

PRoC™ BLE 入門

著者: Sai Prashanth Chinnappalli, Rahul Garg

関連プロジェクト: なし

関連製品ファミリ: CYBL10X6X, CYBL10X7X

ソフトウェア バージョン: PSoC Creator™ 3.2 以降

関連アプリケーション ノート: [ここをクリック](#)本アプリケーション ノートの最新版を入手するには www.cypress.com/go/AN94020 へアクセスしてください。

本アプリケーション ノート (AN94020) では、ARM® Cortex™-M0 ベースのプログラマブル ラジオ オン チップに BLE 無線機能を搭載した製品「PRoC BLE」についてご紹介いたします。PRoC BLE ソリューションおよび開発ツールを探索できる他、PSoC Creator™ という PRoC BLE 用の開発ツールを使用して初めて自分の BLE プロジェクトをビルドする際に有用なご案内も提供いたします。さらに、PRoC BLE に関わる深い知識を迅速に取得できるように提供するリソースへのアクセス方法も記述しております。

目次

1	はじめに	2	11.2	CySmart モバイル アプリケーション	46
2	PSoC リソース	2	12	付録 D: PRoC BLE デバイス	47
2.1	PSoC Creator	3	12.1	ARM Cortex-M0 とメモリ	47
2.2	PSoC Creator ヘルプ	4	12.2	BLE サブシステム	47
2.3	サンプル コード	4	12.3	プログラマブルなデジタル ペリフェラル	48
3	PRoC BLE の機能	6	12.4	設定可能なアナログ ペリフェラル	48
4	BLE の概要	7	12.5	システム全体のリソース	49
4.1	BLE 接続の確立	8	12.6	プログラマブル GPIO	51
4.2	GATT データ形式	9	13	付録 E: BLE プロトコル	52
4.3	BLE プロファイル	11	13.1	概要	52
4.4	BLE コンポーネント	11	13.2	物理層 (Physical Layer; PHY)	52
5	PRoC BLE 開発セットアップ	14	13.3	リンク層 (Link Layer; LL)	52
6	初めての PRoC BLE 設計	16	13.4	ホスト制御インターフェース (Host Control Interface; HCI)	53
6.1	設計について	16	13.5	論理リンク制御および適応プロトコル (Logical Link Control and Adaptation Protocol; L2CAP)	53
6.2	事前準備	16	13.6	セキュリティ マネージャ (Security Manager; SM)	54
6.3	第 1 段階: 設計を作成/設定	17	13.7	アトリビュート プロトコル (Attribute Protocol; ATT)	54
6.4	第 2 段階: アプリケーション コードを書く	28	13.8	一般的アトリビュート プロファイル (Generic Attribute Profile; GATT)	58
6.5	第 3 段階: デバイスをプログラム	34	13.9	一般的アクセス プロファイル (Generic Access Profile; GAP)	58
6.6	第 4 段階: 設計をテスト	35	改訂履歴		61
7	まとめ	39	ワールドワイド販売と設計サポート		62
8	関連アプリケーション ノート	39			
9	付録 A: BLE デバイス ファミリの比較	40			
10	付録 B: サイプレスの専門用語	42			
11	付録 C: サイプレス BLE 開発ツール	43			
11.1	CySmart ホスト エミュレーション ツール	43			

1 はじめに

PRoC BLE は、CapSense®、12 ビットアナログ-デジタル変換器 (ADC)、4 個のタイマー/カウンタ/PWM (TCPWM)、4 個の追加 8/16 ビット PWM、36 本の GPIO、2 個のシリアル通信ブロック (SCB)、液晶ディスプレイ (LCD) ドライバー、および IC 間サウンド (I2S) を搭載する 32 ビットの 48MHz ARM Cortex-M0 BLE チップです。これはヒューマン インターフェース デバイス (HID)、リモコン、玩具、ビーコンおよびワイヤレス充電器に、完全に柔軟なプログラマブルなソリューションを提供します。さらに、PRoC BLE といった低コストで簡単な手段を利用することで BLE 接続をどのシステムにも追加できます。

PRoC BLE は、Bluetooth 4.1 仕様に適合する無償の BLE プロトコル スタック (コントローラーとホスト) からなり、Bluetooth Special Interest Group (BT SIG) 採択済みプロファイルすべてを含む (仕様リビジョンおよびプロファイルの将来の変更に対応できるようにロードマップも用意した) BLE プロファイルのパッケージソフトで提供されます。

PRoC BLE は、時計用水晶発振器 (WCO) 有効の 1.3µA のディープスリープ モード; SRAM 内容、プログラマブルなロジックおよび割り込みによるウェイクアップ機能を維持する 150nA のハイバネート モード; および GPIO 割り込みによるウェイクアップ能力を維持する 60nA のストップ モードといった低電力モードが整備されます。

PRoC BLE 内の CapSense という容量タッチ センシング特長は、ボタン、スライダー、トラック パッドや近接センサーなどの多種多様なセンサー タイプを提供します。このチップの高い信号対ノイズ比および耐水性により誤タッチを回避できます。

PRoC BLE の他、サイプレスは、BLE 無線、プログラム可能なアナログとデジタル周辺機能、メモリ、および ARM Cortex-M0 マイクロコントローラーを単一のチップに集積したプログラマブル組み込みシステムオンチップ「PSoC 4 BLE」製品も提供します。詳細については、[AN91267 - Getting Started with PSoc® 4 BLE アプリケーション ノート](#)をご覧ください。サイプレスの BLE デバイス ファミリー内の各製品間の比較については、[付録 A: BLE デバイス ファミリーの比較](#)をご覧ください。

2 PSoc リソース

サイプレスは、www.cypress.com に大量のデータを掲載しており、ユーザーがデザインに適切な PRoC/PSoC デバイスを選択し、デザインに迅速かつ効率的に統合する手助けをしています。サイプレスの PRoC/PSoC 製品ファミリーの初心者のお客様は、[付録 B: サイプレスの用語](#)に記載する一般的に使用する用語一覧をご覧ください。リソースの総合リストについては、「[KBA86521, How to Design with PSoc 3, PSoc 4, and PSoc 5LP](#)」をご参照ください。

以下は PRoC BLE のリソースの要約です。

- **概要:** [PSoc ポートフォリオ](#)、[PRoC ポートフォリオ](#)
- **製品セクター:** [PSoc 1](#)、[PSoc 3](#)、[PSoc 4](#) または [PSoc 5LP](#)。さらに、[PSoc Creator](#) にはデバイス選択ツールが含まれています。
- **データシート:** [CYBL10X6X](#) と [CYBL10X7X](#) デバイス ファミリー向けの電気的仕様を説明し提供します。
- **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。多くの [アプリケーション ノート](#) はサンプル コードを含んでいます。PSoc Creator は追加のサンプル コードを提供します。詳細は、「[サンプル コード](#)」をご参照ください。
- **テクニカル リファレンス マニュアル (TRM):** 各 PRoC BLE デバイス ファミリーの [アーキテクチャとレジスタ](#) について詳細に説明します。
- **CapSense デザイン ガイド:** PRoC BLE ファミリーの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](#) について説明します。
- **開発ツール**
 - [CY8CKIT-042-BLE Bluetooth Low Energy \(BLE\) Pioneer Kit](#): 使いやすく安価な BLE 向けのプラットフォームです。このキットは、Arduino™ 準拠シールドおよび Digilent® Pmod™ ドーターカード用コネクタを搭載します。
 - [Windows](#)、[iOS](#)、および [Android](#) 用 CySmart BLE Host Emulation Tool: ご使用の BLE ペリフェラル アプリケーションをテストおよびデバッグするための使い勝手の良い GUI です。

その概要については、[付録 C: サイプレス BLE 開発ツール](#)をご覧ください。

■ テクニカル サポート

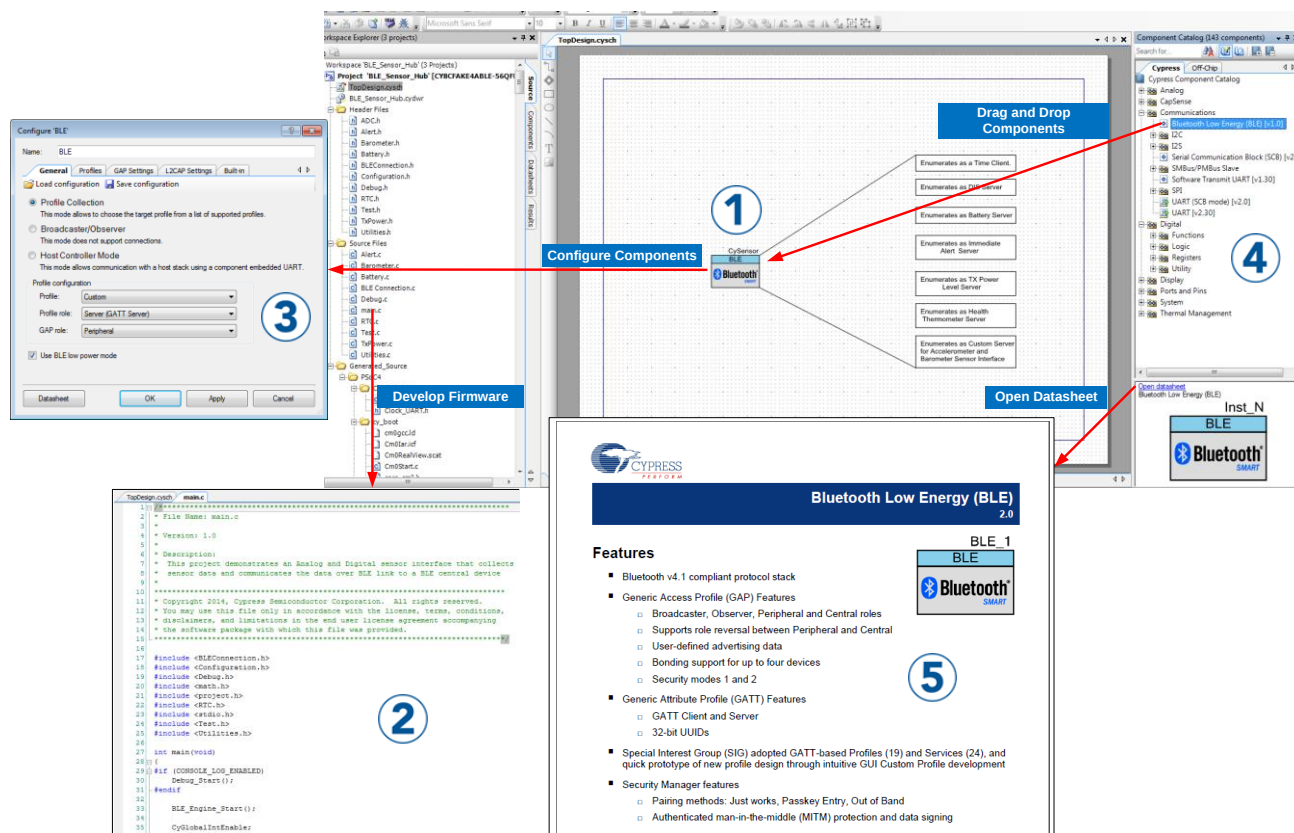
- よくある質問 (FAQ): サイプレスの BLE エコシステムについて詳細に説明します。
- BLE フォーラム: PSoC 4 BLE と PSoC BLE フォーラムをご覧ください。ご質問が既に他の開発者によって解答されているか確認して頂けます。
- サイプレス サポート: まだ困っていますか？サイプレスのサポート ページを訪れられ、テクニカル サポート ケースを作られるか、またはお近くの代理店までお問い合わせください。米国のお客様は、弊社フリーダイヤル (+1-800-541-4736) までお電話いただければ、弊社のテクニカル サポート チームより通話でご対応いたします。プロンプト上のオプション「8」を選択してください。

2.1 PSoC Creator

PSoC Creator は無償の 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 4 BLE、PSoC BLE を基にして、ハードウェアとファームウェアを同時に設計できます。図 1 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。

- コンポーネントをドラッグ アンドドロップして、メイン デザイン ワークスペースでハードウェア システム デザインを構築
- アプリケーションのファームウェアと PSoC ハードウェアを同時に設計
- コンフィギュレーション ツールを用いて、コンポーネントを構成
- 100 以上のコンポーネントを含むライブラリを探索
- コンポーネント データシートを確認

図 1. PSoC Creator の回路図エントリとコンポーネント



2.2 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インストールしてください。次に、PSoC Creator を起動して、以下の項目を開きます。

1. **Quick Start Guide (クイック スタート ガイド):** Help > Documentation > Quick Start Guide を選択します。このガイドは PSoC Creator プロジェクトを開発するための基礎知識を提供します。
2. **Simple Component example projects (簡単なコンポーネント サンプル プロジェクト):** File > Open > Example projects を選択します。これらのサンプル プロジェクトは、PSoC Creator のコンポーネントの設定と使用方法を説明します。
3. **Starter designs (初心者向けの設計):** File > New > Project > PSoC 4 Starter Designs を選択します。これらの初心者向けの設計は、PSoC 4 の独自の機能をデモします。
4. **System Reference Guide (システム リファレンス ガイド):** Help > System Reference > System Reference Guide を選択します。このガイドは、PSoC Creator が提供するシステム機能を記載し説明します。
5. **Component datasheets (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て「データシートを開く」を選択します。
6. **Document Manager (ドキュメント マネージャー):** PSoC Creator が提供するドキュメント マネージャーにより、ドキュメント リソースを容易に検索し、レビューすることができます。Document Manager を開くには、メニューから Help > Document Manager を選択します。

2.3 サンプル コード

PSoC Creator は多数のサンプル コード プロジェクトを提供しています。これらのプロジェクトは、図 2 に示すように、PSoC Creator のスタート ページからアクセスできます。

サンプル プロジェクトは、空のページの代わりに完成した設計で開始して設計時間を短縮させることができます。サンプル プロジェクトは、PSoC Creator コンポーネントを様々なアプリケーションに使用方法も示します。図 3 に示すようにサンプル コードおよびデータシートが含まれています。

図 3 に示す Find Example Project ダイアログにはいくつかのオプションがあります。

- アーキテクチャ、デバイス ファミリ (PSoC 4、PSoC 4 BLE、PSoC 4 BLE など)、カテゴリまたはキーワードに基づいてサンプル プロジェクトを検索します。
- **Filter Options** に基づいて提供されたサンプル プロジェクトのメニューから選択します。図 3 に示すように、20 以上の BLE サンプル プロジェクトが提供されています。選択したプロジェクトのデータシートをレビューします (Documentation タブ)。
- 選択したプロジェクトのサンプル コードをレビューします。このウィンドウからコードをプロジェクトにコピー アンド ペーストして、コード開発時間を短縮させることができます。

さらに、選択に応じて新規プロジェクト (また、必要な場合は新規ワークスペース) を作成することができます。完成した基本的な設計で開始して設計時間を短縮させます。このように、設計をご開発中のアプリケーションに適合させることが可能になります。

図 2. PSoC Creator のサンプル コード

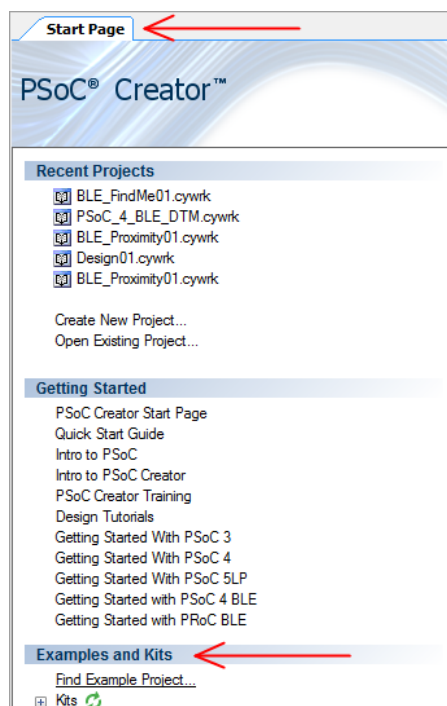
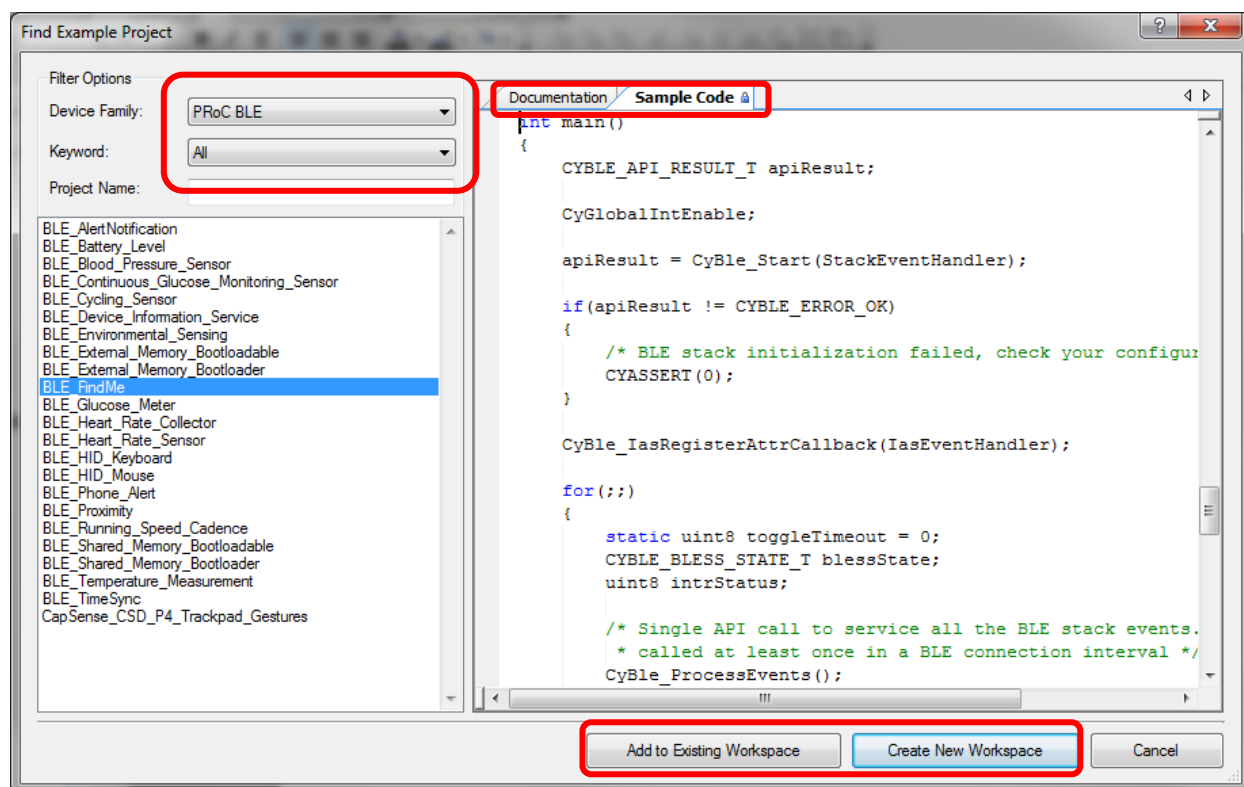


図 3. サンプル プロジェクトおよびサンプル コード



3 PRoC BLE の機能

PRoC BLE は、CPU とメモリ サブシステム、BLE サブシステム、デジタル サブシステム、およびシステム リソースからなる機能豊富なデバイス ファミリです。図 4 と図 5 にはそれぞれ CYBL10X6X と CYBL10X7X デバイス ファミリの可能な機能を示します。

各機能の簡単な説明については、付録 D: PRoC BLE デバイスをご覧ください。また、詳細な情報については、PRoC BLE ファミリー用のデータシート、TRM、およびアプリケーション ノートをご覧ください。

図 4. PRoC BLE アーキテクチャ (CYBL10X6X)

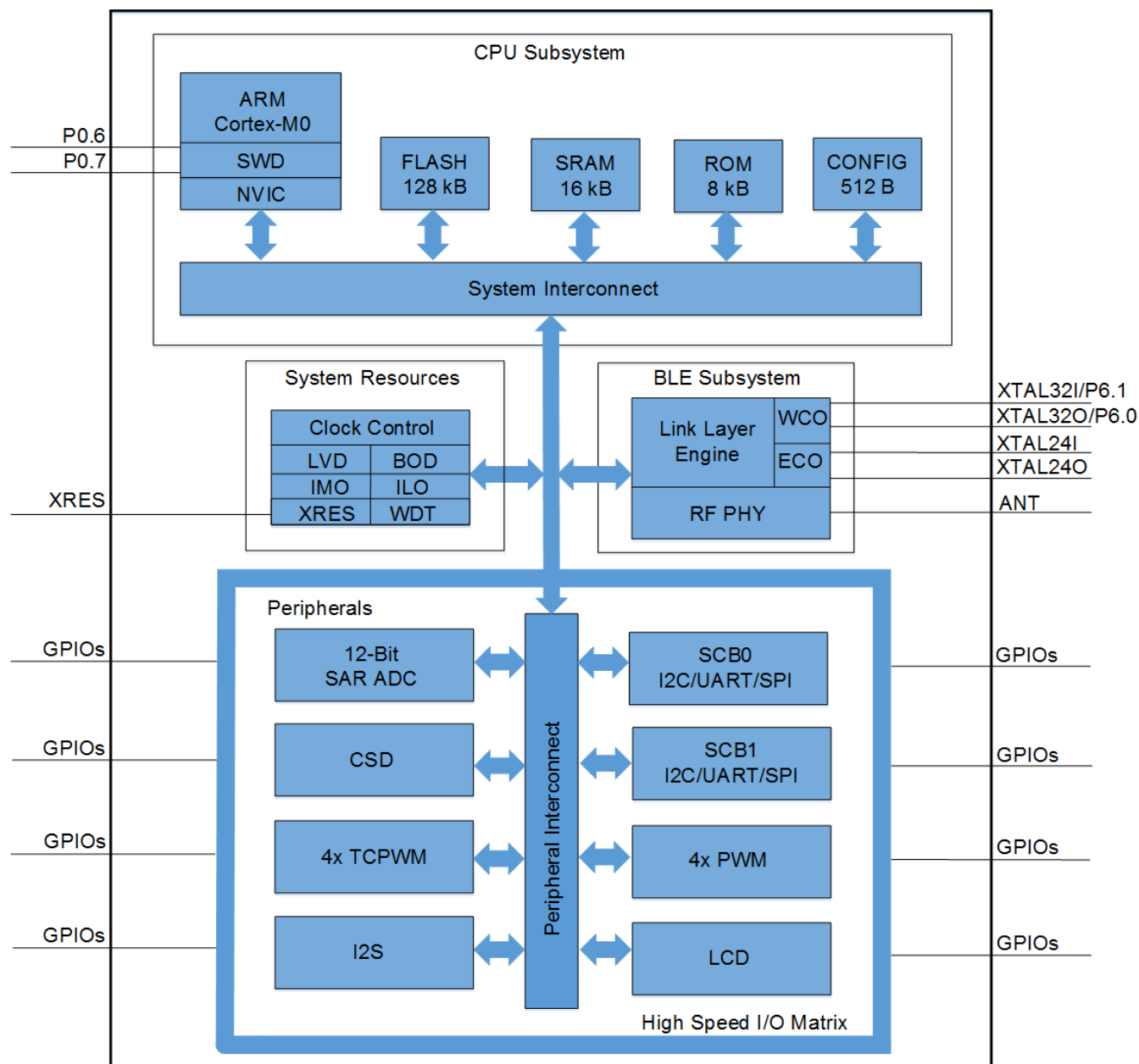
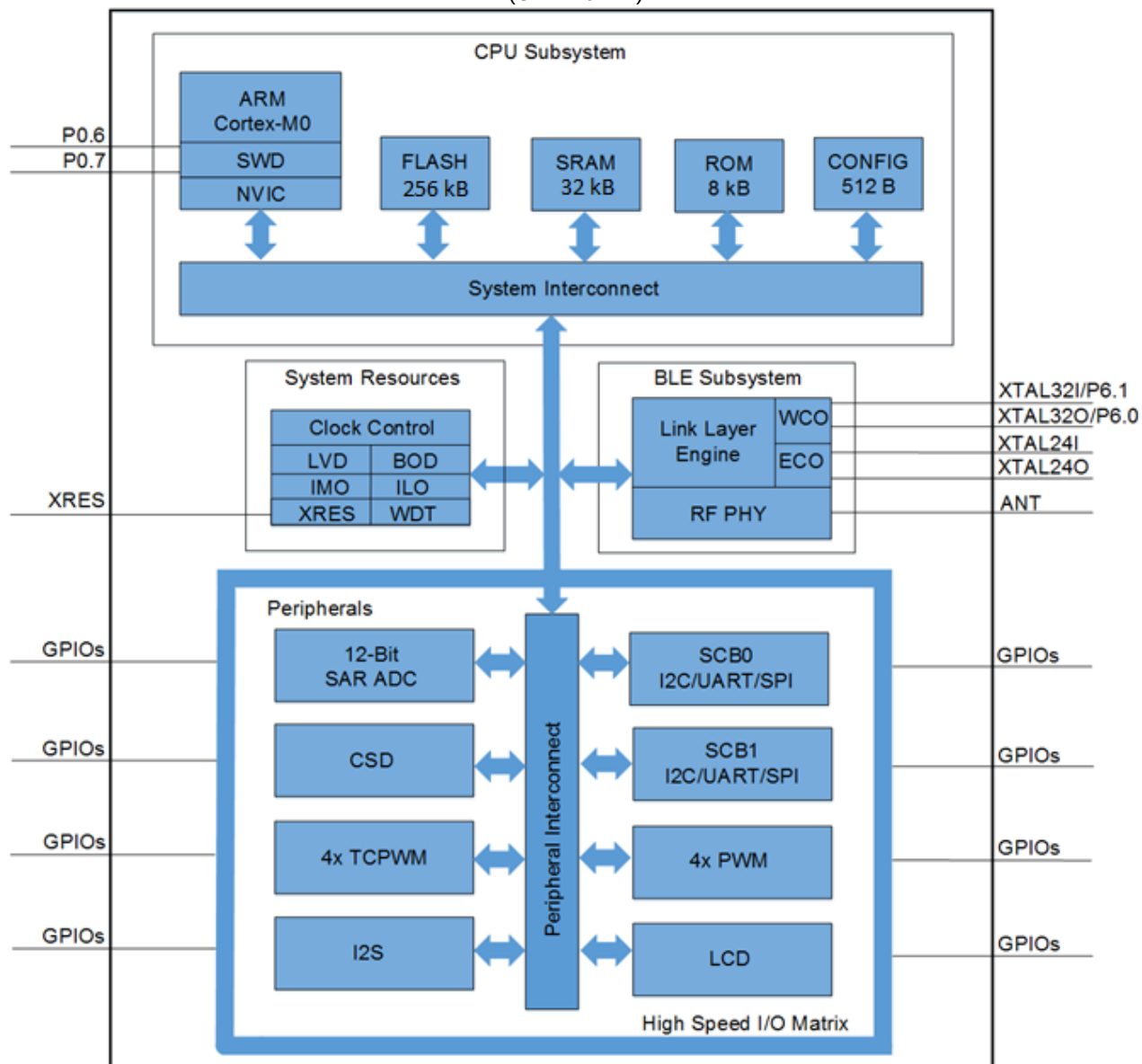


図 5. PSoC BLE アーキテクチャ
 (CYBL10X7X)


4 BLE の概要

BLE または Bluetooth Smart™ は、Bluetooth SIG により策定された、低消費電力、短距離、低データ レートの無線通信プロトコルです。図 6 に示すように、BLE は、低消費電力で小さなデータ チャンクを効率的に転送するために設計された階層式プロトコル スタックを有するもので、バッテリー式デバイス向けの事実上の無線プロトコルとなります。

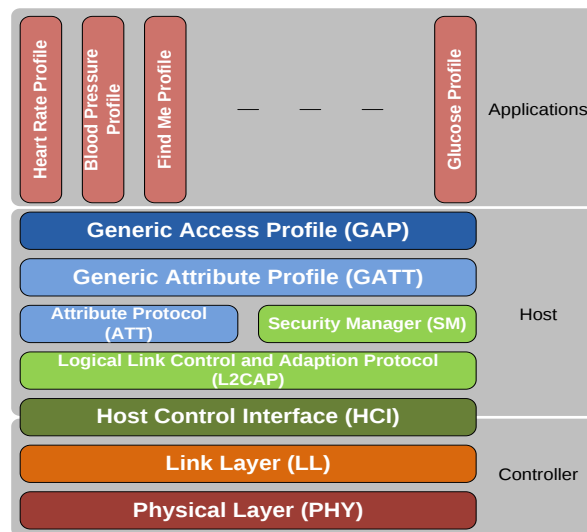
BLE スタックは以下の要素からなります。

- 物理層 (PHY): 2.4GHz RF、データ レートが 1Mbps
- Link Layer (LL): PHY のタイミングおよびパケット形式を定義する

- Host Control Interface (HCI): スタックのハードウェア コントローラー (PHY + LL) の層とファームウェア ホスト層をリンクする
- Logical Link Control および Adaptation Protocol (L2CAP): パケット アセンブリ/ディサセンブリおよびプロトコル マルチプレクサ層として動作する
- Attribute Protocol (ATT): アプリケーション データの構成とアクセス方法を定義する
- Security Manager (SM): BLE 接続を介した安全なデータ転送を行うためのツールボックス
- Generic Attribute Profile (GATT): ATT 層により定義されたデータにアクセスする方式を定義する
- Generic Access Profile (GAP): デバイスが BLE リンク マスターかスレーブかを定義し、それに応じた下層を構成するアプリケーション指向インターフェース

BLE プロトコルの詳細説明については、付録 E: BLE プロトコルをご参照ください。

図 6. BLE アーキテクチャ



BLE を使用したアプリケーションを開発するには、この複雑なプロトコル スタックの実用的な知識は必要ありません。サイプレスは、プロトコルの複雑さを簡素化し、構成しやすい GUI ベースの BLE コンポーネントを提供します。BLE を使い始めるにあたって、以下を理解すれば充分です。

- BLE 接続を確立する手順
- アプリケーション データ表現と抽象
- サイプレスの BLE コンポーネント GUI を使用した GAP と GATT 層構成のアプリケーションの要件マッピング

4.1 BLE 接続の確立

BLE Pioneer Kit とスマートフォンのような 2 つのデバイス間の BLE 接続を確立するにあたって、以下の 2 つの GAP デバイスの役割を理解する必要があります。

- **GAP ペリフェラル:** 自分の存在を広報し、GAP センtral デバイスからの接続を許可するデバイスです。心拍数測定機能を実装する BLE Pioneer Kit が GAP ペリフェラル デバイスの一例です。
- **GAP センtral:** GAP ペリフェラルからの広報をスキャンし、それらのペリフェラルとの接続を確立します。心拍数測定デバイスに接続するスマートフォンが GAP センtral デバイスの一例です。

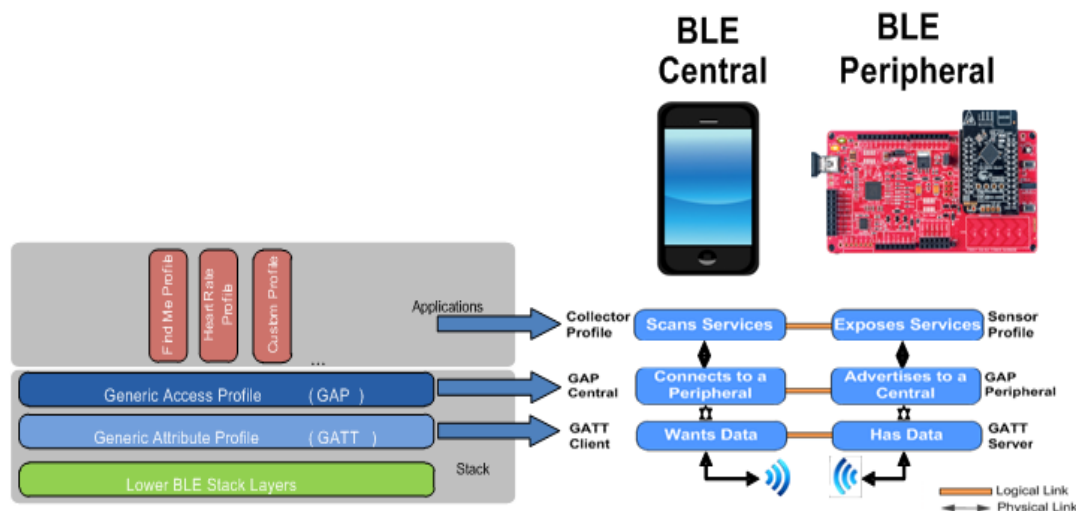
セントラル デバイスがペリフェラルとの接続を確立した後、両方のデバイスは BLE 接続を介して接続されていると言われます。接続された BLE 通信では、GATT は、GAP のロールとは関係なく、データの転送元と転送先に基づいて以下の 2 つのロールを定義します。

- **GATT サーバー:** データまたは状態情報を格納するデバイスです。GATT クライアントにより構成されると、GATT クライアントにデータを送信するか、または自分のローカル状態を変更します。例えば、心拍数測定デバイスは、スマートフォン GATT クライアントに心拍数のデータを送信する GATT サーバーです。
- **GATT クライアント:** GATT サーバーの状態を構成するか、または GATT サーバーからデータを受信します。例えば、心拍数測定デバイスから心拍数情報を受信するスマートフォンは GATT クライアントです。

BLE 接続が確立された後、GATT クライアントは GATT サーバーに存在するすべてのデータを検出します。データを検出すると、GATT クライアントは GATT サーバーを構成する、および／または GATT サーバーに対してデータを読み書きすることができます。

図 7 に、GAP ペリフェラル／ GATT サーバーロールとして構成された BLE Pioneer Kit と、GAP セントラル／GATT クライアントデバイスとして構成されたスマートフォンとの接続の例を示します。各デバイスでの GAP と GATT ロールの構成は、アプリケーションの使用状況と [BLE Profile](#) によって定義されます。例えば、心拍数を測定するデバイスは心拍数データを公開する GATT サーバーのロールを果たし、GAP ペリフェラル デバイスとして構成されます。一方、心拍数測定デバイスから心拍数の情報を読み取るスマートフォンは GATT クライアントのロールを果たし、GAP セントラル デバイスとして構成されます。

図 7. BLE アプリケーションの概要



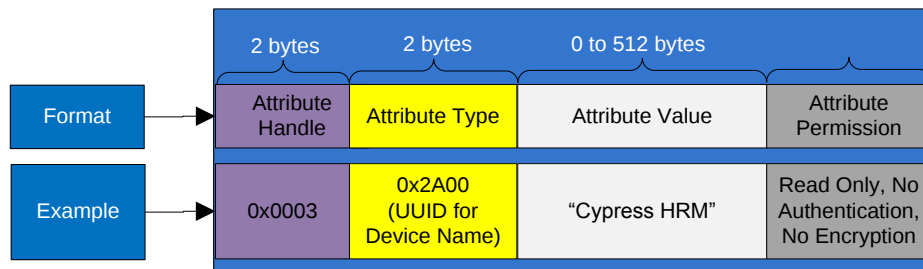
4.2 GATT データ形式

GATT サーバーは、アトリビュート、キャラクタースティック、およびサービスで BLE デバイスのデータを表示および抽象化します。

- **アトリビュート:** ATT/GATT 層の基本的なデータ コンテナであり、一件の個別情報を表します。図 8 に示すように、アトリビュートは以下の要素から構成されています。
 - **アトリビュート ハンドル:** アトリビュートをアドレス指定するために示します。
 - **アトリビュート タイプ:** Bluetooth SIG が割り当てる 16 ビットの UUID (Universally Unique Identifier) であり、アトリビュートに格納されているものを定義します。
 - **アトリビュート バリュー:** 実データを格納します。
 - **アトリビュート パーミッション:** アトリビュートの読み出し／書き込み、およびセキュリティ要件を定義します。
- 心拍数測定、バッテリー残量、バッテリー残量単位、およびデバイス名がアトリビュートの例です。

GATT サーバーには、「アトリビュート データベース」と呼ばれるファームウェア データベースに格納される数多くのアトリビュートが含まれます。GATT クライアントは、アトリビュート ハンドルを使用して GATT サーバーのアトリビュート データベースから 1 つまたは複数のアトリビュートを読み書きします。

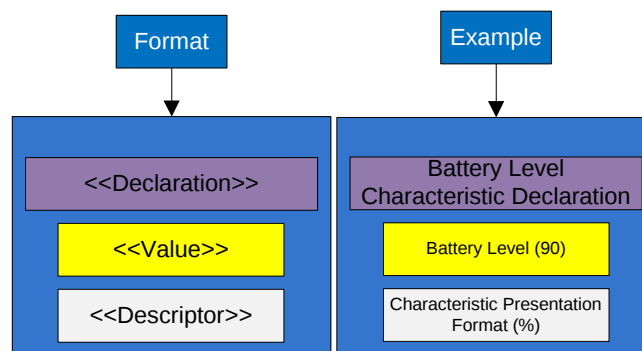
図 8. GATT アトリビュートの例



- **キャラクターリスティック:** 複数の個別アトリビュートで構成されています。これらのアトリビュートを組み合わせるとシステム情報や意味のあるデータを形成します。図 9 に示すように、キャラクターリスティックは、1 個のキャラクターリスティック宣言アトリビュート、1 個のキャラクターリスティック値のアトリビュート、1 つまたは複数 (任意) のキャラクターリスティック ディスクリプタのアトリビュートで構成されています。「90」のバッテリー残量のアトリビュートと、「%」のバッテリー残量ディスクリプタのアトリビュートを組み合わせると、90%のシステムのバッテリー残量情報を作ります。

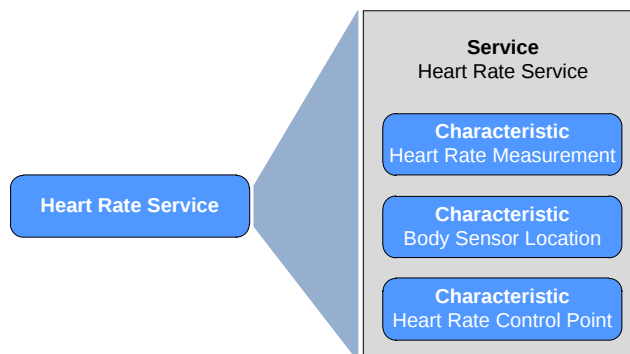
Bluetooth SIG は、ご開発のアプリケーションに有用な定義済みの**標準キャラクターリスティック**一式を提供します。あるいは独自のカスタム キャラクターリスティックを定義することもできます。

図 9. GATT キャラクターリスティックの例



- **サービス:** デバイスの特定の機能または特長を定義する 1 つまたは複数の関連キャラクターリスティックより構成されます。図 10 に、心拍数測定に関連する情報を説明した 3 つのキャラクターリスティックを含む心拍数サービスの例を示します。Bluetooth SIG は、BLE デバイスの一般的な機能を実装するための定義済み**標準サービス**一式を提供しています。あるいは標準またはカスタム キャラクターリスティックからなる独自のカスタム サービスを定義することもできます。

図 10. GATT サービスの例



4.3 BLE プロファイル

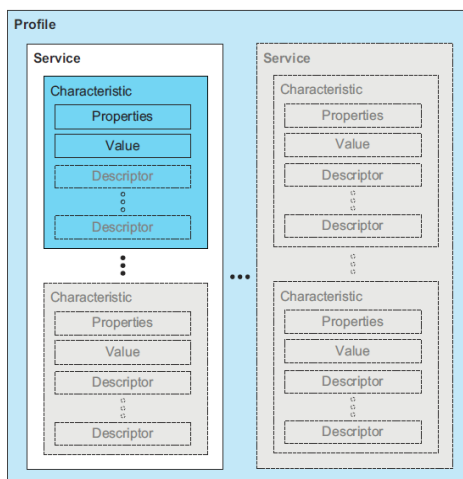
BLE プロファイルは、プロファイル準拠デバイス間のアプリケーション レベルでの相互運用性を確保するための仕様です。特定の最終アプリケーションやユースケースを作成するために、異なる BLE 層のロールと構成、および対応する GATT サービスを定義します。例えば、心拍数監視機器を実装したい場合、BLE 心拍数プロファイルは、相互運用可能な心拍数測定装置を作成するために必要な GAP と GATT のロール、および対応する GATT サービスを定義します。Bluetooth SIG は、一般的な BLE 最終アプリケーション用の定義済み標準プロファイル一式を提供しています。あるいは標準またはカスタム サービスからなる独自のカスタム プロファイルを作成することもできます。

図 7 に示すように、プロファイルは GATT 層と同様に、2 つのアプリケーション ロールを定義します。

- センサーまたはサーバー: データを有するアプリケーション
- コレクターまたはクライアント: データを求めるアプリケーション

図 11 に、BLE デバイスにおけるデータ抽象化とデータ階層の概要を示します。

図 11. BLE データ階層*



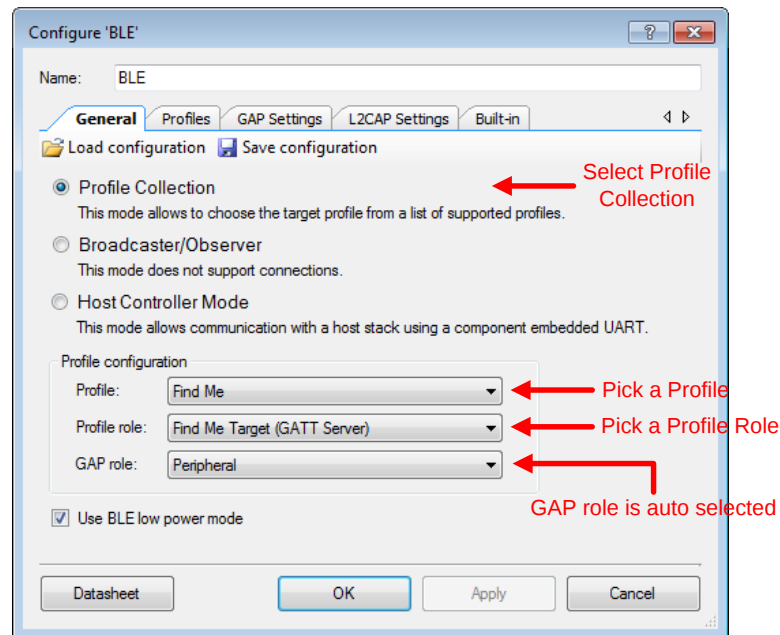
*Bluetooth SIG により提供された無料イメージ

4.4 BLE コンポーネント

PSoC Creator の BLE コンポーネントは、BLE プロトコルの複雑さをシンプルで使いやすい GUI と数個の API で簡素化します。BLE コンポーネントを標準 BLE ペリフェラルになるように構成する手順は次の 5 つのステップです。

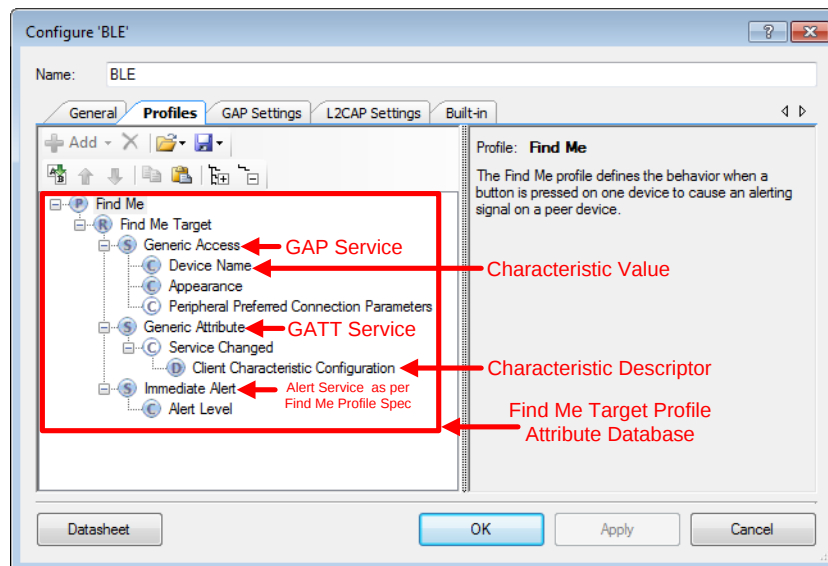
ステップ 1: ご使用のデザインに適切なプロファイルとプロファイル ロールを選択します。図 12 に示すように、BLE コンポーネントは、選択したプロファイル ロールに必要な GAP と GATT ロールを自動的に選択します。

図 12. BLE プロファイルのコンフィギュレーション



ステップ 2: 選択したプロファイル ロールに応じて、サポートされるすべての GATT サービスと該当するキャラクターリスティックがプロファイル仕様に従って自動生成されます。サービスとキャラクターリスティック値を確認し、必要に応じて編集します。**Profiles** タブ内の設定情報は、選択したデザインのアトリビュート データベースになります。

図 13. BLE アトリビュート データベースのコンフィギュレーション



ステップ 3: 図 14 と図 15 に示すように、ご使用のデザインの GAP の一般設定と広報データのコンフィギュレーションを構成します。

図 14. BLE GAP のコンフィギュレーション

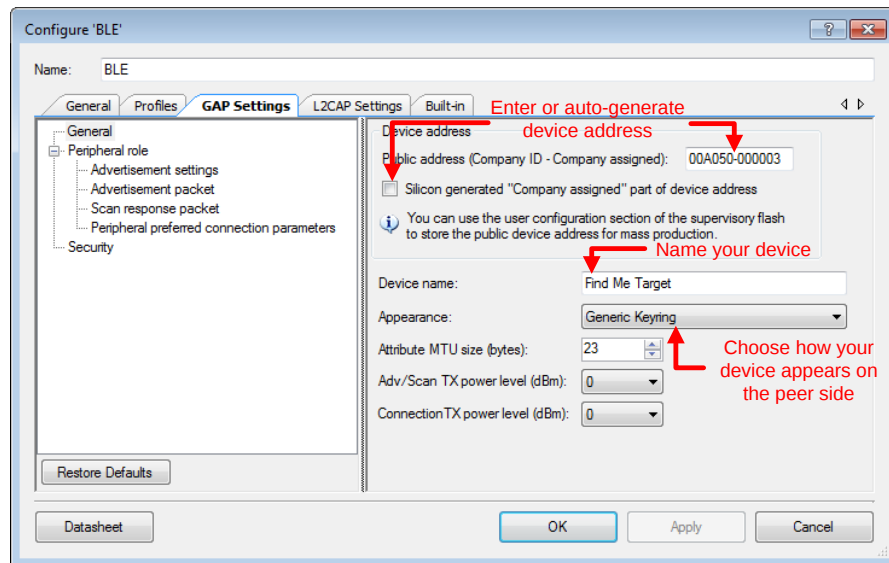
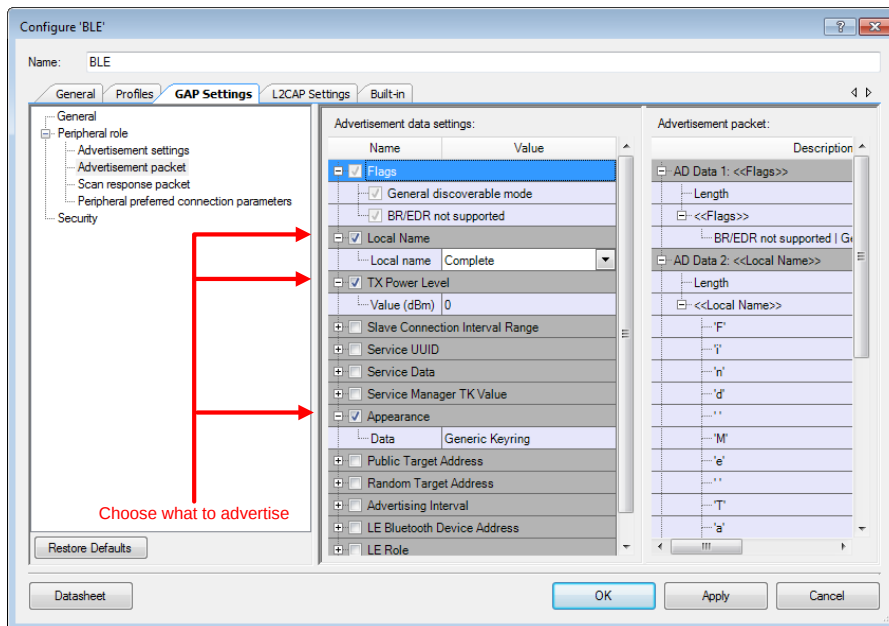


図 15. BLE 広報データのコンフィギュレーション



ステップ 4: ファームウェアを書いて、構成したデザインを初期化します。BLE コンポーネントに対応するアプリケーション層のイベント ハンドラを登録して、BLE コンポーネントからのイベントとデータを受信させます。イベント ハンドラは、GAP、GATT やその他の一般的なイベントに対応する一般イベント ハンドラ、およびサービス固有イベントに対応するサービス固有イベント ハンドラ (対応するサービスごとに 1 つのイベント ハンドラ) の 2 種類です。

ステップ 5: プログラムのメイン ループで待機して、BLE コンポーネントからイベントが発生したら、必要なアクションを取るか、または BLE コンポーネント API を使用してセントラル デバイスにデータを送信します。

5 PSoC BLE 開発セットアップ

図 16 に、PSoC BLE デバイスを用いて BLE 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。BLE Pioneer Kit (図 16 に示す赤色の基板上の黒いモジュール) は、セントラル デバイスとして機能する CySmart iOS/Android アプリまたは CySmart ホスト エミュレーション ツールと通信できるペリフェラルとして機能します。また、CySmart ホスト エミュレーション ツールの動作には BLE ドングル (図 16 に示す黒色の基板) が必要です。図 17 に、BLE 設計のプログラミングとデバッグに必要な開発システムのセットアップを示します。

図 16. BLE 機能的セットアップ

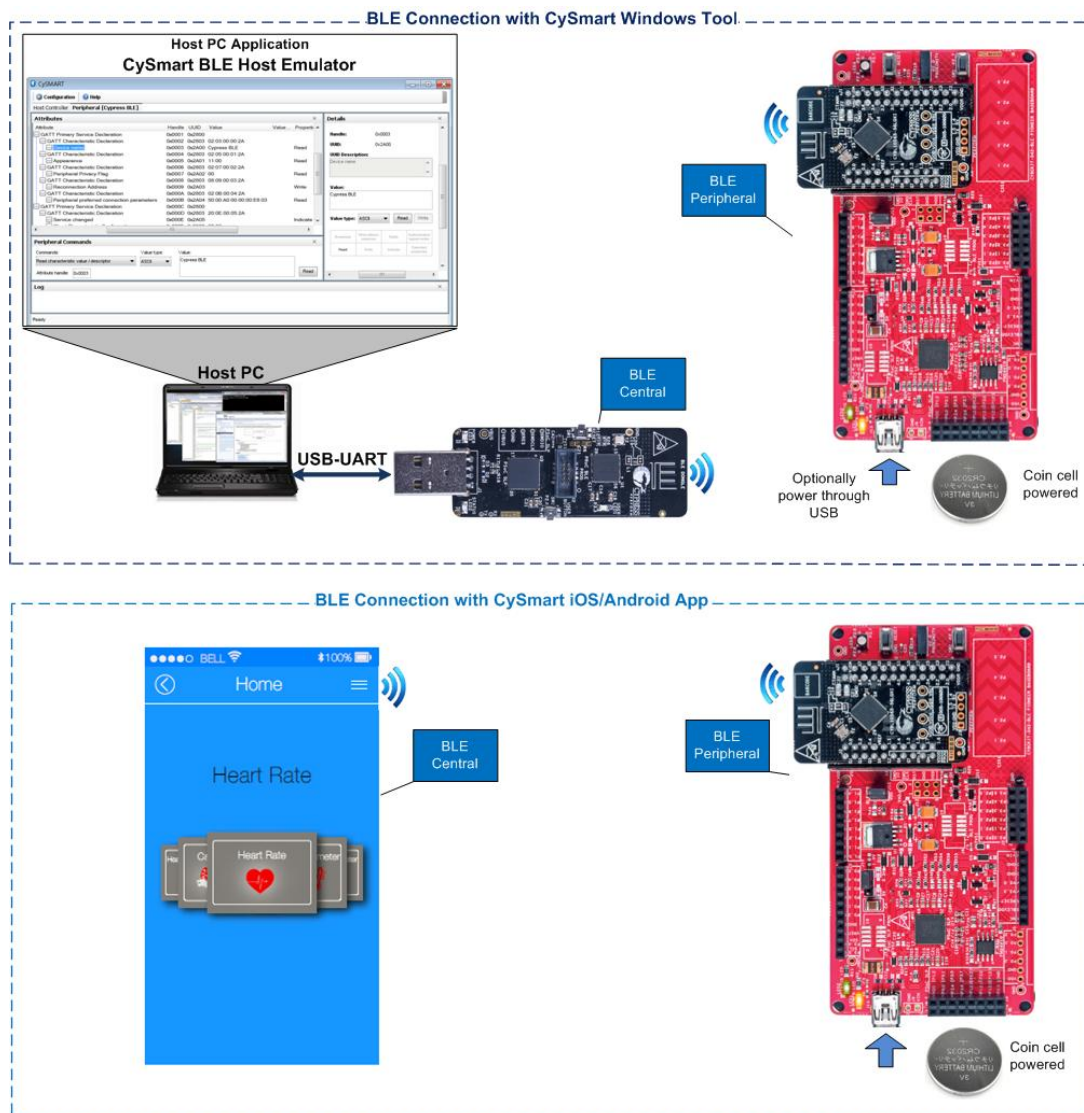
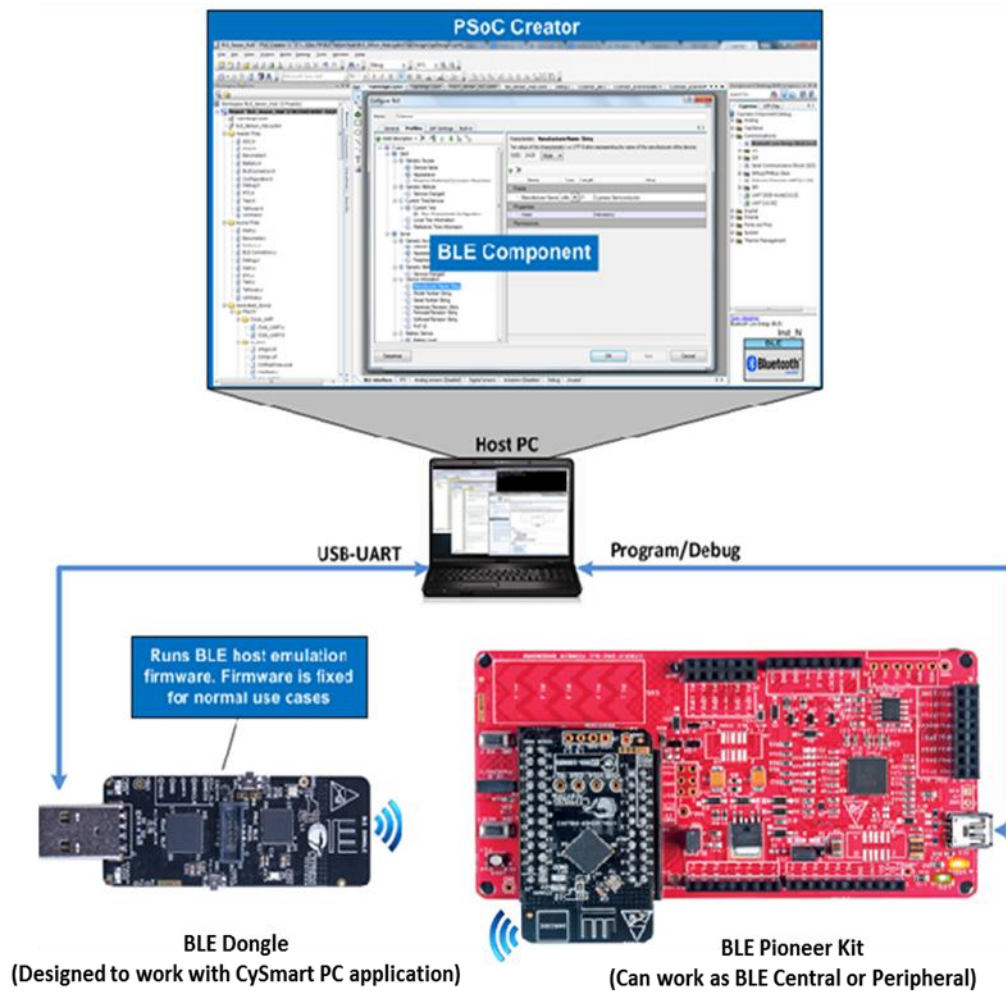


図 17. BLE 開発セットアップ



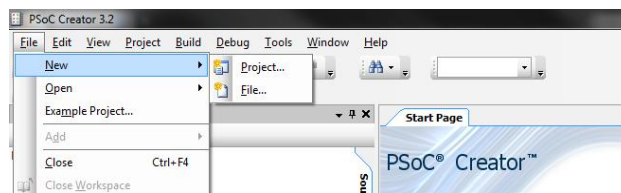
注: 本アプリケーション ノートで説明する BLE サンプル設計の PSoC Creator 機能プロジェクトはアプリケーション ノートのウェブページに掲載されませんが、PSoC Creator のサンプル プロジェクトに含まれています。サンプル プロジェクト (**File > Example Project**) を選択して、**Keyword** を **Find Me > BLE_FindMe** として選択すること で開始することが可能です。サンプル プロジェクトで開始する場合、上記のステップ 1 と 2 (以下の節 6.3 と 6.4) を飛ばしてステップ 3 と 4 (以下の節 6.5 と 6.6) から進んでください。

6.3 第 1 段階: 設計を作成／設定

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

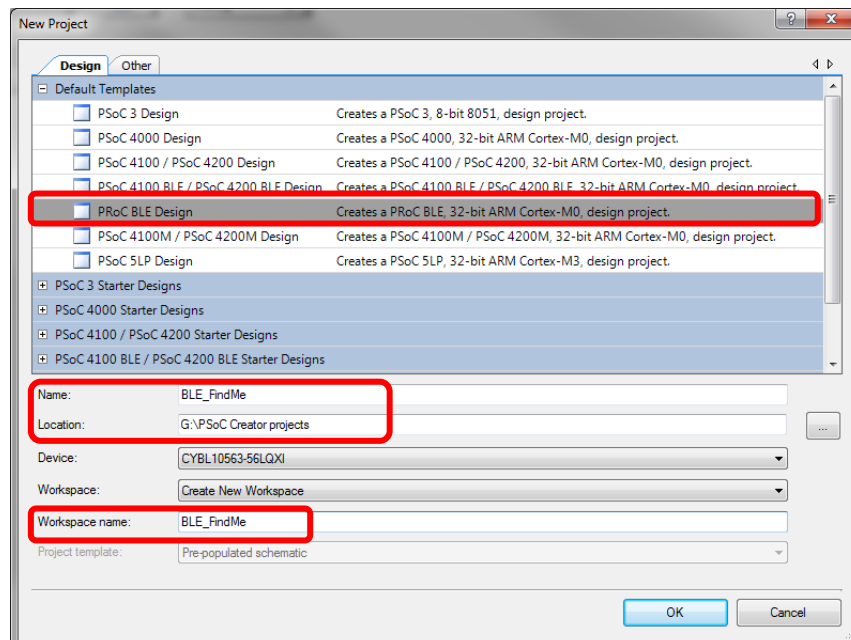
1. **PSoC Creator ホームページ** から PSoC Creator 3.2 またはそれ以降のバージョンをご使用の PC にインストールしてください。
2. PSoC Creator を起動し、[図 19](#) に示すように **File > New > Project** を選択します。

図 19. プロジェクトの新規作成



3. [図 20](#) に示すように、PSoC BLE Design のプロジェクト テンプレートを選択して、プロジェクトに名前を付けます (例: 「BLE_FindMe」)。新規プロジェクトの適切な場所を選択します。

図 20. 新規プロジェクトの命名



4. **Project > Device Selector** を移動して、デバイスを選択します。開発キットを使用する場合、キットの型番を読み、またはキットのガイドを参照して型番を調べてください。また、[図 22](#) に示すように、デバイス セレクタの **Devices** タブのエリアで右クリックして **Select Default Device** を選択して **PRoC BLE** ファミリーをクリックすることで、**BLE Pioneer Kit** に使用する CYBL10563-56LQXI デバイスを選択できます。

注: CY5671 モジュールには CYBL10X6X デバイス ファミリーを選び、CY5676 モジュールには CYBL10X7X デバイス ファミリーを選びます。

図 21. デバイス セクタの起動

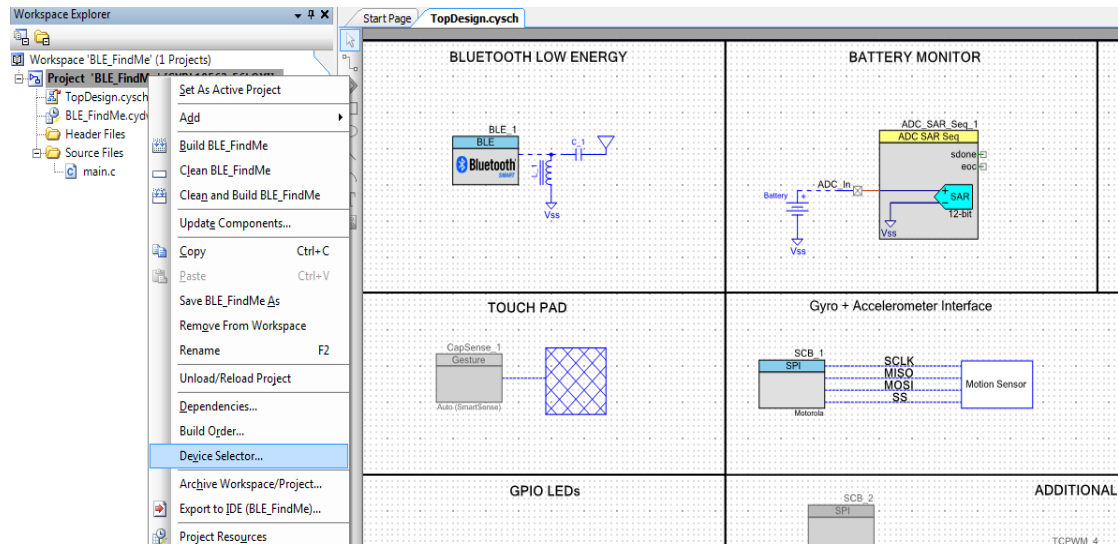
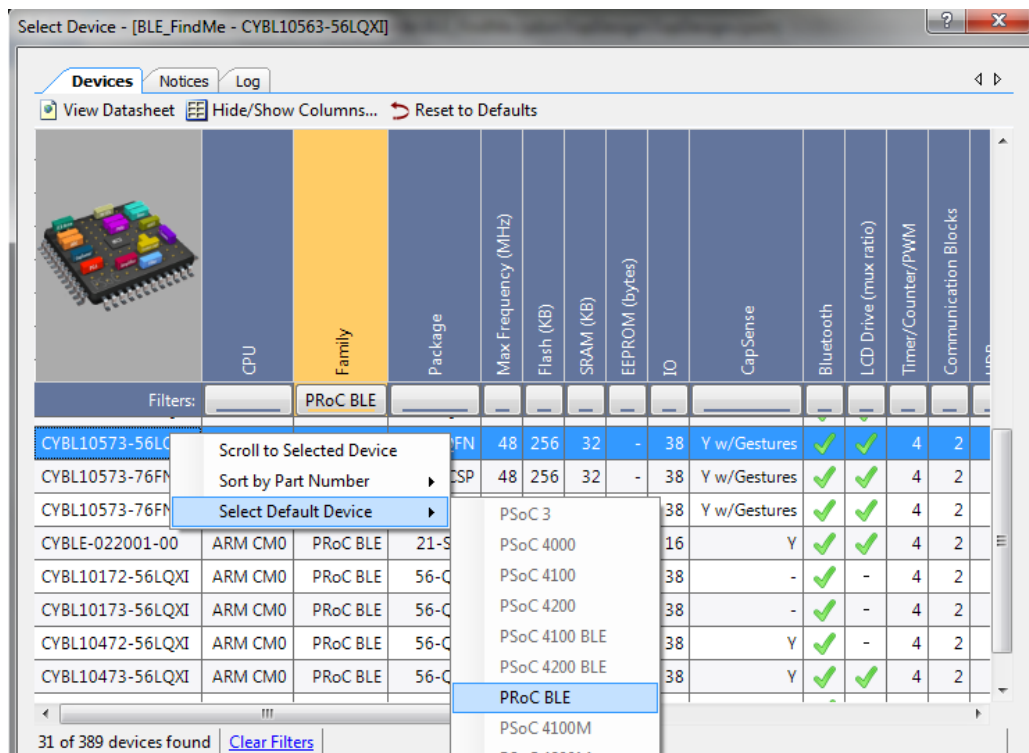


図 22. デバイス セクタ



- 新規プロジェクトを作成すると、ベースライン ファイル一式を格納するプロジェクト フォルダが作成されます。図 23 に示すように、これらのファイルが **Workspace Explorer** ウィンドウに表示されます。*TopDesign.cysch* ファイルをダブルクリックして、プロジェクトの回路図を開きます。

図 23. Top Design 回路図を開く

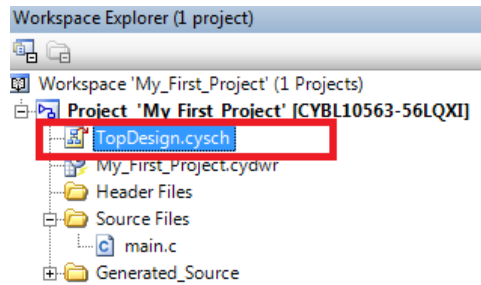
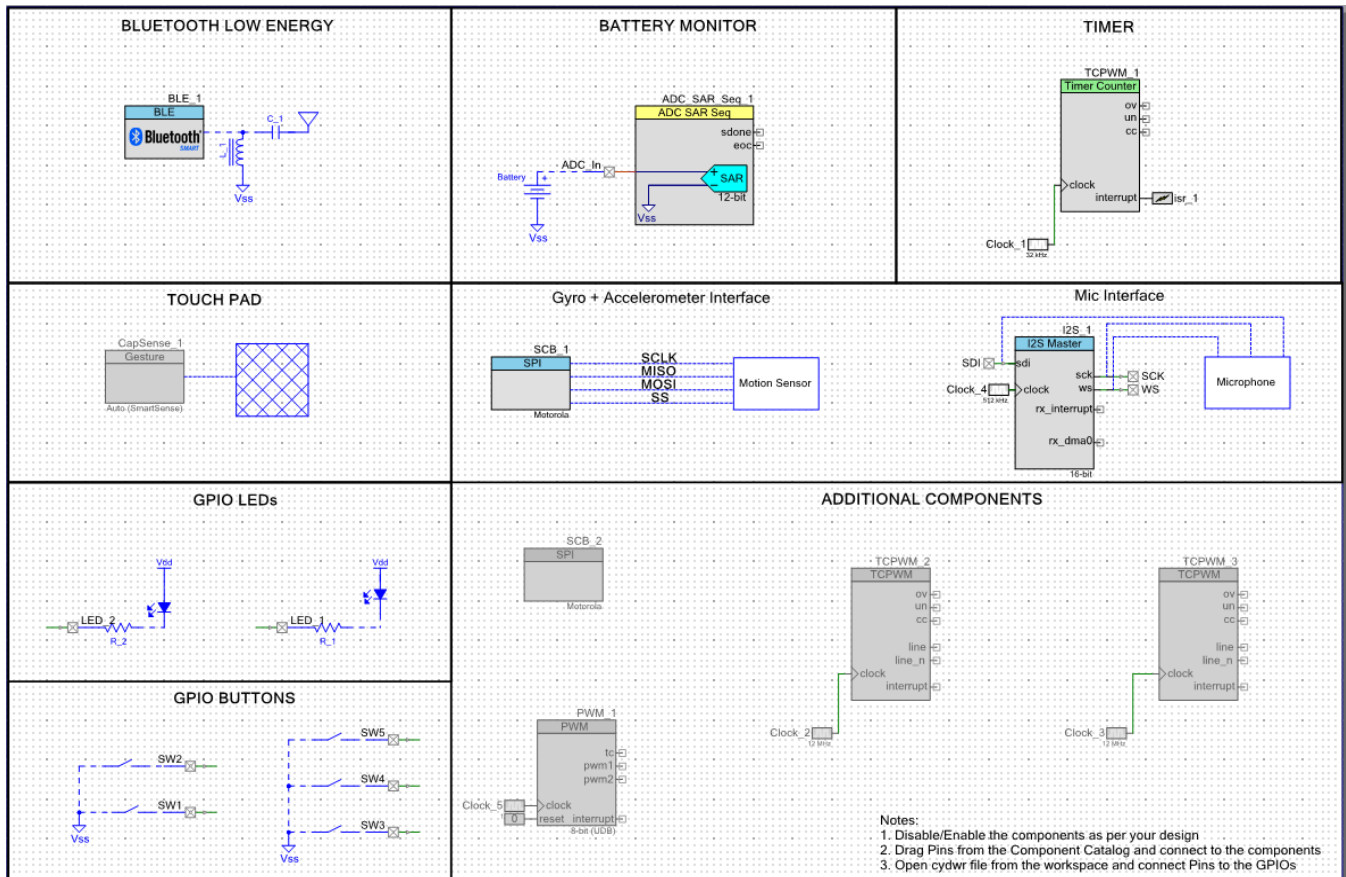


図 24 に示すように、「TopDesign.cysch」には予め用意された回路図が表示されます。

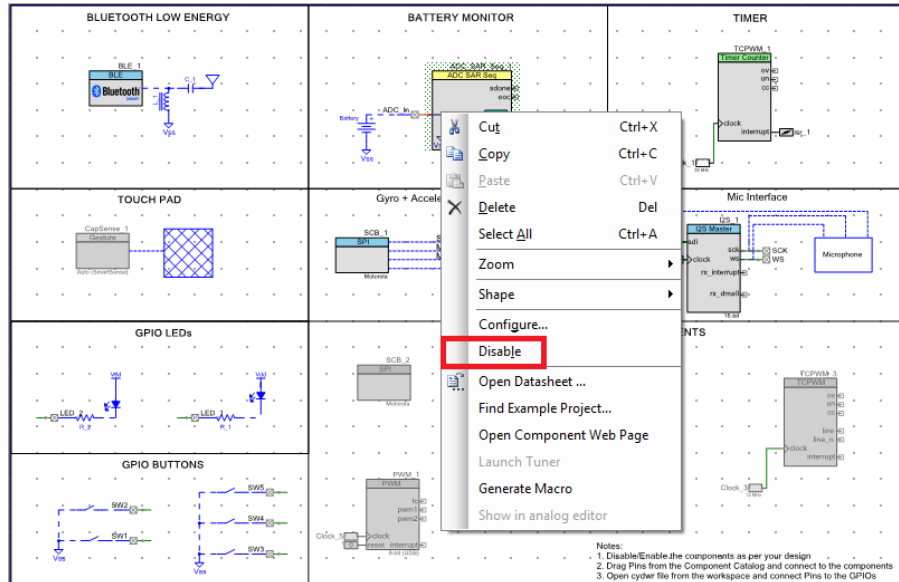
図 24. 事前追加の回路図



6. 各コンポーネントを右クリックして対応するオプションを選択することで該当のコンポーネントを無効にする (または削除する) ことができます (図 25 をご参照ください)。本設計では次のコンポーネントを無効にします。

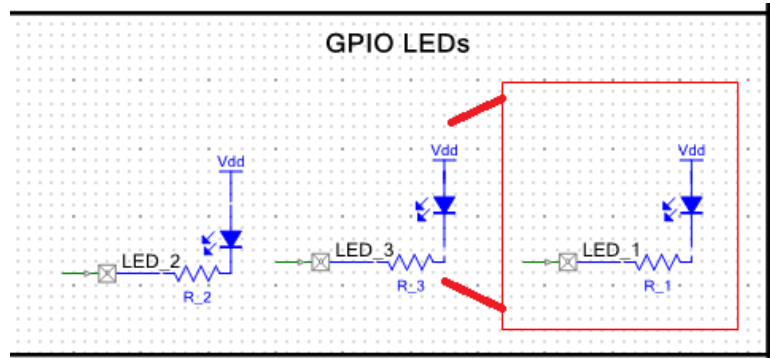
- 「BATTERY MONITOR」セクションからの ADC_SAR_Seq_1 および ADC_In
- すべての GPIO ボタン (SW1~SW5)
- 「TIMER」セクションからの TCPWM_1、Clock_1、および isr_1
- 「Gyro + Accelerometer Interface」セクションからの SCB_1
- 「Mic Interface」セクションからの I2S_1、Clock_4、SCK、SDI、および WS

図 25. コンポーネントを有効／無効にする



7. LED_1 と LED_2 の間で 1 個の LED を追加します。追加するには、まずは、新規 LED に合わせるように LED_1 と LED_2 の回路図の間の間隔を増やします。図 26 に示すように、LED_1 回路図を選択して右クリックして **Copy** を選び、また右クリックして **Paste** を選んで LED_3 を作成します。

図 26. LED をもう 1 つ追加



8. 回路図内の BLE コンポーネントをダブルクリックし、以下の特性を持つ「BLE Find Me Target」に設定します。
 - 図 27 に示すように、GAP の役割 (GAP Role) を「Peripheral」に、プロファイルの役割 (Profile role) を「Find Me Target (GATT server)」に設定します。Find Me プロファイルのサービスとキャラクタースティックは図 28 に示すように (初期設定の値のままに) します。
 - 図 29 に示すように、GAP のデバイス名 (Device name) を「Find Me Target」に、その表示 (Appearance) を「Generic Keying」に設定します。「Silicon generated “Company assigned” part of device address」にチェックを入れます。
 - 図 30 に示すように、広報モードを「Limited」に、広報タイムアウトを「30」秒に、高速広報期間を 20~30ms に設定します。「Slow advertising interval」チェックボックスにチェックを外します。

- 図 31 と図 32 に示すように、Advertisement Packet で「Flags」を選択し、「Service UUID」フィールドの下 Immediate Alert を選んでそのフィールドを有効にし、Scan Response Packet で「Local Name」と「Appearance」フィールドを有効にします。
- 図 33 に示すように、GAP のセキュリティレベルを最低レベル (Mode 1、No Security) に設定すると、データ交換時に認証、暗号化、認可、または接続が不要になります。

図 27. BLE コンポーネントの一般設定

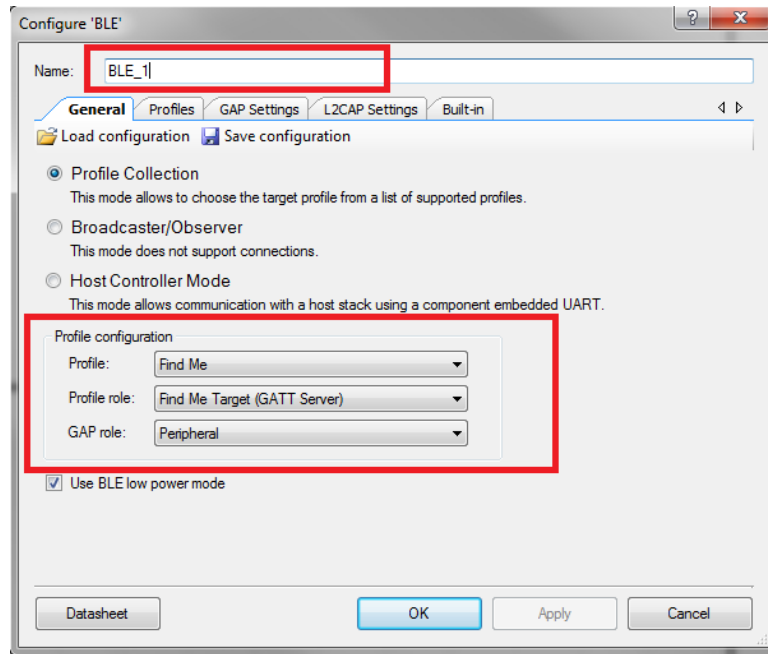


図 28. BLE コンポーネントのプロファイル設定

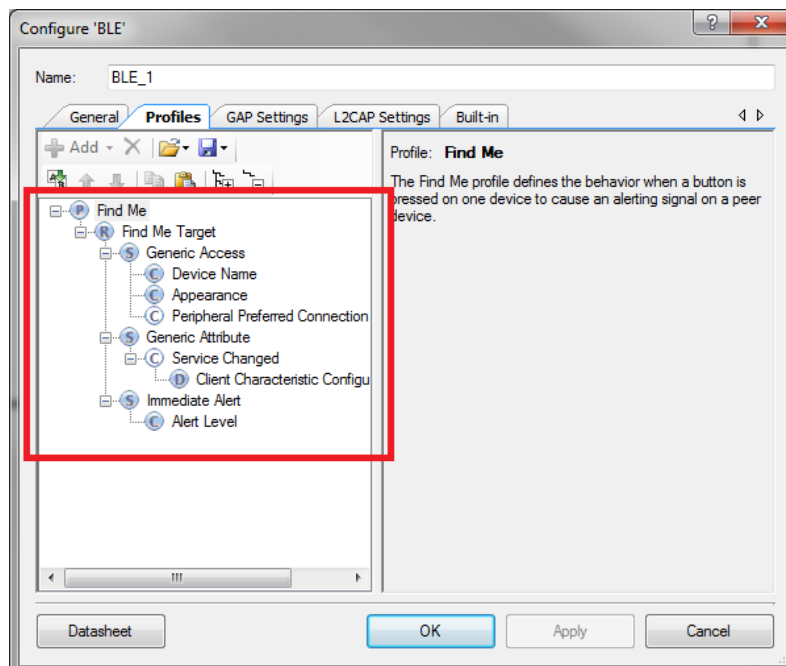


図 29. BLE コンポーネントの GAP 一般設定

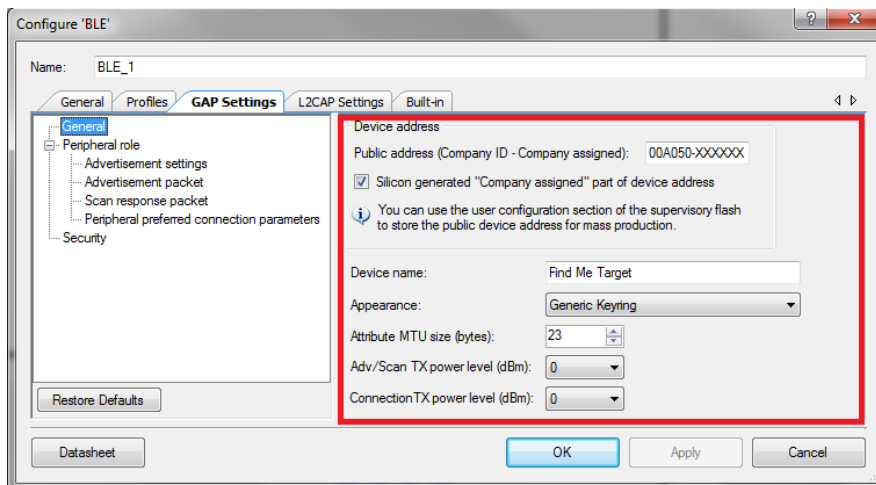


図 30. BLE コンポーネントの GAP 広報設定

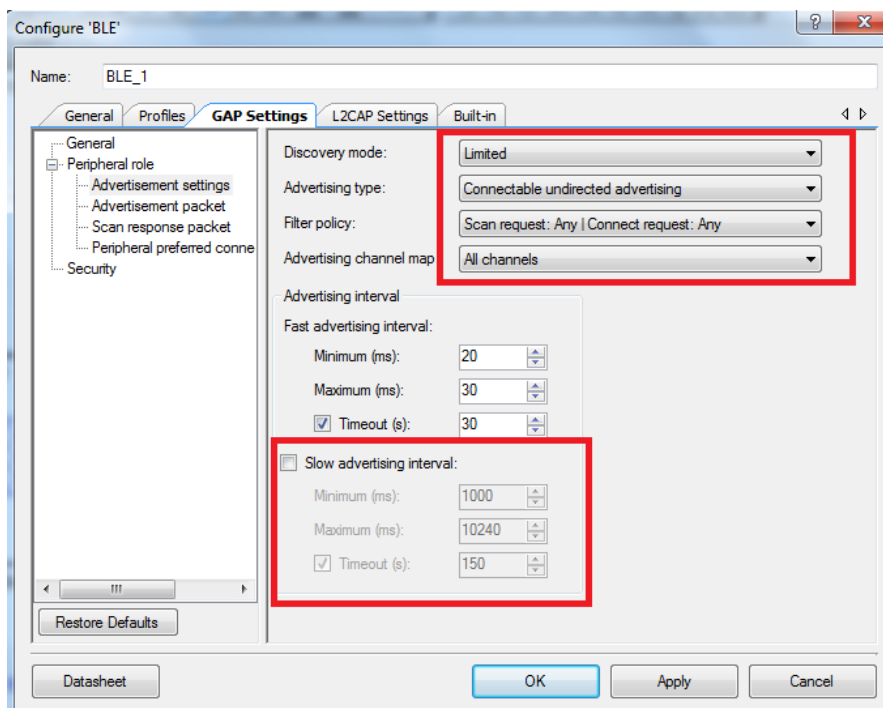


図 31. BLE コンポーネントの GAP 広報パッケージ

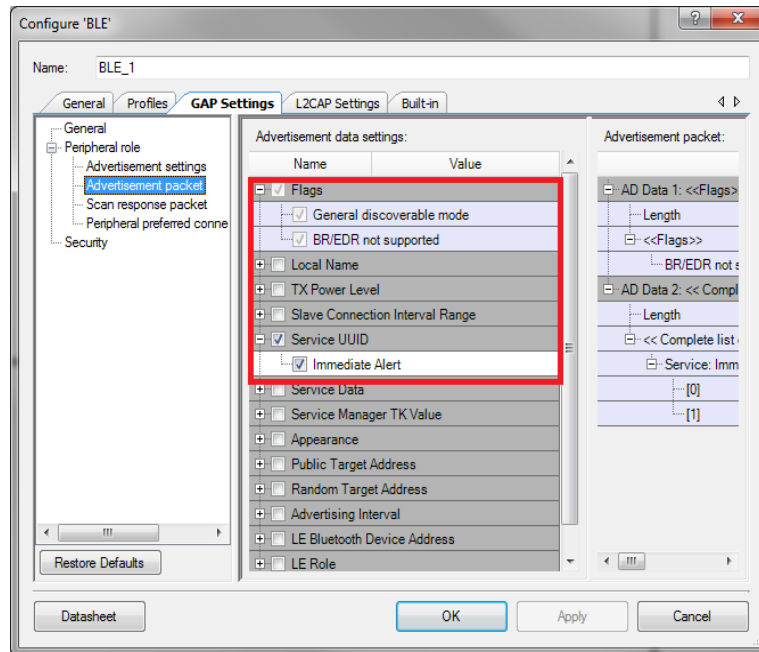


図 32. BLE コンポーネントの GAP スキャン応答パッケージ

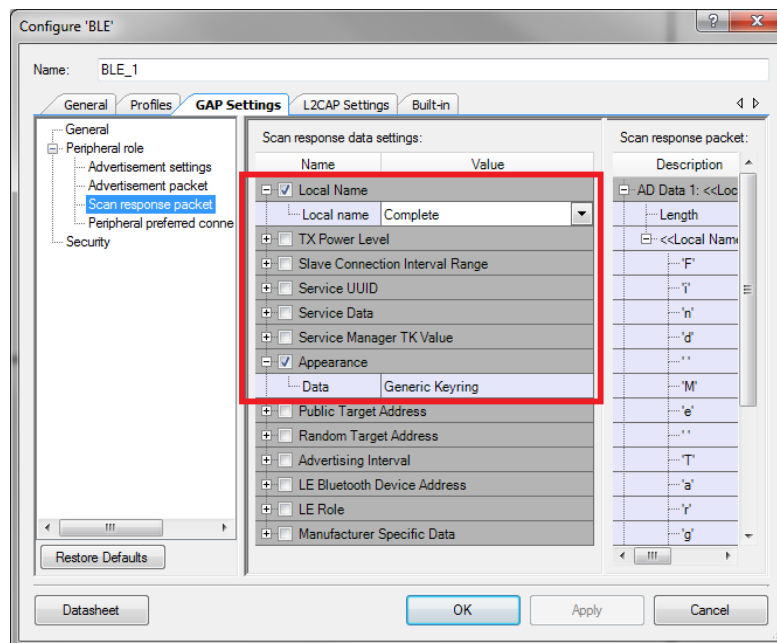
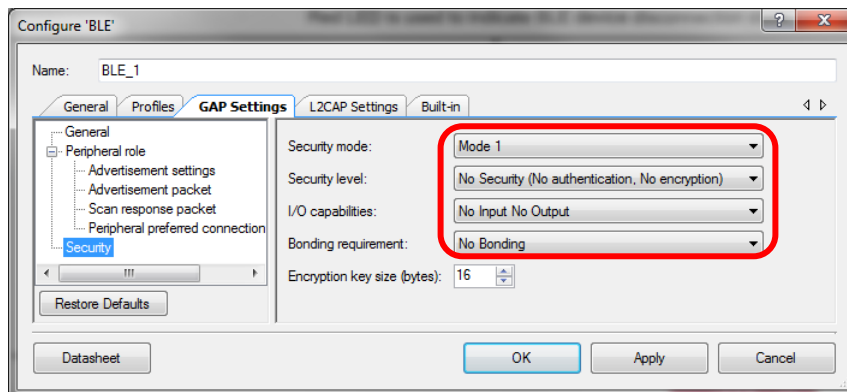


図 33. BLE コンポーネントの GAP セキュリティ設定



9. 図 34 と図 35 に示すように、「External Terminal」が選択されている状態で LED_1 と LED_2 の名前をそれぞれ Advertising_LED と Disconnect_LED に変更します。BLE Pioneer Kit 上の LED はアクティブ LOW のため、「Initial drive state」を「High(1)」にします。すなわち、駆動ピンが HIGH になると、LED は消灯し、駆動ピンが LOW になると、LED は点灯します。これらの LED は BLE 広報と接続切断状態を指示するために使用されます。

図 34. LED_1 コンフィギュレーション

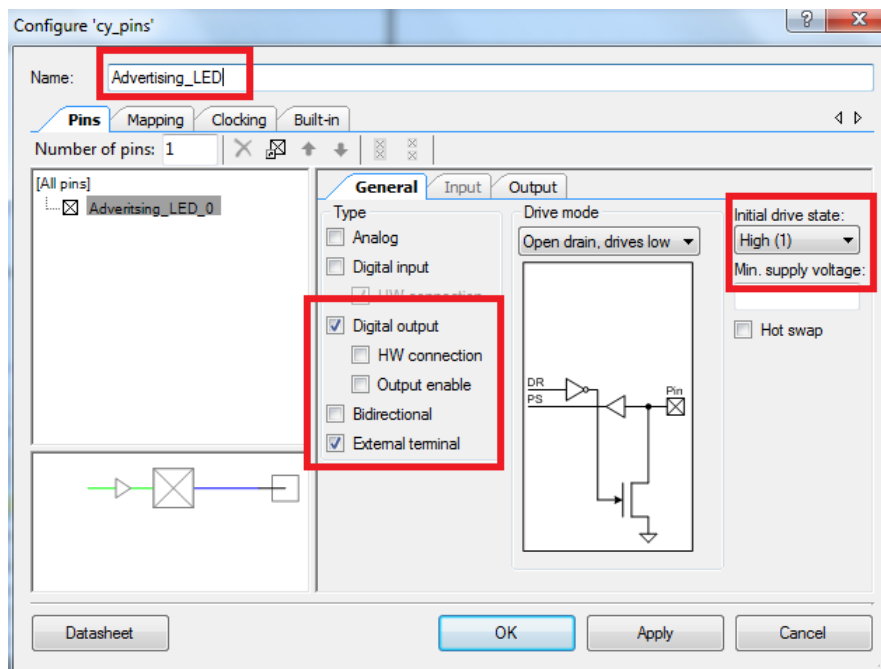
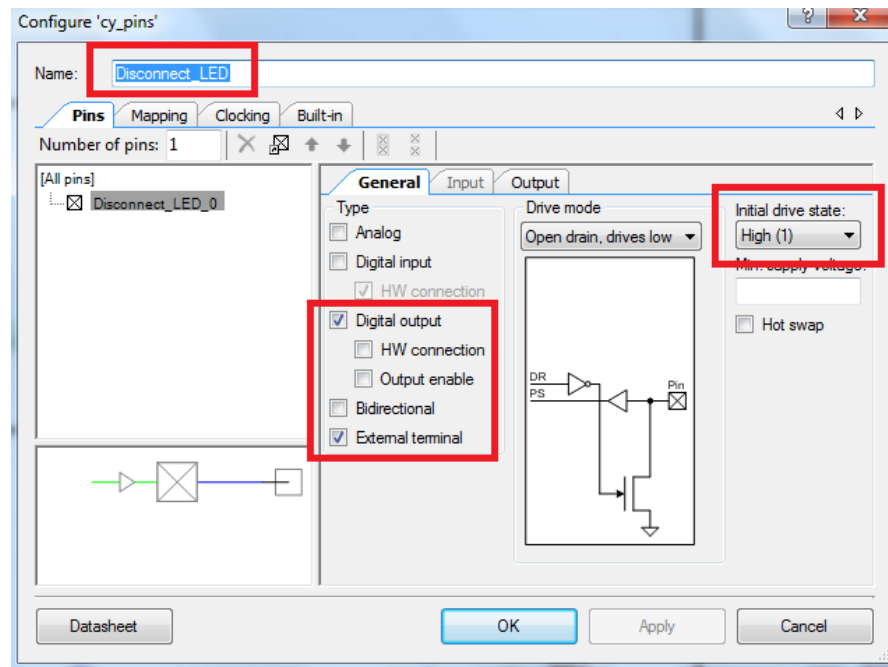
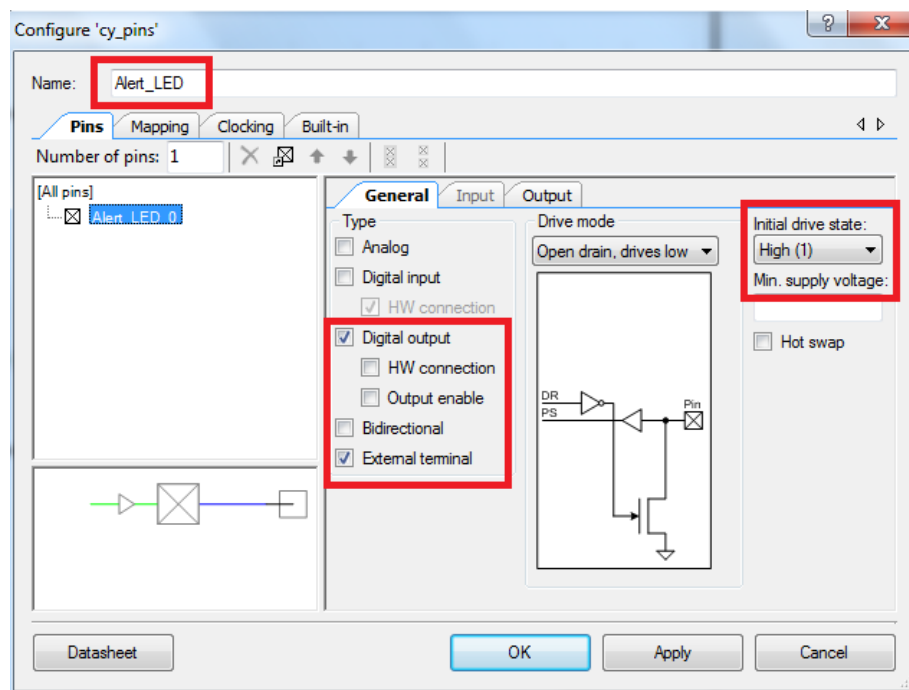


図 35. LED_2 コンフィギュレーション



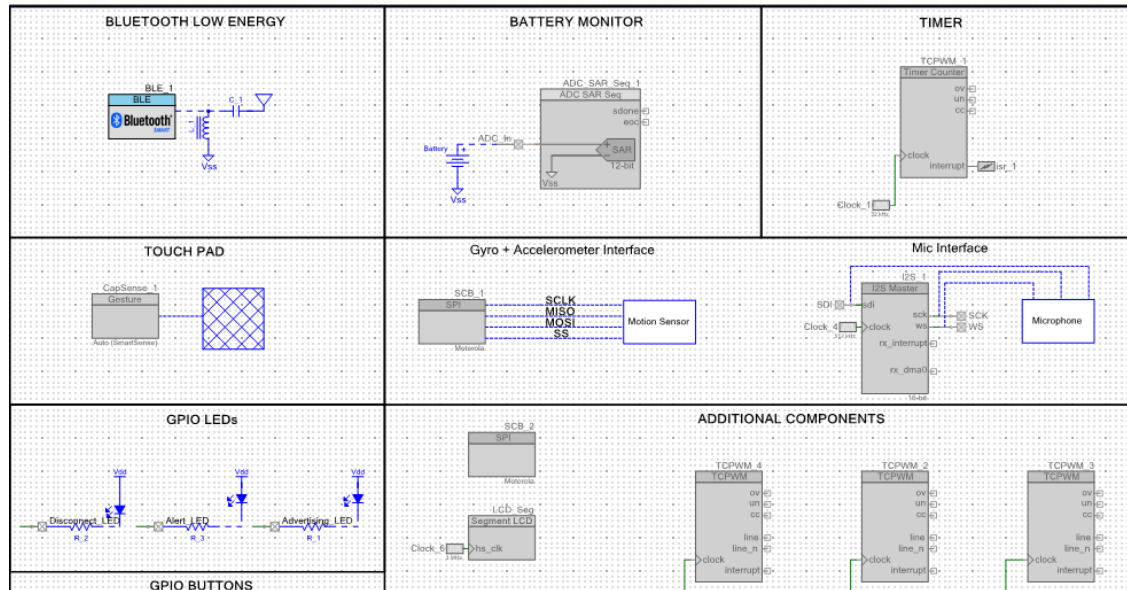
10. 図 36 に示すように、「External Terminal」が選択されている状態で LED_3 の名前を Alert_LED に変更し、「Initial drive state」を「High(1)」に設定します。

図 36. LED_3 コンフィギュレーション



11. 回路図設定を完了した後の設計は図 37 のようになります。

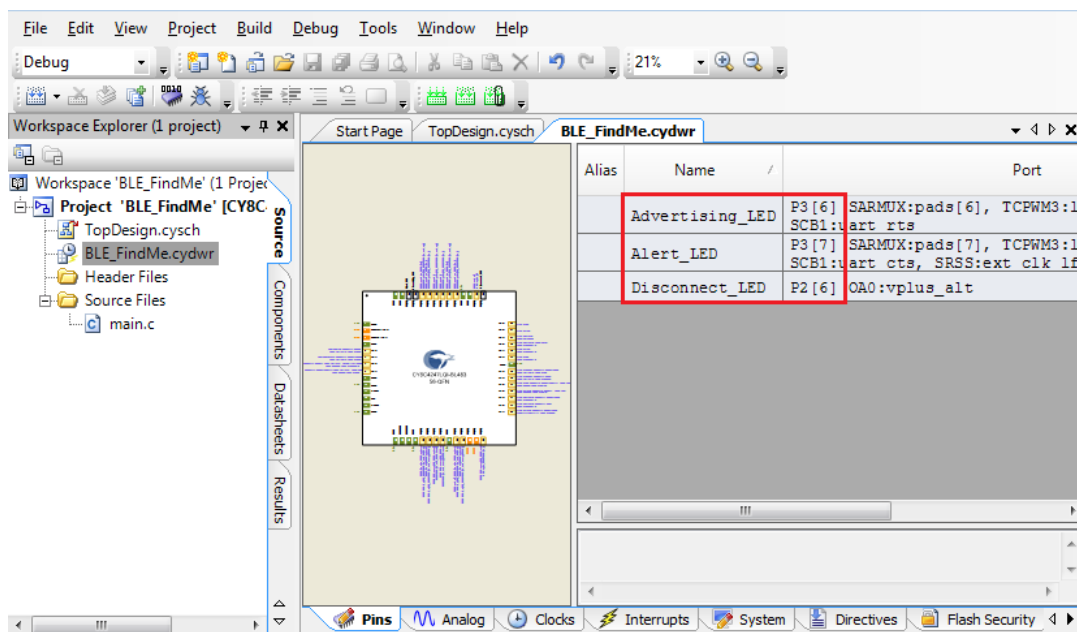
図 37. 回路図のコンフィギュレーション



注: 図 37 に示した青色の点線、LED 記号、抵抗記号はチップ外部の PSoC Creator コンポーネントであり、説明のためにのみ表示され、設計の機能には不要です。チップ外部のコンポーネントを設計に追加したい場合、PSoC Creator のチップ外部コンポーネント カタログから必要なチップ外部コンポーネントをプロジェクト回路図ページにドラッグ アンド ドロップします。

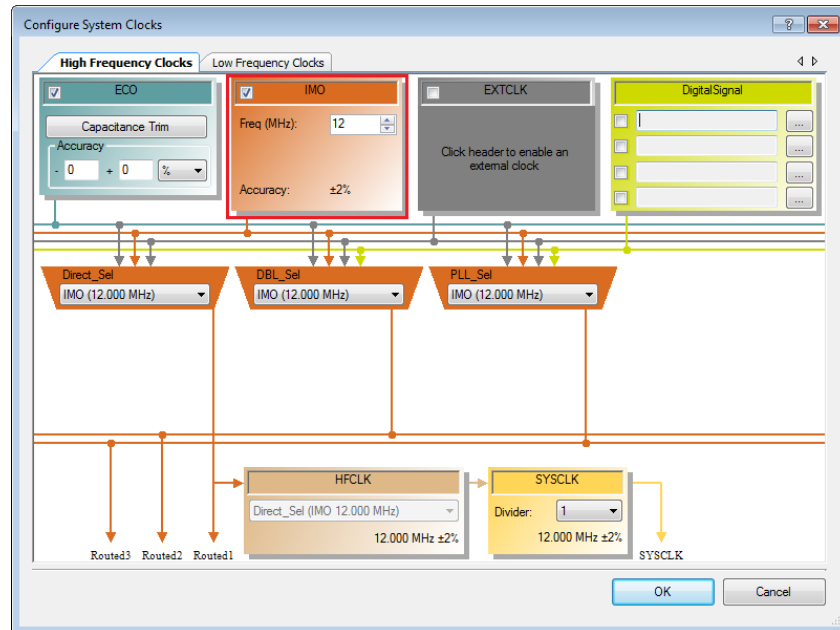
12. **Workspace Explorer** から *BLE_FindMe.cydwr* ファイル (Design-Wide Resources ファイル) を開き、**Pins** タブをクリックします。このタブでは、出力のデバイス ピン (Advertising_LED、Disconnect_LED、Alert_LED) を選択できます。図 38 に、Advertising_LED、Disconnect_LED、Alert_LED のピンを BLE Pioneer Kit 上のそれぞれ緑色 LED、赤色 LED、青色 LED に接続するピンコンフィギュレーションを示します。

図 38. ピンの選択



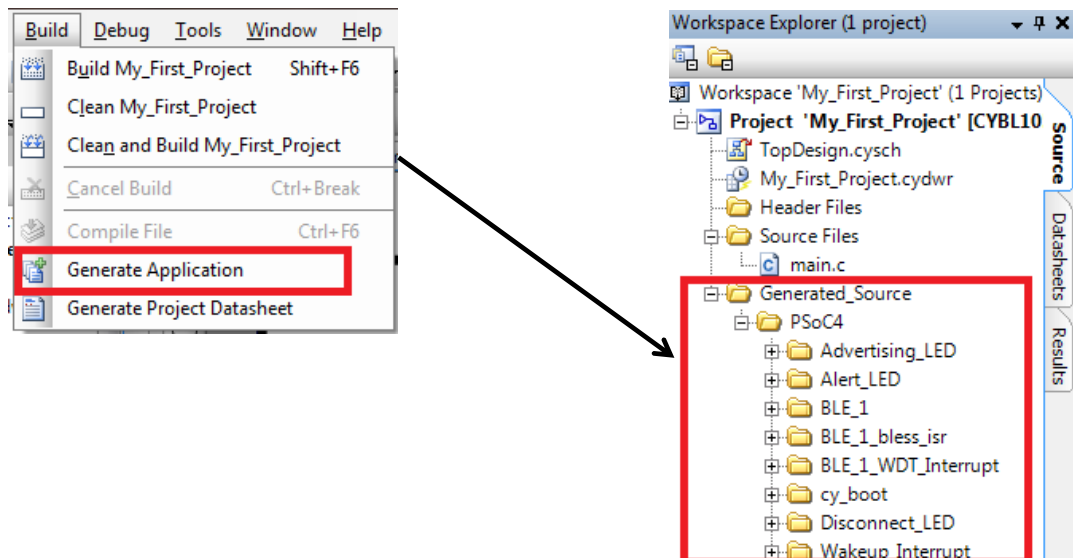
13. 同様に、図 39 に示すように、*BLE_FindMe.cydwr* ファイルの下 **Clocks** タブでは、**IMO** をダブルクリックして内部主発振器 (IMO) を 12MHz に設定します。**OK** をクリックします。

図 39. クロックの設定



14. **Build** メニューから **Generate Application** を選択します。図 40 に示すように、**Workspace Explorer** ウィンドウには、PSoC Creator によって自動的に生成された BLE、クロック、およびデジタル出力／入力ピン コンポーネントのソースコードファイルが表示されています。

図 40. 生成されたソース ファイル



6.4 第 2 段階: アプリケーション コードを書く

PSoC Creator で BLE 標準プロファイルのアプリケーションを設計するには、以下の 4 つの主なファームウェア ブロックが必要です。

- システム初期化
- BLE スタック イベント ハンドラ
- BLE サービス固有イベント ハンドラ
- メイン ループおよび低消費電力実装

本節で、[第 1 段階: 設計を作成／設定](#)で設定した設計に応じて各ブロックの詳細を説明します。

6.4.1 システム 初期化

PRoC BLE デバイスがリセットされると、ファームウェアはシステムの初期化を実行しなければなりません。このプロセスでは、プラットフォームが初期化され、グローバル割り込みおよび設計で使用するすべてのコンポーネントが有効化されます。システムが初期化された後、ファームウェアは BLE コンポーネントを初期化し、その結果、BLE サブシステム全体が内部的に初期化されます。[図 41](#) にシステムの初期化のフロー図を示します。

BLE 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 BLE スタックによって呼び出されるイベント ハンドラ関数を登録する必要があります。[図 44](#) に示す BLE スタック イベント ハンドラは BLE の初期化の一部として登録されます。BLE 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 固有イベント用のイベント ハンドラを登録し、制御権をメイン ループに切り替えます。メインループ コードのセグメントは[節 6.4.4 メインループおよび低消費電力動作](#)に説明されます。[図 42](#) にシステム初期化のファームウェア ソースコードを示します。

図 41. システム初期化のフロー図

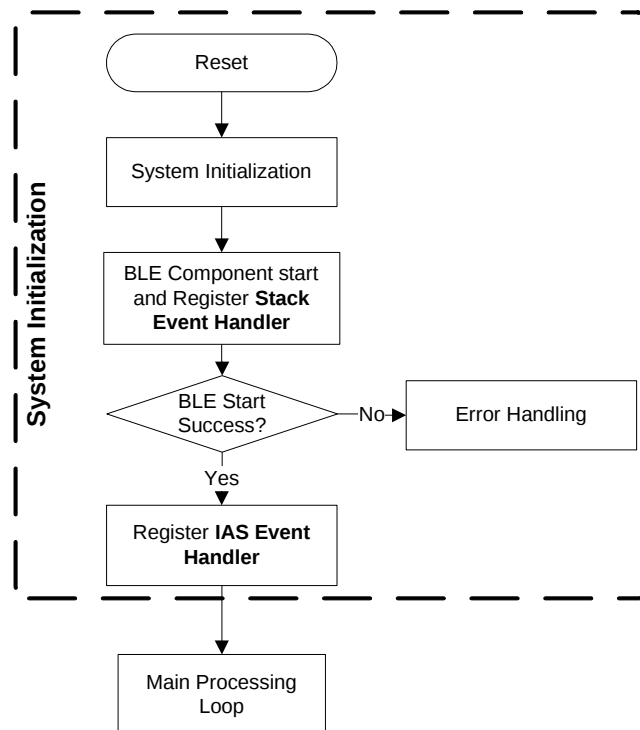


図 42. システム初期化のファームウェア

```
#include <project.h>

#define LED_ON                (0u)
#define LED_OFF               (1u)

#define NO_ALERT              (0u)
#define MILD_ALERT            (1u)
#define HIGH_ALERT            (2u)

#define LED_TOGGLE_TIMEOUT    (100u)

void StackEventHandler(uint32 event, void *eventParam);
void IasEventHandler(uint32 event, void *eventParam);

uint8 alertLevel;

int main()
{
    CYBLE_API_RESULT_T apiResult;

    CyGlobalIntEnable;

    apiResult = CyBle_Start(StackEventHandler);

    if(apiResult != CYBLE_ERROR_OK)
    {
        /* BLE stack initialization failed, check your configuration */
        CYASSERT(0);
    }

    CyBle_IasRegisterAttrCallback(IasEventHandler);

    /* Place the main application loop here */
}
```

6.4.2 BLE スタック イベント ハンドラ

BLE コンポーネント内の BLE スタックはイベントを生成し、CyBle_Start API 呼び出しで登録された BLE スタック イベント ハンドラを介して BLE インターフェースの状態とデータをアプリケーション ファームウェアに提供します。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し、そのスタックを設定して BLE 接続を確立および維持します。本書に述べる Find Me アプリケーションでは、BLE スタック イベント ハンドラは表 1 で説明するすべてのイベントを処理します。図 43 と図 44 に、BLE スタック イベント処理のフロー図とファームウェアを示します。

表 1. BLE スタック イベント

BLE 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
CYBLE_EVT_STACK_ON	BLE コンポーネント内の BLE ファームウェア スタックの初期化が正常に完了	広報を開始し、広報状態を LED に反映
CYBLE_EVT_GAP_DEVICE_DISCONNECTED	ピア デバイスとの BLE 接続切断	広報を再実行し、広報状態を LED に反映
CYBLE_EVT_GAP_DEVICE_CONNECTED	ピア デバイスとの BLE 接続確立	BLE 接続状態を LED に更新
CYBLE_EVT_GAPP_ADVERTISEMENT_START_STOP	BLE スタックの広報開始／停止イベント	広報がタイムアウトした場合、デバイスをストップ モードに設定

図 43. BLE スタック イベント ハンドラのフロー図

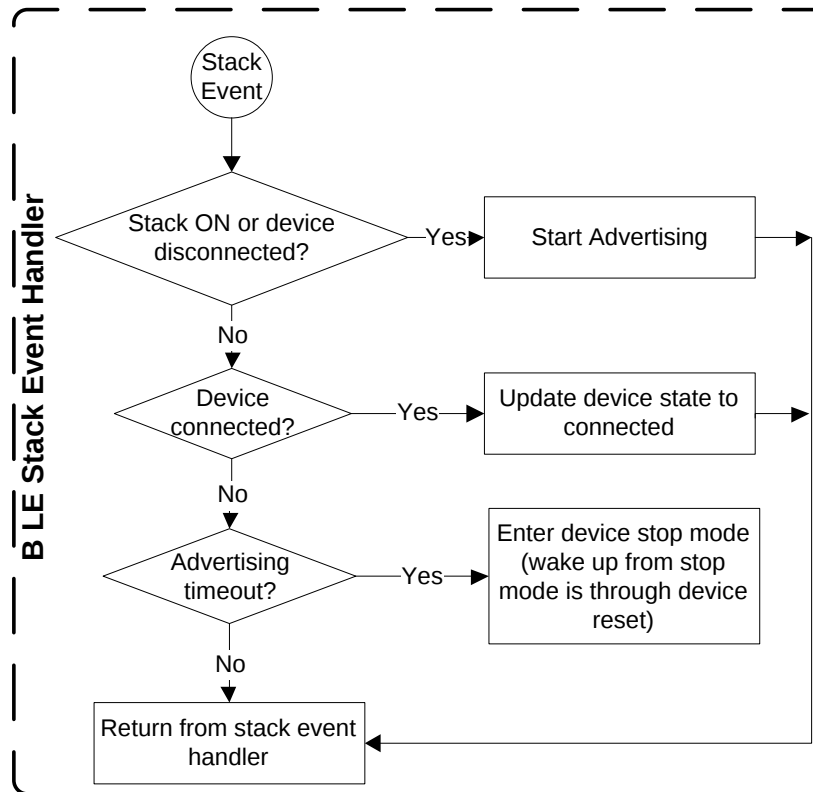


図 44. BLE スタック イベント ハンドラのファームウェア

```

void StackEventHandler(uint32 event, void *eventParam)
{
    switch(event)
    {
        /* Mandatory events to be handled by Find Me Target design */
        case CYBLE_EVT_STACK_ON:
        case CYBLE_EVT_GAP_DEVICE_DISCONNECTED:
            /* Start BLE advertisement for 30 seconds and update link
             * status on LEDs */
            CyBle_GappStartAdvertisement(CYBLE_ADVERTISING_FAST);
            Advertising_LED_Write(LED_ON);
            alertLevel = NO_ALERT;
            break;

        case CYBLE_EVT_GAP_DEVICE_CONNECTED:
            /* BLE link is established */
            Advertising_LED_Write(LED_OFF);
            Disconnect_LED_Write(LED_OFF);
            break;

        case CYBLE_EVT_GAPP_ADVERTISEMENT_START_STOP:
            if(CyBle_GetState() == CYBLE_STATE_DISCONNECTED)
            {

```

```

    /* Advertisement event timed out, go to low power
    * mode (Stop mode) and wait for device reset
    * event to wake up the device again */
    Advertising_LED_Write(LED_OFF);
    Disconnect_LED_Write(LED_ON);
    CySysPmSetWakeupPolarity(CY_PM_STOP_WAKEUP_ACTIVE_HIGH);
    CySysPmStop();

    /* Code execution will not reach here */
  }
  break;

default:
  break;
}
}

```

6.4.3 BLE サービス固有イベント ハンドラ

BLE コンポーネントはまた、設計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。ご作成中の Find Me Target アプリケーションでは、BLE コンポーネントはアラート レベルのキャラクタリスティックが新しい値に更新されるかをアプリケーションに通知する IAS イベントを生成します。図 45 と図 46 に BLE IAS イベントのフロー図とファームウェアを示します。

図 45. BLE IAS イベント ハンドラのフロー図

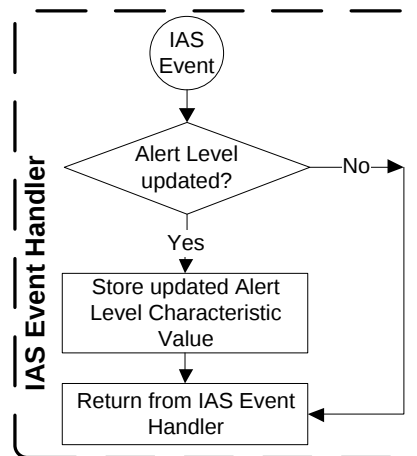


図 46. BLE IAS イベント ハンドラのファームウェア

```

void IasEventHandler(uint32 event, void *eventParam)
{
    /* Alert Level Characteristic write event */
    if(event == CYBLE_EVT_IASS_WRITE_CHAR_CMD)
    {
        /* Read the updated Alert Level value from the GATT database */
        CyBle_IassGetCharacteristicValue(CYBLE_IAS_ALERT_LEVEL,
            sizeof(alertLevel), &alertLevel);
    }
}

```

6.4.4 メインループおよび低消費電力動作

設計のメイン ループ ファームウェアは、BLE スタック処理イベントを定期的に処理し、IAS のアラート レベルのキャラクターシティック値に応じて青色 LED を更新し、接続期間との間に BLESS ブロックと PRoC BLE システムを低消費電力モードに設定します。図 47 および図 48 にメイン ループのフロー図とファームウェアを示します。このメイン ループのコードは、サービス固有イベントハンドラを登録するために *main* 関数内の呼び出し関数の次に配置することを推奨します。

図 47. メイン ループのフロー図

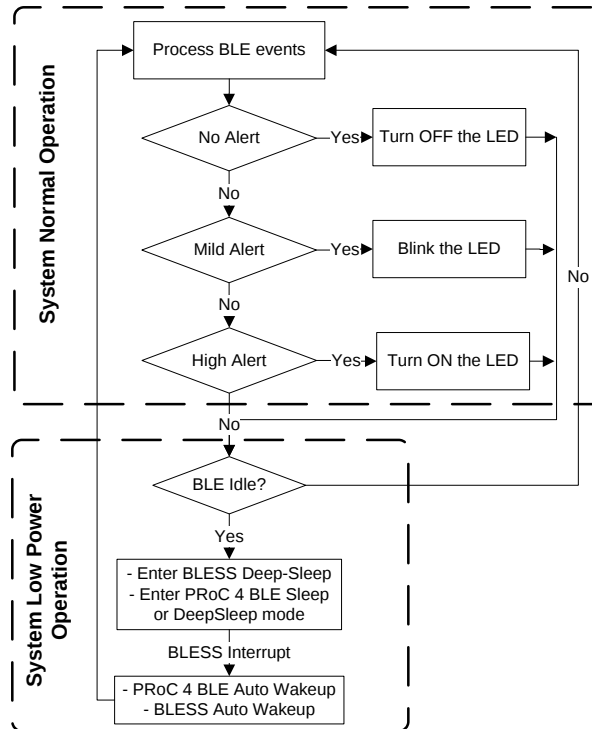


図 48. ファームウェアのメインループ

```

for (;;)
{
    static uint8 toggleTimeout = 0;
    CYBLE_BLESS_STATE_T blessState;
    uint8 intrStatus;

    /* Single API call to service all the BLE stack events. Must be
     * called at least once in a BLE connection interval */
    CyBle_ProcessEvents();

    /* Update Alert Level value on the blue LED */
    switch(alertLevel)
    {
        case NO_ALERT:
            Alert_LED_Write(LED_OFF);
            break;

        case MILD_ALERT:
            toggleTimeout++;
            if(toggleTimeout == LED_TOGGLE_TIMEOUT)
            {

```

```
        /* Toggle alert LED after timeout */
        Alert_LED_Write(Alert_LED_Read() ^ 0x01);
        toggleTimeout = 0;
    }
    break;

    case HIGH_ALERT:
        Alert_LED_Write(LED_ON);
        break;
}

/* Configure BLESS in Deep-Sleep mode */
CyBle_EnterLPM(CYBLE_BLESS_DEEPSLEEP);

/* Prevent interrupts while entering system low power modes */
intrStatus = CyEnterCriticalSection();

/* Get the current state of BLESS block */
blessState = CyBle_GetBleSsState();

/* If BLESS is in Deep-Sleep mode or the XTAL oscillator is turning
on,
 * then PSoC 4 BLE can enter Deep-Sleep mode (1.3uA current
consumption) */
if(blessState == CYBLE_BLESS_STATE_ECO_ON ||
    blessState == CYBLE_BLESS_STATE_DEEPSLEEP)
{
    CySysPmDeepSleep();
}
else if(blessState != CYBLE_BLESS_STATE_EVENT_CLOSE)
{
    /* If BLESS is active, then configure PSoC 4 BLE system in
 * Sleep mode (~1.6mA current consumption) */
    CySysPmSleep();
}
else
{
    /* Keep trying to enter either Sleep or Deep-Sleep mode */
}
CyExitCriticalSection(intrStatus);

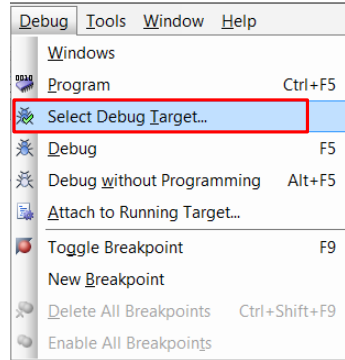
/* BLE link layer timing interrupt will wake up the system from
Sleep
 * and Deep-Sleep modes */
}
```

6.5 第 3 段階: デバイスをプログラム

本節は、PSoC Creator を使用してデバイスをプログラムする方法を示します。プログラマ内蔵の開発キットを使用する場合、USB ケーブルでキット基板をコンピュータに接続します。その他のキットについては、キット ガイドをご参照ください。自分のハードウェアで開発する場合、デバイスをプログラムするためにサイプレス [CY8CKIT-002 MiniProg3](#) などのハードウェアデバッガが必要です。

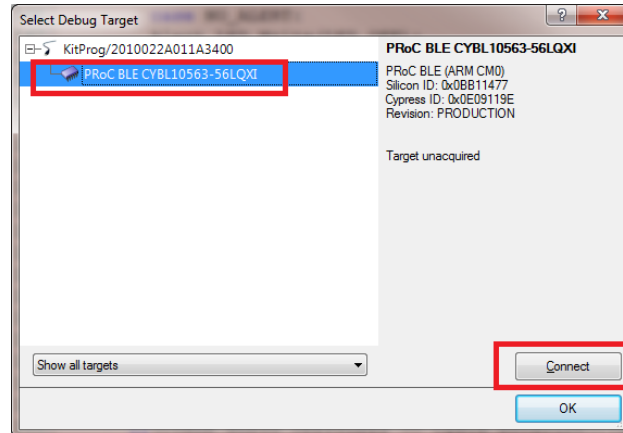
図 49 に示すように、PSoC Creator で、**Debug > Select Debug Target** を選択します。

図 49. デバッグ対象の選択



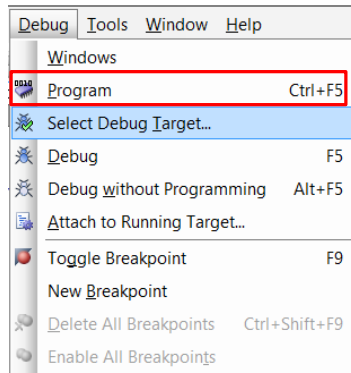
1. 図 50 に示すように、**Select Debug Target** ダイアログ ボックスで **Port Acquire** をクリックして **Connect** をクリックします。**OK** をクリックして、ダイアログ ボックスを閉じます。

図 50. デバイスに接続



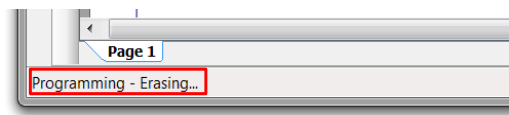
2. 図 51 に示すように、**Debug > Program** を選択してプロジェクトをデバイスにプログラムします。

図 51. デバイスのプログラム



3. 図 52 に示すように、プログラミングの状態は PSoC Creator のステータス バー (画面の左下隅) に表示されます。

図 52. プログラミングの状態



6.6 第 4 段階: 設計をテスト

本節では、CySmart モバイル アプリと PC アプリを用いて BLE 設計をテストする方法を説明します。図 16 に、BLE Pioneer Kit を用いた設計テストのセットアップを示します。

1. 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。
2. CySmart アプリを起動します。
3. BLE Pioneer Kit 上のリセット スイッチを押してご開発の設計からの BLE 広報を開始します。
4. CySmart アプリのホーム画面をプルダウンし、BLE ペリフェラルのスキャンを開始します。これで、ご使用のデバイスは CySmart アプリのホーム画面に表示されます。ご使用のデバイスを選択して BLE 接続を確立します。
5. カラーセル画面で「Find Me」プロフィールを選択します。
6. Find Me **Profile** 画面でアラート レベルの 1 つを選択します。ご覧のように、デバイス上の LED 状態が選択に応じて変化します。

図 53 と図 54 に、CySmart モバイル アプリの設定手順を示します。

図 53. iOS アプリ CySmart でのテスト

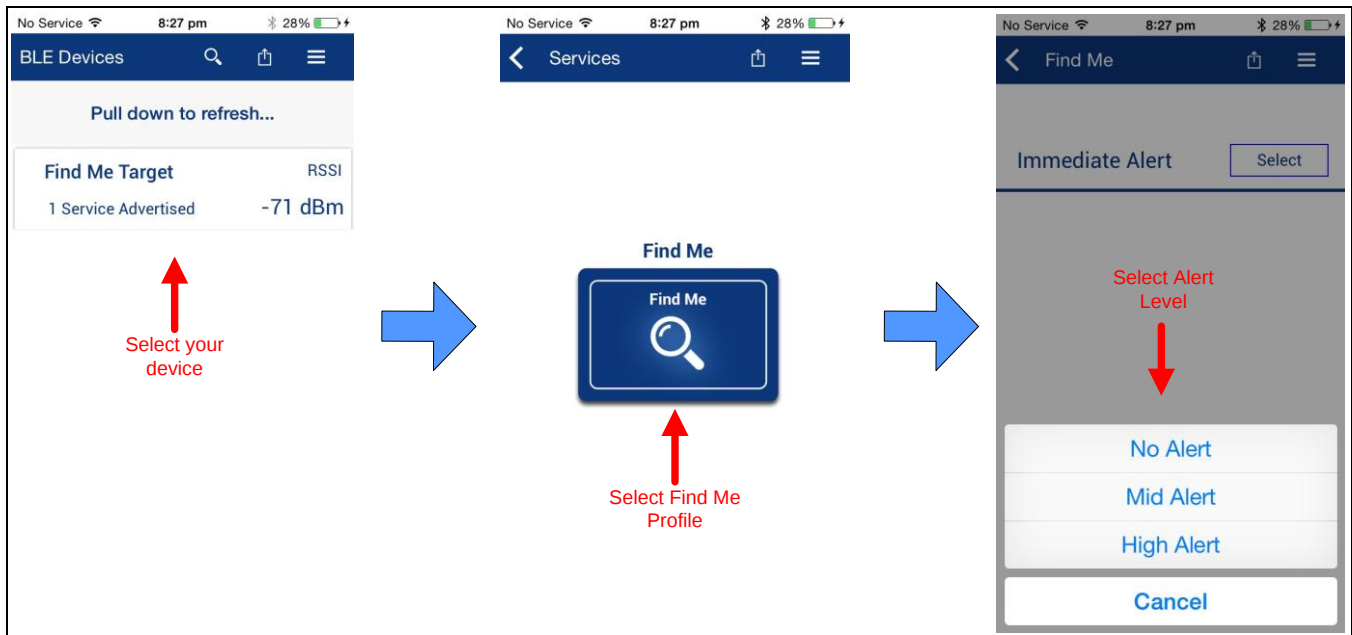
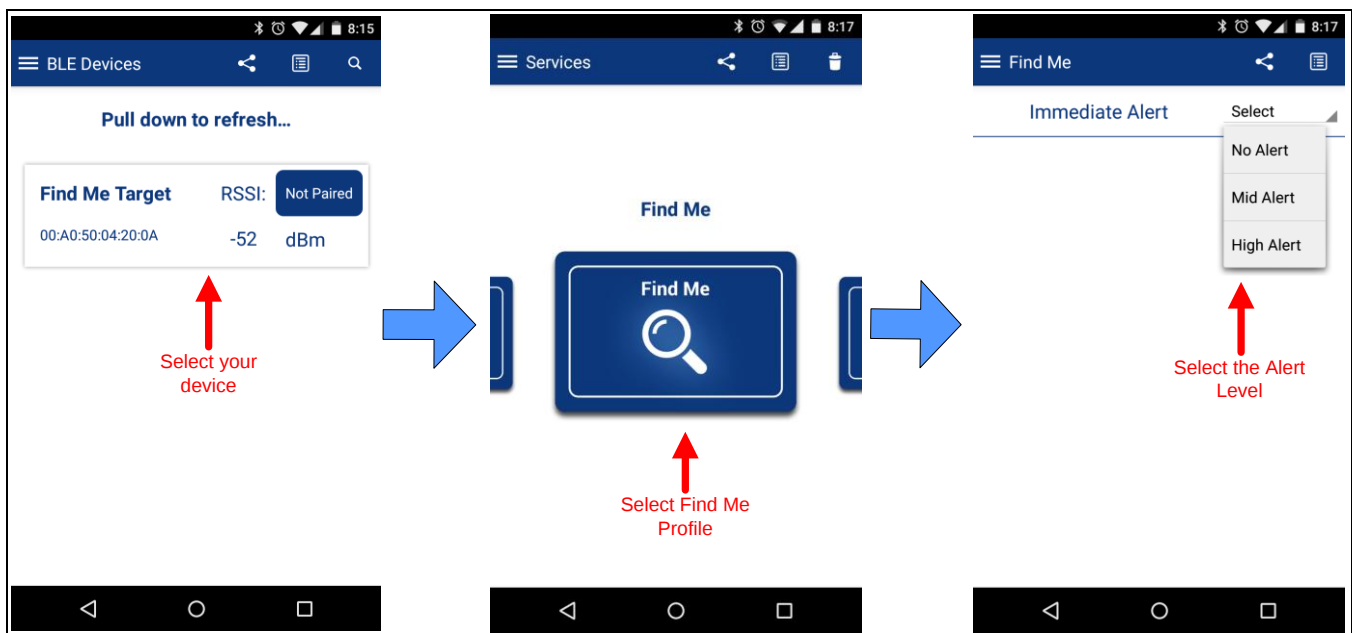


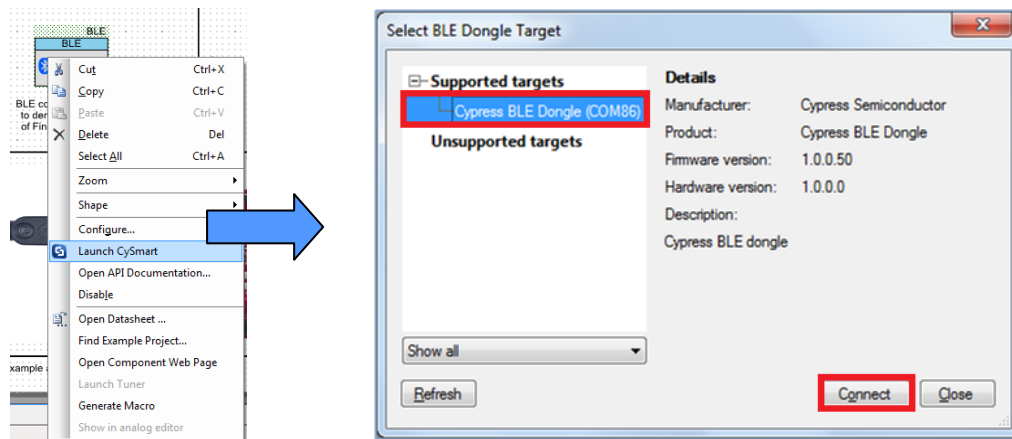
図 54. Android アプリ CySmart でのテスト



CySmart モバイル アプリと同様に、CySmart ホスト エミュレーション ツールを使用してご開発の設計との BLE 接続を確立し、BLE キャラクタースティックに対して読み書き処理を実行できます。

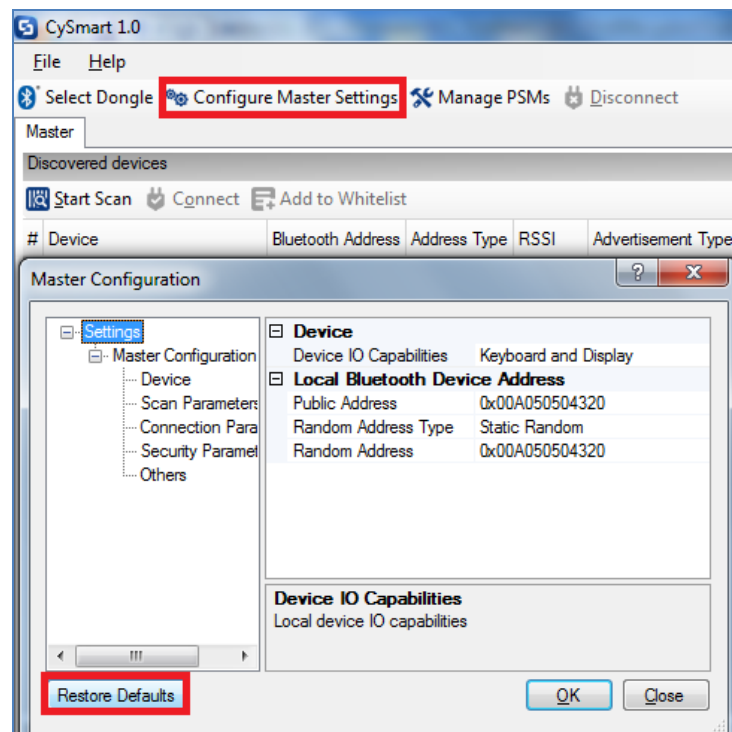
1. BLE ドングルをご使用の Windows マシンに接続します。ドライバーのインストールが完了するまで待ちます。
2. CySmart ホスト エミュレーション ツールを起動します。ツールは自動的に BLE ドングルを検出します。BLE ドングルが **Select BLE Dongle Target** ポップアップ ウィンドウに表示されない場合、**Refresh** をクリックします。図 55 に示すように **Connect** をクリックします。

図 55. CySmart での BLE ドングル選択



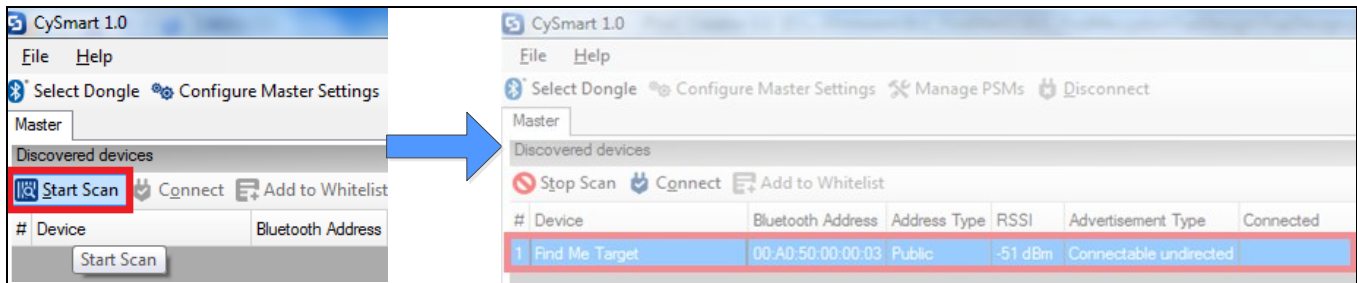
3. 図 56 に示すように、**Configure Master Settings** を選択して初期設定値を復元します。

図 56. 「CySmart Master Settings」コンフィギュレーション



4. BLE Pioneer Kit 上のリセット スイッチを押してご開発の設計からの BLE 広報を開始します。
5. 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、ご使用のデバイスが **Discovered devices** リストに表示されます。

図 57. CySmart でのデバイス検出



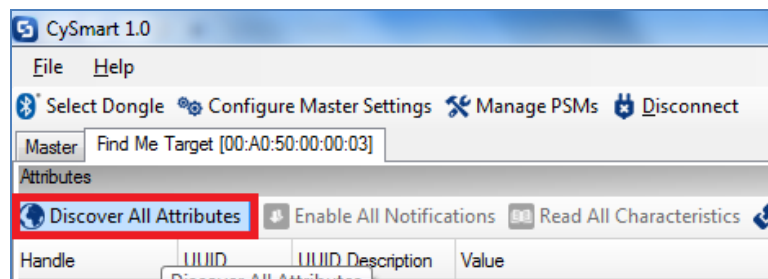
- 図 58 に示すように、ご使用のデバイスを選択し、**Connect** をクリックして CySmart ホスト エミュレーション ツールとの BLE 接続を確立します。

図 58. CySmart でのデバイス接続



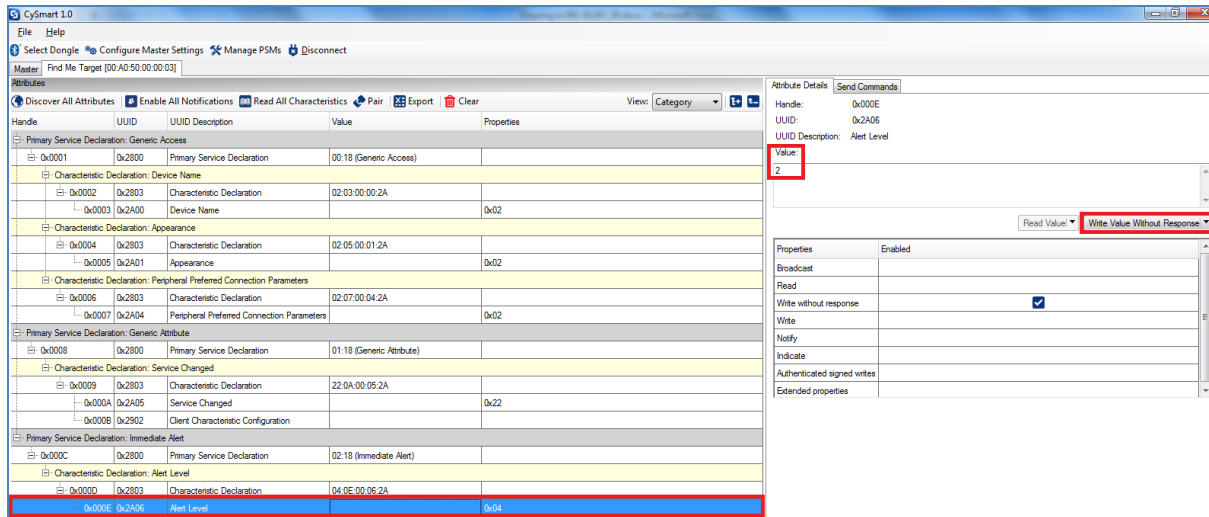
- 接続が確立された後、図 59 に示すように、CySmart ホスト エミュレーション ツールでご開発の設計のすべてのアトリビュートを検出します。

図 59. CySmart での属性検出



- 図 60 に示すように、Immediate Alert Service の下の Alert Level Characteristic に値 0、1、2 のいずれかを書きます。ご覧のように、ご使用のデバイス上の LED 状態がアラートレベル特性の設定に応じて変化します。

図 60. CySmart ホスト エミュレーション ツールでのテスト



7 まとめ

本アプリケーション ノートは BLE プロトコルと PRoC BLE デバイスのアーキテクチャおよび開発ツールの基礎について説明しました。PSoC 4 BLE は、BLE 無線ブロック、デジタルのペリフェラル機能、メモリ、ARM Cortex-M0 マイクロコントローラーを単一チップに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。集積した機能および低電力モードが整備されたため、PRoC BLE はバッテリー駆動ウェアラブル機器、健康関連製品の BLE アプリケーションに最適になります。

本アプリケーション ノートはまた、PRoC BLE を迅速に深く理解できるように総合的なリソース収集へのアクセスも提供しました。

8 関連アプリケーション ノート

- [AN91445](#) – Antenna Design Guide
- [AN91267](#) – Getting Started with PSoC® 4 BLE
- [AN91184](#) – PSoC 4 BLE - Designing BLE Applications
- [AN91162](#) – Creating a BLE Custom Profile
- [AN92584](#) – Designing for Low Power and Estimating Battery Life for BLE Applications
- [AN96841](#) – Getting Started With EZ-BLE™ PRoC™ Module

著者について

氏名: Sai Prashanth Chinnapalli (CSAI)

役職: アプリケーション エンジニア

経歴: インドの イラーハーバードのモオール・ネイルー・ナショナル工科大学、電気工学理学士

氏名: Rahul Garg (RAHU)

役職: シニア システム エンジニア

経歴: インドの ニューデリーの Bharathi Vidyapeeth's College of Engineering、技術の学士号

9 付録 A: BLE デバイス ファミリの比較

表 2 はサイプレスの BLE デバイス ファミリの特長と機能をまとめます。

表 2. BLE デバイス ファミリ

機能	デバイス ファミリ					
	CY8C41x7-BL	CY8C42x7-BL	CYBL10X6X	CYBL10X7X	CY8C41x8-BL	CY8C42x8-BL
BLE サブシステム	Bluetooth 4.1 対応のプロトコルスタックを持つ BLE 無線とリンク層ハードウェアブロック	Bluetooth 4.1 対応のプロトコルスタックを持つ BLE 無線とリンク層ハードウェアブロック	Bluetooth 4.1 対応のプロトコルスタックを持つ BLE 無線とリンク層ハードウェアブロック	Bluetooth 4.1 対応のプロトコルスタックを持つ BLE 無線とリンク層ハードウェアブロック	Bluetooth 4.1 対応のプロトコルスタックを持つ BLE 無線とリンク層ハードウェアブロック	Bluetooth 4.1 対応のプロトコルスタックを持つ BLE 無線とリンク層ハードウェアブロック
CPU	24MHz ARM Cortex-M0 CPU、シングルサイクルの乗算に対応	48MHz ARM Cortex-M0 CPU、シングルサイクルの乗算に対応	48MHz ARM Cortex-M0 CPU、シングルサイクルの乗算に対応	48MHz ARM Cortex-M0 CPU、シングルサイクルの乗算に対応	24MHz ARM Cortex-M0 CPU、シングルサイクルの乗算に対応	48MHz ARM Cortex-M0 CPU、シングルサイクルの乗算に対応
フラッシュ メモリ	128KB	128KB	128KB	256KB	256KB	256KB
SRAM	16KB	16KB	16KB	32KB	32KB	32KB
GPIO	最大 36	最大 36	最大 36	最大 36	最大 36	最大 36
Ca pSense	最大 35 個のセンサー	最大 35 個のセンサー	最大 35 個のセンサー	最大 35 個のセンサー	最大 35 個のセンサー	最大 35 個のセンサー
CapSense ジェスチャー	一部のデバイスのみ	一部のデバイスのみ	一部のデバイスのみ	一部のデバイスのみ	一部のデバイスのみ	一部のデバイスのみ
ADC	シーケンサーを備えた 12 ビット、806ksps SAR ADC	シーケンサーを備えた 12 ビット、1Msps SAR ADC	シーケンサーを備えた 12 ビット、1Msps SAR ADC	シーケンサーを備えた 12 ビット、1Msps SAR ADC	シーケンサーを備えた 12 ビット、806ksps SAR ADC	シーケンサーを備えた 12 ビット、1Msps SAR ADC
オペアンプ	ディープスリープモードでアクティブな 2 個のプログラマブルなオペアンプ	ディープスリープモードでアクティブな 4 個のプログラマブルなオペアンプ	なし	なし	ディープスリープモードでアクティブな 2 個のプログラマブルなオペアンプ	ディープスリープモードでアクティブな 4 個のプログラマブルなオペアンプ
コンパレータ	ウェイクアップ機能を備えた 2 個の低消費電力コンパレータ	ウェイクアップ機能を備えた 2 個の低消費電力コンパレータ	なし	なし	ウェイクアップ機能を備えた 2 個の低消費電力コンパレータ	ウェイクアップ機能を備えた 2 個の低消費電力コンパレータ
電流 DAC	7 ビット (1 個)、8 ビット (1 個)	7 ビット (1 個)、8 ビット (1 個)	なし	なし	7 ビット (1 個)、8 ビット (1 個)	7 ビット (1 個)、8 ビット (1 個)
電源供給範囲	1.9V~5.5V	1.9V~5.5V	1.9V~5.5V	1.9V~5.5V	1.9V~5.5V	1.9V~5.5V
低消費電力モード	ディープスリープモード: 1.3μA を消費 ハイバネートモード: 150nA を消費 ストップモード: 60nA を消費	ディープスリープモード: 1.3μA を消費 ハイバネートモード: 150nA を消費 ストップモード: 60nA を消費	ディープスリープモード: 1.3μA を消費 ハイバネートモード: 150nA を消費 ストップモード: 60nA を消費	ディープスリープモード: 1.3μA を消費 ハイバネートモード: 150nA を消費 ストップモード: 60nA を消費	ディープスリープモード: 1.3μA を消費 ハイバネートモード: 150nA を消費 ストップモード: 60nA を消費	ディープスリープモード: 1.3μA を消費 ハイバネートモード: 150nA を消費 ストップモード: 60nA を消費

機能	デバイス ファミリ					
	CY8C41x7-BL	CY8C42x7-BL	CYBL10X6X	CYBL10X7X	CY8C41x8-BL	CY8C42x8-BL
セグメント LCD 駆動	一部のデバイスでの 4 COM、32 セグメントの LCD 駆動	一部のデバイスでの 4 COM、32 セグメントの LCD 駆動	一部のデバイスでの 4 COM、32 セグメントの LCD 駆動	一部のデバイスでの 4 COM、32 セグメントの LCD 駆動	一部のデバイスでの 4 COM、32 セグメントの LCD 駆動	一部のデバイスでの 4 COM、32 セグメントの LCD 駆動
シリアル通信	プログラマブルな I ² C、SPI、または UART を備えた 2 個の個別のシリアル通信ブロック (SCB)	プログラマブルな I ² C、SPI、または UART を備えた 2 個の独立した SCB	プログラマブルな I ² C、SPI または UART を備えた 1 個または 2 個の独立した SCB	プログラマブルな I ² C、SPI または UART を備えた 1 個または 2 個の独立した SCB	プログラマブルな I ² C、SPI、または UART を備えた 2 個の独立したシリアル通信ブロック (SCB)	プログラマブルな I ² C、SPI、または UART を備えた 2 個の独立した SCB
タイマー／カウンタ／パルス幅変調器 (TCPWM)	4	4	4	4	4	4
汎用デジタルブロック (UDB)	なし	4 個、各々は 8 つのマクロセルおよび 1 つのデータパスを備える追加のデジタル ペリフェラル (タイマー、カウンタ、PWM) または通信 インターフェース (UART、SPI) を合成するのに使用可能	なし	なし	なし	4 個、各々は 8 つのマクロセルおよび 1 つのデータパスを備える追加のデジタル ペリフェラル (タイマー、カウンタ、PWM) または通信 インターフェース (UART、SPI) を合成するのに使用可能
追加のデジタルペリフェラル (I ² S、PWM)	なし	あり (一部のデバイスでの UDB ベースのデジタルペリフェラル)	あり (一部のデバイスでの固定機能ブロック)	あり (一部のデバイスでの固定機能ブロック)	なし	あり (一部のデバイスでの UDB ベースのデジタルペリフェラル)
クロック	IMO: 3MHz～24MHz ILO: 32kHz ECO: 24MHz WCO: 32kHz	IMO: 3MHz～48MHz ILO: 32kHz ECO: 24MHz WCO: 32kHz	IMO: 3MHz～48MHz ILO: 32kHz ECO: 24MHz WCO: 32kHz	IMO: 3MHz～48MHz ILO: 32kHz ECO: 24MHz WCO: 32kHz	IMO: 3MHz～24MHz ILO: 32kHz ECO: 24MHz WCO: 32kHz	IMO: 3MHz～48MHz ILO: 32kHz ECO: 24MHz WCO: 32kHz
電源電圧監視	パワーオンリセット (POR) 電圧低下検出 (BOD) 低電圧検出 (LVD)	POR BOD LVD	POR BOD LVD	POR BOD LVD	POR BOD LVD	POR BOD LVD
パッケージ	56-QFN (7.0 × 7.0 × 0.6mm) および 68-WLCSP (3.52 × 3.91 × 0.55mm)	56-QFN (7.0 × 7.0 × 0.6mm) および 68-WLCSP (3.52 × 3.91 × 0.55mm)	56-QFN (7.0 × 7.0 × 0.6mm) および 68-WLCSP (3.52 × 3.91 × 0.55mm)	56-QFN (7.0 × 7.0 × 0.6mm) および 68-WLCSP (3.52 × 3.91 × 0.55mm)	56-QFN* (7.0 × 7.0 × 0.6mm) および 76-WLCSP (4.04 × 3.87 × 0.55mm)	56-QFN* (7.0 × 7.0 × 0.6mm) および 76-WLCSP (4.04 × 3.87 × 0.55mm)

10 付録 B : サイプレスの専門用語

本節では、サイプレスの PSoC ファミリのデバイスで操作する際によく出てくる用語について説明します。

コンポーネント コンフィギュレーション ツール: PSoC Creator の簡単な GUI で、各コンポーネントに組み込まれています。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。

コンポーネント: PSoC Creator ソフトウェア上にアイコンとして表示する無償の組み込み IC です。これらのコンポーネントは複数の IC とシステム インターフェースを、本質的にメイン システム バスを介して MCU に接続される 1 個の PSoC コンポーネントに統合するのに使用されます。例えば、BLE コンポーネントで Bluetooth Smart 製品をほんの数分で作成できます。同様に、プログラム可能なアナログ コンポーネントをセンサー用に使用できます。

MiniProg3: 開発用のプログラミング ハードウェアであり、内蔵のプログラマーがサポートされていないカスタム基板または PSoC 開発キットの上に PSoC デバイスをプログラムするために使用されます。

PSoC BLE: Bluetooth 4.1 仕様に対応するロイヤリティフリーの BLE プロトコル スタックを含む BLE 無線ブロックを統合した固定機能の SoC です。

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、および PSoC 5LP の統合設計環境 (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を同時に設計するか、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可能にします。

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ム アプリケーションです。PSoC Programmer は PSoC Creator と統合され、PSoC 3、PSoC 4、PSoC 5LP、および PSoC 5LP 設計をプログラムします。

11 付録 C: サイプレス BLE 開発ツール

PRoC BLE 開発キット

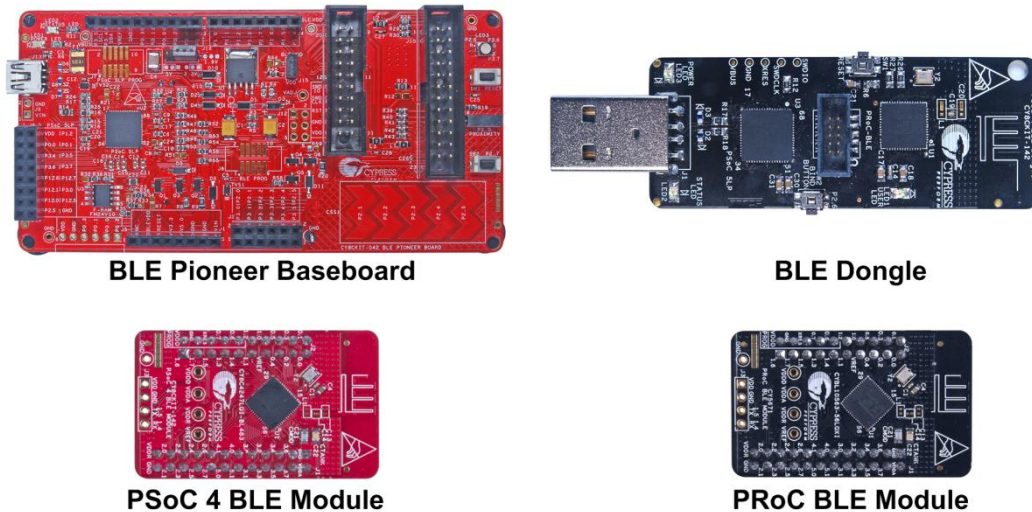
図 61 に示す BLEPioneer Kit は、PSoC BLE と PRoC BLE 両方のデバイス ファミリに対応しているサイプレスの BLE 開発キットです。このキットは BLE Pioneer ベースボードに接続した取り外し可能な PRoC BLE (および PSoc 4 BLE) モジュールから成ります。このキットはコイン電池から、または USB インターフェースを介して電源供給を受けます。

BLE Pioneer ベースボードと BLE モジュールを組み合わせて使用すれば、標準 Arduino Uno コネクタ準拠のシールドと併用する低消費電力のバッテリー式 BLE デバイス、または CapSense ユーザー インターフェースなどの基板搭載 PRoC BLE デバイス機能を開発できます。

BLE Pioneer ベースボードはオンボードプログラムとデバッグ インターフェースを搭載しているので、PSoC Creator IDE を介した BLE 設計のデバッグ処理は手早く簡単になります。BLE Pioneer Kit は Digilent®などの第三者ベンダーからの Pmod™ドーターカードとのインターフェースでの接続をサポートする追加のヘッダーを備えています。このキットは、BLE 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 ツールと連動して、非 BLE Windows PC で BLE ホスト エミュレーション プラットフォームを実現する BLE ドングルも内蔵しています。

このキットには、ユーザーが独自の BLE 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一式が含まれています。キットの最新版、キット デザイン、サンプル プロジェクトやドキュメント ファイルを入手するには、www.cypress.com/go/CY8CKIT-042-BLE にアクセスしてください。

図 61. BLE Pioneer Kit



注: PRoC BLE 開発キットでは、PRoC BLE デバイス モジュール (「図 61. BLE Pioneer Kit」での黒色のモジュール) は 128KB のフラッシュ メモリを持つ CYBL10X6X デバイスを内蔵しています。256KB のフラッシュ メモリを持つデバイス (CYBL10X7X) を内蔵しているモジュールはサイプレス ウェブサイトの [CY5676](http://www.cypress.com/go/CY5676) ウェブページで個別に注文できます。

11.1 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 BLE Pioneer Kit の BLE ドングルを使用して BLE センtral デバイスをエミュレートするための Windows アプリケーションです。このアプリケーションは BLE Pioneer Kit の一部としてインストールされ、BLE コンポーネントの右クリック オプションから起動できます。これは、ペリフェラルの BLE サービス、キャラクタースティック、およびアトリビュートを調べ、構成することで、GATT または L2CAP 接続指向のチャネルを介して、ご使用の PSoc 4 BLE ペリフェラル実装をテスト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 ツールを使用して実現できる動作は以下が含まれますが、これらに限定されません。

- BLE 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する。
- サービスとキャラクタースティックなどの、接続したペリフェラル デバイスの BLE アトリビュートを検出する。
- キャラクタースティック バリューとディスクリプタに基づいて読み書きを行う。
- 接続したペリフェラル デバイスからのキャラクタースティック通知および指示を受ける。
- BLE 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 デバイスとの結び (ボンド) を確立する。
- ペリフェラル デバイスとの BLE L2CAP 接続指向のセッションを設定し、Bluetooth 4.1 仕様に従ってデータを交換する。

図 62 と図 63 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については、**Help** メニュー内の CySmart ユーザー ガイドをご参照ください。

図 62. 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

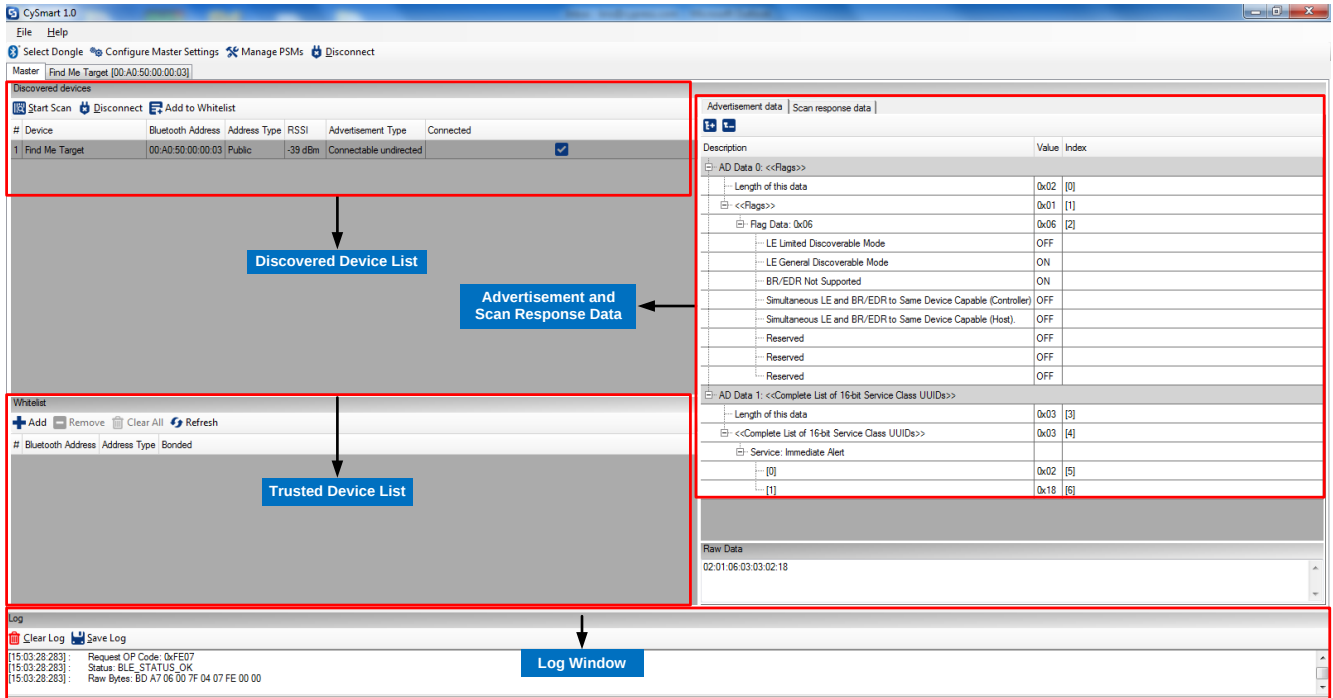
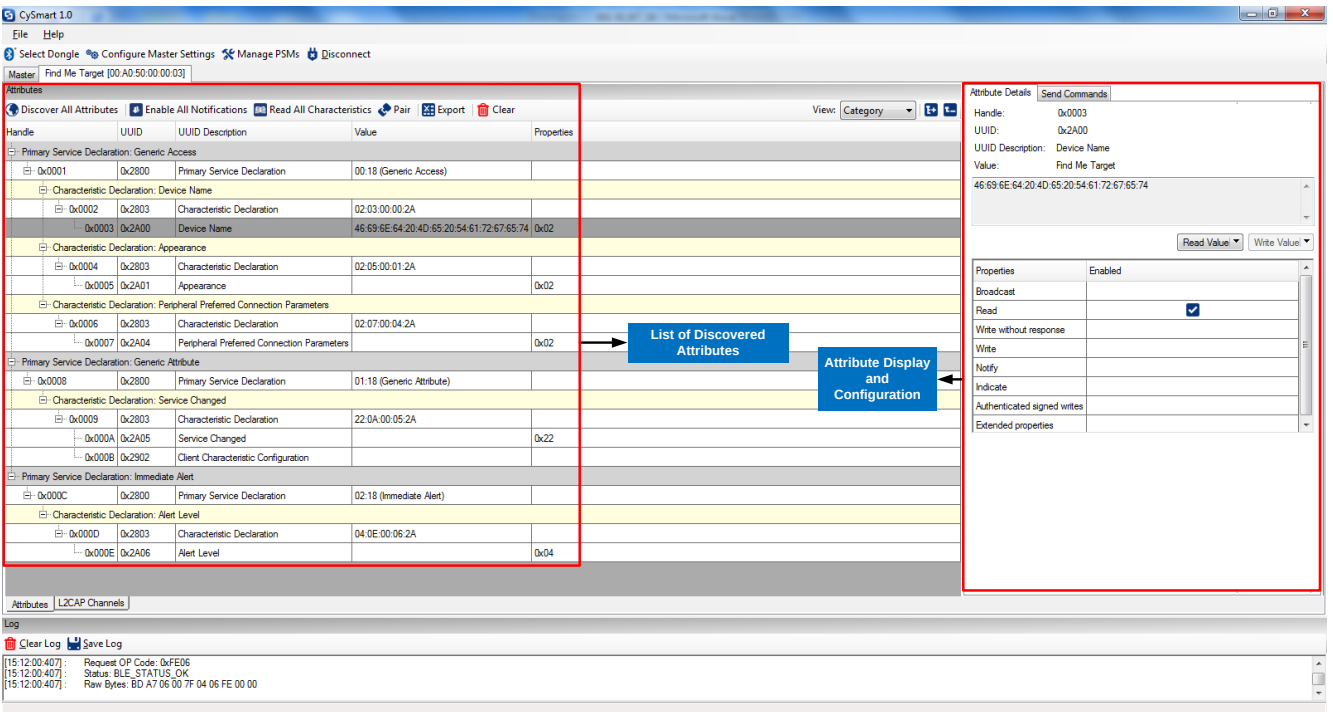


図 63. CySmart ホスト エミュレーション ツールのペリフェラル デバイス アトリビュート タブ



11.2 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 ストアからの iOS や Android 用の CySmart モバイル アプリをダウンロードすることもできます。このアプリは、BLE 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 デバイスとして構成できるよう、それぞれの iOS コアの Bluetooth フレームワークと BLE 用の Android の組み込みプラットフォーム フレームワークを使用しています。

モバイル アプリは、直観的な GUI を使用して SIG 採用の BLE 標準プロファイルをサポートし、基本的な BLE サービスおよびキャラクタースティックの詳細を簡略化します。BLE 標準プロファイルに加えて、アプリはサイプレス社の 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実証します。図 64 と図 65 に、CySmart アプリにおける心拍数プロファイル ユーザー インターフェースの一連のスクリーンショットを示しています。BLE Pioneer Kit のプロジェクト例を使用してアプリケーションを実行する方法については、BLE Pioneer Kit ガイドを参照してください。

図 64. CySmart iOS アプリ心拍数プロファイルの例

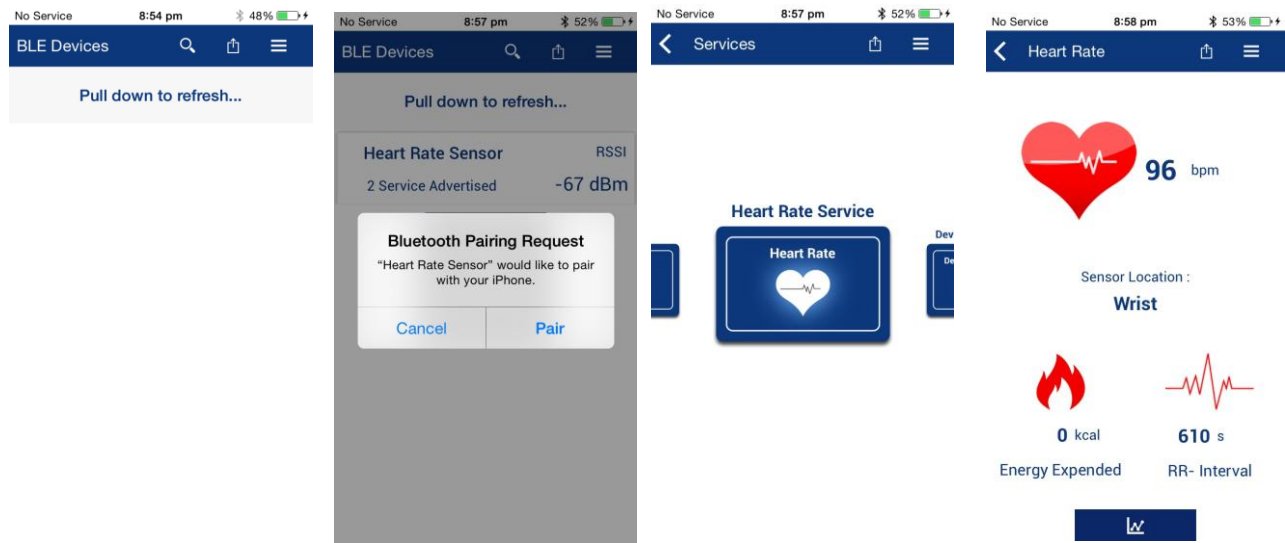
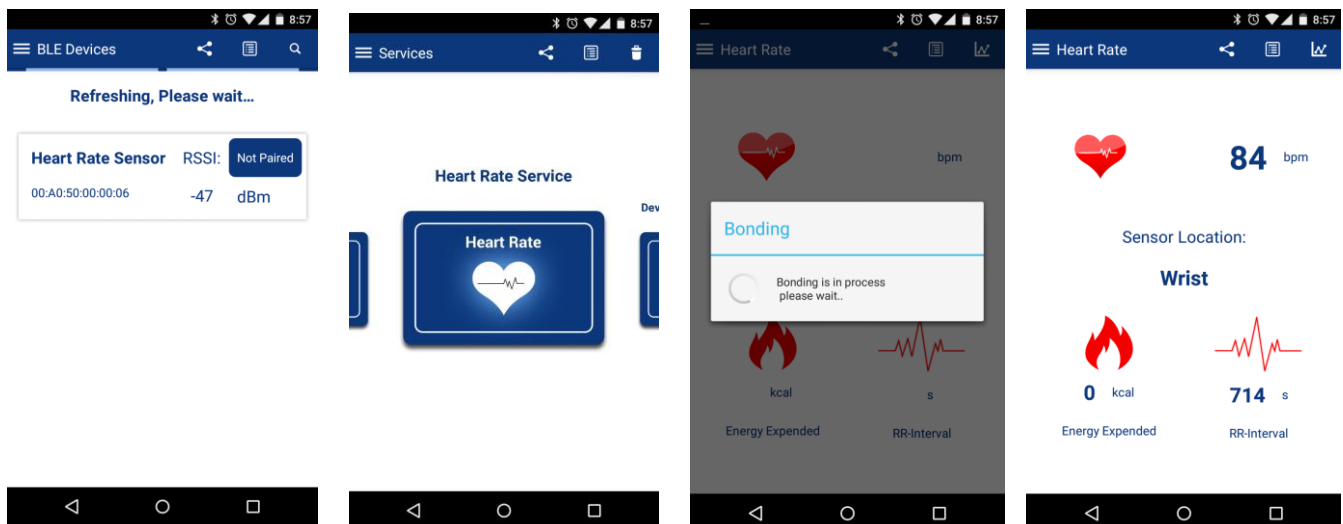


図 65. CySmart Android アプリ心拍数プロファイルの例



12 付録 D: PRoC BLE デバイス

12.1 ARM Cortex-M0 とメモリ

PRoC BLE は、32 ビットの ARM Cortex-M0 CPU を内蔵し、最大 48MHz の周波数で動作可能で、43-DMIPS 性能を実現します。Cortex-M0 CPU はシングル サイクルの 32 ビット乗算をサポートします。PRoC BLE は最大 32KB の SRAM および最大 256KB のフラッシュ メモリを内蔵しています。フラッシュ メモリは平均 85%のシングル サイクル SRAM アクセス性能を実現する読み出し加速装置を備えています。

12.2 BLE サブシステム

BLE サブシステムは BLE リンク層エンジンおよび物理層から成ります。リンク層エンジンはマスターとスレーブ両方のロールに対応しています。リンク層エンジンは、タイム クリティカルなデバイス検出と接続関連の手順、およびハードウェアでの AES 暗号化を実行して、消費電力を削減し、プロセッサ介入を最小限にしながら高性能を実現します。リンク層制御やホスト プロトコルなどの主なプロトコル要素はファームウェアで実装されます。ダイレクト テスト モード (DTM) は、標準 Bluetooth テスタを使用して無線性能をテストするために含まれています。物理層はモデムおよび、2.4GHz ISM の周波数帯と 1Mbps のデータ レートで BLE パケットを送受信する RF トランシーバーから成ります。送信では、このブロックが GFSK 変調を行った後、これらの BLE パケットのデジタル ベースバンド信号を無線周波数に変換しアンテナから送信します。受信では、このブロックが GFSK 復調を行った後、アンテナからの RF 信号をデジタル ビット ストリームに変換します。

RF トランシーバーはバランを内蔵し、整合回路を介して 50Ω アンテナを駆動するシングルエンド RF ポート ピンを備えています。出力電力は、消費電流を異なるアプリケーション毎に最適化するため、-18 dBm~+3dBm でプログラム可能です。

BLE プロトコル スタックは以下の機能を実現します。

- リンク層 (LL)
 - マスターとスレーブのロール
 - 128ビットのAES暗号化
 - 低デューティ比通知 (Bluetooth 4.1の機能)
 - 低エネルギーPing (Bluetooth 4.1の機能)
- 以下のものを備えているBLE 4.1シングル モード ホスト スタック
 - 論理リンク制御と適応プロトコル (L2CAP)、アトリビュート (ATT) およびセキュリティ マネージャ (SM) プロトコル
 - 一般アトリビュート プロファイル (GATT) と一般アクセス プロファイル (GAP)
- GATT、GAP、L2CAPへのAPIアクセス
- L2CAP接続向けチャネル (Bluetooth 4.1の機能)
- GAP機能
 - ブロードキャスター、オブザーバー、ペリフェラル、セントラルのロールおよび手順
 - セキュリティ モード1: レベル1、2、3
 - セキュリティ モード2: レベル1、2
 - ユーザー定義の通知データ
 - 複数の接続に対応
- GATT機能
 - GATTクライアントとサーバー
 - GATTサブプロシージャに対応
 - 32ビットUUID (Bluetooth 4.1の機能)
- セキュリティ
 - ペアリング方法: Just Works、Passkey Entry、Out of Band
 - 認証済みの中間者攻撃 (MITM) 保護とデータ署名
 - BLEプライバシーおよび署名データ
- すべてのSIG採用BLEプロファイルに対応

サイプレスが提供するソフトウェア (PSoC Creator) により、BLE ソリューションの設計は非常に簡単になります。この GUI ベースのコンフィギュレーション ツールを使用すれば、BLE プロトコル スタックのロール、機能およびパラメーターを構成できます。詳細については、BLE コンポーネント データシートをご参照ください。

12.3 プログラマブルなデジタル ペリフェラル

PRoC BLE は豊富なデジタルおよびアナログ ペリフェラルを提供します。デジタル ペリフェラルは SCB、TCPWM および I2S を含んでいます。アナログ ペリフェラルは高速 12 ビットの逐次比較レジスタ アナログ-デジタル コンバーター (SAR ADC)、静電容量タッチ センシング (CapSense)、およびセグメント LCD 直接駆動を含みます。

12.3.1 プログラマブルな SCB

PRoC BLE は 2 個の独立して実行時にプログラマブルな SCB を内蔵しています。これらのブロックは I²C、SPI または UART インターフェースを備えています。これらの SCB ブロックは以下の機能をサポートします。

- Motorola、Texas Instruments、National Semiconductor のプロトコルと互換性のある標準的な SPI マスターと SPI スレーブ機能
- SmartCard リーダー、ローカル相互接続ネットワーク (LIN)、および赤外線データ協会 (IrDA) プロトコルと互換性のある標準的な UART 機能
- 標準的な I²C マスターおよびスレーブ機能
- CPU の介入なしに動作が可能な SPI および I²C モード
- SPI および I²C プロトコル用の低電力 (ディープ スリープ) 動作モード (外部クロックを使用)

詳細については、[SCB コンポーネント データシート](#)をご参照ください。

12.3.2 プログラマブルな TCPWM/PWM

PRoC BLE はプログラマブルな 16 ビットの TCPWM ブロックを持っています。各 TCPWM は、16 ビット タイマー、カウンタ、PWM、または直交デコーダを実装できます。TCPWM は、相補出力および選択可能なスタート、リロード、ストップ、カウント、およびキャプチャ イベントの信号を提供します。PWM モードは中央揃え、エッジ、および擬似ランダム動作をサポートします。

詳細については、[TCPWM コンポーネント データシート](#)をご参照ください。

ファミリの一部のデバイスは TCPWM の他に、4 個の追加の PWM を備えています。PWM ペリフェラルは 8 ビットまたは 16 ビット分解能で設定できます。PWM には、ハードウェアで単一または連続タイミングと制御信号を生成するために比較出力があります。また CPU の介入を最小限にして複雑なリアルタイム イベントを正確に発生させる簡単な方法も提供します。最大 4 個の 8 ビット PWM または最大 2 個の 16 ビット PWM があります。

詳細については、[PWM コンポーネント データシート](#)をご参照ください。

12.3.3 IC 間サウンドバス ブロック

PRoC BLE はマスター モードで動作し、独立したデータ バイト ストリームを持つトランスミッター (TX) とレシーバー (RX) に対応する I2S ブロックを内蔵しています。このブロックはサンプル当たり 8~32 データ ビット、および 16、32、48 または 64 ビットワード選択期間をサポートします。データレートは最大 96kHz まで (64 ビットワード選択期間) です。独立型左右チャンネル FIFO またはインターリーブ型ステレオ FIFO が使用可能です。

詳細については、[I2S コンポーネント データシート](#)をご参照ください。

12.4 設定可能なアナログ ペリフェラル

12.4.1 ハードウェア シーケンサーを備えた SAR ADC

PRoC BLE は 12 ビット、1Msps の SAR ADC を内蔵しています。入力 チャンネルは、プログラマブルな分解能およびシングル エンドまたは差動入力オプションをサポートします。チャンネル数は使用可能な GPIO 数のみによって制限されます。

SAR ADC は、CPU の介入無しに最大 8 個のチャンネルで自動スキャンを実行できるハードウェア シーケンサーを備えています。SAR ADC は、またこれらの 8 個のチャン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。

詳細については、[逐次比較型コンポーネント データシート](#)をご参照ください。

12.4.2 静電容量タッチ センシング (CapSense)

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 センサーは機械式ボタンに比べると、審美的に優れており、使い易く、長寿命のものです。

PRoC BLE の CapSense 機能は高い信号対ノイズ比、クラス最高の耐水性を実現し、ボタン、スライダ、トラックパッドや近接センサーなどの多種多様なセンサーのタイプをサポートします。

ジェスチャーに対応した CapSense トラックパッドは以下の特長を持っています:

- 1 本指と 2 本指タッチのアプリケーションに対応
- 最大 35 個の X/Y センサー入力に対応
- ジェスチャー検出ライブラリを含む:
 - 1 本指タッチ: トレース、パン、クリック、ダブルクリック
 - 2 本指タッチ: パン、クリック、ズーム

サイプレスが提供するソフトウェア コンポーネントを使用すれば、静電容量センシングの設計が簡単になります。本コンポーネントは、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ます。

詳細については、[CapSense 設計ガイド](#)をご参照ください。

12.4.3 セグメント LCD 直接ドライバー

血糖値計、マルチメータやリモコンなどのほとんどの低消費電力のポータブルな携帯機器はセグメント LCD を使って情報を表示します。セグメント LCD はマイクロコントローラと接続するために外部ドライバーを必要とします。PRoC BLE はセグメント LCD ガラスを直接駆動できる低消費電力 LCD ドライバーを内蔵しています。

PRoC BLE は最大 4 つのコモンラインと 32 セグメント電極を持つ LCD を駆動できます。セグメント LCD ドライバーは、ディープスリープのシステム電力モードで最低 7μA のシステム電流消費量でスタティックディスプレイを保持できます。

詳細については、[AN87391 – PSoc 4 セグメント LCD 直接駆動](#)をご参照ください。

12.4.4 プログラマブル GPIO

PRoC BLE は最大 36 本のプログラマブルな 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。PRoC BLE の GPIO は多くの駆動モード、駆動能力およびスルーレートに対応します。

PRoC BLE は内部信号を GPIO に接続する多様な選択肢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簡単になります。

詳細については、[PRoC™ BLE テクニカル リファレンス マニュアル](#)をご参照ください。

12.5 システム全体のリソース

12.5.1 電源と電圧の監視

PRoC BLE は 1.9V~5.5V 電源電圧で動作できます。さまざまな電力モードをサポートするためのアクティブ レギュレータ、静音レギュレータ、ディープスリープレギュレータやハイバネートレギュレータなどの多数の内部レギュレータがあります。

PRoC BLE はパワーオンリセット (POR)、電圧低下検出 (BOD) および低電圧検出 (LVD) の 3 種類の電圧監視機能を持っています。

詳細については、[PRoC™ BLE テクニカル リファレンス マニュアル](#)をご参照ください。

12.5.2 クロック供給システム

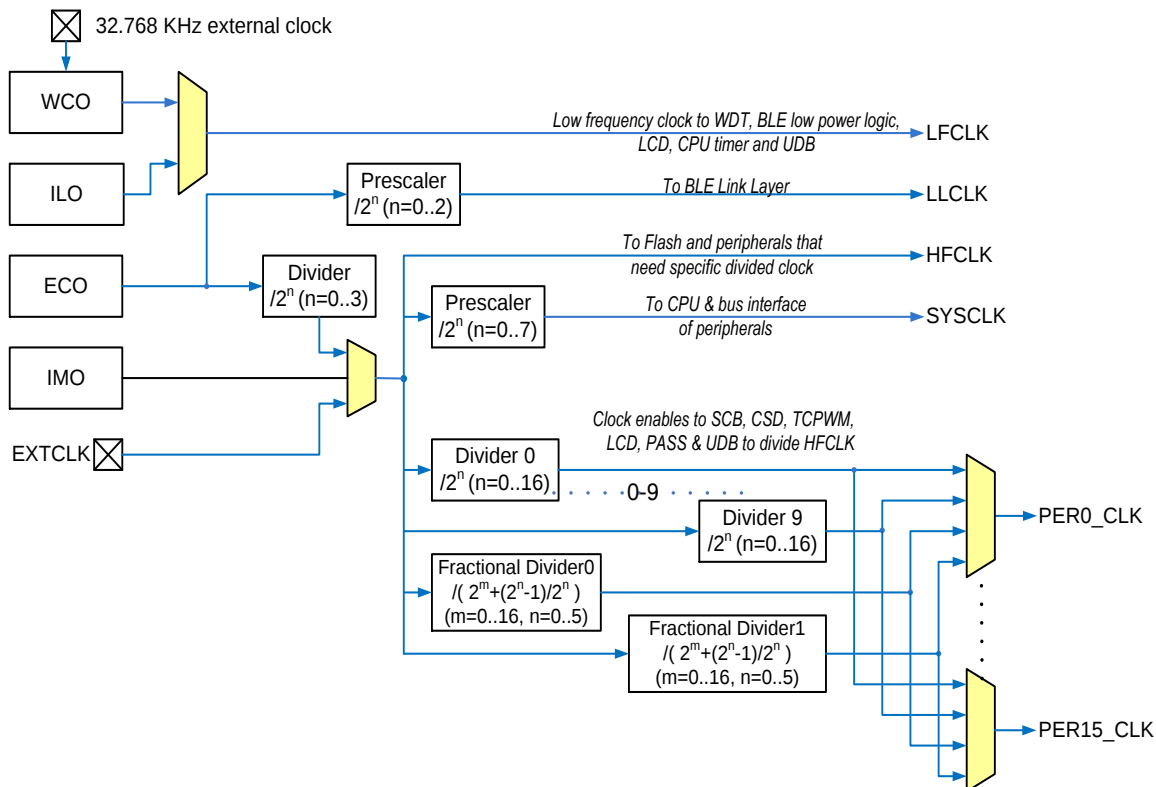
PRoC BLE は以下のクロック ソースを備えています。

- **内部主発振器 (IMO):** IMO は PRoC BLE の内部クロック供給の主なソースです。CPU およびすべての高速ペリフェラルは IMO からクロックで動作します。PRoC BLE は IMO から動作する多数のクロック分周器を含んでいます。これらの分周器は高速のペリフェラル用のクロックを生成します。
- **内部低速発振器 (ILO):** ILO は超低消費電力発振器であり、主にディープスリープ モードで動作する低速ペリフェラル用にクロックを生成します。
- **外部水晶発振器 (ECO):** ECO は Bluetooth 4.1 仕様で指定された±50ppm クロック精度要件を満たすために BLE サブシステムのアクティブ クロックとして使用されます。ECO は、実際のクロック周波数を測定することで水晶クロック周波数を調整するための調整可能な負荷コンデンサを備えています。高精度 ECO クロックをシステム クロックとしても使用できます。

- **時計用水晶発振器 (WCO):** WCO は Bluetooth 4.1 仕様で指定された±500ppm クロック精度要件を満たすために BLE サブシステムのスリープ クロックとして使用されます。スリープ クロックは正確なスリープ タイミングを提供し、指定された通知と接続間隔でのウェイクアップを可能にします。WCO およびファームウェアにより、正確なリアルタイム クロック (32.768kHz 水晶発振器の精度範囲以内) は実現できます。

図 66 に、PRoC BLE デバイスのクロッキング アーキテクチャを示しています。詳細については、PRoC™ BLE テクニカル リファレンス マニュアルをご参照ください。

図 66 PRoC BLE クロック供給システム



12.5.3 低消費電力モード

PRoC BLE は表 3 に示すように以下の 5 つの電力モードをサポートします。

注: これらのモードは PRoC BLE デバイスの電力モードであり、BLE サブシステムで記述される電力モードと異なります。

- **アクティブ モード:** これは主要な動作モードです。このモードでは、すべてのペリフェラルが有効です。
- **スリープ モード:** このモードでは、CPU がスリープ状態になり、SRAM が保持状態になり、すべてのペリフェラルが有効です。割り込みが発生すると、CPU がウェイクアップして、システムがアクティブ モードに戻ります。
- **ディープ スリープ モード:** このモードでは、高周波数クロック (IMO) およびすべての高速ペリフェラルが無効です。WDT、LCD、I²C/SPI、リンク層および低周波数クロック (32kHz ILO) は有効です。GPIO、WDT または SCB からの割り込みは、ディープスリープになったコンポーネントをウェイクアップさせることができます。このモードでは、PRoC BLE ファミリのすべてのデバイスの電流消費量は 1.3μA です。
- **ハイバネート モード:** この電力モードは、SRAM、プログラム可能な論理、および GPIO から生成された割り込みによりウェイクアップする機能を保持しながら、150nA というクラス最高の電流消費量を実現します。
- **ストップ モード:** この電力モードでは GPIO 状態が保持されます。固定の「WAKEUP」ピンからのウェイクアップは可能です。このモードでは、電流消費量は 60nA のみです。

表 3. PRoC BLE 低消費電力モード

BLESS モード	PRoC BLE システム電力モード				
	アクティブ	スリープ	ディープスリープ	ハイバネート	ストップ
送信	✓	✓	✗	✗	✗
受信	✓	✓	✗	✗	✗
アイドル	✓	✓	✗	✗	✗
スリープ	✓	✓	✗	✗	✗
ディープスリープ	✓	✓	✓	✗	✗
パワー	✗	✗	✗	✓	✓

12.5.4 デバイス セキュリティ

PRoC BLE はフラッシュ メモリを不正アクセスあるいはコピーから保護するオプションをいくつか提供します。フラッシュの各行に 1 つの保護ビットがあります。これらのビットは監視フラッシュ行に格納されています。

12.6 プログラマブル GPIO

I/O システムは CPU やペリフェラルと外部間のインターフェースを提供します。PRoC BLE は最大 36 本のプログラマブルな 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。PRoC BLE の GPIO は多くの駆動モード、駆動能力およびスルーレートに対応します。

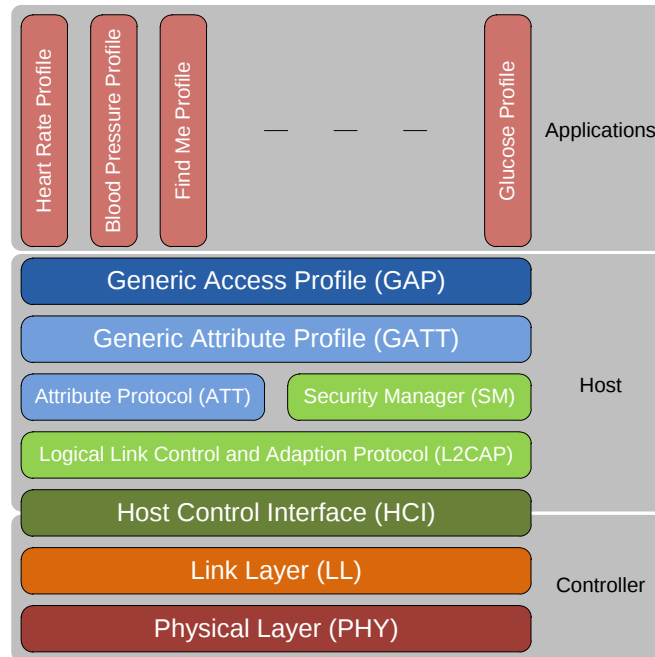
PRoC BLE は内部信号を GPIO に接続する多様な選択肢を与えるインテリジェントな配線システムを提供します。この柔軟な配線により、回路設計と基板レイアウトが簡単になります。

13 付録 E: BLE プロトコル

13.1 概要

BLE (Bluetooth Smart とも呼ばれている) は 2.4GHz ISM 帯で動作する低消費電力の無線標準として Bluetooth SIG により策定されました。図 67 に BLE スタックを示しています。

図 67. BLE アーキテクチャ



BLE スタックは次の 3 グループに分けられます。

- **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。信号の受信時にこの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、アトリビュート 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使用例です。

下記の節は、一般的な心拍数監視サービスと電池通知サービスを例として取り上げて BLE スタックの複数の層の概要について説明します。BLE アーキテクチャの詳細説明については、[Bluetooth Developer](#) のサイトに搭載されている Bluetooth 4.1 仕様または学習ビデオをご覧ください。

13.2 物理層 (Physical Layer; PHY)

この層は、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 1Mbps でデジタル データを送受信します。この BLE 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 個で宣伝チャネルは 3 個です。

13.3 リンク層 (Link Layer; LL)

この層では、(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 BLE 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。

- アドバタイジング (宣伝)、スキャンニング、接続の作成および維持: 物理リンクを確立できます。
- 24 ビット CRC および 128 ビット AES 暗号化: データ交換時の堅牢性と安定性を確保できます。
- 高速接続の確立および低デューティ比の宣伝: 低消費電力動作を実現できます。
- AFH (Adaptive Frequency Hopping): 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できます。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に設定する例です。
- **スレーブ**: 心拍数監視機器はリンク層をスレーブに設定する例です。

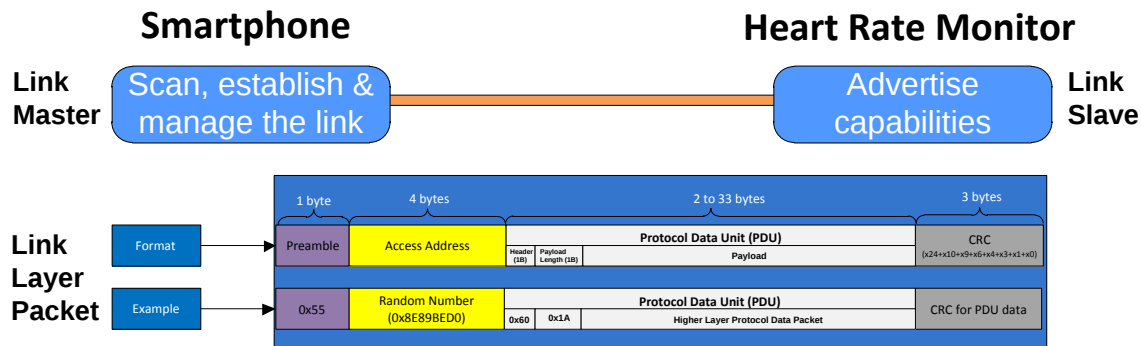
PSoC/PRoC BLE デバイスは両方の設定で動作できます。

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宣伝するものです。リンク層マスターは宣伝パケットを受信し、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 68 をご覧ください)。この心拍数監視アプリケーションの実装例では、心拍数監視機器はスレーブとして動作し、データをマスターとしてのスマートフォンに送信します。スマートフォンのアプリケーションはその後スマートフォン上に読み出し結果を表示します。

PSoC/PRoC BLE デバイスは、宣伝、CRC や AES 暗号化方式などタイム クリティカルでプロセッサが集中するリンク層の機能をハードウェアで実装します。宣伝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 68 に、BLE のリンク層パケットの構造およびリンク層パケット内の各フィールドのサイズを示しています。リンク層パケットはすべての上位層のデータを自体のペイロード フィールドに格納します。物理リンク上の通信を識別し同じ RF チャネルで動作する近隣 BLE 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 アドレスを用意します。24 ビット CRC によりデータの堅牢性を確保できます。

図 68. BLE リンク層プロトコル



13.4 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として定義された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 に物理的転送方法を整備する必要があります。

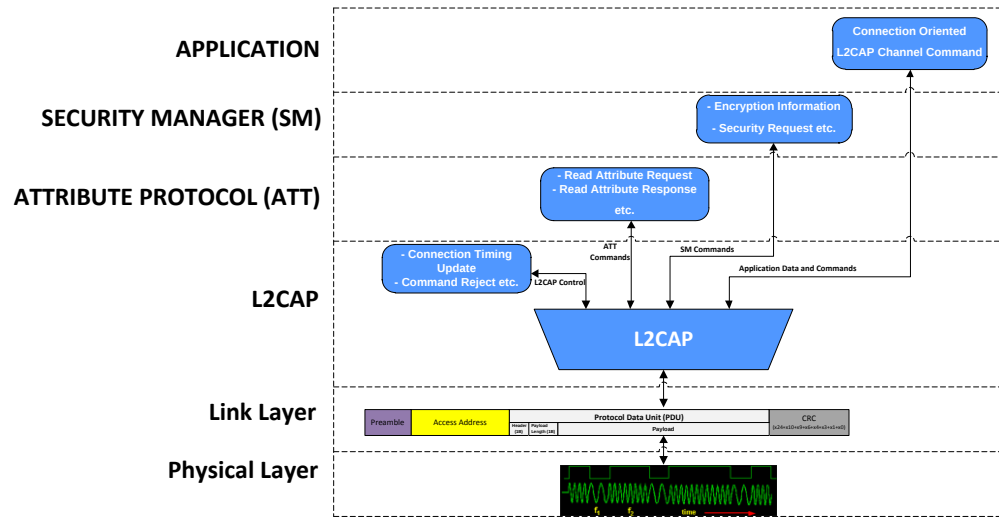
PSoC/PRoC BLE デバイスでは、HCI が単なるファームウェア プロトコル層で、メッセージとイベントをコントローラーとホスト間に転送するだけです。

13.5 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、再構成サービスを上位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多くのより小さいパケットに分けますが、再構成サービスでは、リンク層から受信した分割済みより小さいパケットを組み合わせて有意義のパケットを構成します。L2CAP 層は、図 69 のように、**アトリビュート プロトコル (Attribute Protocol; ATT)**、**セキュリティ マネージャ (Security Manager; SM)**、および L2CAP 自体の制御用に 3 つのプロトコル チャネル ID を提供します。Bluetooth 4.1 仕様により、これらのプロトコル チャネルの最上位にある L2CAP 接続指向チャネルから直接接続データ チャネルを作成できます。

PSoC/PRoC BLE では L2CAP とそれにある各層がファームウェアで実装されます。

図 69. BLE L2CAP の層



13.6 セキュリティ マネージャ (Security Manager; SM)

SM 層はペアリング、暗号化、および鍵の配布用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 チャンネル上でスパイにより探られることなく、BLE インターフェースを介してデータが安全に交換できるようになります。
- **ボンディング**は、ペアリング プロセスで交換された鍵と識別情報が保存されるプロセスです。デバイスが一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がありません。

BLE はデータ暗号化に 128 ビット AES 方式を用いています。

13.7 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:** データや情報を格納します。GATT クライアントから要求を受信しデータで返答します。例えば、心拍数監視の GATT サーバー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 サーバーからデータを要求／受信します。例えば、スマートフォンは 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 BLE 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 BLE 通信の基盤となります。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アトリビュートを検索してアクセスすることが可能になります。GATT クライアントと GATT サーバーのアーキテクチャの詳細については、[一般的アトリビュート プロファイル \(Generic Attribute Profile; GATT\)](#) をご参照ください。

アトリビュートは次の内容を含む ATT/GATT 層の基本的なデータ コンテナです。

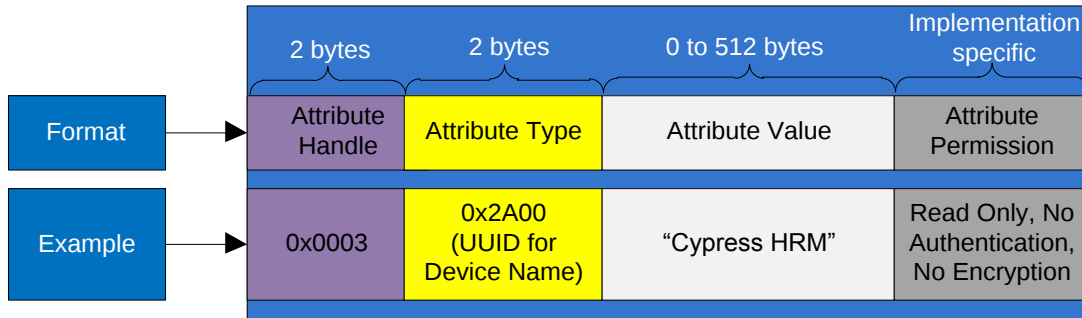
- **アトリビュート ハンドル:** 1 つのアトリビュートをアドレス指定してそれにアクセスする 16 ビット アドレスです。
- **アトリビュート タイプ:** アトリビュートに格納されるデータの種類を指定します。Bluetooth SIG が定義した 16 ビット UUID で表示されます。

例えば、心拍数管理サービスの 16 ビット UUID は 0x180D で、デバイス名アトリビュートの UUID は 0x2A00 です。SIG が指定した 16 ビット UUID の一覧については Bluetooth の[ホームページ](#)をアクセスしてご参照ください。

- **アトリビュート バリュー:** アトリビュートに格納される実際の値です。
- **アトリビュート パーミッション:** アトリビュート アクセス、認証、および権限の要件を指定します。これはより上位層の仕様により設定され、アトリビュート プロトコルを通してアクセスできません。

図 70 にデバイス名アトリビュートの構造を例として取り上げます。

図 70. アトリビュート フォーマットの例



13.7.1 アトリビュート階層

アトリビュートはデータを表示する ATT/GATT 層での構成要素です。アトリビュートは大きく次の 2 グループに分けられ、データ階層とデータ抽象化を提供します。

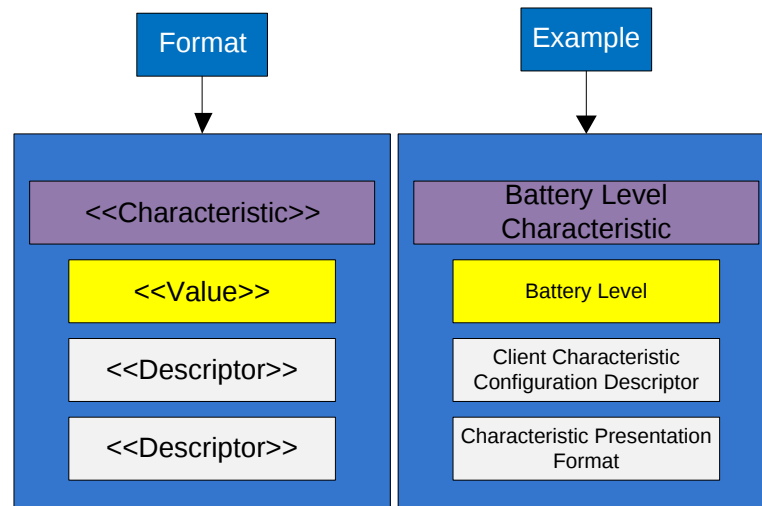
- **キャラクターリスティック:** アトリビュートのコレクションであり、システム情報や有意義のデータを表します。1 つのキャラクターリスティックは次のアトリビュートからなります。
 - **キャラクターリスティック宣言アトリビュート:** キャラクターリスティックの開始を定義します。
 - **キャラクターリスティック バリュー アトリビュート:** 実際のデータを格納します。
 - **キャラクターリスティック ディスクリプタ アトリビュート:** これらは任意のアトリビュートであり、キャラクターリスティック値の追加情報を提供します。

「電池レベル」は電池通知サービス (BAS) のキャラクターリスティックの 1 つの例です。電池レベルをパーセント値で表示するのはキャラクターリスティック ディスクリプタの 1 つの例です。

図 71 に例として電池レベルのキャラクターリスティックの構造を示しています。

- キャラクターリスティックの最初の部分はそのキャラクターリスティックの宣言 (キャラクターリスティックの開始を示す) であり、図 71 の例に「電池レベル キャラクターリスティック」として表示されています。
- 次は実際のキャラクターリスティック値または実際の値であり、例の電池レベル キャラクターリスティックでは当時の電池レベルです。電池レベルは「65」、「90」など全測定範囲に対する割合 (パーセント) で表示されます。
- キャラクターリスティック ディスクリプタはキャラクターリスティックの値の意味を解明するのに必要な追加情報を提供します。例えば、電池レベルの表示形式のキャラクターリスティック ディスクリプタは電池レベルがパーセントで表示されることを示します。よって、「90」が読み出されると、GATT クライアントは 90mV や 90mAh ではなく 90%と理解します。同様に、(図 71 に表示されない) 有効範囲のキャラクターリスティック ディスクリプタは電池レベルの範囲が 0~100 パーセントであることを示します。
- クライアント キャラクターリスティック コンフィギュレーション ディスクリプタ (CCCD) は別の一般的なキャラクターリスティック ディスクリプタであり、GATT クライアントによる GATT サーバー上のキャラクターリスティックの動作設定を可能にするものです。GATT クライアントが 0x01 の値を任意のキャラクターリスティックの CCCD に書き込むと、(後節で説明するように) GATT サーバーからの通知が非同期に送信されます。電池レベル キャラクターリスティックの場合は、電池レベル CCCD に 0x01 の値を書き込むと、電池通知サービスが自体の電池レベルを定期的または電池レベルの変更時に通知することが可能になります。

図 71. キャラクタースティックのフォーマットと例

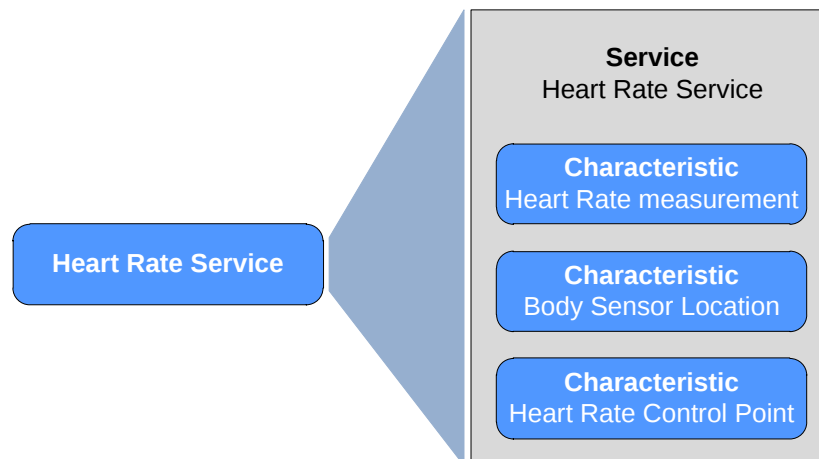


- **サービス:** GATT サーバーが実行する機能を定義するアトリビュートの一種です。サービスはキャラクタースティックのコレクションであり、他のサービスを含むこともあります。サービスの概念は関連するデータ グループを設立し、データ階層を提供するのに使用されます。心拍数管理サービス (HRS) の例は図 72 をご覧ください。

サービスは一次サービスと二次サービスの 2 種類に分けられています。一次サービスはデバイスの主な機能を提供し、二次サービスは追加機能を提供します。例えば、心拍数監視デバイスの場合には、HRS は一次サービスで、BAS は二次サービスです。

サービスは GATT サーバー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。

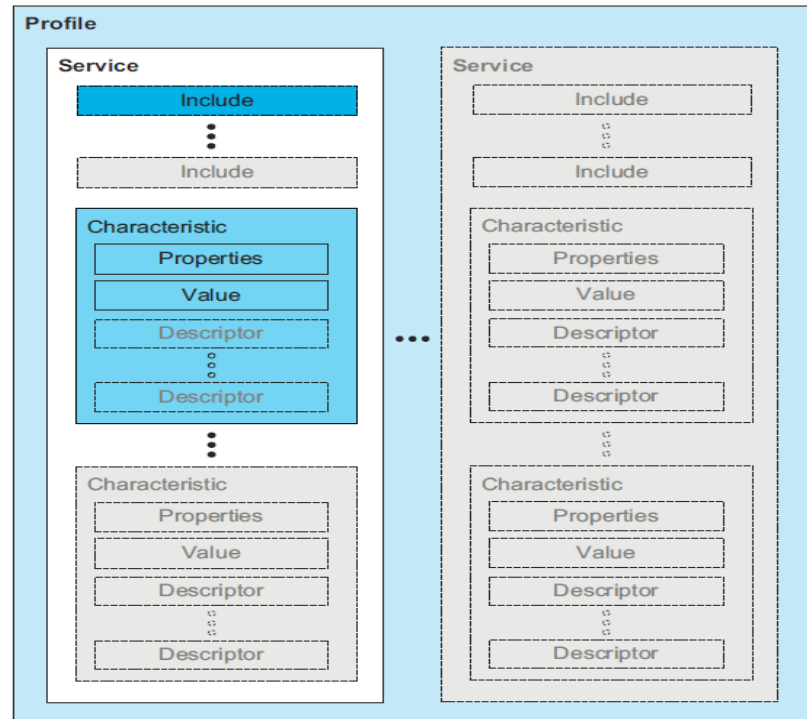
図 72. BLE 心拍数サービスの例



BLE の「プロファイル」という言葉は特定のエンド アプリケーションを共に実現するサービスとその動作のコレクションです。心拍数プロファイル (HRP) は BLE プロファイルの例で、心拍数監視デバイスの作成に必要なすべてのサービスを定義します。詳細については[一般的アクセス プロファイル \(Generic Access Profile; GAP\)](#) 節をご覧ください。

図 73 には、本節の上記に定義されたアトリビュート、キャラクタースティック、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ています。

図 73. BLE データ階層*



*Bluetooth SIG からの無料イメージ

13.7.2 アトリビュート動作

前節で定義したアトリビュートは次の 5 つの基本的な方法でアクセスされます。

- **読み出し要求:** GATT クライアントはアトリビュート バリューの読み出しのためにこの要求を GATT サーバーに送信します。各要求ごとに GATT サーバーは GATT クライアントに応答を送信します。この読み出し要求の例は、スマートフォンが心拍数監視デバイスの電池レベル キャラクタリスティックを読み出すことです (図 71 をご覧ください)。
- **書き込み要求:** GATT クライアントはアトリビュート バリューの書き込みのためにこの要求を GATT サーバーに送信します。GATT サーバーはその値が書き込んだか否かを GATT クライアントに返答します。書き込み要求の例はスマートフォンが 0x01 の値を電池レベル キャラクタリスティック CCCD に書き込んで通知を有効にすることです。
- **書き込みコマンド:** GATT クライアントはアトリビュート バリュー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。このコマンドに対して GATT サーバーは何の返答も送信しません。例えば、BLE 即時アラート サービス (IAS) は (スマートフォンなどの) IAS 検知器から (LED をオンにする、ブザーを鳴らす、振動モーターを駆動することなどの) アラートを (BLE キーフォブなどの) 対象の IAS デバイス上でトリガーするために書き込みコマンドを使用します。
- **通知:** GATT サーバーはあるアトリビュートの新しい値を通知するためにこの通知を GATT クライアントに送信します。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、心拍数監視デバイスはその 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 サーバーはこの種類のメッセージを送信します。GATT クライアントは常にその確認を行います。例えば、BLE 健康体温計サービス (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。

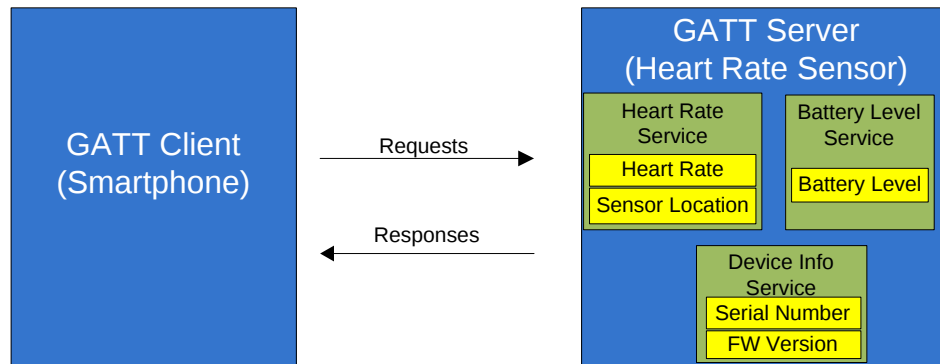
13.8 一般的アトリビュート プロファイル (Generic Attribute Profile; GATT)

GATT はアトリビュートの検出と使用方法を定義します。GATT は次の 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 (例えば、スマートフォン) です。
- **GATT サーバー:** データを提供するデバイス (例えば、心拍数モニター) です。

図 74 には心拍数監視デバイスを例として GATT 層内のクライアントとサーバーのアーキテクチャを示しています。心拍数監視デバイスは多くのサービス (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 71 に示すようにキャラクタリスティック値とキャラクタリスティック ディスクリプタを含む 1 つ以上のキャラクタリスティックを持っています。

図 74. GATT クライアントとサーバーのアーキテクチャ



BLE 接続がリンク層レベルで確立された後、(接続された BLE デバイスについて最初に何も分からない) GATT クライアントは「サービス発見」というプロセスを開始します。サービス発見の一部として、GATT サーバーでの全ての有効なサービス、キャラクタリスティック、アトリビュート一覧を得るために GATT クライアントは GATT サーバー側に多くの要求を送信します。サービス発見が完了した後、GATT クライアントは、前述したアトリビュート動作を利用することで 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得ました。

13.9 一般的アクセス 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の情報、および発見／接続／接着 (ボンディング) 方法を提供します。このプロファイルは、デバイスの発見／接続方法、使用可能なサービス一覧、およびサービスの使用方法を定義します。図 76 に心拍数プロファイルの例を示します。

GAP 層は次の 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宣伝のロールです。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、遠隔装置に測定した心拍数を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:** 宣伝をスキャンして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) から心拍数計測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宣伝ロールです。BLE 接続を確立できず、データ交換にも関与しません (要求／応答動作がありません)。このロールは、聞き手が有無かを問わ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る一方データ通信です。何の応答も求めなく情報を連続して放送するビーコンは GAP ブロードキャスターの典型的な例です。
- **オブザーバー:** 宣伝をスキャンしますが宣伝デバイスとの接続を実現しない聞き取りロールです。これはブロードキャスター ロールの逆のロールです。情報を連続して聞き取るが、情報源に通信できないラジオ受信機と同様に動作します。ビーコンからの信号を連続的に聞き取るスマートフォン アプリケーションは GAP オブザーバーの典型的な例として取り上げられます。

図 75 に、サイプレスの BLE Pioneer Kit がペリフェラルでスマートフォンがセントラル デバイスである一般的な BLE システムを示します。また、BLE プロトコルの層間の相互作用、およびセントラルとペリフェラル デバイス上に該当するロールも示しています。

図 75. BLE システムの設計

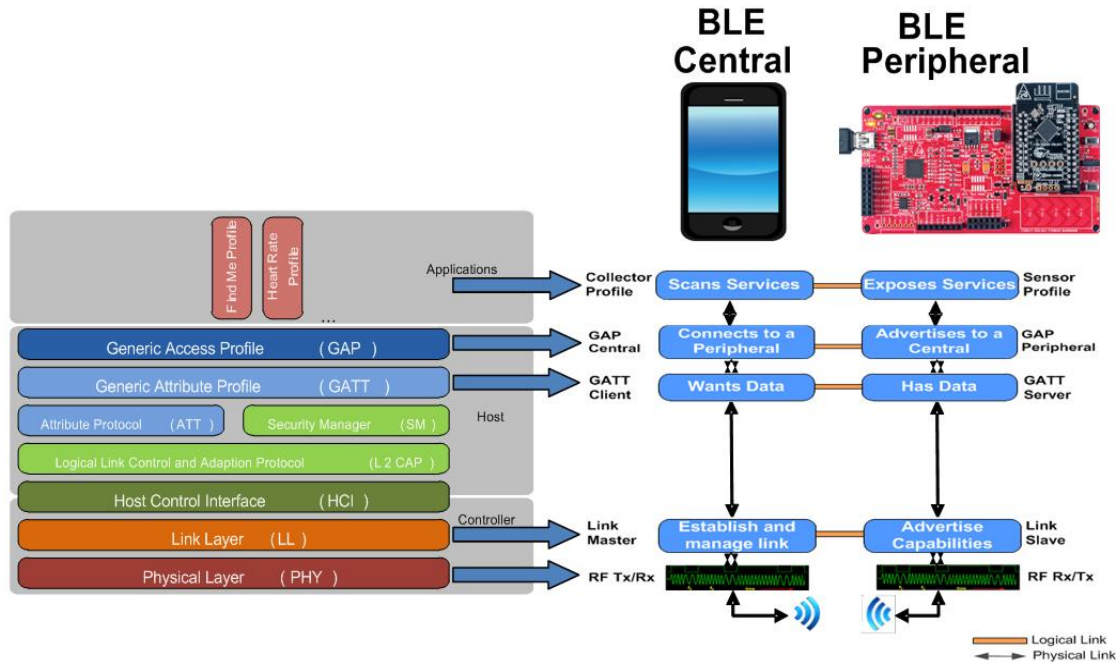


図 76 に、心拍数アプリケーションを内蔵するスマートフォンがセントラルとして動作し、心拍数センサーがペリフェラルとして動作する例を示します。心拍数監視デバイスは心拍数センサー プロファイルを実装し、データを受信するスマートフォンは心拍数コレクター プロファイルを実装します。

この例には、心拍数センサー プロファイルは次の 2 つの標準サービスを実装します。一つ目は心拍数サービスで、3 つのキャラクターリスティック (心拍数計測キャラクターリスティック、身体センサー位置キャラクターリスティック、および心拍数制御点キャラクターリスティック) からなります。第 2 のサービスはデバイス情報サービスです。リンク層では、心拍数計測デバイスがスレーブで、スマートフォンがマスターです。心拍数のサービスとプロファイルの詳細については Bluetooth developer portal サイトにアクセスしてご覧ください。

図 76. BLE 心拍数監視システム



改訂履歴

文書名: AN94020 –PRoC™ BLE 入門

文書番号: 002-03261

版	ECN	変更者	発行日	変更内容
**	4955078	HZEN	10/09/2015	これは英語版 001-94020 Rev. *B を翻訳した日本語版 002-03261 Rev. **です。
*A	5705729	AESATP12	04/25/2017	Updated logo and copyright.

ワールドワイド販売と設計サポート

サイプレスは、事業所、ソリューション センター、メーカー代理店、および販売代理店の世界的なネットワークを保持しています。お客様の最寄りのオフィスについては、[サイプレスのロケーション ページ](#)をご覧ください。

製品

ARM® Cortex® Microcontrollers

cypress.com/arm

車載用

cypress.com/automotive

クロック&パッファ

cypress.com/clocks

インターフェース

cypress.com/interface

IoT (モノのインターネット)

cypress.com/iot

メモリ

cypress.com/memory

マイクロコントローラ

cypress.com/mcu

PSoC

cypress.com/psoc

電源用 IC

cypress.com/pmic

タッチ センシング

cypress.com/touch

USB コントローラー

cypress.com/usb

ワイヤレス/RF

cypress.com/wireless

PSoC® ソリューション

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6

サイプレス開発者コミュニティ

[フォーラム](#) | [WICED IOT Forums](#) | [Projects](#) | [ビデオ](#) | [ブログ](#) | [トレーニング](#) | [Components](#)

テクニカルサポート

cypress.com/support

All other trademarks or registered trademarks referenced herein are the property of their respective owners.



Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

© Cypress Semiconductor Corporation, 2015-2017. 本書面は、Cypress Semiconductor Corporation 及び Spansion LLC を含むその子会社（以下、「Cypress」という。）に帰属する財産である。本書面（本書面に含まれ又は言及されているあらゆるソフトウェア又はファームウェア（以下、「本ソフトウェア」という。）を含む）は、アメリカ合衆国及び世界のその他の国における知的財産法令及び条約に基づき、Cypress が所有する。Cypress はこれらの法令及び条約に基づく全ての権利を留保し、また、本段落で特に記載されているものを除き、Cypress の特許権、著作権、商標権又はその他の知的財産権のライセンスを一切許諾していない。本ソフトウェアにライセンス契約書が伴っておらず、かつ、あなたが Cypress との間で別途本ソフトウェアの使用方法を定める書面による合意をしていない場合、Cypress は、あなたに対して、（1）本ソフトウェアの著作権に基づき、（a）ソースコード形式で提供されている本ソフトウェアについて、Cypress ハードウェア製品と共に用いるためにのみ、組織内部でのみ、本ソフトウェアの修正及び複製を行うこと、並びに（b）Cypress のハードウェア製品ユニットに用いるためにのみ、（直接又は再販売者及び販売代理店を介して間接のいずれかで）エンドユーザーに対して、バイナリーコード形式で本ソフトウェアを外部に配布すること、並びに（2）本ソフトウェア（Cypress により提供され、修正がなされていないもの）に抵触する Cypress の特許権のクレームに基づき、Cypress ハードウェア製品と共に用いるためにのみ、本ソフトウェアの作成、利用、配布及び輸入を行うことについての非独占的で譲渡不能な一身専属的ライセンス（サブライセンスの権利を除く）を付与する。本ソフトウェアのその他の使用、複製、修正、変換又はコンパイルを禁止する。

適用される法律により許される範囲内で、Cypress は、本書面又はいかなる本ソフトウェアに関しても、明示又は黙示をとわず、いかなる保証（商品性及び特定の目的への適合性の黙示の保証を含むがこれらに限られない）も行わない。適用される法律により許される範囲内で、Cypress は、別途通知することなく、本書面を変更する権利を留保する。Cypress は、本書面に記載のあるいかなる製品又は回路の適用又は使用から生じる一切の責任を負わない。本書面で提供されたあらゆる情報（あらゆるサンプルデザイン情報又はプログラムコードを含む）は、参照目的のためのみに提供されたものである。この情報で構成するあらゆるアプリケーション及びその結果としてのあらゆる製品の機能性及び安全性を適切に設計し、プログラムし、かつテストすることは、本書面のユーザーの責任において行われるものとする。Cypress 製品は、兵器、兵器システム、原子力施設、生命維持装置若しくは生命維持システム、蘇生用の設備及び外科的移植を含むその他の医療機器若しくは医療システム、汚染管理若しくは有害物質管理の運用のために設計され若しくは意図されたシステムの重要な構成部分として用いるため、又はシステムの不具合が人身傷害、死亡若しくは物的損害を生じさせることになるその他の使用（以下、「本目的外使用」という。）のためには、設計、意図又は承認されていない。重要な構成部分とは、装置又はシステムのその構成部分の不具合が、その装置若しくはシステムの不具合を生じさせるか又はその安全性若しくは実効性に影響すると合理的に予想できる、機器又はシステムのあらゆる構成部分をいう。Cypress 製品のあらゆる本目的外使用から生じ、若しくは本目的外使用に関連するいかなる請求、損害又はその他の責任についても、Cypress はその全部又は一部をとわず一切の責任を負わず、かつ、あなたは Cypress をそれら一切から免除するものとし、本書により免除する。あなたは、Cypress 製品の本目的外使用から生じ又は本目的外使用に関連するあらゆる請求、費用、損害及びその他の責任（人身傷害又は死亡に基づく請求を含む）から Cypress を免責補償する。Cypress、Cypress のロゴ、Spansion、Spansion のロゴ及びこれらの組み合わせ、WICED、PSoC、CapsSense、EZ-USB、F-RAM、及び Traveo は、米国及びその他の国における Cypress の商標又は登録商標である。Cypress の商標のより完全なリストは、cypress.com を参照のこと。その他の名称及びブランドは、それぞれの権利者の財産として権利主張がなされている可能性がある。