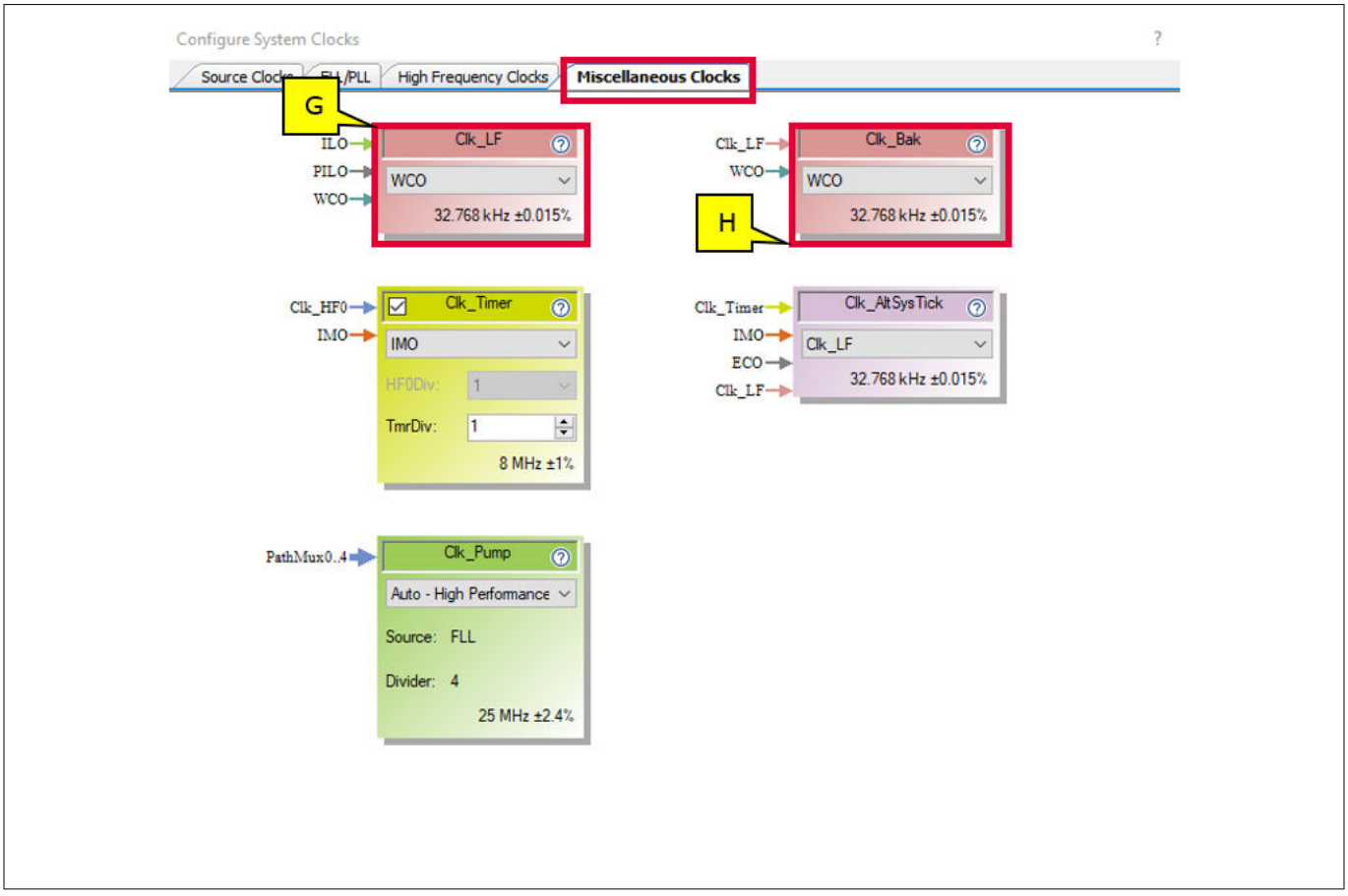


図 39　　クロックの設定 - Miscellaneous clock

- **システム割込みの設定**

このステップでは、システムの割込みを設定します。図 40 を参照してください。

- **Workspace Explorer** ペインで、**Design Wide Resources** の下にある **Interrupts** 項目をダブルクリックします。割込み一覧が表示されます。
- CM0+ CPU で **BLE_bless_isr** が有効にされ、優先度が **1**、ベクトルが **3** にセットされていることを確認します。このように、Bluetooth® LE コントローラーの割込みが CM0+によって最優先で確実に処理され、またディープスリープモードにおいて確実にも処理されるようにします。
- CM4 では **MCWDT_isr** を有効にします。
- 両方のコアで **UART_DEBUG_SCB_IRQ** 割込みを無効にします。
- 対応するチェックボックスのチェックを外します。このデザインでは、UART_DEBUG コンポーネントに関連する割込みを使用しないため、その割込みは無視できます。

「パート 3: ソース コードの生成」でコードを生成すると、割込み番号が PSoC™ Creator によって自動的に生成されます。

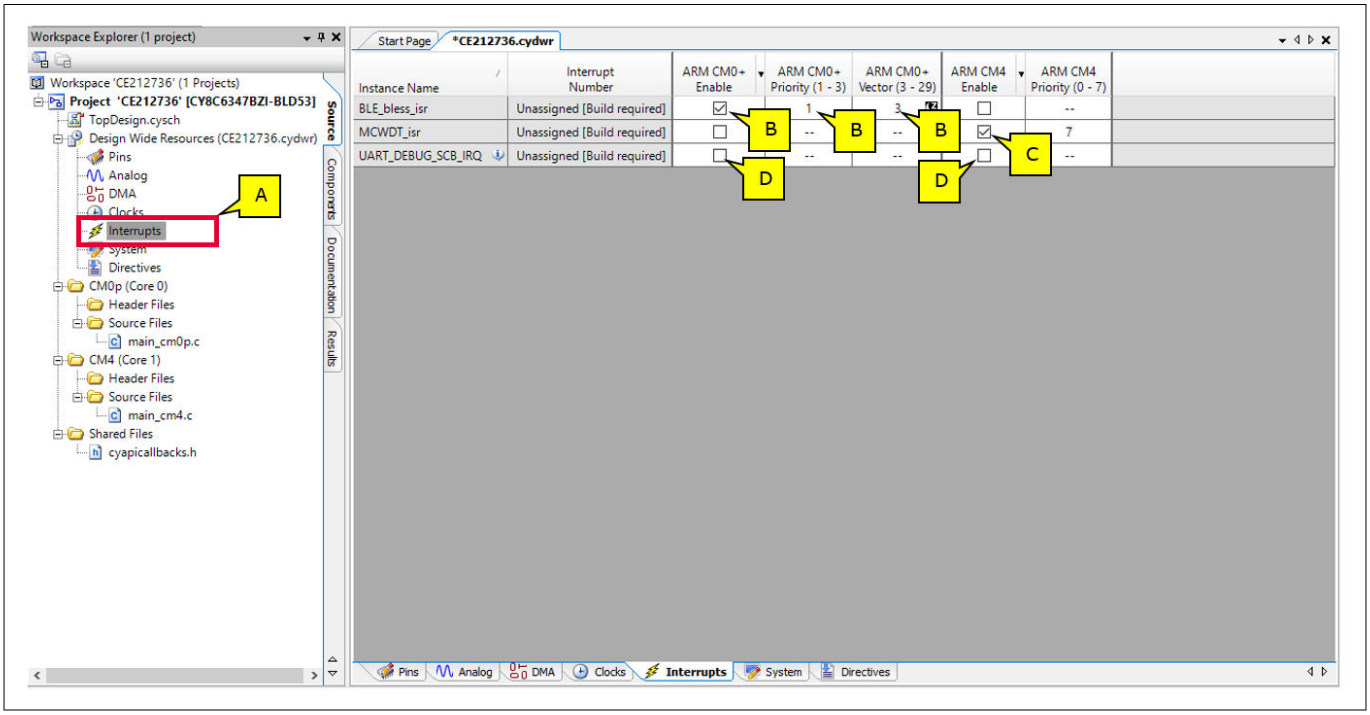


図 40　　割込みの設定

開発プロセスの次の段階はコードの生成です。

注:　　この演習では、対象の IDE へのエクスポートの詳細を説明しません。ただし、対象の IDE を使用する場合は、コードの生成前に、正しい対象の IDE が選択されていることを確認するワークフロー内の要点です。他の IDE のサポートを参照してください。

5.6　　パート 3: ソース コードの生成

PSoC™ Creator はデザインに基づいてソース コードを生成します。推奨するワークフローとしては、ファームウェアを書く前にコードを生成します。PSoC™ Creator は、ファームウェアで使用されるマクロ、定数および API 呼び出しを自動的に作成します。

このパートは非常に簡単で、パスはどなたでも同じです。

パス	最初からの作業サンプル コードを参考とする	サンプル コードの使用 PSoC™ Creator または Bluetooth® LE に詳しくない	サンプル コードを使用 PSoC™ Creator と Bluetooth® LE に精通している
アクション	唯一のステップを行う	唯一のステップを行う	唯一のステップを行う

- **アプリケーションを生成**

Build > **Generate Application** を選択します。PSoC™ Creator はデザインに基づいてソース コードを生成し、ファイルを *Generated_Source* フォルダに保存します。図 41 を参照してください。PSoC™ Creator は、発生する可能性があるエラーや問題について警告します。最初から作業を行い、かつエラーが発生した場合、「パート 2: 設計の実装」での設定手順を再検討して、それらが正しく行われたことを確認してください。

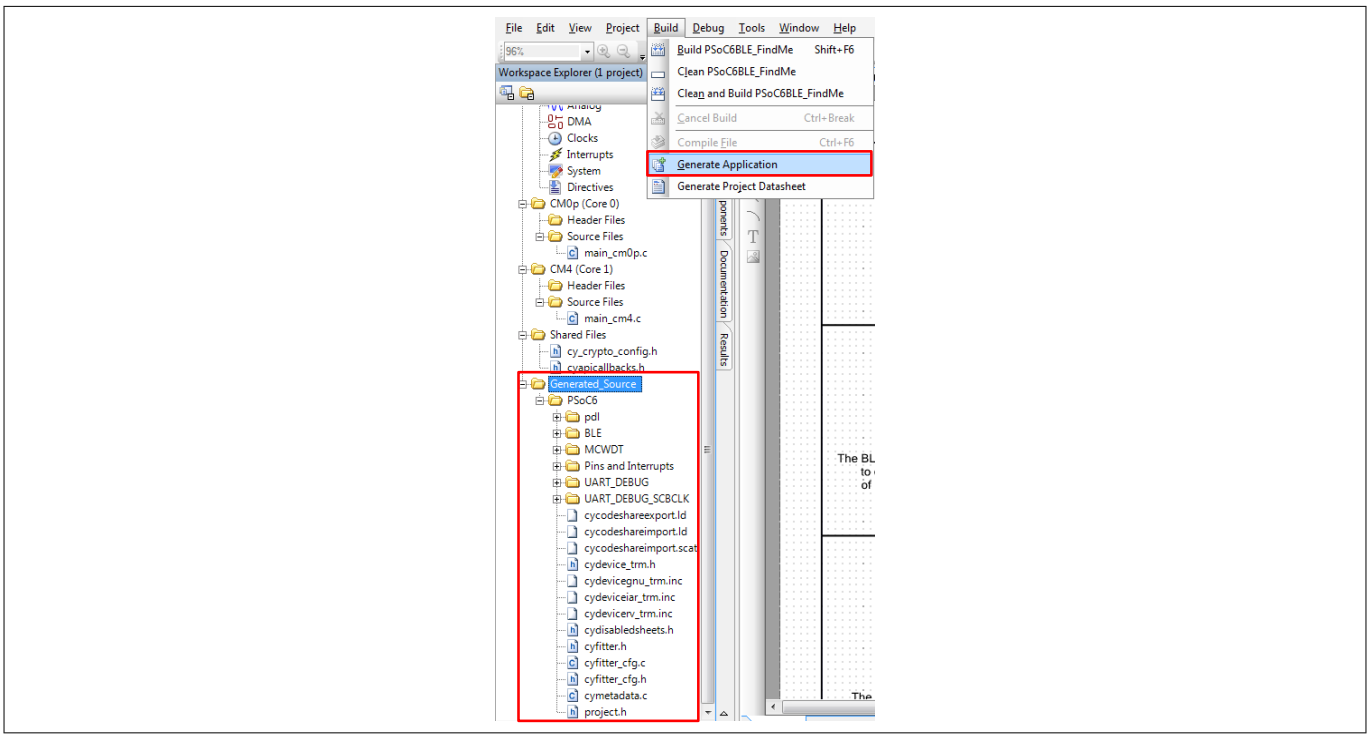


図 41　　アプリケーションの生成

バックグラウンド: PSoC™ 6-BLE はデュアル CPU プラットフォームです。ファームウェアは、CM4 または CM0+のいずれかで実行するようにターゲット設定できます。これは、ファイル プロパティへアクセスすることでソースファイル レベルで設定します。ソースファイルを右クリックして、**Properties** を選択します。

図 41 に、**Properties** ダイアログ ウィンドウを示します。このサンプル コードは、CM4 コアを対象とします。

デフォルトでは、*main_cm0p.c* ファイルは CM0+を、*main_cm4.c* ファイルは CM4 を対象とします。その他のファイルに対しては、プロパティを修正する必要はありません。それらは既にサンプル コードで設定されています。

通常、CM0+向けのファイルは *CM0p* フォルダに、CM4 向けのファイルは *CM4* フォルダに配置されていますが、プロパティも適切に設定する必要があります。

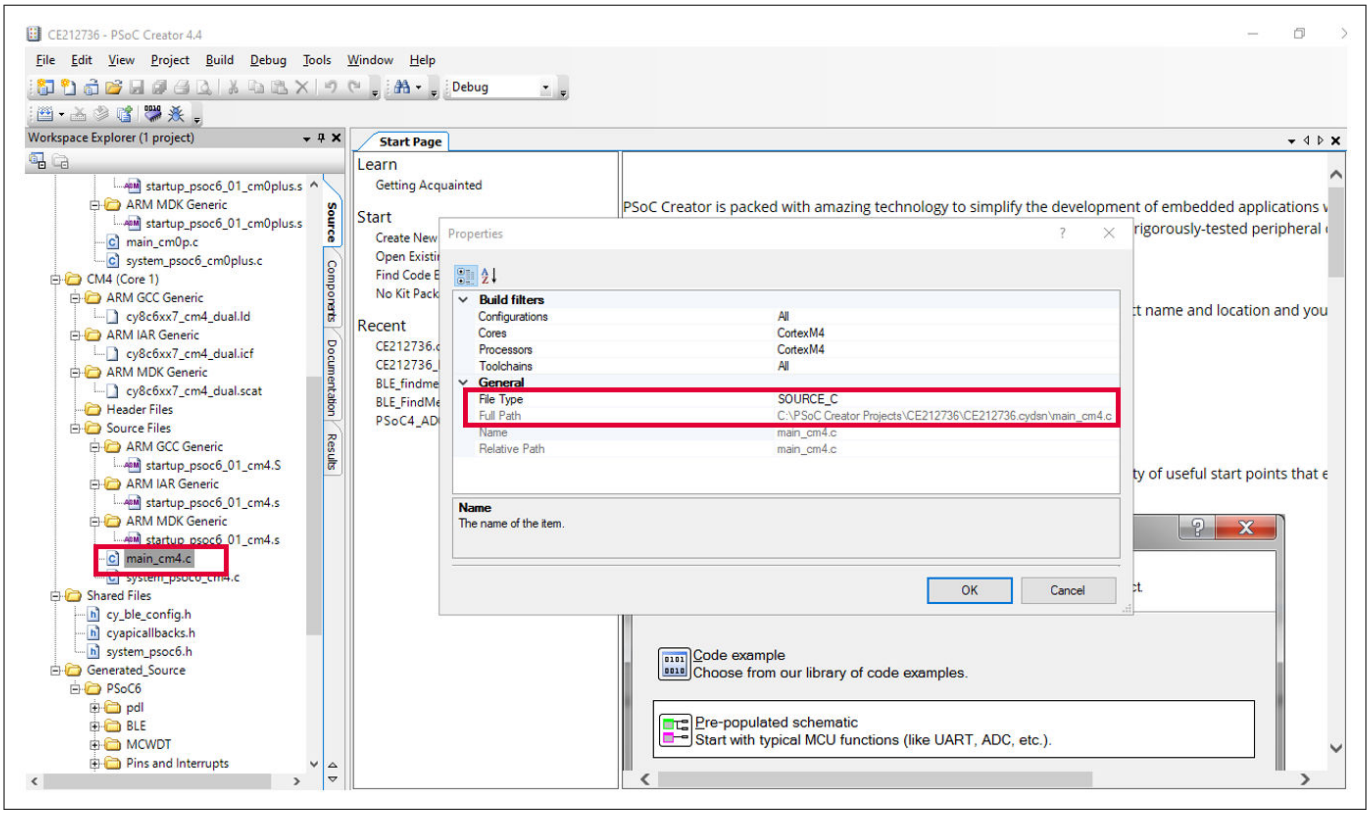


図 42　　ソース C ファイル用のターゲットプロセッサを設定

5.7　パート 4: ファームウェアの記述

開発プロセスのこの時点では、プロジェクトの作成、ハードウェア設計の実装およびコードの生成が行われました。このパートでは、アプリケーションに Bluetooth® LE 機能を実装するファームウェアを検討します。

ファームウェアは Bluetooth® LE 標準のプロファイル アプリケーションを実装するために、以下を含むいくつかのタスクを遂行しなければなりません。

- システム初期化を実行
- BLE スタック イベント ハンドラを実装
- BLE サービス固有イベント ハンドラを実装
- メイン ループを実行
- 低消費電力の性能を実装 (任意)

このパートの手順では、「パート 2: 設計の実装」で設定したデザイン向けのファームウェアを説明します。コードの 1 つ 1 つの行を検討しませんが、重要な機能を実装するコード内の重要な要素を指摘します。

パス	最初からの作業 サンプル コードを参考とする	サンプル コードの使用 PSoC™ Creator または Bluetooth® LE に詳しくない	サンプル コードの使用 PSoC™ Creator と Bluetooth® LE に精通している
アクション	すべてのステップを行う	ステップ 1 を飛ばす。 残りのステップを行う	この部分を飛ばすことができる。パート 5: プロジェクトのビルド、デバイスのプログラミングへ進む

サンプル コードにはすべての必要なコードがあります。最初から作業を行う場合、ステップ 1 でサンプル コードのプロジェクトからソース ファイルをコピーします。サンプル コードを使用する場合、それらのファイルは既にプロジェクトに含まれているため、ステップ 1 を飛ばしても構いません。

- **プロジェクトにファイルを追加**

サンプル コードを使用する場合、このステップを飛ばしても構いません。サンプル コードには既に必要なソース ファイルがあります。

最初から作業を行う場合、必要なソース コード ファイルはご使用のプロジェクトにありません。

- サンプル コード用のソースファイルを格納する *CE212736.cydsn* フォルダを探します。このフォルダは、以前ダウンロードしたサンプル コード ワークスペース アーカイブにあります。
- これらのファイルを *CE212736.cydsn* フォルダからプロジェクトの*.cydsn* フォルダにコピーします。既存のファイルを置き換えます。
  - BLEFindMe.h
  - debug.h
  - LED.h
  - BLEFindMe.c
  - debug.c
  - main_cm0p.c
  - main_cm4.c
- これらのファイルをプロジェクトに追加します。このためには、ファイルを *Windows* フォルダから Workspace Explorer にドラッグし、ワークスペース内の *CM4* フォルダの場所にドロップします。図 43 を参照してください。
  - BLEFindMe.h
  - debug.h
  - LED.h
  - BLEFindMe.c
  - debug.c

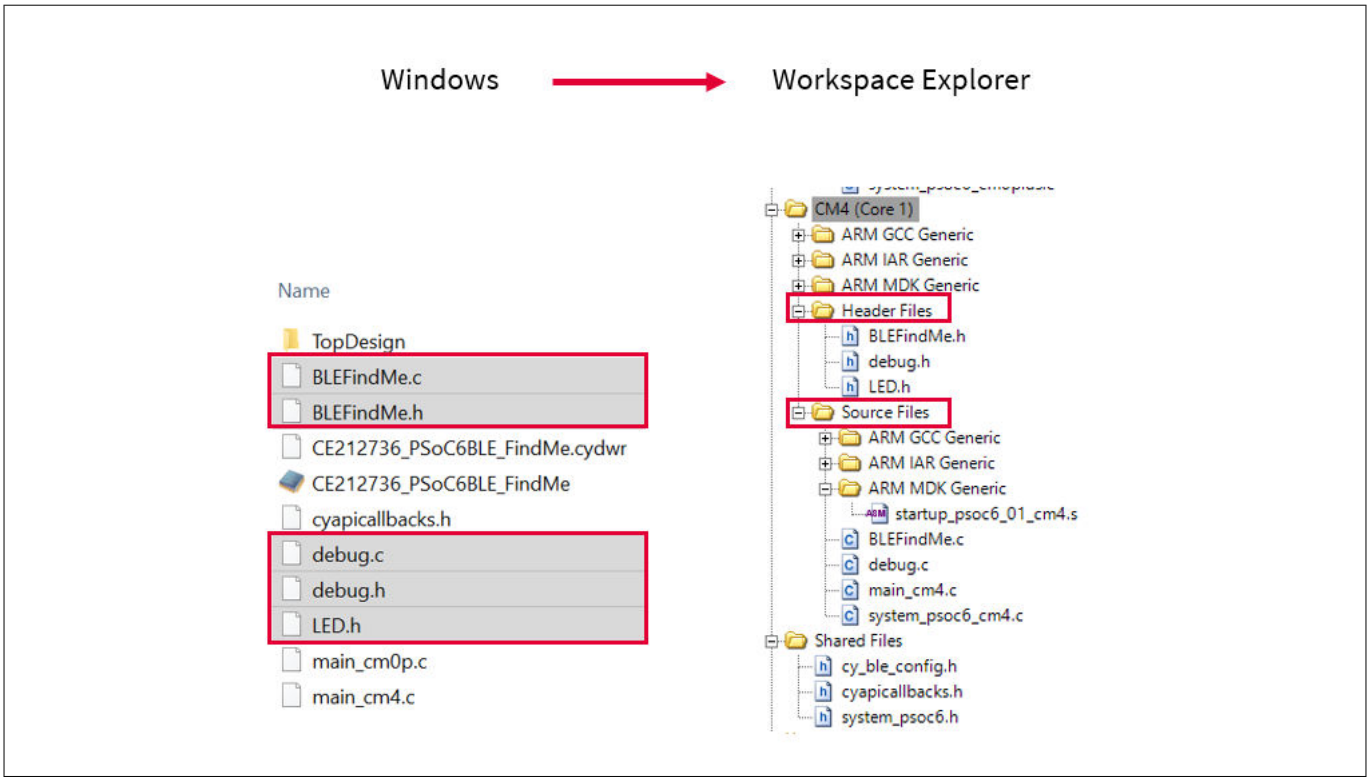


図 43　　プロジェクトにファイルを追加

プロジェクトには、デフォルトで *main_cm0p.c* と *main_cm4.c* ファイルがあるため、それらを追加する必要はありません。プロジェクトが新しいバージョンを使用できるように単に *project* フォルダでそれらを置き換えます。

- **システムを初期化**

残りのステップで、*main_cm0p.c* および *main_cm4.c* ファイル内のコードを検討します。明確にするために、プログラムコードのデバッグ プリント文が削除される場合があります。コードを完全に理解するために、実際のソースファイルを参照してください。

図 44 に、システムを初期化するプロセスの手順を示します。PSoC™ 6-BLE デバイスがリセットされると、ファームウェアはまずシステムの初期化を実行します。このプロセスでは、実行用の CPU コアのセットアップ、グローバル割込みおよび設計で使用するコンポーネントを有効にします。初期化プロセスは CPU コア間で分割されます。CM0p CPU はリセットを終了し、Bluetooth® LE コントローラー部分の開始を試みます。成功したら、CM0p CPU は CM4 CPU を有効にします。CM4 CPU は Bluetooth® LE ホストの部分を開始し、アプリケーション側の必要なハンドラ関数を登録します。

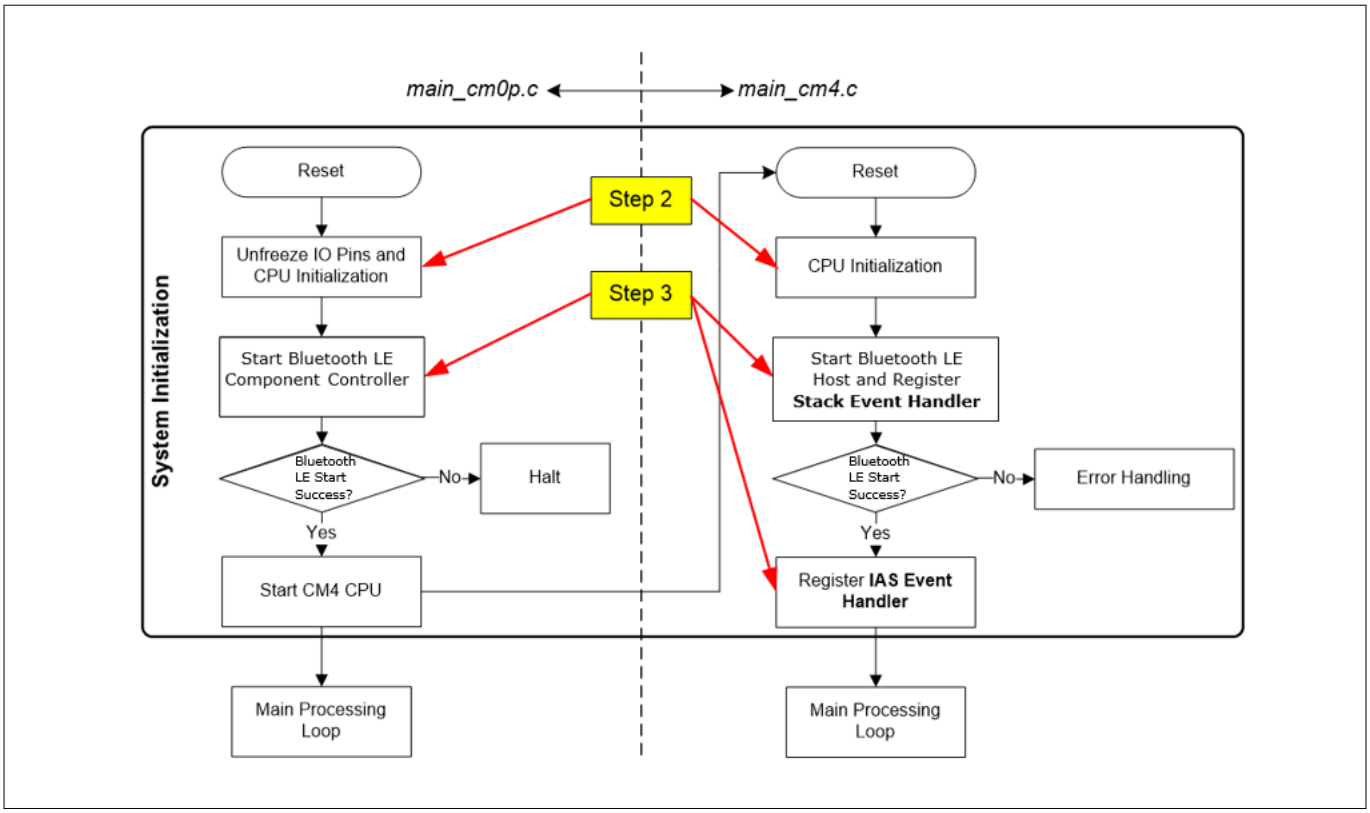


図 44　　システム初期化のフローチャート – CM4

main_cm0p.c のコードは、Bluetooth® LE API 呼び出しからの返り値を保持するローカル変数を宣言します。主なタスクは、Bluetooth® LE コントローラーを有効にし、アプリケーションコードを実行するための CM4 をセットアップ

することです。メインループでは、CM0p CPU はコントローラーで保留中の Bluetooth® LE イベントを処理します。保留中のイベントがない時、CM0p はディープスリープモードに入ります。

```
int main(void)
{
  cy_en_ble_api_result_t          apiResult;
  __enable_irq();    /* Enable global interrupts. */
  /* Unfreeze IO if device is waking up from hibernate */
  if(Cy_SysPm_GetIoFreezeStatus())
  {
    Cy_SysPm_IoUnfreeze();
  }
  /* Start the Controller portion of BLE. Host runs on the CM4 */
  apiResult = Cy_BLE_Start(NULL);
  if(apiResult == CY_BLE_SUCCESS)
  {
    /* Enable CM4 only if BLE Controller started successfully.
    * CY_CORTEX_M4_APPL_ADDR must be updated if CM4 memory layout
    * is changed. */
    Cy_SysEnableCM4(CY_CORTEX_M4_APPL_ADDR);
  }
  else
  {
    /* Halt CPU */
    CY_ASSERT(0u != 0u);
  }
  for(;;)
  {
    /* Place your application code here. */
    /* Put CM0p to deep sleep. */
    Cy_SysPm_DeepSleep(CY_SYSPM_WAIT_FOR_INTERRUPT);

    /* Cy_Ble_ProcessEvents() allows BLE stack to process pending events */
    /* The BLE Controller automatically wakes up host if required */
    Cy_BLE_ProcessEvents();
}
```

main_*cm4.c* のコードは、CM4 が使用する主なコンポーネントを初期化し、Bluetooth® LE アプリケーションプロセスを引き続き実行します。BleFindMe_Init() ルーチンは、CM4 割込みのセットアップを含む、すべてのコンポーネントを初期化および起動を行います。主なタスクは Bluetooth® LE ホストを有効にすることです。

アプリケーションプロセスでは、CM4 CPU はホスト上で保留中の Bluetooth® LE イベントを処理します。保留中のイベントがない場合、CM4 はディープスリープ低消費電力モードに入ります。

```
int main(void)
{
  __enable_irq(); /* Enable global interrupts. */

  /* Initialize BLE */
  BleFindMe_Init();

  for(;;)
  {
    BleFindMe_Process();
  }
}
```

- **Bluetooth® LE コンポーネントを起動し、イベントハンドラを登録**

CPU が初期化された後、ファームウェアは Bluetooth® LE コンポーネントを初期化し、その結果、Bluetooth® LE サブシステムが完全にセットアップされます。BleFindMe_Init()サブルーチンが実行します。主なタスクに集中するために、デバッグ プリント文は削除されます。完全なコードを検証するために、ソース ファイルを参照してください。

Bluetooth® Low Energy (Bluetooth® LE) コンポーネントの初期化プロセスの一部として、保留中のイベントを通知するために Bluetooth® LE スタックによって呼び出されるイベント ハンドラ関数を正常に実行する必要があります。Bluetooth® LE コンポーネントの初期化が正常に完了すると、ファームウェアは IAS 固有イベント用の第 2 のイベント ハンドラを登録します。

コードは PDL API 関数の呼び出しを使用してアプリケーションを設定します。まず、Bluetooth® LE コンポーネントを起動しますパラメーターはスタック イベント ハンドラ関数のアドレスです。

コードはスタックバージョンも取得します。デバッグポートが有効な場合、バージョンはシリアル通信ウィンドウに表示されます。

その後、即時アラートサービス関連のイベントを処理するための IAS イベントハンドラを登録します。最後に、250ms ごとに割込みを 1 回トリガできるよう MCWDT を設定します。

```
void BleFindMe_Init(void)
{
  cy_en_ble_api_result_t apiResult;
  cy_stc_ble_stack_lib_version_t stackVersion;

  /* Configure switch SW2 as hibernate wake up source */
  Cy_SysPm_SetHibWakeupSource(CY_SYSPM_HIBPIN1_LOW);

  /* Start Bluetooth® Low Energy Component and register generic event handler */
  apiResult = Cy_BLE_Start(StackEventHandler);
  apiResult = Cy_BLE_GetStackLibraryVersion(&stackVersion);


  /* Register IAS event handler */
  Cy_BLE_IAS_RegisterAttrCallback(IasEventHandler);

  /* Enable 4 Hz free-running MCWDT counter 0*/
  /* MCWDT_config structure is defined by the MCWDT_PDL component based
     on parameters entered in the customizer. */
  Cy_MCWDT_Init(MCWDT_HW, &MCWDT_config);
  Cy_MCWDT_Enable(MCWDT_HW, CY_MCWDT_CTR0, 93 /* 2 LFCLK cycles */);

  /* Unmask the MCWDT counter 0 peripheral interrupt */
  Cy_MCWDT_SetInterruptMask(MCWDT_HW, CY_MCWDT_CTR0);

  /* Configure ISR connected to MCWDT interrupt signal*/
  /* MCWDT_isr_cfg structure is defined by the SYSINT_PDL component based
     on parameters entered in the customizer. */
  Cy_SysInt_Init(&MCWDT_isr_cfg, &MCWDT_Interrupt_Handler);

  /* Clear CM4 NVIC pending interrupt for MCWDT */
  NVIC_ClearPendingIRQ(MCWDT_isr_cfg.intrSrc);

  /* Enable CM4 NVIC MCWDT interrupt */
  NVIC_EnableIRQ(MCWDT_isr_cfg.intrSrc);
}
```

- **スタック イベント ハンドラを実装**

Bluetooth® LE コンポーネント内の Bluetooth® LE スタックはイベントを生成します。それらのイベントは Bluetooth® LE スタックイベントハンドラをとおして、アプリケーションファームウェアにステータスおよびデータを提供します。図 45 に、特定のイベントを表す簡素化したフローチャートを示します。

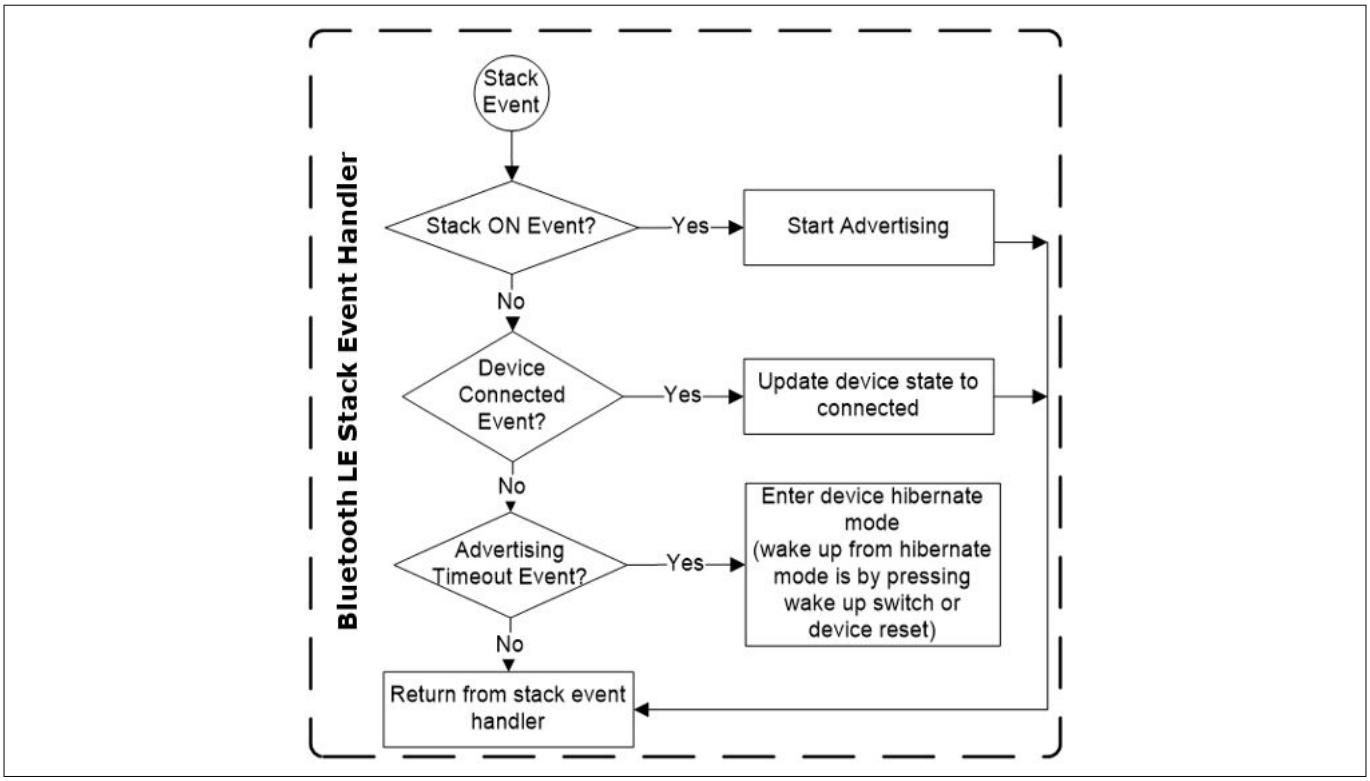


図 45　　BLE スタックイベントハンドラのフローチャート

イベント ハンドラはスタックからのいくつかの基本的なイベントを処理する必要があります。このサンプル コードでのファインド ミー ターゲット アプリケーションに対しては、BLE スタック イベント ハンドラは表 2 で説明するイベントを処理しなければなりません。実際のコードは追加のイベントを認識し、それらに応答しますが、このアプリケーションには必須ではありません。

表 2　　Bluetooth® LE スタック イベント

Bluetooth® LE スタック イベント名	イベントの説明	イベント ハンドラの応答処理
CY_BLE_EVT_STACK_ON	Bluetooth® LE スタックの初期化が正常に完了した	アドバタイズを開始し、アドバタイズ状態を LED に反映
CY_BLE_EVT_GAP_DEVICE_DISCONNECTED	ピア デバイスとの Bluetooth® LE 接続が切断	アドバタイズを再始動し、アドバタイズ状態を LED に反映
CY_BLE_EVT_GAP_DEVICE_CONNECTED	ピア デバイスとの Bluetooth® LE 接続が確立	BLE 接続状態を LED に更新
CY_BLE_EVT_GAPP_ADVERTISEMENT_START_STOP	Bluetooth® LE スタックのアドバタイズ開始/停止イベント	Bluetooth® LE スタックをシャットダウン
CY_BLE_EVT_HARDWARE_ERROR	Bluetooth® LE ハードウェアエラー	ハードウェア エラーを反映するように LED 状態を更新し、CPU を停止
CY_BLE_EVT_STACK_SHUTDOWN_COMPLETE	BLE スタックがシャットダウン	ハイバネートモードにデバイスを設定し、ウェイクアップピン上のイベントを待機

以下のプログラムコードは、イベントハンドラが特定のイベントにどのように応答するかを示す 2 つの例です。完全に理解するために、実際のソースコードを参照してください。

このプログラムコードで、ハンドラは「アドバタイズの開始/停止」イベントに応答します。コードは LED を適切に切り替えます。アドバタイズが開始すると、アドバタイズ LED は点灯します。デバイスがアドバタイズを開始し、接続

の準備ができたため、切断 LED は消灯します。アドバタイズが停止したら、コードは LED を適切にセットし、ハイバネートモードに移行するためのフラグをセットします。

```
/* This event indicates peripheral device has started/stopped advertising */
case CY_BLE_EVT_GAPP_ADVERTISEMENT_START_STOP:
  DEBUG_PRINTF("CY_BLE_EVT_GAPP_ADVERTISEMENT_START_STOP: ");
  if(Cy_BLE_GetAdvertisementState() == CY_BLE_ADV_STATE_ADVERTISING)
  {
    DEBUG_PRINTF("Advertisement started \r\n");
    Cy_GPIO_Write(Advertising_LED_0_PORT, Advertising_LED_0_NUM, LED_ON);
    Cy_GPIO_Write(Disconnect_LED_0_PORT, Disconnect_LED_0_NUM, LED_OFF);
  }
  else if(Cy_BLE_GetAdvertisementState() == CY_BLE_ADV_STATE_STOPPED)
  {
    DEBUG_PRINTF("Advertisement stopped \r\n");
    Cy_GPIO_Write(Advertising_LED_0_PORT, Advertising_LED_0_NUM, LED_OFF);
    Cy_GPIO_Write(Disconnect_LED_0_PORT, Disconnect_LED_0_NUM, LED_ON);


    /* Advertisement event timed out before connection, shutdown BLE
     * stack to enter hibernate mode and wait for device reset event
     * or SW2 press to wake up the device */
    Cy_BLE_Stop();
  }
break;
```

このプログラムコードでは、ハンドラは「切断」イベントに応答します。LED を正しくセットし、ハイバネートフラグをセットします。

これらのプログラムコードは、イベント ハンドラがイベントにどのように応答するかを示します。各イベントの処理方法を確認するために、実際の関数を調べてください。

```
/* This event is generated when disconnected from remote device or
failed to establish connection. */

case CY_BLE_EVT_GAP_DEVICE_DISCONNECTED:
  if(Cy_BLE_GetConnectionState(appConnHandle) == CY_BLE_CONN_STATE_DISCONNECTED)
  {
    DEBUG_PRINTF("CY_BLE_EVT_GAP_DEVICE_DISCONNECTED %d\r\n", CY_BLE_CONN_STATE_DISCONNECTED);
    alertLevel = CY_BLE_NO_ALERT;
    Cy_GPIO_Write(Advertising_LED_0_PORT, Advertising_LED_0_NUM, LED_OFF);
    Cy_GPIO_Write(Disconnect_LED_0_PORT, Disconnect_LED_0_NUM, LED_ON);

    /* Enter into discoverable mode so that remote device can search it */
    apiResult = Cy_BLE_GAPP_StartAdvertisement(CY_BLE_ADVERTISING_FAST,
CY_BLE_PERIPHERAL_CONFIGURATION_0_INDEX);
    if(apiResult != CY_BLE_SUCCESS)
    {
      DEBUG_PRINTF("Start Advertisement API Error: %d \r\n", apiResult);
      ShowError();

      /* Execution does not continue beyond this point */
    }
    else
    {
      DEBUG_PRINTF("Start Advertisement API Success: %d \r\n", apiResult);
      Cy_GPIO_Write(Advertising_LED_0_PORT, Advertising_LED_0_NUM, LED_ON);
      Cy_GPIO_Write(Disconnect_LED_0_PORT, Disconnect_LED_0_NUM, LED_OFF);
    }
  }
break;
```

- **サービス固有のイベント ハンドラを実装**

Bluetooth® LE コンポーネントはまた、デザインでサポートされる各サービスに対応するイベントを生成します。ファインド ミー ターゲット アプリケーションでは、Bluetooth® LE コンポーネントはアラートレベル特性が新しい値で更新されたことをアプリケーションに知らせる IAS イベントを生成します。イベント ハンドラは新しい値を取得し、それを alertLevel 変数に格納します。メイン ループは現時点のアラート レベルに基づいてアラート LED を切り替えます。

図 46 に IAS イベント ハンドラ フローチャートを示します。

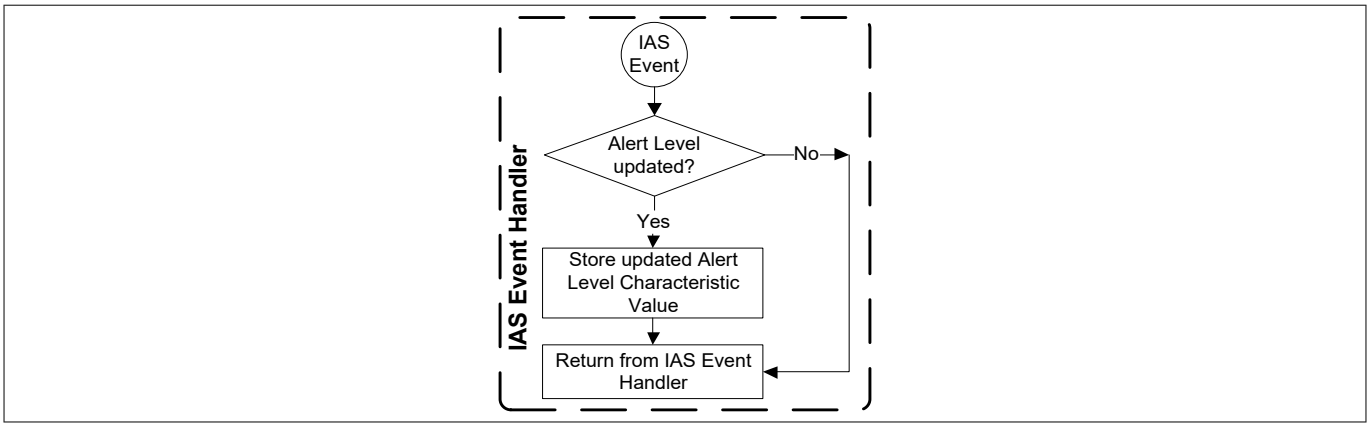


図 46 Bluetooth® LE IAS イベントハンドラのフローチャート

このプログラムコードは、ファームウェアがこのタスクをどのように遂行するかを示します。

```
void IasEventHandler(uint32 event, void *eventParam)
{

  /* Alert Level Characteristic write event */
  if(event == CY_BLE_EVT_IASS_WRITE_CHAR_CMD)
  {
    /* Read the updated Alert Level value from the GATT database */
    Cy_BLE_IASS_GetCharacteristicValue(CY_BLE_IAS_ALERT_LEVEL,
         sizeof(alertLevel), &alertLevel);
  }

  /* To remove unused parameter warning */
  eventParam = eventParam;
}
```

- **発生したイベントを処理 (メイン ループ)**

メインループは単に BleFindMe_Process() を呼び出します。図 47 に BleFindMe_Process() フローチャートを示します。

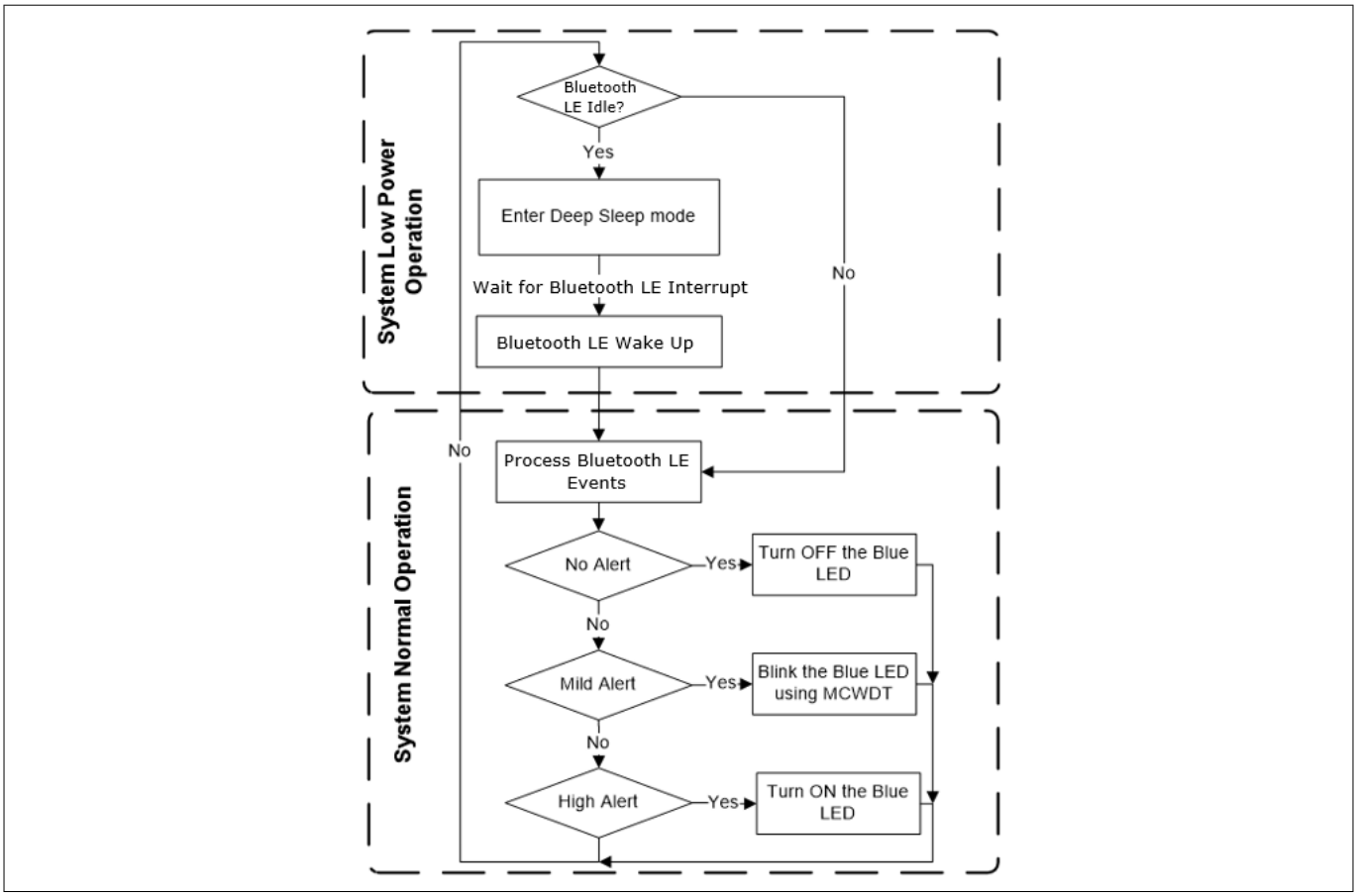


図 47　　ファームウェアのメインループのフローチャート

保留中の Bluetooth® LE ホストイベントがなく、アクティブな相互作用がない場合、アプリケーションは低消費電力モードに入ります。その後、Bluetooth® LE にイベントを処理するよう指示し、アラートレベルに基づいて LED を更新します。

```
        /* The call to EnterLowPowerMode also causes the device to enter hibernate mode if the
BLE is disconnected. */

  EnterLowPowerMode();
  /* Cy_Ble_ProcessEvents() allows BLE stack to process pending events */
  Cy_BLE_ProcessEvents();
  /* Update Alert Level value on the Blue LED */
  switch(alertLevel)
  {
    case CY_BLE_NO_ALERT:
      /* Disable MCWDT interrupt at NVIC */
      NVIC_DisableIRQ(MCWDT_isr_cfg.intrSrc);
      /* Turn the Blue LED OFF in case of no alert */
      Cy_GPIO_Write(Alert_LED_0, LED_OFF);
      break;

/* Use the MCWDT to blink the Blue LED in case of mild alert */
case CY_BLE_MILD_ALERT:
    /* Enable MCWDT interrupt at NVIC */
    NVIC_EnableIRQ(MCWDT_isr_cfg.intrSrc);
    /* The MCWDT interrupt handler will take care of LED blinking */
    break;
case CY_BLE_HIGH_ALERT:
    /* Disable MCWDT interrupt at NVIC */
    NVIC_DisableIRQ(MCWDT_isr_cfg.intrSrc);
    /* Turn the Blue LED ON in case of high alert */
    Cy_GPIO_Write(Alert_LED_0, LED_ON);
    break;
/* Do nothing in all other cases */
default:
    break;
}
```

以上、サンプル コードを用いてファームウェアの動作概要を説明しました。理解を深めるために、ご自由にソース ファイルをご活用ください。

5.8 パート 5: プロジェクトのビルド、デバイスのプログラミング

ここでは、PSoC™ 6-BLE デバイスのプログラム方法を説明します。内蔵プログラマを備えた開発キット (CY8CKIT-062-BLE PSoC™ 6-BLE Pioneer Kit など) をご使用の場合、USB ケーブルを使用して基板をコンピュータに接続します。ご自身でハードウェア開発をされている場合、CY8CKIT-002 MiniProg3 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。

パス	最初からの作業 サンプル コードを参考とする	サンプル コードの使用 PSoC™ Creator または Bluetooth® LE に詳しくない	サンプル コードの使用 PSoC™ Creator と Bluetooth® LE に精通している
アクション	すべてのステップを行う		

最初から作業を行い、かつエラーが発生した場合、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。これらの最終ステップでは、エラーを解決するか、サンプル コードに切り替えます。

注: *CY8CKIT-062-BLE キットには Kitprog2 が同梱されています。キットをプログラミングする前に、ボードが最新の Kitprog3 ファームウェアに更新されていることを確認してください。Kitprog のファームウェアのアップデート方法については、デバッギングを参照してください。*

- **デバッグ ターゲットを選択**

PSoC™ Creator は 1 度に 1 つのコアをデバッグできます。

- 図 48 に示すように、PSoC™ Creator で **Debug** > **Select Debug Target** を選択します。

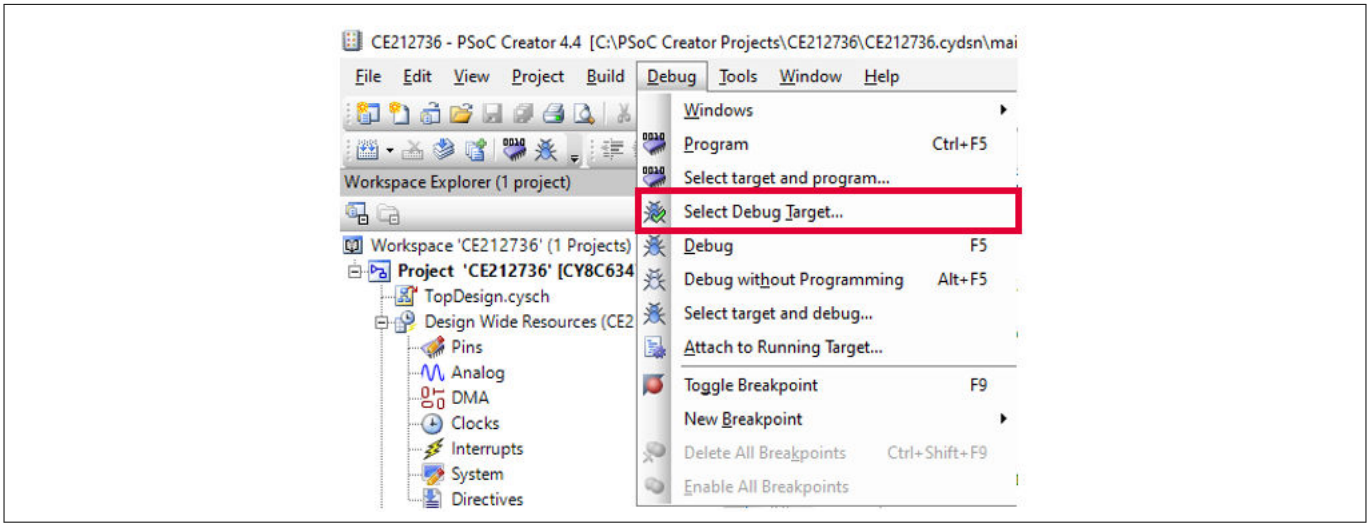


図 48 デバッグ ターゲットを選択

- **基板に接続します。**

図 49 に示すように、**Select Debug Target** ダイアログボックスで CM4 ターゲットを選択し、**OK** または **Connect** をクリックします。

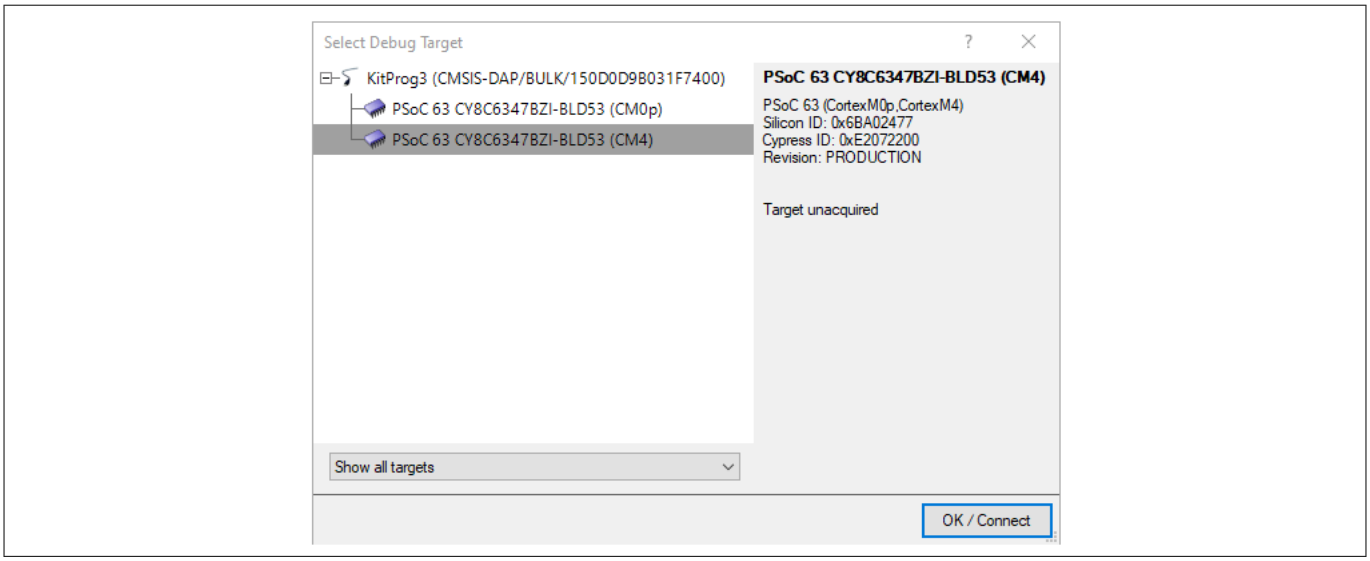


図 49　　デバイスへ接続

ヒント:　基板をプログラムする際に、いずれかのターゲットを選択できます。コアは同じメモリ空間を共有します。いずれかのコアのプログラミングは、両方のコアがプログラムされます。しかし、デバッグの場合は、この選択が重要です。デバッガには、接続するコアだけが表示されます。これらの命令ではデバッガを使用しません。

- **基板をプログラム**

図 50 に示すように、**Debug** > **Program** を選択し、プロジェクトをデバイスにプログラムします。

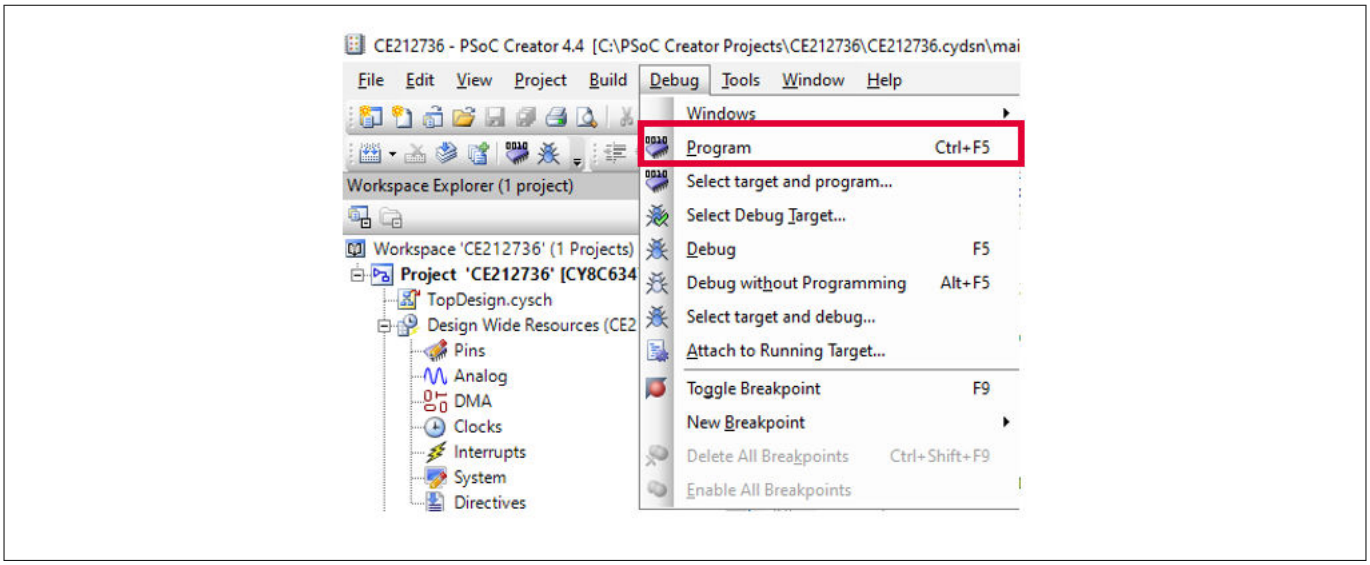


図 50　　デバイスをプログラム

図 51 に示すように、PSoC™ Creator ウィンドウの左下隅にプログラミング状態が表示されます。

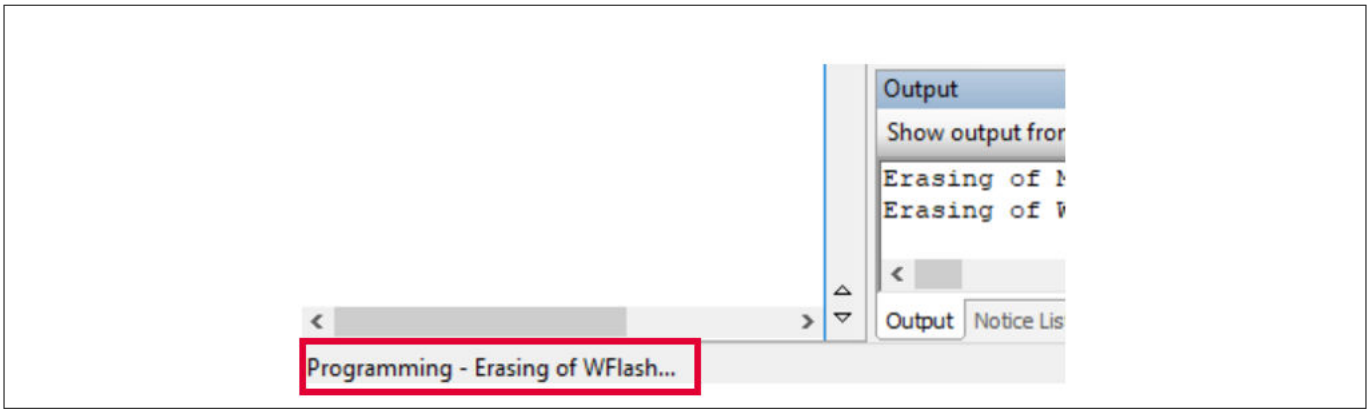


図 51　　プログラミングの状態

デュアルコア アプリケーションでは、リンカ ファイルは各実行ファイルをメモリの正しい場所に配置します。実行は CM0+コアで開始され、次に CM4 コアが有効にされます。

プログラミングが完了すると、アプリケーションは実行します。LED が緑色に点灯し、ターゲットがアドバタイズしていることを示します。アドバタイズ タイムアウトが発生した後、LED は赤色に変わり、切断されていることを示します。

ヒント:　***Debug*** *>* ***Debug*** *コマンドを使用しても基板をプログラム できます。コードを生成またはリビルドする必要がある場合、* ***Program*** *または* ***Debug*** *コマンドを発行するとコードの生成またはリビルドは自動的に実行されます。また、基板をプログラミングせずにデバッグもできます。ただし、これらのインストラクションではデバッガを使用しません。*

注:　*キットの KitProg2 ファームウェアはアップデートが必要な場合があります。ファームウェアを更新する手順については、キットのユーザーガイドを参照してください。*

5.9　パート 6: デザインをテストする

ここでは、付録 D.3 または付録 D.2 を使用して Bluetooth® LE 設計をテストする方法について説明します。図 10 に、Bluetooth® LE Pioneer Kit を使用して設計をテストするための設定を示します。

パス	最初からの作業 サンプル コードを参考とする	サンプル コードの使用 PSoC™ Creator または Bluetooth® LE に詳しくない	サンプル コードの使用 PSoC™ Creator と Bluetooth® LE に精通している
アクション	ステップ 1 またはステップ 2 のいずれかを行う		

- **CySmart モバイル アプリ使用によるテスト**
  - ご使用の iOS または Android モバイル機器の Bluetooth®機能をオンにします。
  - CySmart アプリケーションを起動します。
  - Bluetooth® LE Pioneer Kit のリセットスイッチを押して、開発するデザインから Bluetooth® LE アドバタイズを開始します。CySmart アプリがデバイスを認識できるようにするためには、緑色の LED が点灯している必要があります。
  - CySmart アプリのホーム画面をプルダウンして、BLE ペリフェラルのスキャンを開始します。ファインド ミー ターゲットは、CySmart アプリのホーム画面に Bluetooth® LE デバイスとして表示されます。タップして Bluetooth® LE 接続を確立します。モバイル機器がターゲットを見つけられない場合、リセットボタンをもう一度押すか、CySmart ホストエミュレーションツールを試してください (ステップ 2)。

- カルーセルビューから「Find Me」プロファイルを選択します。
- **Find Me** プロファイル画面で、**Alert Level** 値を選択し、ご使用の機器上の青色 LED の状態が選択に応じて変化することを観察します。

図 52 には、iOS アプリを使用したこのプロセスを示します。図 53 には、Android アプリを使用したこのプロセスを示します。

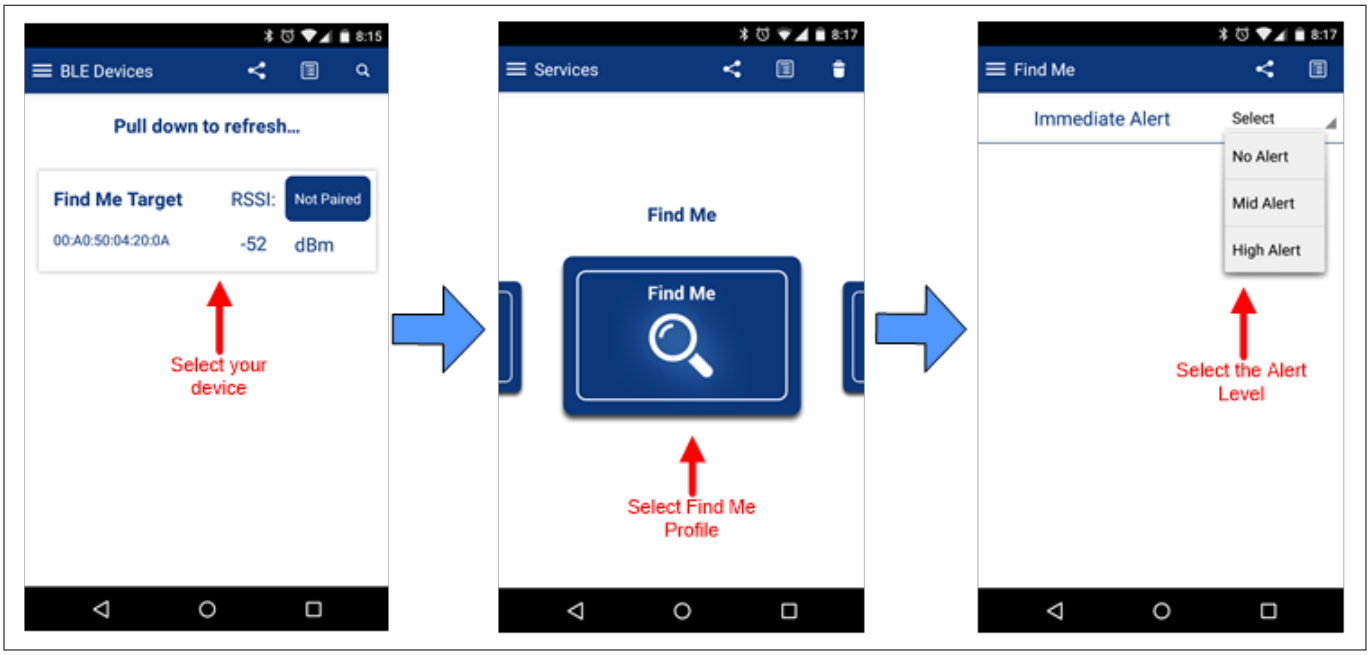


図 52　　iOS 向け CySmart アプリでのテスト

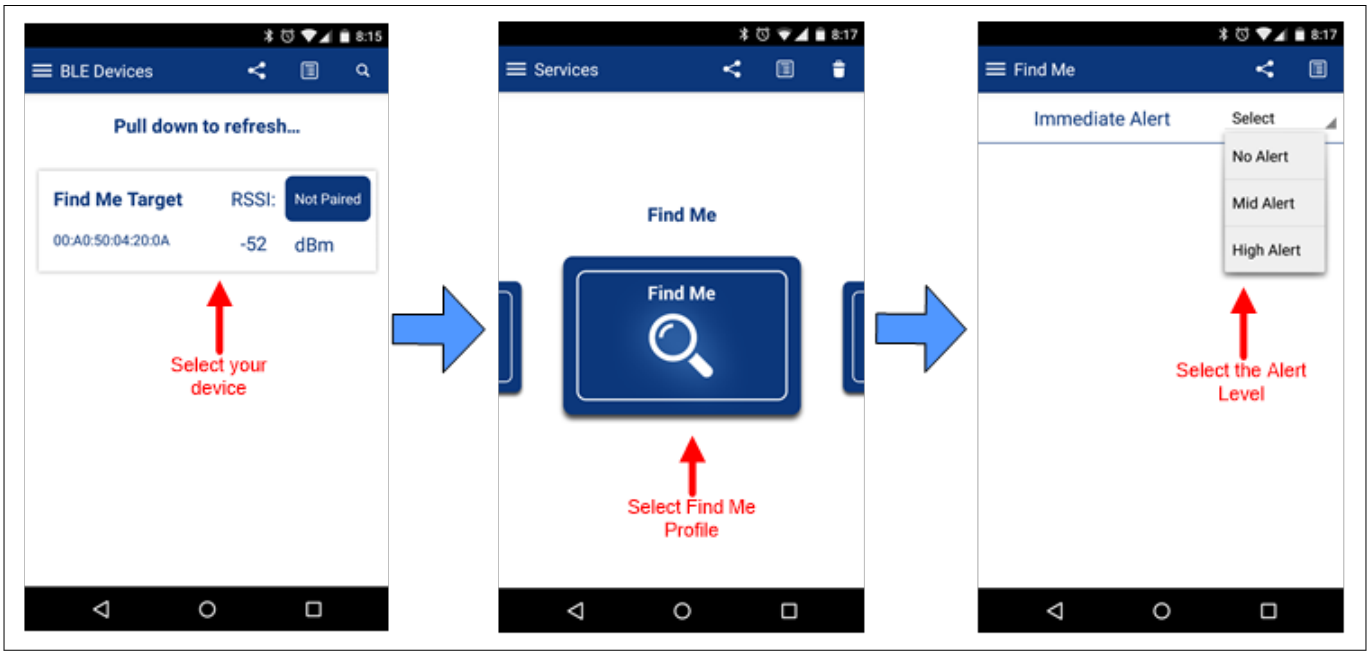


図 53　　Android 向け CySmart アプリでのテスト

- **CySmart ホスト エミュレーション ツール使用によるテスト**

CySmart モバイルアプリの代わりに、CySmart ホストエミュレーションツールを使用し、開発するデザインとの Bluetooth® LE 接続を確立し、Bluetooth® LE 特性への読み書き操作を実行できます。

- インストールされていない場合、CySmart ホスト エミュレーション ツールをインストールします。

図 54 に示すように、トップ回路図上の Bluetooth® LE コンポーネントシンボルを右クリックし、**Download CySmart** を選択します。インストーラーの指示に従ってツールをインストールします。

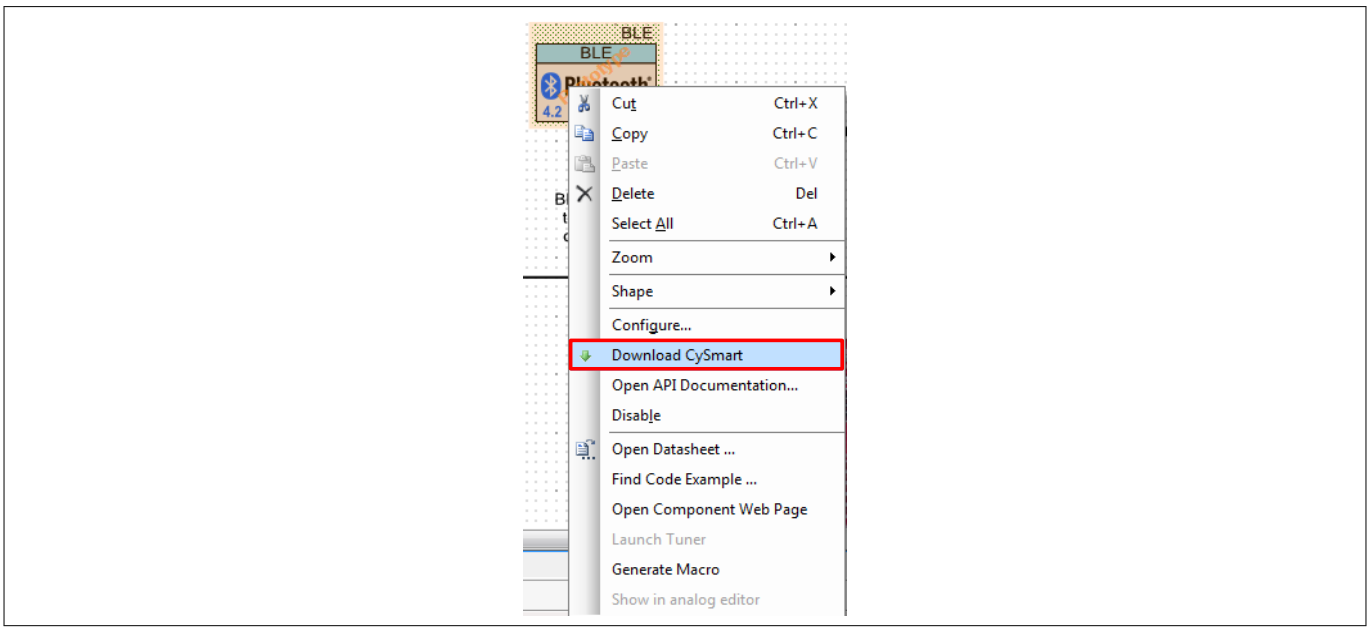


図 54 CySmart のダウンロード

- Bluetooth® LE ドングルをご使用の Windows PC に接続します。ドライバー インストールが完了するまでお待ちください。
- CySmart ホストエミュレーションツールを起動します。

ツールは自動で Bluetooth® LE ドングルを検出します。Bluetooth® LE ドングルが **Select BLE Dongle Target** ポップアップウィンドウに表示されない場合、**Refresh** をクリックします。図 55 に示すように、**Connect** をクリックします。

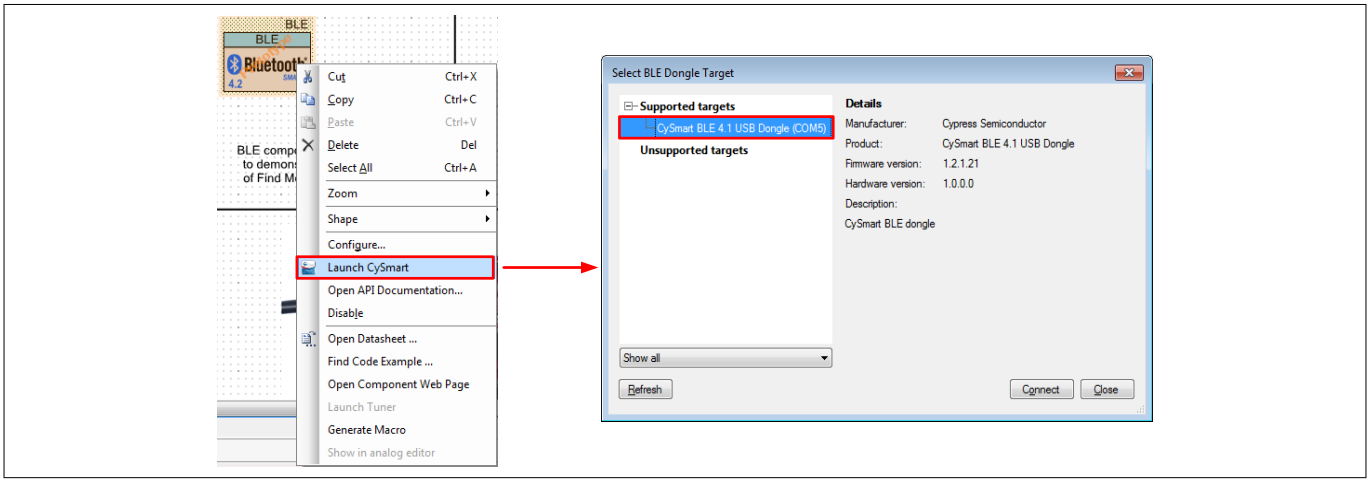


図 55 CySmart での Bluetooth® LE ドングル選択

注: ドングルのファームウェアが古い場合は、警告が表示されます。このステップを完了する前に、ファームウェアをアップグレードする必要があります。ウィンドウの指示に従って、ドングルのファームウェアを更新します。

- 図 56 に示すように、**Configure Master Settings** をクリックし、**Restore Defaults** をクリックします。その後、**OK** をクリックします。

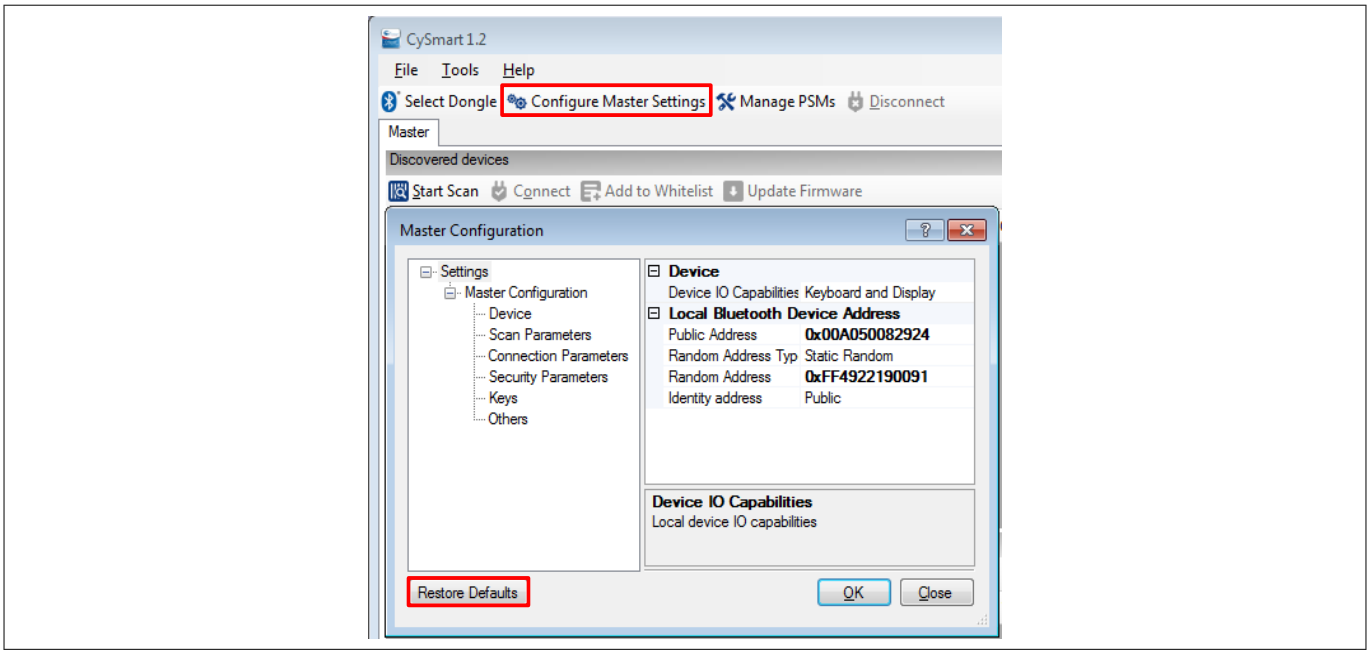


図 56　　CySmart Master Settings 設定

- Bluetooth® LE Pioneer Kit のリセットスイッチを押して、開発するデザインから BLE アドバタイズを開始します。LED は緑色に変わり、アドバタイズしていることを示します。
- CySmart ホスト エミュレーション ツールで、**Start Scan** をクリックします。図 57 に示すように、ご使用機器の名称 (**Find Me Target** として設定されている) が **Discovered devices** リストに表示されます。

注:　　ターゲットが見つけられず、スキャン プロセスがタイムアウトした場合、LED は赤色に変わります。再びリセットボタンをクリックし、アドバタイズを再始動します。

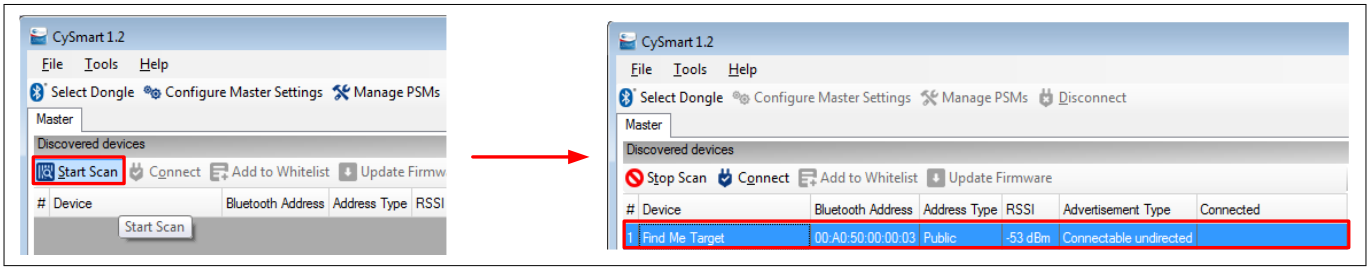


図 57　　CySmart でのデバイス検出

- 図 58 に示すように、**Find Me Target** を選択し、**Connect** をクリックして、CySmart ホストエミュレーションツールとの Bluetooth® LE 接続を確立します。

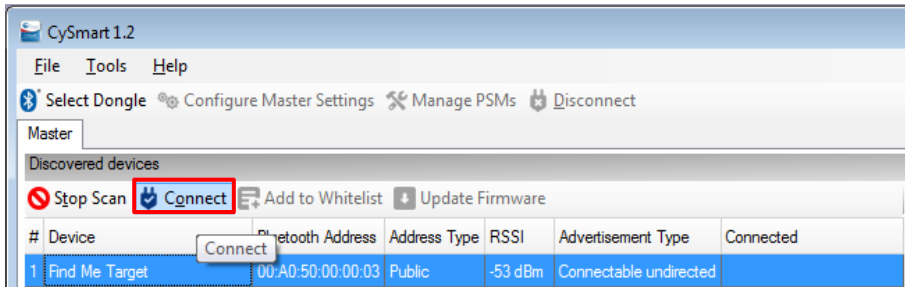


図 58　　CySmart でのデバイス接続

- 図 59 に示すように、接続後、**Find Me Target** デバイスタブに切替え、CySmart ホストエミュレーション ツールで開発するデザインのすべての属性を検出します。

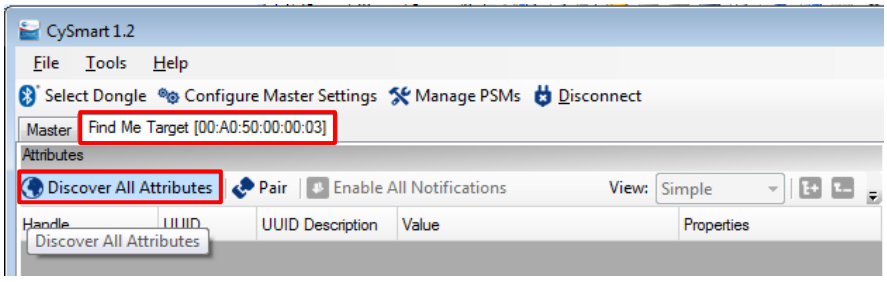


図 59　　CySmart での属性検出

- **Attributes** ウィンドウをスクロールダウンし、**Immediate Alert** サービスのフィールドを探します。図 60 に示すように、**Immediate Alert** サービスの下の **Alert Level** 特性に 0、1、または 2 の値のいずれかを書き込みます。アラートレベル特性の設定ごとに、ご使用のデバイス上の LED 状態変化を観察します。

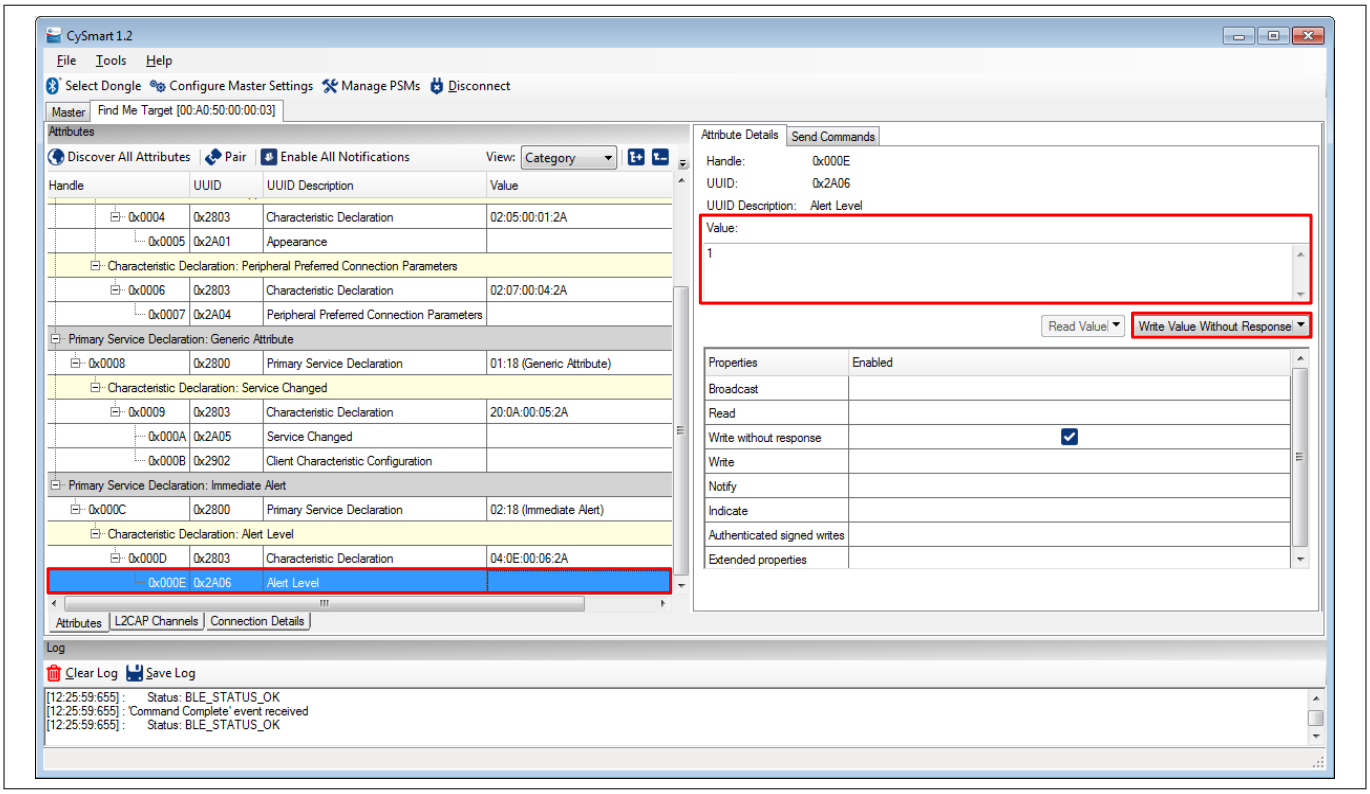


図 60 CySmart ホスト エミュレーション ツールでのテスト

- 検索が終了したら、CySmart アプリケーションの Disconnect ボタンをクリックします。LED は赤色に変わり、切断されたことを示します。

このプロセスを繰り返すためには、基板上のリセット ボタンを押してアドバタイズを再始動します。

6 まとめ

本アプリケーション ノートは PSoC™ 6-BLE デバイス アーキテクチャおよび関連する開発ツールを説明しました。

PSoC™ 6-BLE は、単一のチップ上に低消費電力 Bluetooth® LE 無線ブロック、コンフィギュレーション可能なアナログとデジタル ペリフェラル機能、メモリおよびデュアル CPU の CPU システムを集積した、真のプログラマブル組込みシステムオンチップです。統合された機能および低消費電力モードを備えるため、PSoC™ 6-BLE はバッテリ駆動のウェアラブル機器、ヘルスケアやフィットネスの Bluetooth® LE アプリケーションに最適です。

7 関連アプリケーションノートとサンプル コード

PSoC™ 6 MCU サンプル コードの完全で最新のリストについては、サンプル コードのウェブページをご覧ください。PSoC™ 6 MCU 関連のドキュメントについては、PSoC™ 6 MCU 製品のウェブページをご覧ください。

表 3 は、PSoC™ 6-BLE と PSoC™ Creator の学習の次段階のために推奨されるシステム レベルおよび一般的なアプリケーション ノートを一覧にします。

表 3 一般的なシステムレベルのアプリケーションノート、サンプル コード

文書番号	文書名
AN221774	PSoC™ Creator での PSoC™ 6 MCU 入門
CE221773	PSoC™ 6 MCU Hello World Example
AN218241	PSoC™ 6 MCU ハードウェア設計上の注意事項
AN219434	PSoC™ 6 MCU — 生成コードの IDE へのインポート
AN219528	PSoC™ 6 MCU 低消費電力モードおよび節電技術

表 4 は、デバイスの特定ペリフェラルおよびアプリケーションのアプリケーション ノートおよびサンプル コード (CE) を示します。

表 4 PSoC™ 6-BLE 機能に関連する資料

文書番号	文書名
プログラマブル デジタル	
Bluetooth® Smart	
AN91162	BLE カスタム プロファイルの作成について
AN91445	アンテナの設計および高周波レイアウト ガイドライン
AN92584	BLE アプリケーション向け低消費電力の設計およびバッテリ寿命推測
CE218463	Bluetooth® Low Energy（BLE）アラート通知クライアント/サーバー
CE218464	Bluetooth® Low Energy（BLE）Phone Alert クライアント/サーバー
システム リソース、CPU、割込み	
AN215656	PSoC™ 6 MCU デュアル CPU システム設計
AN217666	PSoC™ 6 MCU 割込み
CE216795	PSoC™ 6 MCU Dual-CPU Basics
CE216825	PSoC™ 6 MCU Real-Time Clock Basics
CE218129	PSoC™ 6 MCU Wake up from Hibernate Using Low Power Comparator
CE218541	PSoC™ 6 MCU Fault-Handling Basics
CE218542	PSoC™ 6 Custom Tick Timer Using RTC Alarm Interrupt
CE218552	PSoC™ 6 MCU UART to Memory Buffer Using DMA
CE218964	PSoC™ 6 MCU RTC Daily Alarm
CE219339	PSoC™ 6 MCU MCWDT and RTC Interrupts (Dual Core)

(続く)

7 関連アプリケーションノートとサンプル コード

表 4 (続き) PSoC™ 6-BLE 機能に関連する資料

文書番号	文書名
CE219521	PSoC™ 6 MCU GPIO Interrupt
CE219881	PSoC™ 6 MCU Switching Power Modes
CE220060	PSoC™ 6 MCU Watchdog Timer
CE220061	PSoC™ 6 MCU Multi-Counter Watchdog Interrupts
CE220120	PSoC™ 6 MCU Blocking Mode Flash Write
CE220169	PSoC™ 6 MCU Periodic Interrupt Using TCPWM
GPIO	
CE219490	PSoC™ 6 Breathing LED Using SMART IO
CE219506	PSoC™ 6 Clock Buffer Using SMART IO
CE220263	PSoC™ 6 MCU GPIO Pins Example
CAPSENSE™	
AN92239	CAPSENSE™による近接センシング
AN85951	PSoC™ 4 および PSoC™ 6 MCU CAPSENSE™デザインガイド
SMIF	
ブートローダ	
AN213924	MCU デバイス ファームウェア アップデート ソフトウェア開発キット ガイド
CE213903	PSoC™ 6 MCU Basics Bootloaders
通信	
CE220541	PSoC™ 6 MCU SCB EzI2C
セグメント LCD	
オーディオ	
CE218636	PSoC™ 6 MCU Inter-IC Sound (I2S) Example
CE219431	PSoC™ 6 MCU PDM-to-PCM Example
RTOS	
CE217911	PSoC™ 6 FreeRTOS™ Example Project
セキュリティ	
CE220465	PSoC™ 6 MCU Cryptography – AES Demonstration
CE220511	PSoC™ 6 MCU Cryptography – SHA Demonstration

A Appendix A. 用語集

ここでは、PSoC™デバイス ファミリで操作する際によく出てくる用語について説明します。

Component Customizer: PSoC™ Creator のシンプルな GUI で各コンポーネントに埋め込まれています。コンポーネント パラメーターをカスタマイズするために使用され、コンポーネントを右クリックすることでアクセスされます。

Components: コンポーネントは、メインシステムバスを介して MCU に本質的に接続されている 1 つの PSoC™コンポーネントに複数の IC およびシステムインターフェイスを統合するために使用されます。例えば、Bluetooth® LE コンポーネントは Bluetooth® Smart 製品を数分で作成します。同様に、プログラマブル アナログ コンポーネントをセンサー用に使用できます。

MiniProg3 開発用プログラミングハードウェアであり、開発する基板または内蔵プログラマを備えない PSoC™開発キット上の PSoC™デバイスをプログラミングするために使用されます。

PSoC™: 32 ビット CM4 などの 1 つ以上の CPU を含み、アナログとデジタル ブロックを内蔵するプログラム可能な組込み設計プラットフォームです。タッチセンシングなどの信頼性が高く使いやすいソリューションで組込みシステム設計を加速させ、低消費電力の設計を可能にします。

PSoC™ Creator: 開発を行うお客様の PC にインストールされた PSoC™ 3、PSoC™ 4、PSoC™ 5LP、および PSoC™ 6-BLE IDE ソフトウェアであり。PSoC™システムのハードウェアとファームウェアの同時に設計するか、またはハードウェア設計後に一般的な IDE へエクスポートすることをを可能にします。

Peripheral Driver Library: Peripheral Driver Library (PDL) は、PSoC™ 6 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 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、使用可能な広範な周辺機器のソフトウェア開発を容易にします。

PSoC™ Programmer: PSoC™デバイスをプログラムするための柔軟性のある統合プログラミング アプリケーションです。PSoC™ Programmer は PSoC™ Creator と統合され、PSoC™ 3、PSoC™ 4、PRoC、PSoC™ 5LP、および PSoC™ 6 MCU デザインをプログラムします。

B Appendix B. Bluetooth® LE プロトコル

B.1 概要

BLE (Bluetooth® Smart とも呼ばれている) は、2.4 GHz ISM 帯で動作する低消費電力の無線規格として Bluetooth® SIG により策定されました。図 61 に BLE プロトコル スタックを示します。

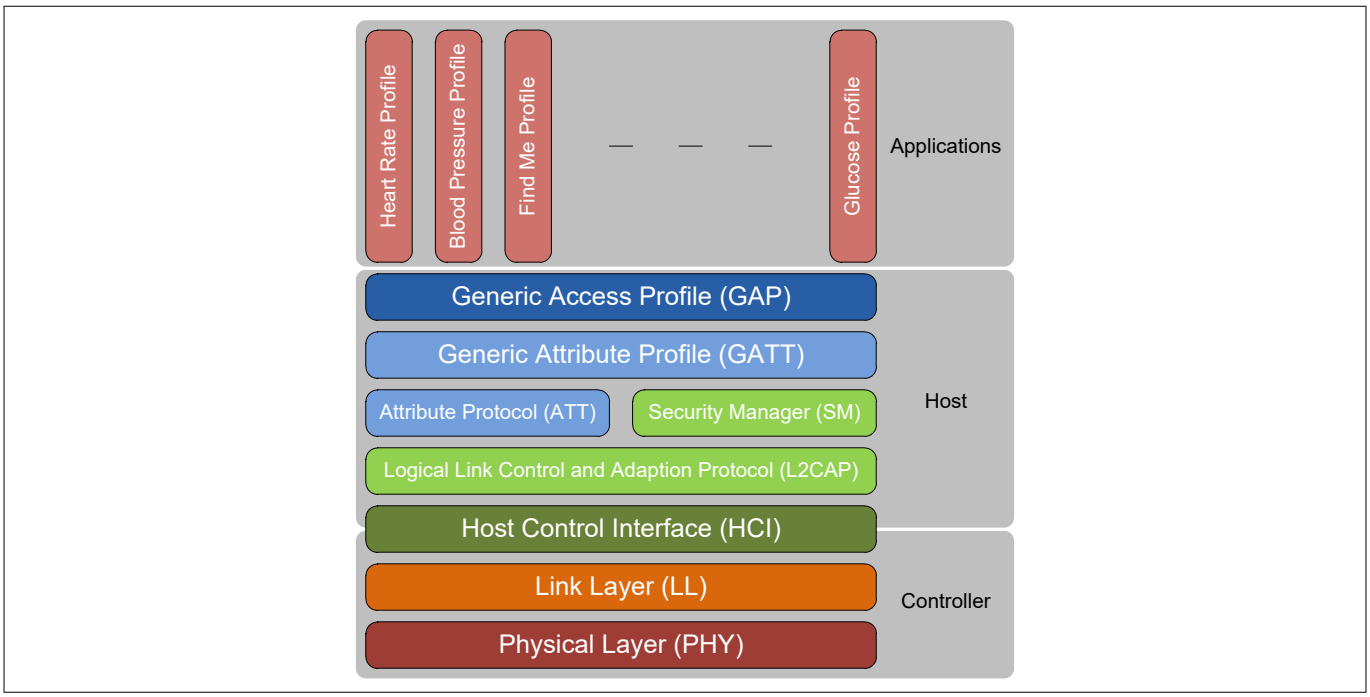


図 61 Bluetooth® LE アーキテクチャ

BLE スタックは以下の 3 グループに分けられています。

- **コントローラー:** 物理デバイスで、パケットを暗号化して無線信号として送信します。信号の受信時、コントローラーは無線信号を復号してパケットを復元します。
- **ホスト:** セキュリティ マネージャ、属性プロトコルなどの様々なプロトコルとプロファイルを含むソフトウェア スタックであり、2 つ以上のデバイスがどのように他のデバイスに通信するかを管理します。
- **アプリケーション:** ソフトウェア スタックとコントローラーを使用して特定の機能を実装するユースケースです。

下記では、標準的な心拍数監視サービスとバッテリ サービスを例として取り上げて Bluetooth® LE スタックの複数の層の概要について説明します。Bluetooth® LE アーキテクチャの詳細説明については、Bluetooth® Core Specification をご覧ください。

B.2 物理層 (Physical Layer; PHY)

物理層は、2.4GHz ISM 帯のガウス型周波数偏移変調 (GFSK) という変調方式を利用して 1 Mbps でデジタル データを送受信します。この Bluetooth® LE の物理層は ISM 帯を 2MHz ごと 40 個の RF チャネルに分けており、その内データ チャネルは 37 個でアドバタイズ チャネルは 3 個です。

B.3 リンク層 (Link Layer; LL)

リンク層では、(認識およびフロー制御に基づいたアーキテクチャを利用して) 信頼性のある物理リンクを成立するための重要な手続き、および堅牢で低消費電力の Bluetooth® LE プロトコルを実現する機能を実装します。以下はリンク層の機能の一部です。

- アドバタイズ、スキャン、接続の作成および維持:物理リンクを確立
- 24 ビット CRC および AES-128 ビット暗号化:データ交換時の堅牢性と安全さを確保
- 高速接続の確立および低デューティ比のアドバタイズ:低消費電力動作を実現
- アダプティブ周波数ホッピング (Adaptive Frequency Hopping;AFH): パケット送信用の通信チャネルを変更し、他のデバイスからの干渉を軽減

リンク層では、次の 2 つのロールが定義されます。

- **マスター:** スマートフォンは、リンク層をマスターコンフィギュレーションに設定する例です。
- **スレーブ:** 心拍数モニターデバイスは、リンク層をスレーブコンフィギュレーションに設定する例です。

PSoC™ 6-BLE デバイスはどちらのコンフィギュレーションでも動作できます。

リンク層スレーブは自体の存在を他のリンク層マスターにアドバタイズするものです。リンク層マスターはアドバタイズ パケットを受信し、アプリケーションの要求に応じてスレーブとの接続を決定します (図 62 を参照してください)。この心拍数モニター アプリケーションの実装例では、心拍数モニター デバイスはスレーブとして動作し、データをマスターとして動作するスマートフォンに送信します。スマートフォンアプリケーションは、スマートフォン上に読み出し結果を表示します。

PSoC™ 6-BLE デバイスは、アドバタイズ、CRC、および AES 暗号化などタイムクリティカルでプロセッサに集中するリンク層の機能をハードウェアで実装します。アドバタイズ状態に移行するか暗号化を開始するなどのリンク層制御動作はファームウェアで実装されます。

図 62 に、Bluetooth® LE のリンク層パケットの構造および各々のフィールドのサイズを示します。リンク層パケットは上層のすべてのデータをペイロード フィールドに格納します。物理リンク上の通信を一意に識別し、同じ RF チャネルで動作する近くの Bluetooth® LE デバイスからのパケットを無視するために 4 バイトのアクセスアドレスを持っています。24 ビット CRC によりデータの堅牢性を確保できます。

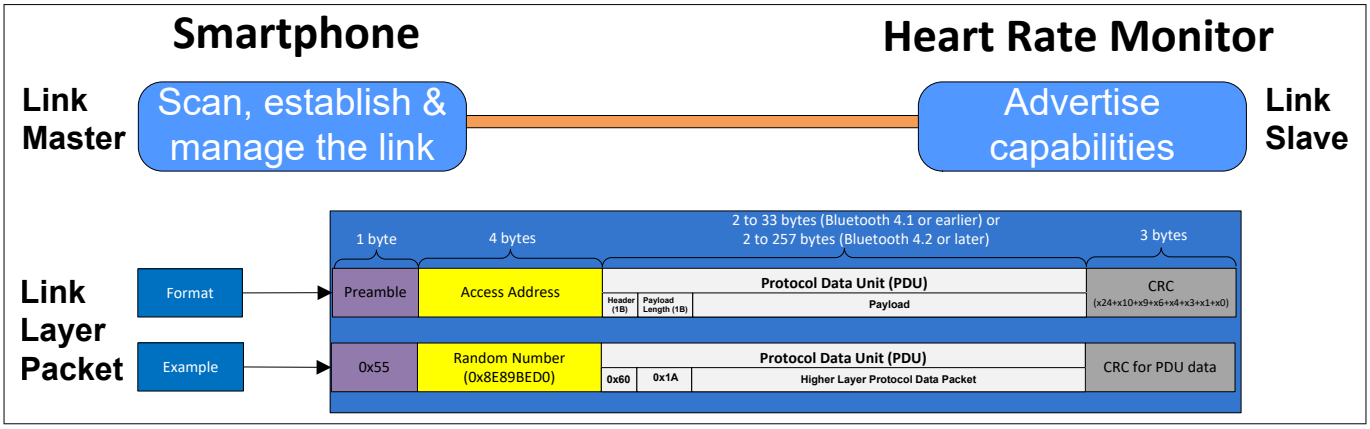


図 62 Bluetooth® LE リンク層プロトコル

B.4 ホスト制御インターフェース (Host Control Interface; HCI)

HCI は、ホストとコントローラー間の標準定義のインターフェースです。これによりホストとコントローラーがコマンド、データ、イベントなどの情報を USB や UART のような異なる物理的転送方式で相互交換できます。コントローラーとホストは異なるデバイスの場合にのみ HCI に物理的転送が必要です。

PSoC™ 6-BLE デバイスでは、HCI はコントローラーとホスト間でメッセージとイベントを転送するファームウェアプロトコル層にすぎません。

B.5 論理リンク制御および適応プロトコル (Logical Link Control and Adaptation Protocol;L2CAP)

L2CAP はプロトコルの多重化、分割、再構成のサービスを上層のプロトコルに提供します。分割サービスでは、上位層から受信したパケットをリンク層が送信できるより小さいパケットに分割します。再構成サービスでは、リ

ンク層から受信した小さいパケットを組み合わせて意味のあるパケットに構成します。L2CAP 層は、図 63 に示すように、付録 B.7 および L2CAP 制御用に 3 つのプロトコル チャネル ID を提供します。Bluetooth 4.2 仕様により、これらのプロトコル チャネルの上部にある L2CAP 接続指向チャネルから直接データ チャネルを作成できます。

PSoC™ 6-BLE では、L2CAP とその上部にある各層がファームウェアで実装されます。

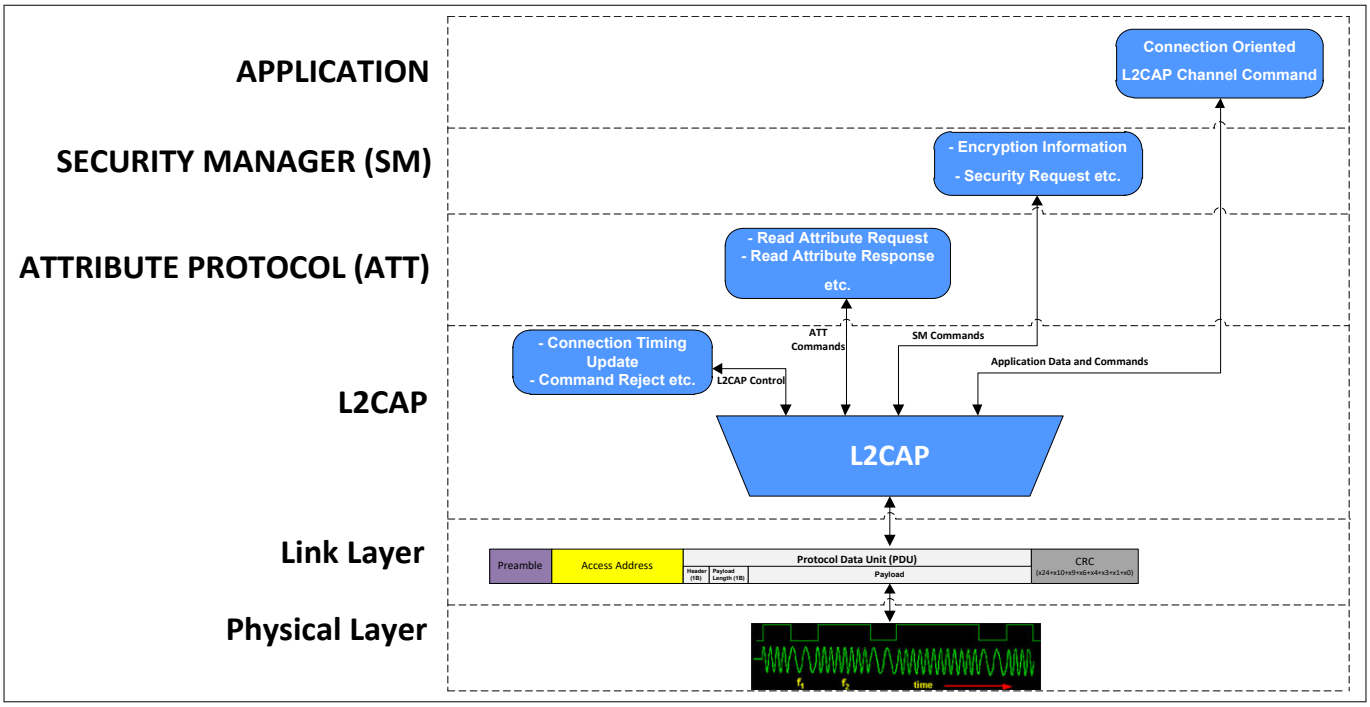


図 63　　Bluetooth® LE L2CAP 層

B.6　セキュリティ管理 (Security Manager; SM)

SM 層は、ペアリング、暗号化、およびキーの配布の方法を定義します。

- **ペアリング**は、セキュリティ機能を有効にするプロセスです。このプロセスでは、2 つのデバイスが認証され、そのリンクが暗号化されてから、暗号化の鍵が交換されます。これにより、RF チャネル上でスパイによって探られることなく、Bluetooth® LE インターフェースを介してデータが安全に交換できます。
- **ボンディング**は、ペアリングプロセスで交換されたキーと識別情報が保存されるプロセスです。デバイスが 1 度ボンディングされたら、再接続する際に再度ペアリング プロセスを経過する必要はありません。

Bluetooth® LE はデータ暗号化に 128 ビット AES を使用します。

B.7　アトリビュート プロトコル (Attribute Protocol; ATT)

BLE の GATT 仕様には、ATT と GATT 層を理解するために把握する必要がある 2 つのロールがあります。

- **GATT サーバー:** GATT サーバーはデータや情報を格納します。GATT クライアントからリクエストを受信し、データで返答します。例えば、心拍数モニターの GATT サーバーには心拍数の情報を格納します。BLE 型 HID キーボードの GATT サーバーは、ユーザーがキーを押す情報を格納します。
- **GATT クライアント:** GATT クライアントは、GATT サーバーにデータをリクエストまたは受信します。たとえば、スマートフォンは心拍数モニターの GATT サーバーから心拍数情報を受信する GATT クライアントです。ノートパソコンは、Bluetooth® LE キーボードからキー押下情報を受信する GATT クライアントです。

ATT は Bluetooth® LE 通信の基礎となっています。このプロトコルにより、GATT クライアントが GATT サーバーでデータや属性を検索してアクセスできます。GATT クライアントとサーバーのアーキテクチャの詳細については、「付録 B.8」を参照してください。

属性は以下の内容を含む ATT/GATT 層の基本的なデータコンテナーです。

- **属性ハンドル:** 属性のアドレス指定し、そこにアクセスする 16 ビットアドレスです。
- **属性タイプ:** 属性に格納されるデータのタイプを指定します。これは Bluetooth® SIG が定義した 16 ビット UUID で表わされます。

例えば、心拍数管理サービスの 16 ビット UUID は 0x180D で、デバイス名属性の UUID は 0x2A00 です。SIG によって割り当てられた 16 ビット UUID の一覧については、Bluetooth®のウェブページをご利用ください。

- **属性値:** 属性に格納される実際の値です。
- **属性権限:** 属性へのアクセス、認証、および権限付与の要件を指定します。これは上層の仕様によって設定され、属性プロトコルをとおしてアクセスできません。

図 64 にデバイス名属性の構造を例として取り上げます。

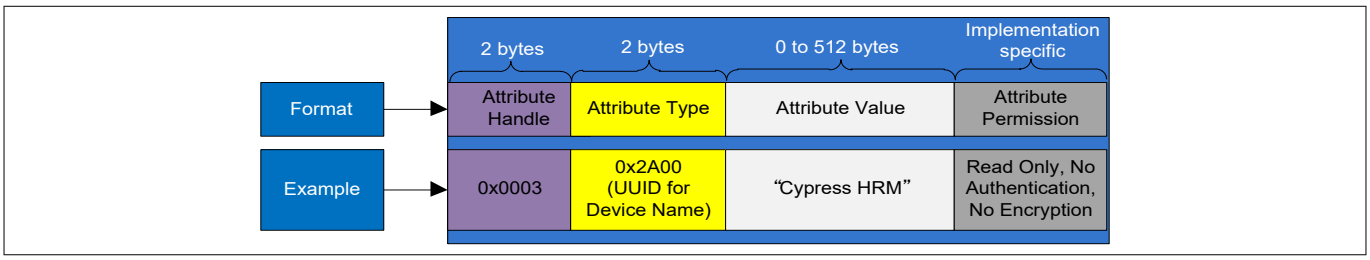


図 64　　属性フォーマットの例

B.7.1　　属性階層

属性はデータを表示する ATT/GATT での構成要素です。属性は次の 2 グループに大きく分けられ、データ階層とデータ抽象化を提供します。

- **キャラクタリスティック (Characteristic):** システム情報または意味のあるデータを表わす属性のコレクションです。キャラクタリスティックは、以下の属性を含みます。
  - キャラクタリスティック宣言属性 (Characteristic Declaration Attribute): キャラクタリスティックの開始を定義します。
  - キャラクタリスティック値属性 (Characteristic Value Attribute): 実際のデータを格納します。
  - キャラクタリスティック ディススクリプタ属性: これらはオプションの属性であり、キャラクタリスティック値の追加情報を提供します。

「バッテリレベル」は、バッテリサービス (BAS) のキャラクタリスティックの 1 例です。バッテリレベルをパーセント値で表示するのは、キャラクタリスティック ディスクリプタの 1 例です。

図 65 に、例としてバッテリレベルのキャラクタリスティックの構造を示します。

- キャラクタリスティックの最初の部分は、その特性の宣言 (特性の開始) であり、図 65 にバッテリレベル特性で示されています。
- 次は実際のキャラクタリスティック値 (実際のデータ) です。バッテリレベル特性の場合は現時点でのバッテリレベルです。バッテリ レベルは「65」、「90」などフル スケールに対する割合で表わされます。
- キャラクタリスティック ディスクリプタは、キャラクタリスティック値を理解するために必要な追加情報を提供します。例えば、バッテリ レベルのキャラクタリスティック表示形式ディスクリプタはバッテリ レベルをパーセントで表わされることを示します。よって、「90」を読み出すと、GATT クライアントは 90 mV やや 90 mAh ではなく 90 %と理解します。同様に、(図 65 に示されていない) 有効範囲のキャラクタリスティック ディスクリプタはバッテリ レベルの範囲が 0～100 %であることを示します。
- クライアント キャラクタリスティック コンフィギュレーション ディスクリプタ (CCCD) は、GATT クライアントが GATT サーバー上の特性の動作を設定できるようにする、もう 1 つの一般的に使用されるキャラクタリスティック ディスクリプタです。GATT クライアントが 0x01 の値を特性の CCCD に書き込むと、GATT サーバーから非同期通知 (次のセクションで説明) を送信できるようになります。バッテリ レベル キャラクタリスティックの

場合、バッテリレベル CCCD に 0x01 を書き込むと、バッテリ サービスがバッテリレベルを定期的に、またはバッテリレベル値の変化に応じて通知できます。

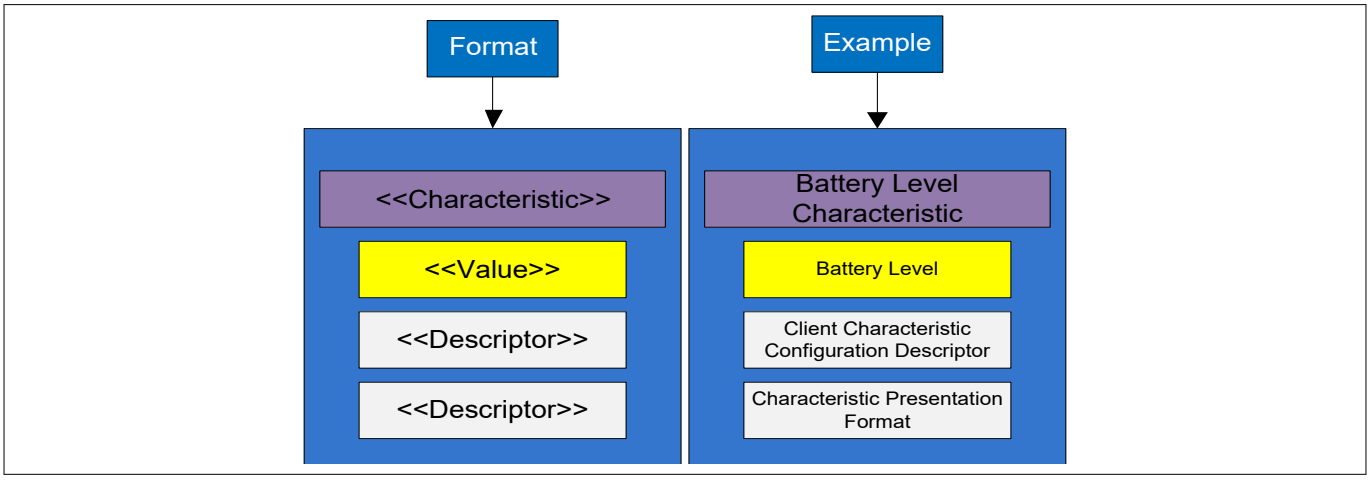


図 65　　キャラクタリスティックのフォーマットと例

- **サービス:** GATT サーバーによって実行される機能を定義する属性の 1 種です。サービスはキャラクタリスティックの集合であり、他のサービスを含むこともあります。サービスの概念は、関連のあるデータのグループを確立し、データ階層を提供するために使用されます。心拍数サービス (HRS) の例は、図 66 を参照してください。

サービスは、1 次サービスと 2 次サービスの 2 種類に分けられています。1 次サービスはデバイスの主な機能を提供し、2 次サービスは追加機能を提供します。例えば、心拍数モニターデバイスの場合、HRS は 1 次サービスで、BAS は 2 次サービスです。

サービスには、GATT サーバー上にある他のサービスも含むことがあります。含まれているすべてのサービスは新しいサービスの一部になります。

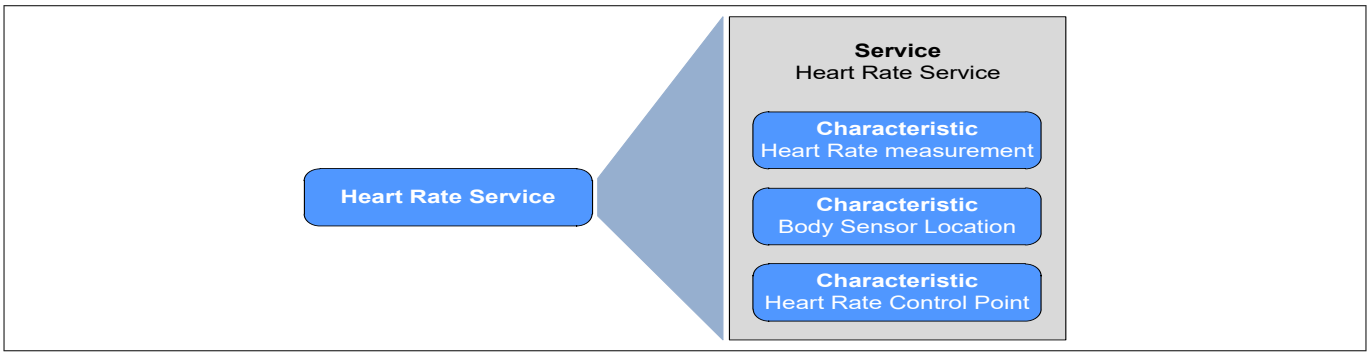


図 66　　Bluetooth® LE 心拍数サービスの例

Bluetooth® LE の「プロファイル」という言葉は、特定の最終アプリケーションを一緒に実行する一連のサービスとその動作です。心拍数プロファイル (HRP) は、心拍数モニターデバイスに必要なすべてのサービスを定義する Bluetooth® LE プロファイルの 1 例です。詳細は「付録 B.9」を参照してください。

図 67 に、このセクションの前で定義された属性、キャラクタリスティック、サービス、およびプロファイルを使用するデータ階層を示します。

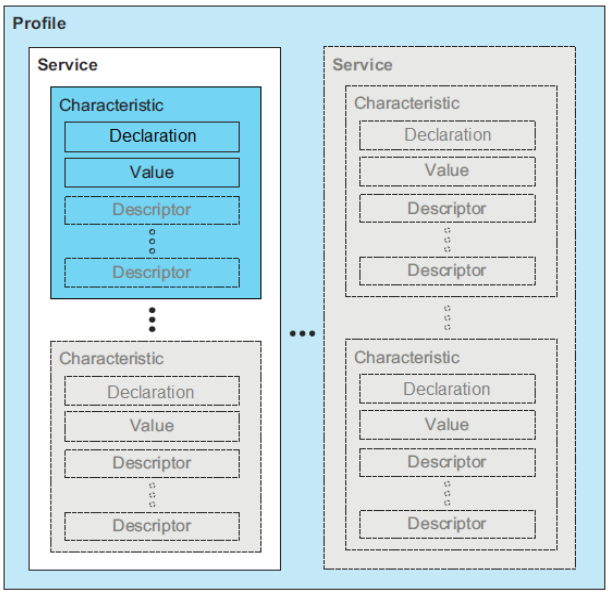


図 67 Bluetooth® LE データ階層

注: Bluetooth® SIG から画像提供

B.7.2 属性の操作

前節で定義された属性は、以下の 5 つの基本的方法でアクセスされます。

- **読み出しリクエスト:** GATT クライアントはこのリクエストを GATT サーバーに送信して、属性値を読み出します。リクエストごとに、GATT サーバーは GATT クライアントに応答を送信します。この読み出しリクエストの 1 例は、スマートフォンが心拍数モニターデバイスのバッテリレベル特性 (図 65 を参照) を読み出すことです。
- **書き込みリクエスト:** GATT クライアントはこのリクエストを GATT サーバーに送信して、属性値を書き込みます。GATT サーバーは GATT クライアントに応答し、値が書き込まれたかどうかを示します。書き込みリクエストの 1 例は、スマートフォンが通知を有効にするためにバッテリレベル特性の CCCD に 0x01 の値を書き込むことです。
- **書き込みコマンド:** GATT クライアントは属性値の書き込みのために、このコマンドを GATT サーバーに送信します。GATT サーバーはこのコマンドに対して何も応答を送信しません。例えば、Bluetooth® LE 即時アラートサービス (IAS) は、IAS ロケーター (スマートフォンなど) から IAS ターゲットデバイス (Bluetooth® LE キー フォブなど) のアラート (LED のオンにする、ブザーを鳴らす、振動モーターを駆動するなど) をトリガするために書き込みコマンドを使用します。
- **通知:** GATT サーバーは属性の新しい値を知らせるために、GATT クライアントに通知を送信します。GATT クライアントはこの通知に対して何の確認も送信しません。例えば、心拍数モニター デバイスはその CCCD に 0x01 が書き込まれると心拍数測定通知をスマートフォンに送信します。
- **指示:** GATT サーバーはこのタイプのメッセージを送信します。GATT クライアントはそれに対して常に確認を行います。例えば、Bluetooth® LE 健康体温計サービス (Health Thermometer Service; HTS) は測定した体温の値を、スマートフォンのような健康体温計コレクターに確実に送信するためにこれらの指示を使用します。

B.8 汎用属性プロファイル (Generic Attribute Profile; GATT)

GATT は、属性の検出と使用の方法を定義しています。GATT は以下の 2 つのロールのいずれかで動作します。

- **GATT クライアント:** データを要求するデバイス (例:スマートフォン)
- **GATT サーバー:** データを提供するデバイス (例:心拍数モニター)

図 68 に、心拍数モニター デバイスを例として GATT 層のクライアントとサーバーのアーキテクチャを示します。心拍数モニターデバイスは、複数のサービス (HRS、BAS、およびデバイス情報サービス) を提供し、それぞれのサービスは図 65 に示すように、特性値とディスクリプタを持つ 1 つ以上の特性を持っています。

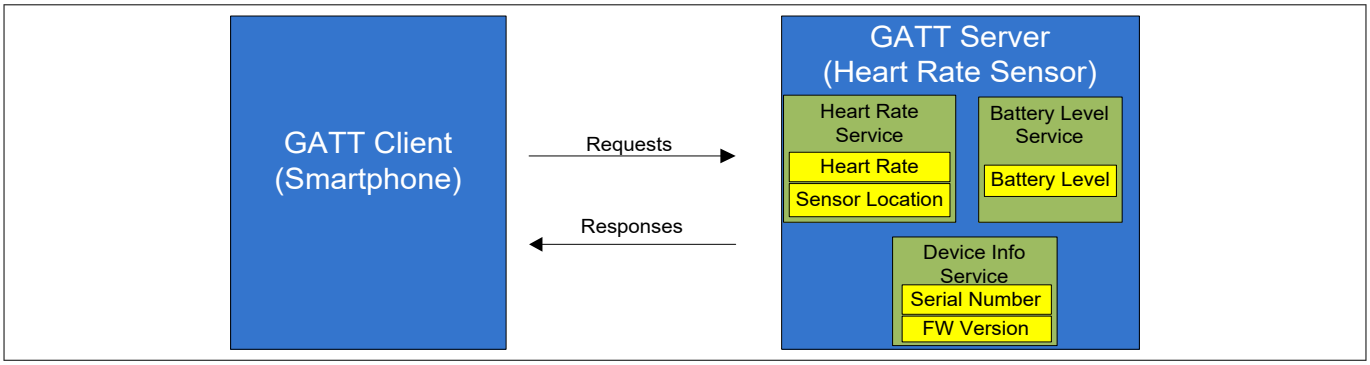


図 68 GATT クライアントとサーバーのアーキテクチャ

Bluetooth® LE 接続がリンク層レベルで確立された後、GATT クライアントは (最初に接続された Bluetooth® LE デバイスについて何も分からず)、「サービス検出」というプロセスを開始します。サービス検出の一部として、GATT クライアントは複数のリクエストを GATT サーバーに送信して、GATT サーバーで利用可能なすべてのサービス、特性、および属性のリストを取得します。サービス検出が完了すると、GATT クライアントは、前述の属性操作を使用して GATT サーバーが提供する情報の読み出しや修正に必要な情報を取得します。

B.9 汎用アクセス プロファイル (Generic Access Profile; GAP)

GAP 層はデバイス アドレス、デバイス名などのデバイス固有情報、および検出/接続/ボンディング方法を提供します。プロファイルは、デバイスの検出方法、接続方法、使用可能なサービスのリスト、およびサービスの使用方法を定義します。図 70 に心拍数プロファイルの例を示します。

GAP 層は以下の 4 つのロールのいずれかで動作します。

- **ペリフェラル:** デバイスを GAP セントラルに接続することを可能にするアドバタイズ ロールです。セントラルとの接続が確立された後、デバイスはスレーブとして動作します。例えば、測定された心拍数をリモート デバイスに報告する心拍数センサーは GAP ペリフェラルとして動作します。
- **セントラル:** アドバタイズをスキャンし、ペリフェラルとの接続を開始する GAP ロールです。この GAP ロールは、ペリフェラルとの接続が確立した後、マスターとして動作します。例えば、ペリフェラル (心拍数センサー) から心拍数測定データを受信したスマートフォンは、GAP セントラルとして動作します。
- **ブロードキャスター:** データを放送するのに使用されるアドバタイズ ロールです。Bluetooth® LE 接続を確立できず、データ交換にも関与しません (要求/応答動作なし)。このロールは、聞き手の有無に係わらずデータを連続して送信するラジオ放送局と同様の役割を果たしています。これは一方向データ通信です。GAP ブロードキャスターの典型的な 1 例は、情報を継続的に放送するが、いかなる応答も期待しないビーコンです。
- **オブザーバー:** アドバタイズをスキャンするリスニング ロールですが、アドバタイズ デバイスには接続しません。これはブロードキャスター ロールの逆ロールです。情報を継続的に聞き取りますが、情報源と通信できないラジオ受信機と同様に動作します。GAP オブザーバーの典型的な 1 例は、継続的にビーコンを聞くスマートフォン アプリです。

図 69 に、Bluetooth® LE Pioneer Kit をペリフェラルとして、スマートフォンをセントラルデバイスとして使用した一般的な Bluetooth® LE システムを示します。また、Bluetooth® LE プロトコル層とセントラルおよびペリフェラル デバイスのロールの間の相互作用も示します。

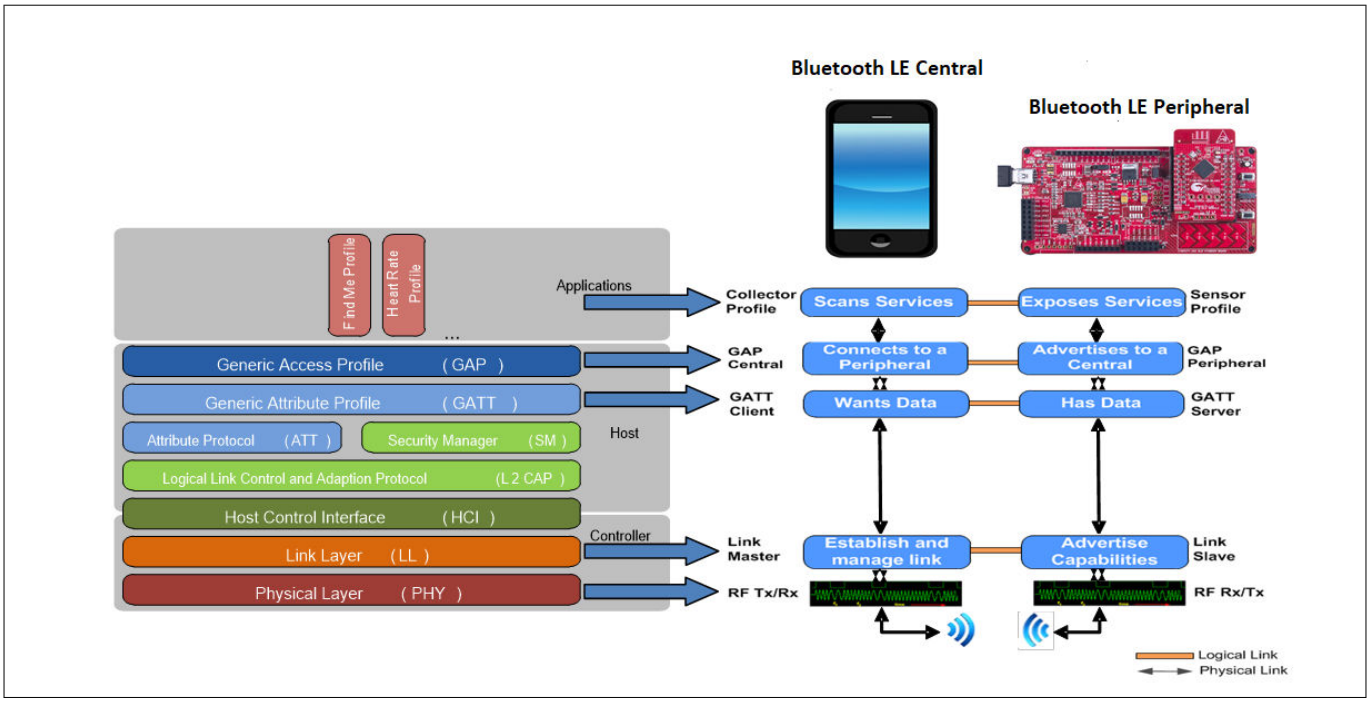


図 69　　Bluetooth® LE システムの設計

図 70 は心拍数アプリを備えたスマートフォンがセントラルとして動作し、心拍数センサーはペリフェラルとして動作する例を示します。心拍数モニターデバイスは心拍数センサープロファイルを実装し、データを受信するスマートフォンは心拍数コレクタープロファイルを実装します。

この例では、心拍数センサープロファイルは 2 つの標準サービスを実装しています。1 つ目は、3 つの特性 (心拍数測定特性、身体センサーの位置特性、および心拍数制御点特性) で構成される心拍数サービスです。2 つ目のサービスは、デバイス情報サービスです。Link Layer で、心拍数測定デバイスはスレーブであり、スマートフォンはマスターです。心拍数サービスとプロファイルの詳細は Bluetooth®開発者ポータルを参照してください。

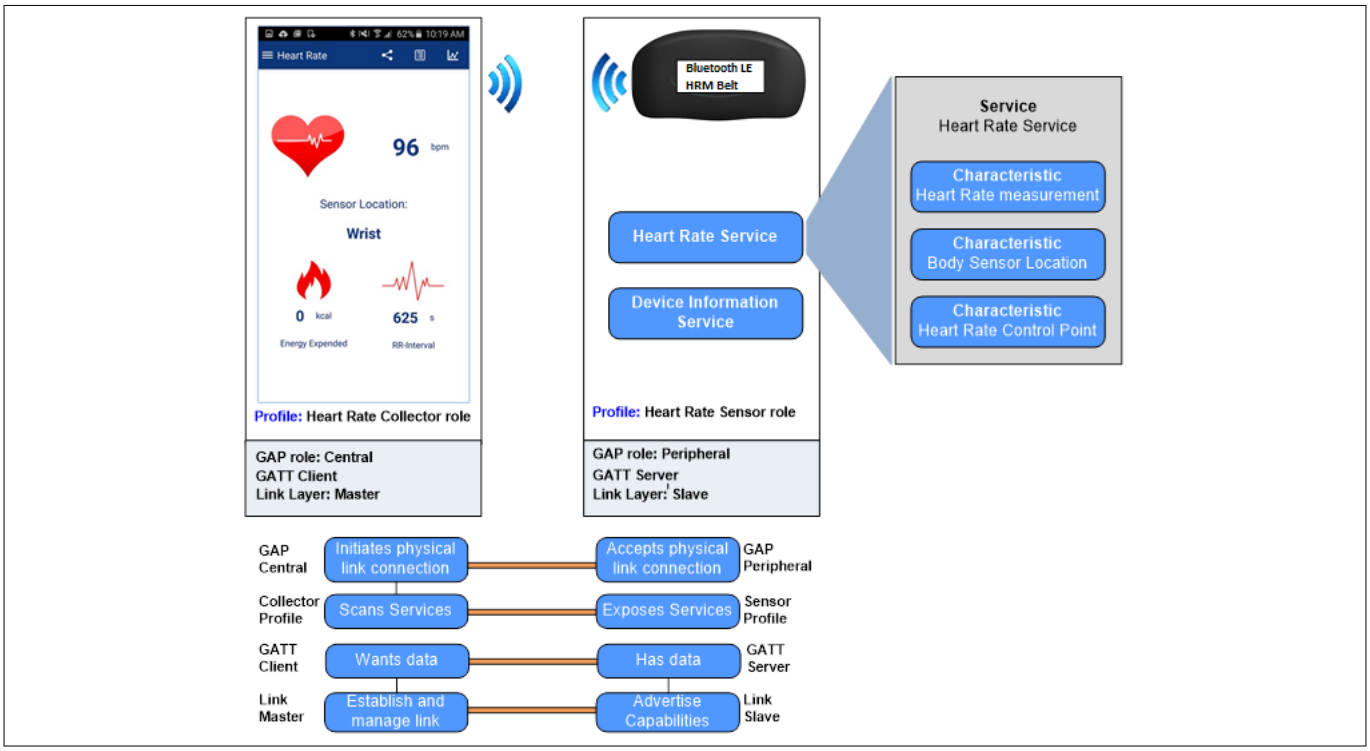


図 70　　Bluetooth® LE 心拍数モニターシステム

C Appendix C. デバイスの特長

C.1 システム ワイド リソース

C.1.1 CPU サブシステム: CM4 および CM0

PSoC™ 6-BLE の CPU サブシステムは、2 つの Cortex®コアで構成されています。150 MHz の最大周波数で動作可能な単精度浮動小数点ユニットを備えた CM4。および最大周波数 100 MHz で動作可能な CM0+です。両方のコアでメモリ保護ユニット (MPU) があります。さらに、ペリフェラル保護ユニット (PPU) と呼ばれるペリフェラルに接続された保護ユニットと、共有メモリ領域用の共有メモリ保護ユニット (SMPU) があります。

CM0+は、セキュアブート機能も提供します。この機能によって、ブート後にシステムの整合性が保護され、ユーザー アプリケーションの実行に先立つ特権が適用されます。

C.1.2 IPC

プロセッサ間通信 (IPC) は 2 つのコアが通信を行い、アクティビティを同期させるための機能を提供します。IPC ハードウェアは PSoC™ 6-BLE のレジスタ構造を使用して実装されます。これらのレジスタ構造は、イベントを同期させ、IPC チャネルの「通知」イベントまたは「解放」イベントをトリガするために使用されます。PSoC™ 6-BLE は最大 16 チャネルがあり、CPU コア間でメッセージを受け渡すことができ、相互排除によるロックをサポートします。

C.1.3 メモリ システム

CPU コアにはメモリとペリフェラルへの共有アクセスを可能にする固定メモリ アドレス マップを持ちます。コードは両方のコアでフラッシュと RAM の両方から実行できます。

PSoC™ 6-BLE ファミリには、最大 1 MB のフラッシュメモリと、EEPROM エミュレーションに使用できる追加の 32 KB のフラッシュがあります。追加の 32 KB の監視フラッシュもあります。さらに、フラッシュは Read-While-Write (RWW) 動作をサポートし、これにより、CPU が命令を実行している時にフラッシュに書き込みができます。ブート ルーチンとコンフィギュレーション ルーチンを含む 128 KB ROM もあります。これにより、エンド アプリケーションにユーザー フラッシュの認証が必要な場合、セキュア ブート動作が確実に行われます。

PSoC™ 6-BLE には最大 288 KB の SRAM メモリがあり、完全に保持することも、ユーザー指定の 32 KB ブロック単位で保持することもできます。

C.1.4 DMA

CPU サブシステムには 2 つの独立した DMA コントローラーが含まれており、それぞれ 32 チャネルで実行できます。コントローラーは、Arm®標準の Advanced Microcontroller Bus Architecture (AMBA) 高性能バス (AHB) を使用して、周辺機器への独立したアクセスをサポートします。

DMA はメモリ、ペリフェラルおよびレジスタとの間でデータを転送します。これらの転送は、CPU から独立して行われます。DMA チャネルは以下をサポートします。

- ソース側とデスティネイション側で 8 ビット、16 ビット、32 ビットのデータ幅
- チャネルごとに 4 つの優先度レベル
- 各 DMA ディスクリプタで設定可能な割込み
- ディスクリプタ チェーン

C.1.5 クロック システム

PSoC™ 6-BLE は以下のクロック ソースがあります。

- 内部主発振器 (IMO): IMO は、PSoC™ 6-BLE 内部クロックの 1 次ソースです。CPU およびすべての高速ペリフェラルは IMO または外部水晶発振器 (WCO) から動作できます。PSoC™ 6-BLE は IMO または ECO のいずれかから動作する多数のペリフェラル クロック分周器があります。これらの分周器は高速のペリフェラルにクロックを生成します。IMO は 8MHz クロックを±1%の精度で生成でき、アクティブモードでのみ使用できます。
- 外部水晶発振器 (ECO): PSoC™ 6 MCU デバイスは、正確なクロックソースのために 4MHz から 33.33MHz の外部水晶を駆動する発振器を含みます。

さらに、調整可能な組込みの水晶負荷容量を備えた Bluetooth® Le サブシステムの外部水晶発振器を使用して、高精度の 32 MHz クロックを生成します。主に、RF クロックを生成する Bluetooth® LE サブシステムのクロックに使用されます。高精度 ECO クロックは、PSoC™ 6-BLE デバイスの高周波クロック (CLK_HF) のクロックソースとしても使用でき、AltHF クロックとして指定されています。

- 外部クロック (EXTCLK): 外部クロックはメガヘルツ範囲のクロックで、指定された I/O ピンの信号から供給できます。このクロックは PLL または FLL のソース クロックとして使用でき、また高周波クロックによって直接使用もできます。
- 内部低速発振器 (ILO): ILO は非常に低消費電力の 32 kHz 発振器であり、主にすべての電力モードで動作する低速周辺機器用のクロックを生成します。
- 高精度内部低速発振器 (PILO): PILO は、ILO よりも正確な 32.768 kHz クロックを提供できる追加のソースです。PILO はディープ スリープ以上のモードで動作します。
- 時計用水晶発振器 (WCO): 32.768 kHz WCO は、ILO/PILO とともに低周波数クロックツリー (CLK_LF) のソースの 1 つとして使用されます。WCO は Bluetooth® LE アドバタイズおよび接続イベントの時間間隔を正確に維持するために使用されます。ILO と同様に WCO はハイバネートとストップ モードを除くすべてのモードで使用可能です。

PSoC™ Bluetooth® LE クロック生成システムには、位相ロックループ (PLL) および周波数ロックループ (FLL) ブロックがあり、これらを使用して高周波クロック (CLK_HF) を生成できます。これらの高周波クロックは、CPU コア クロックとペリフェラル クロック分周器を駆動します。

PSoC™ 6-BLE は 5 つの高周波ルート クロック (CLK_HF[0～4]) を持っています。各 CLK_HF には、シリアルメモリインターフェイスなどのペリフェラルなどのデバイス上の宛先があります。

PSoC™ 6-BLE は、高周波クロックパス 0 (CLK_HF [0]) および WCO にクロック監視機能を備えており、クロックが停止したかどうかを検出して、割込みまたはシステムリセット、あるいはその両方をトリガできます。

C.1.6 システム割込み

PSoC™ 6-BLE は、両方の CPU コア (CM4 および CM0+) での割込みおよび例外をサポートします。コアは、割込み/例外を処理するための独立したベクトル テーブルを提供します。PSOC™ 6-BLE は、CM4 で最大 139 の割込み、CM0+で最大 32 の割込みをサポートできます。最大 33 個の割込みにより、デバイスをディープスリープ電力モードからウェイクアップできます。

C.1.7 電源と電圧監視

PSoC™ 6-BLE デバイス ファミリは、1.71V～3.6V の動作電圧をサポートします。これにはデバイス内のブロックに電力を供給するためにシングル入力マルチ出力 (SIMO) 降圧コンバータを内蔵しています。コアの動作電圧は、0.9V～1.1V の間でユーザーが選択できます。デバイスファミリは、V_{DDD}、V_{DDA}、V_{DDIO}、および V_{BACKUP} の複数の電源レールをサポートします。さらに、Bluetooth® LE 無線操作用の電源レール– V_{DDR} があります。アプリケーションの電源レール/ピンの利用可能性については、選択されたデバイス パッケージによって異なります。

このデバイスには、RTC や WCO などの周辺機器の小さなセットに電力を供給する V_{BACKUP} 電源ピンが含まれています。このレールは他のすべてのレールとは独立しており、他のレールがない場合でも提供できます。これらのブロックへの電源は専用の V_{BACKUP} ピンから供給されるため、これらのブロックは、デバイスの電源が切断されているか、リセット状態に保持されている場合でも動作し続けます。バックアップ ドメインにある RTC は、任意の消費電力モードからデバイスをウェイク アップするオプションを提供します。

C Appendix C. デバイスの特長

PSoC™ 6-BLE ファミリは、パワーオンリセット (POR)、電圧低下検出 (BOD)、過電圧保護 (OVP)、および電源監視用の低電圧検出 (LVD) 回路をサポートし、フェイルセーフ回復を実装します。

PSoC™ 6-BLE の電源の詳細については、PSoC™ 6 MCU: PSoC™ 63 with BLE Architecture Technical Reference Manual を参照してください。

C.1.8 電力モード

消費電力の大きい順での PSoC™ 6-BLE がサポートする消費電力モードは以下のとおりです。

- アクティブ モード: これは動作の主モードです。このモードでは、すべてのペリフェラルがアクティブになり、利用可能です。
- 低消費電力アクティブモード: このモードはアクティブモードに似ており、ほとんどの周辺機器は限られた機能で動作します。CPU コアが利用可能です。性能のトレードオフには、動作クロック周波数の低下、周波数が制限された高周波クロックソース、およびコア動作電圧の低下が含まれます。
- スリープ モード: このモードでは、CPU がスリープ モードであり、SRAM は保持状態、すべてのペリフェラルが利用可能です。割込みが発生すると、いずれかの CPU を起動し、システムをアクティブモードに戻します。CM4 と CM0+はどちらも独立の CPU スリープモードをサポートしており、各 CPU は他の CPU の状態に関係なくスリープ状態にできます。両方のコアがスリープ状態である場合、デバイスはスリープ モードになります。
- 低消費電力スリープモード: ほとんどのペリフェラルが制限された機能で動作し、CPU コアは使用できません。
- ディープ スリープ モード: ディープ スリープ モードでは、すべての高速クロック ソースがオフされます。これにより、ディープスリープでは高速ペリフェラルは使用できなくなります。低周波数クロックで動作するペリフェラルのみが使用可能です。
- ハイバネートモード: デバイスと GPIO はフリーズ状態であり、デバイスはウェイクアップ時にリセットされます。

バッテリ駆動の Bluetooth® LE システムで、スリープ、ディープ スリープ、ハイバネート モードを組み合わせた使用により、バッテリ寿命を延ばし、最高のシステム消費電力を達成できます。表 5 に、PSoC™ 6-BLE システム電力モードと BLESS 電力モード間の依存関係を示します。表 5 でのチェックマークのあるセルは、システムにあたえられた電力モードにあるときに、BLESS が指定されたタスクを実行できることを示します。例えば、BLESS は、システムが低消費電力アクティブモードのときにディープスリープモードにできます。「データ保持」のセルは、システムがディープ スリープ モードに切り替わった時に BLESS 動作コンテキストが保持されていることを示します。

注: PSoC™ 6-BLE デバイスでサポートされるコモンとセグメントの数は、デバイス ファミリとデバイス パッケージによって異なります。詳細は関連するデバイス データシートを参照してください。

表 5 PSoC™ 6-Bluetooth® LE 消費電力モード

BLESS モード	PSoC™ 6-Bluetooth® LE システム電力モード					
	アクティブ	低消費電力アクティブ	スリープ	低消費電力スリープ	ディープ スリープ	ハイバネート
送信	✓	✓	✓	✓	データ保持	オフ
受信	✓	✓	✓	✓	データ保持	オフ
アイドル	✓	✓	✓	✓	データ保持	オフ
ディープ スリープ	✓	✓	✓	✓	データ保持	オフ

C.2 セキュア ブート

セキュアブートには、市場/アプリケーション固有の規格で定義されているキーベースのセキュリティプロトコルを使用し、アプリケーションのフラッシュイメージを認証する必要があります。セキュアブートプロセスは、SROM および PSoC™ 6-BLE デバイスの個別の監視フラッシュに実装されます。ブートコードは、ブートコードのチェックサムを計算し、それを eFuse の値と比較します。これらの値が一致しない場合、ブート プロセスは失敗します。ブートコードは eFuse で指定されたデバッグ アクセス制限も適用します。セキュアブートはオプション機能であり、エンドユーザーが有効にする必要があります。

C.3 プログラマブル デジタル ペリフェラル

C.3.1 UDB

図 71 に示すように、UDB は CPLD および FPGA ブロックと同様な機能を提供するプログラマブル ロジック ブロックです。UDB を使用すると、タイマー、カウンター、PWM、疑似ランダム シーケンス (PRS) 、CRC、シフトレジスタ、SPI、UART、I^2S、カスタムの組合せおよびシーケンシャルロジック回路など、さまざまなデジタル機能を作成できます。

各 UDB は 2 個のプログラマブル ロジック デバイス (PLD) を持ち、それぞれ 12 本の入力と 8 つのプロダクト タームがあります。PLD は登録済みまたは組合せ積和ロジックを形成できます。さらに、「データパス」として知られている 8 ビットのシングル サイクル算術ロジック ユニット (ALU) が各 UDB にあります。データパスは、タイマー、カウンター、PWM、および CRC などの機能の効率的な実装に役立ちます。

また、UDB はペリフェラルやポートからの信号を UDB との間でやり取りして通信や制御を行えるスイッチド デジタル シグナル インターコネクト (DSI) ファブリックを提供します。

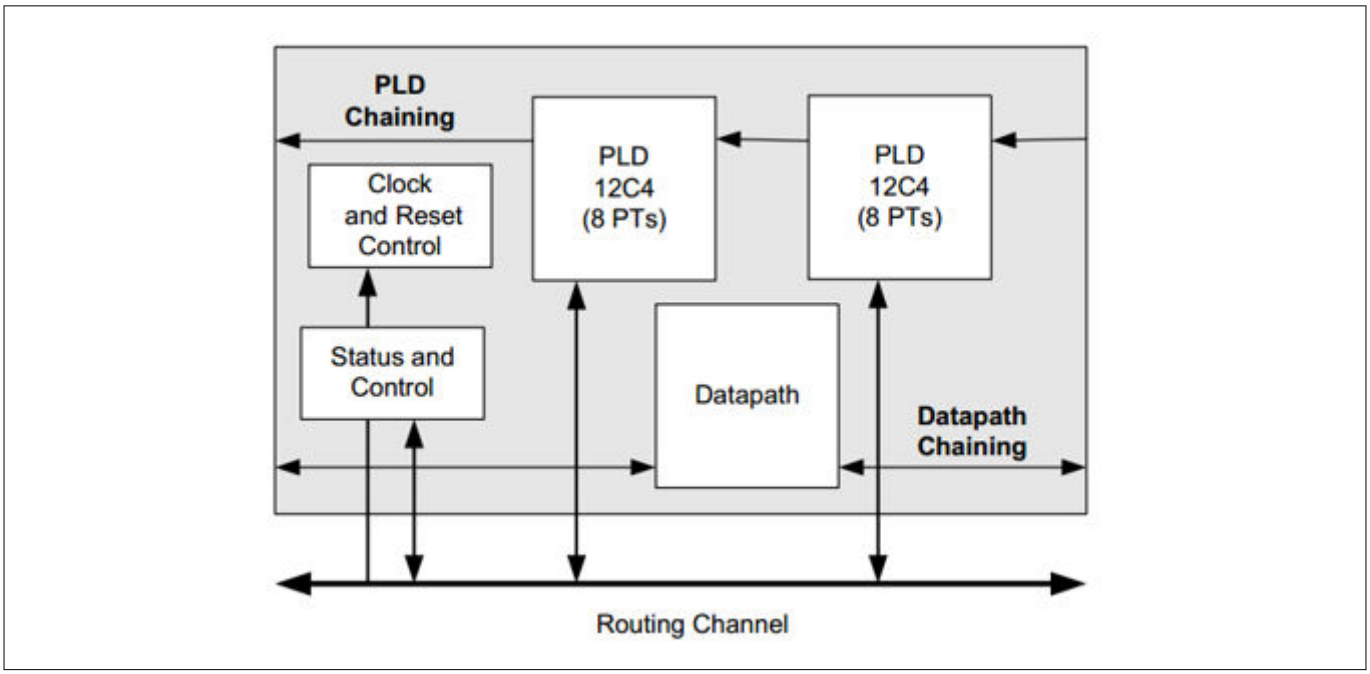


図 71 汎用デジタル ブロックダイヤグラム

UDB を使用するために必ずしもハードウェア記述言語 (HDL) を知る必要はありません。PSoC™ 6 BLE 用開発ツールである PSoC™ Creator は、回路図から必要な機能を生成できます。必要に応じて、上級ユーザーは Verilog を使用して UDB にカスタムロジックを実装できます。

PSoC™ 6-BLE は最大 12 の UDB を持ちます。UDB の詳細については、次のアプリケーションノートを参照してください。

- AN62510 - Implementing state machines with PSoC™ 3, PSoC™ 4, and PSoC™ 5LP
- AN82156 - PSoC™ 3, PSoC™ 4, and PSoC™ 5LP - UDB データパスを用いた PSoC™ Creator コンポーネントの設計
- AN82250 - PSoC™ 3, PSoC™ 4, and PSoC™ 5LP - Verilog によるプログラマブル ロジック設計の実装方法

C.3.2 プログラマブル TCPWM

PSoC™ 6-BLE は 32 のプログラマブル TCPWM ブロックを持ちます。各 TCPWM はタイマー、カウンター、PWM または直交デコーダーとして実装できます。TCPWM はデッド バンド プログラマブル相補 PWM 出力と選択可能な起動、リロード、停止、カウントおよびキャプチャ イベント信号を提供します。PWM モードは、中央揃え、エッジ、擬似乱数操作をサポートします。またキル入力もあり、強制的に出力を既定の状態にできます。

詳細は PSoC™ 6-BLE TCPWM コンポーネント データシートを参照してください。

C.3.3 SCB

PSoC™ 6-BLE には、I²C、SPI、または UART を備えた最大 9 つの独立したランタイムプログラマブル SCB があります。SCB は以下の機能に対応します。

- Motorola®、Texas Instruments®のセキュア シンプル ペアリング (SSP) および National Semiconductor® Microwire プロトコルで標準 SPI マスターおよびスレーブの機能
- スマートカードリーダー、シングルワイヤー ローカル相互接続ネットワーク (LIN) インターフェース、SmartCard (ISO7816) 、および赤外線データアソシエーション (IrDA) プロトコルを備えた標準 UART 機能 (最大 1Mbps ボーレート)
- 動作速度最大 1Mbps の標準 I2C マスターおよびスレーブ機能
- CPU 介入なしで動作可能な EzSPI および EzI2C モード
- 1 つの SCB は外部クロックでディープ スリープ モードで動作するように設定可能。低消費電力 (ディープスリープ) モードは、スレーブモードでのみ SPI および I2C プロトコル (外部クロックを使用) でサポートされる

詳細については、PSoC™ 6-BLE SCB コンポーネントデータシートを参照してください。

C.3.4 BLESS

PSoC™ 6-BLE は Bluetooth® 4.2 仕様で指定されている BLE リンク層と物理層を実装する Bluetooth®スマート サブシステムが組み込まれています。Bluetooth® LE サブシステムには、AES-128 セキュリティエンジンが組み込まれた物理層 (PHY) とリンク層エンジンが含まれています。サブシステムは、Bluetooth® SIG で定義されているすべての Bluetooth® LE プロファイルをサポートします。

物理層は Bluetooth® 4.2 仕様に準拠したデジタル PHY および RF トランシーバで構成されます。トランシーバは 2.4 GHz ISM バンド 1 Mbps で GFSK パケットを送受信します。ベースバンド コントローラーはマスターとスレーブ モードの両方に対応する複合ハードウェア／ファームウェアでの実装です。HCI やリンク制御などの重要なプロトコル要素はファームウェアで実装されていますが、暗号化、CRC、データ ホワイトニング、アクセス コード相関などのタイムクリティカルな機能はハードウェアで実装されます。

BLESS は Bluetooth® 4.2 準拠であり、Bluetooth® 4.0 仕様のすべての機能と、低デューティサイクル アドバタイズ、LE ping、L2CAP 接続指向チャネル、リンク層プライバシー、リンク層データ長拡張および LE 安全接続などの Bluetooth® 4.2 仕様のいくつかの追加機能をサポートします。BLESS ブロックには、正確な RF 周波数を生成し、Bluetooth® LE 接続での連続する接続間隔間の時間をそれぞれ維持するために必要な ECO および WCO も含んでいます。

Bluetooth® LE サブシステムは、最大 4 つの同時接続をサポートします。ディープスリープ、アイドル、送信、および受信の、4 つの機能電力モードをサポートしています。

注: *ここで説明する電力モードは BLESS ブロックの固有のモードです。PSoC™ 6-BLE システムの電力モードについては、「付録 C.1.8」セクションを参照してください。*

C.3.4.1 ディープ スリープ モード:

ディープ スリープ モードは BLESS によりサポートされるもっとも低い消費電力モードです。このモードでは、無線ブロックはオフです。パケットの送受信が完了した後、アドバタイズまたは接続間隔の間で最大の節電のため、このモードになります。節電のためにこのモードで ECO をオフにできます。低周波数クロックである WCO は、BLE リンク層のタイミング基準ロジックを維持するためにオンになります。アプリケーション ファームウェアは、この状態に入ることと出ることを制御します。

C.3.4.2 スリープ モード

スリープ モードでは、無線ブロックはオフです。ブロックはすべての設定を保持します。ECO と WCO はオンですが、コア BLESS ロジックへのクロックはオフです。アプリケーション ファームウェアは、この状態に入ることと出ることを制御します。

C.3.4.3 アイドル モード

アイドル モードは送信と受信状態の準備状態です。このモードでは、無線ブロックはオフですが、CPU がプロトコル ステート マシンを起動するためにリンク層ロジックのリンク層クロックは有効にされています。

C.3.4.4 送信モード

送信モードはアクティブ機能モードです。このモードで、BLESS のすべてのブロックはオンになります。リンク層クロックは、リンク層および RF-PHY 内のロジックを完成させるために有効にされます。このモードでは、RF-PHY はリンク層から最大 2 Mbps のシリアルデータを取得し、2.4 GHz GFSK 変調データをアンテナ ポートに送信します。BLESS はアイドル モードから送信モードに入ります。

C.3.4.5 受信モード

このモードは、BLESS が受信状態に入り、BLE 固有のレシーバー動作を実行できるようにします。RF-PHY は、RF アナログ ブロックから受信した 1Mbps データを変換し、復調後にリンク層コントローラーに転送します。BLESS 電力モードと動作サブブロックの概要を表 6 に示します。

表 6 BLESS 電力モード

BLESS 電力モード	ECO	WCO	RF Tx	RF Rx	BLESS コア
ディープ スリープ	オフ	オン	オフ	オフ	オフ
スリープ	オン	オン	オフ	オフ	オフ
アイドル	オン	オン	オフ	オフ	オン
送信	オン	オン	オン	オフ	オン
受信	オン	オン	オフ	オン	オン

C.3.5 オーディオ サブシステム

PSoC™ 6-BLE は、I²S ブロックと 2 つの PDM チャネルで構成されるオーディオサブシステムを持ちます。PDM チャネルは、デジタル マイクのビットストリーム出力とインターフェースします。PDM 処理チャネルはドループ補正を提供し、384kHz～3.072MHz のクロック速度で動作でき、最大 48ksps のオーディオ サンプリング速度で 16～24 ビットのワード長を生成します。

I²S インターフェースは、8 ビット～32 ビットのワードを送信する時、最大 192ksps のワードクロックレートでマスターモードとスレーブモードの両方をサポートします。

C.3.6 シリアル メモリ インターフェース

PSoC™ 6-BLE のシリアル メモリ インターフェース (SMIF) は、さまざまなタイプのメモリと最大 4 つのメモリとのインターフェースが可能です。SMIF はオクタル SPI (8 ビット/サイクルスループット)、デュアルクアッド SPI (8 ビット/サイクルスループット)、クアッド SPI (4 ビット/サイクルスループット)、デュアル SPI (2 ビット/サイクルスループット)、および SPI (1 ビット/サイクルのスループット)。このブロックは、CPU コアが外部メモリから直接コードを実行できる実行インプレース (XIP) モードもサポートしています。PSoC™ Creator で用意されたソフトウェア モジュールとともに SMIF ブロックを使用すると、アプリケーションで適格な所定のメモリ デバイス セットを使用できます。

C.3.7 eFUSE

eFuse はセキュリティ関連の設定をプログラムするために使用されるワンタイム プログラマブルです。1 度プログラムされ、後で使用されるアプリケーション設定を保存するためにも使用できます。

C.3.8 セグメント LCD

PSoC™ 6-BLE のセグメント LCD 駆動は以下の特長があります。

- 最大 8 コモン (COM) と 64 セグメント (SEG) 電極をサポート
- プログラマブル GPIO により、COM と SEG の電極のフレキシブルな選択が可能
- 14 セグメントと 16 セグメント構成のキャラクタ ディスプレイ、7 セグメント構成のディスプレイ、ドット マトリクスおよび特殊記号をサポート
- 2 つの駆動モード:デジタル相関および PWM
- ハイバネートを除くすべてのシステム電源モードで動作
- 1.8 ボルト VDD から 3 ボルトのディスプレイを駆動可能
- デジタル コントラスト制御

C.4 プログラマブル アナログ ペリフェラル

C.4.1 連続時間ブロック オペアンプ

PSoC™ 6-BLE デバイスは、入力と出力が固定位置ピンに接続された連続時間ブロック (CTBm) ベースのオペアンプのペアを持ちます。オペアンプは、ハイバネートを除くすべての電力モードをサポートしています。しかし、オペアンプは、ディープスリープモードで利得帯域幅積を低下させて動作します。一般的に、これらのオペアンプの出力は、SAR 入力のバッファとして使用できます。

C.4.2 低消費電力コンパレータ

PSoC™ 6-BLE は、ディープスリープとハイバネートモードで動作もできる低消費電力コンパレータの 1 対を内蔵しています。コンパレータは低消費電力モードで 300 nA 未満の電流を消費します。低消費電力の設計では、デバ

イスが低消費電力モードに入った時、低消費電力コンパレータを使用してアナログ入力を監視し、システムを復帰させられる割込みを生成できます。

詳細は PSoC™ 6-BLE Low-Power Comparator Component datasheet を参照してください。

C.4.3 SAR ADC

PSoC™ 6-BLE は、プログラム可能な解像度とシングルエンドまたは差動入力オプションをサポートする入力チャネルを備えた 12 ビット、1 Msps 逐次比較レジスタ (SAR) ADC を内蔵しています。実装可能な ADC 入力チャネルの数は GPIO の数に依存します。SAR ADC は高速クロックを必要とするため、ディープ スリープ モードとハイバネート モードに対応していません。

SAR ADC は、CPU の介入無しに最大 8 本のチャネルで自動スキャンを実行できるハードウェア シーケンサを備えています。SAR ADC は 8 入力シーケンサを介して、決められたピンに接続されます。シーケンサは、スイッチング オーバーヘッドなしで選択されたチャネルを自動的に (シーケンサ スキャン) 循環します。また、これらの 8 つのチャネルで出力データの累積や平均化などの前処理動作をサポートします。

ファームウェアやタイマー、ピン、UDB などを使用してスキャンをトリガでき、追加の設計の柔軟性を与えます。

ノイズの多い状況でのパフォーマンスを改善するために、内部基準アンプに対して、外部バイパスコンデンサを固定したピン位置に設定できます。SAR ADC の詳細については、PSoC™ 6-BLE SAR ADC コンポーネントデータシートを参照してください。

PSoC™ 6-BLE には、温度を測定するためのオンチップ温度センサーを内蔵しています。温度センサーは SAR ADC に接続できます。SARADC は、校正と線形化を含むソフトウェアコンポーネントを使用し、読み出し値をデジタル化し、温度値を生成します。

C.4.4 DAC

PSoC™ 6-BLE は 12 ビット電圧モード連続時間 DAC (CTDAC) を持ち、セトリング時間は 2µs です。12 ビット DAC は外部のサンプルとホールド (S/H) 回路を必要とせずに連続時間出力を提供します。DAC 制御インターフェースにより、CPU と DMA を介して DAC 出力を制御できます。これはダブルバッファ DAC 電圧制御レジスタ、プログラマブル更新レート用のクロック入力、CPU に DAC の空バッファの割込みおよび DMA へのトリガが含まれます。このため、DAC はユーザー定義の波形を生成するために DMA コントローラーにより駆動されます。

DAC の詳細については、PSoC™ 6-BLE DAC コンポーネントデータシートを参照してください。

C.4.5 CAPSENSE™

PSoC™ 6-BLE デバイスの第 4 世代 CAPSENSE™は、自己容量と相互容量ベースのタッチセンシングをサポートします。CAPSENSE™はすべてのピンでサポートされます。

静電容量タッチ センサーは、人体の静電容量を使ってセンサー上またはセンサーの近くの指の存在を検出します。静電容量センサーは審美的に優れており、使い易く、長寿命です。PSoC™ 6-BLE の CAPSENSE™機能はクラス最高の SNR、クラス最高の耐液性、および広く多様なセンサータイプ (ボタン、スライダー、トラックパッド、および近接センサーなど) を提供します。

ソフトウェア コンポーネントは、静電容量センシングの設計を簡単にします。このコンポーネントは、SmartSense というハードウェア自動チューニング機能をサポートし、トラックパッドと近接センサーのジェスチャー認識ライブラリを提供します。

詳細については、AN85951 – PSoC™ 4 および PSoC™ 6 MCU CAPSENSE™デザインガイドを参照してください。

CAPSENSE™ブロックには 2 つの 7 ビット IDAC があります。CAPSENSE™がそれらを使用していない場合、これらは汎用目的に使用できます。1 つの IDAC の使用により、(低速) 10 ビット スロープ ADC を実現できます。スロープ ADC の使用方法の詳細は PSoC™ 6-BLE CAPSENSE™コンポーネント データシートを参照してください。

C.5 プログラマブル GPIO

I/O システムは、CPU コア、ペリフェラル、および外部デバイス間のインターフェースを提供します。PSoC™ 6-BLE は最大 104 本のプログラマブル GPIO ピンを備えています。CAPSENSE™、LCD、アナログ、またはデジタル信号用に GPIO を設定できます。PSoC™ 6-BLE の GPIO は多くの駆動モード、駆動強度およびスルーレートに対応しています。

PSoC™ 6-BLE は内部信号を GPIO に接続する多数のオプションを与えるインテリジェントな配線システムを提供します。この柔軟な配線により、回路設計と基板レイアウトが容易になります。

さらに、PSoC™ 6-BLE には最大 2 つのスマート I/O ポートがあり、GPIO ピンに入出力信号に対してブール演算を実行するために使用できます。スマート I/O ポートは、ディープスリープモードでも動作します。

D Appendix D. IoT 開発ツール

D.1 CY8CKIT-062-BLE PSoC™ 6-BLE Pioneer Kit

図 72 に示す PSoC™ 6-BLE Pioneer Kit は、PSoC® 6-BLE デバイス ファミリに対応している Bluetooth™ LE 開発キットです。

以下に PSoC™ 6-BLE Pioneer Kit ベースボードの機能を示します。

- コイン型電池または Type-C USB インターフェースを介して受電することが可能。Type-C USB インターフェースは、最大 12V、3A の電力供給 (PD) のコンシューマおよびプロバイダ プロファイルもサポート
- 標準の Arduino Uno コネクタ準拠のシールド、CAPSENSE™ユーザーインターフェイスやシリアルメモリ インターフェースなどのオンボード PSoC™ 6-BLE デバイス機能と連携して動作する、バッテリ駆動の低消費電力 Bluetooth® LE 設計の開発を可能する
- Cortex® Debug/ETM コネクタを使用し、サードパーティのプログラミング、デバッグおよびトレースをサポート
- Digilent などのサードパーティベンダーの Pmod™ドーターカードとのインターフェースをサポートする追加ヘッダを含む
- ボイスオーバー Bluetooth® LE 機能用の PDM-PCM マイクをサポート
- QSPI NOR フラッシュと F-RAM を含む

本キットには以下が含まれます。

- Bluetooth® LE リンクマスターとして機能し、CySmart ホストエミュレーションツールと連携して非 Bluetooth® LE Windows PC に Bluetooth® LE ホストエミュレーションプラットフォームを提供する USB-Bluetooth® LE ドングル
- E-Ink ディスプレイ

このキットには、ユーザーが独自の Bluetooth® LE アプリケーションを開発できるように、BLE サンプル プロジェクトと資料 1 式が含まれます。キットの最新版、キットデザイン、サンプルプロジェクト、およびドキュメントファイルを入手するには CY8CKIT-062-BLE PSoC™ 6 BLE Pioneer Kit ウェブページにアクセスしてください。

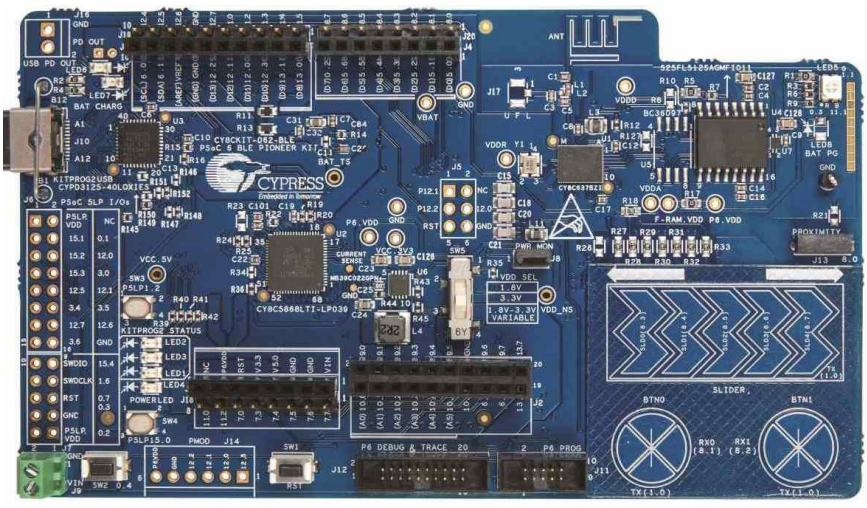


図 72 CY8CKIT-062-BLE PSoC™ 6-BLE Pioneer Kit

D.2 CySmart ホスト エミュレーション ツール

CySmart ホストエミュレーションツールは、PSoC™ 6-BLE Pioneer Kit のドングルを使用して Bluetooth® LE セントラル デバイスをエミュレートする Windows アプリケーションです。図 73 を参照してください。このアプリケーションは Bluetooth® LE Pioneer Kit の一部としてインストールされます。Bluetooth® LE コンポーネントを右クリックして

D Appendix D. IoT 開発ツール

起動できます。ペリフェラルの Bluetooth® LE サービス、特性、および属性を調べて設定することで、GATT または L2CAP 接続指向のチャネルを介して、ご使用の PSoC™ 6-BLE ペリフェラル実装をテストするためのプラットフォームを提供します。

CySmart ホスト エミュレーション ツールを使用し、実現できる動作は以下に含まれますが、これらに限定されません。

- Bluetooth® LE ペリフェラルをスキャンして、接続できるデバイスを検出
- サービスや特性などの、接続したペリフェラルデバイスの Bluetooth® LE 属性を検出
- 特性値とディスクリプタにもとづいた読み書き
- 接続したペリフェラル デバイスからの特性通知および指示を受ける
- Bluetooth® LE セキュリティ マネージャを使用して、接続したペリフェラル デバイスとの結合を確立
- ペリフェラル デバイスとの Bluetooth® LE L2CAP 接続指向のセッションを確立し、Bluetooth® 4.2 仕様に従ったデータ交換を行う
- Bluetooth® LE ペリフェラル デバイスの Over-the-air (OTA) ファームウェア アップグレード

図 73 と図 74 に、CySmart ホスト エミュレーション ツールのユーザー インターフェースを示します。このツールのセットアップおよび使用方法の詳細は、**Help** メニューの CySmart ユーザー ガイドを参照してください。

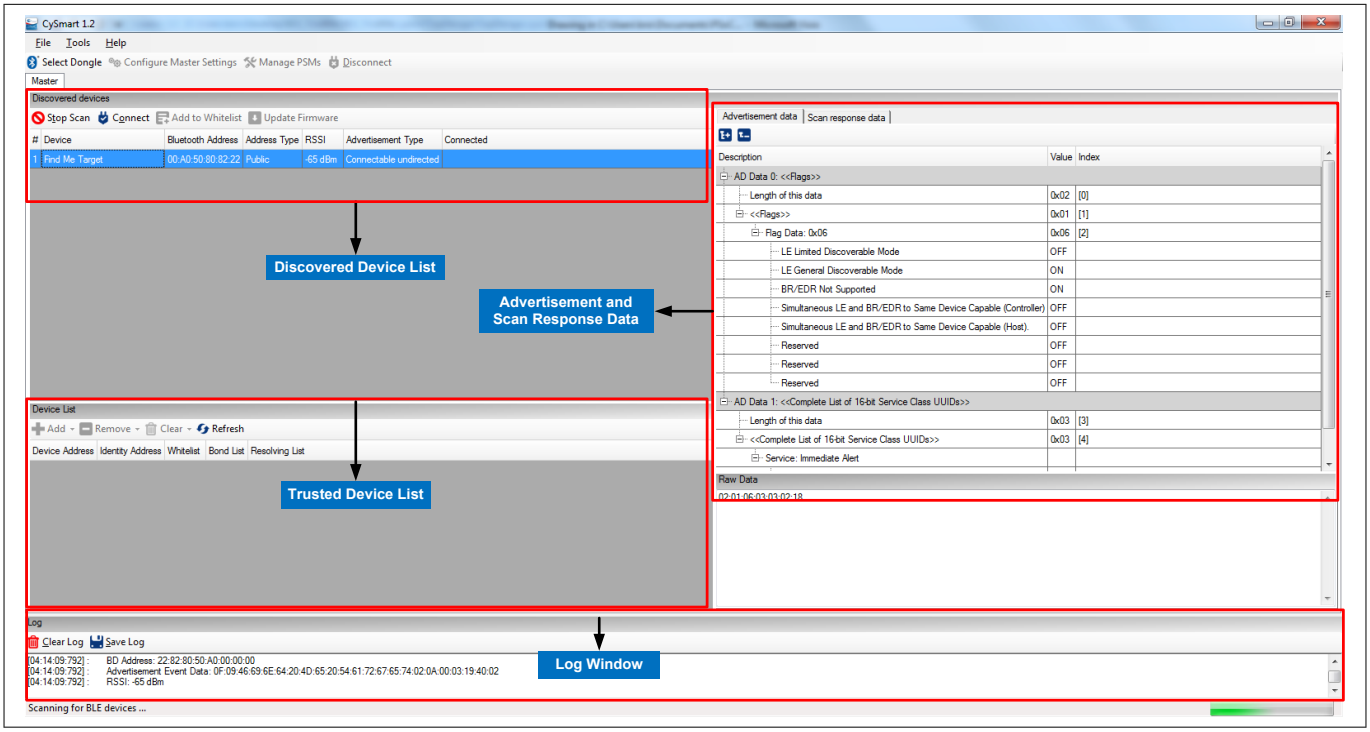


図 73　　CySmart ホスト エミュレーション ツールのマスター デバイス タブ

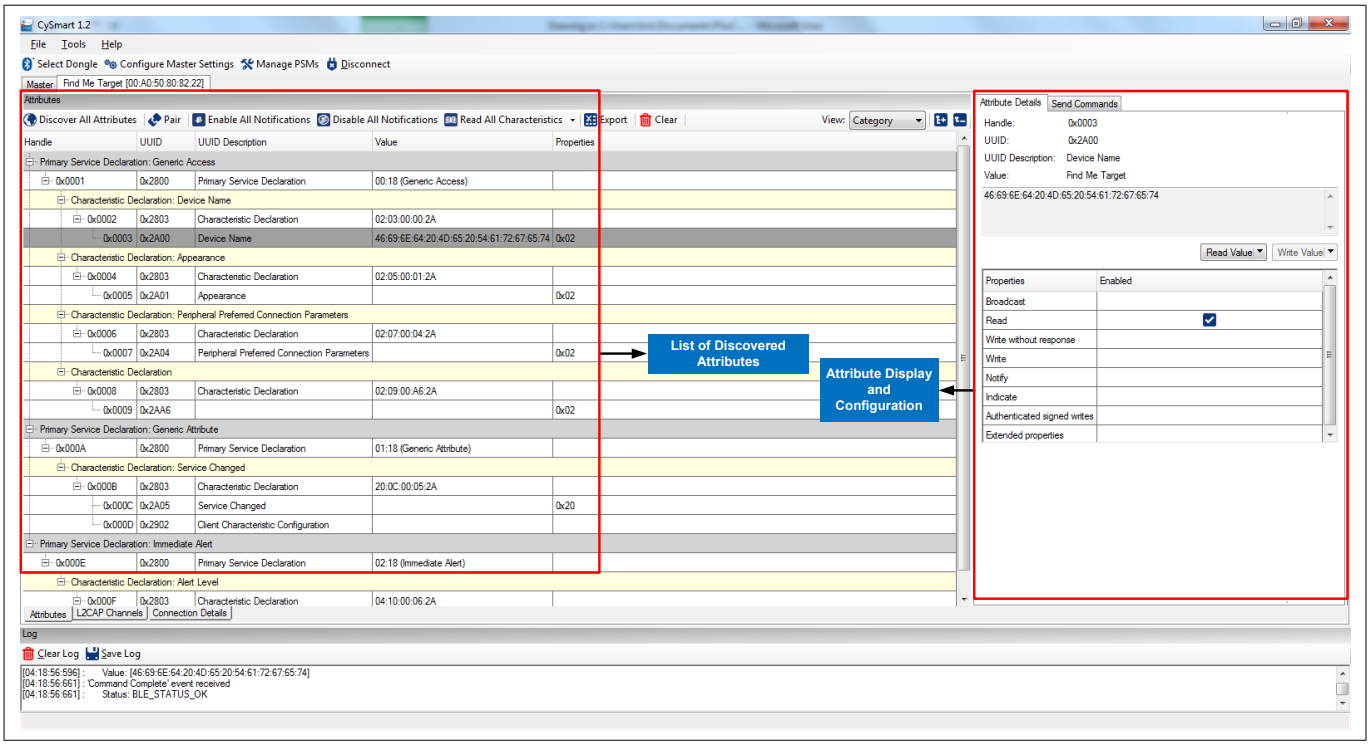


図 74　　CySmart ホスト エミュレーション ツールのペリフェラル デバイス属性タブ

D.3　CySmart モバイル アプリケーション

PC ツールの他に、それぞれのアプリケーション ストアから iOS または Android 向けの CySmart モバイル アプリをダウンロードできます。このアプリは、Bluetooth® LE 用に iOS コア Bluetooth®フレームワークおよび Android 組込みプラットフォームフレームワークをそれぞれ使用します。これは Bluetooth® LE 対応のスマートフォンを、ペリフェラル デバイスをスキャンして接続できるセントラル デバイスとして設定します。

このアプリは、直感的な GUI を使用して SIG 採用の Bluetooth® LE 標準プロファイルをサポートし、基本的な Bluetooth® LE サービスおよび特性の詳細を抽象化します。Bluetooth® LE 標準プロファイルに加えて、アプリは LED と CAPSENSE™実証例を使用してカスタムプロファイルの実装をデモします。図 75 と図 76 に、CySmart アプリにおける心拍数プロファイルのユーザー インターフェースのスクリーンショットを示します。Bluetooth® LE Pioneer Kit のプロジェクト例を使用し、アプリケーションを実行する方法については Bluetooth® LE Pioneer Kit ガイドを参照してください。

D Appendix D. IoT 開発ツール

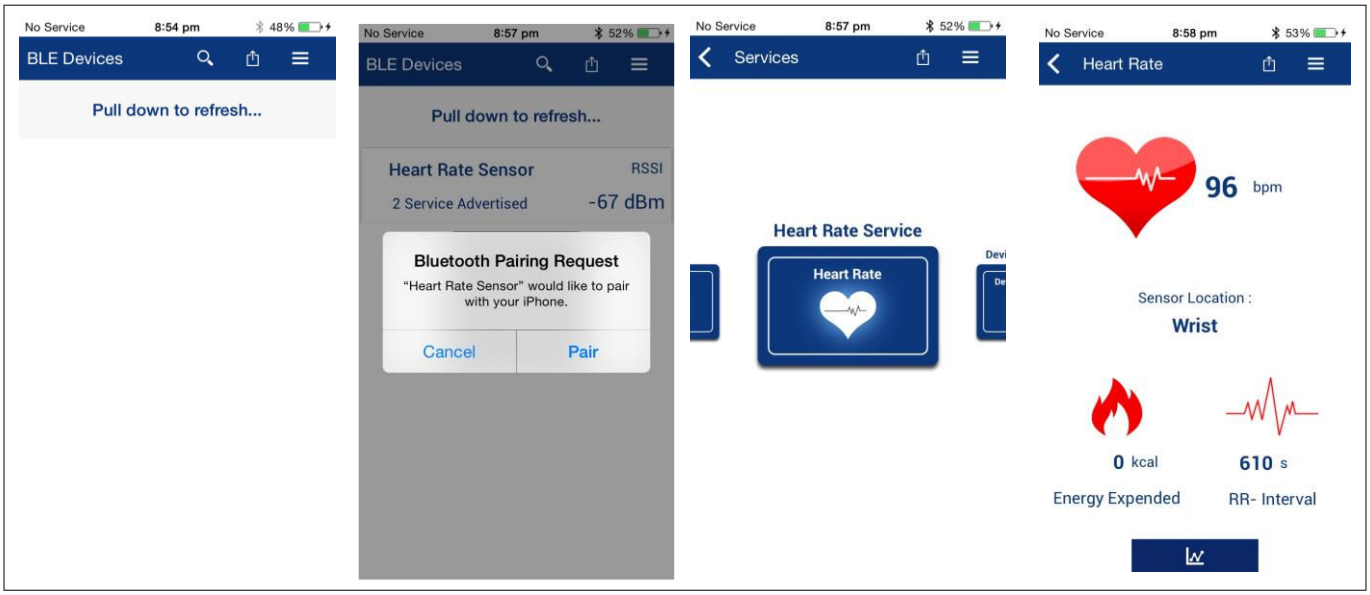


図 75　　CySmart iOS アプリの心拍数プロファイルの例

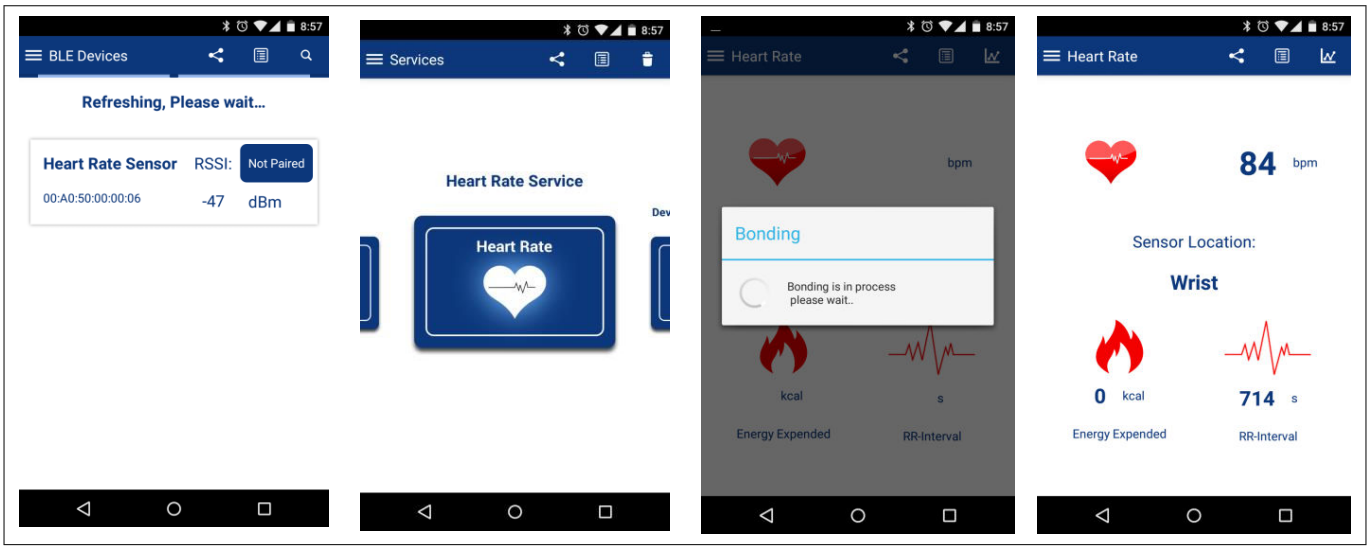


図 76　　CySmart Android アプリの心拍数プロファイルの例

改訂履歴

版数	発行日	変更内容
英語版**	-	本版は英語版のみの発行です。
英語版*A	-	本版は英語版のみの発行です。 Rewrite of application note. Updated UDB template
**	2018-07-05	これは英語版 002-10781 Rev. *B を翻訳した日本語版 Rev. **です。
英語版*C	-	本版は英語版のみの発行です。 Partial rewrite of application note. Updated template, abstract, and figures
*A	2020-04-28	これは英語版 002-10781 Rev. *D を翻訳した日本語版 Rev. *A です。
*B	2022-12-19	これは英語版 002-10781 Rev. *E を翻訳した日本語版 Rev. *B です。 ・BLE を Bluetooth® LE または Bluetooth® Low Energy に、また PSoC™ 6 BLE を PSoC™ 6-BLE に更新。 ・ソフトウェアのバージョンを PSoC™ Creator 4.4 に更新。 ・ドキュメント内のハイパーリンクを修正。

Trademarks

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

Edition 2022-12-19

Published by

Infineon Technologies AG

81726 Munich, Germany

© 2022 Infineon Technologies AG

All Rights Reserved.

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

Email: erratum@infineon.com

Document reference
IFX-vai1649675934009

重要事項

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

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

警告事項

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

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