サイプレスはインフィニオン テクノロジーズになりました

この表紙に続く文書には「サイプレス」と表記されていますが、これは同社が最初にこの製品を開発したからです。新規および既存のお客様いずれに対しても、引き続きインフィニオンがラインアップの一部として当該製品をご提供いたします。

文書の内容の継続性

下記製品がインフィニオンの製品ラインアップの一部として提供されたとしても、それを理由としてこの文書に変更が加わることはありません。今後も適宜改訂は行いますが、変更があった場合は文書の履歴ページでお知らせします。

注文時の部品番号の継続性

インフィニオンは既存の部品番号を引き続きサポートします。ご注文の際は、データシート記載の注文部品番号をこれまで通りご利用下さい。

AN91184

PSoC® 4 BLE - BLE アプリケーションの設計

著者: Uday Agarwal, Ajay Sahu

関連プロジェクト: あり

関連製品ファミリ: CY8C4XX7-BL、CY8C4XX8-BL、CYBL1XX6X、CYBL1XX7X

ソフトウェア バージョン: PSoC Creator™ 4.2

関連アプリケーションノート: 関連文書を参照してください

本アプリケーションノートの最新版を入手するには、http://www.cypress.com/go/AN91184 へアクセスしてください。

その他のサンプルコードが必要な場合は、以下を参照してください。

PSoC のサンプルコード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 ライブラリはここからご覧ください。

本資料 (AN91184) は PSoC Creator の BLE コンポーネントに含まれている Bluetooth® SIG による定義済みの標準プロファイルを使用して、PSoC4 BLE に基づいて Bluetooth 低エネルギー (BLE) アプリケーションを設計する方法について説明します。同資料は CY8CKIT-042-BLE キット上に BLE 健康温度計プロファイルを使用してアプリケーションを構築する方法を示します。

目次

1 はじめに 1
2 PSoC リソース 2
3 PSoC Creator 3
4 カスタム サービスおよび標準サービス 4
4.1 BLE 健康体温計 4
5 PSoC Creator プロジェクト: 健康温度計 5
5.1 コンポーネントを設定 5
5.2 ファームウェアを設定 18
5.3 ハードウェア構成 23
5.4 デバイスをビルドしてプログラム 24
6 アプリケーションのテスト 26
6.1 CySmart セントラル エミュレーション ツール 26
6.2 CySmart モバイル アプリケーション 29
6.3 まとめ 32
7 関連文書 33
付録 A. サンプルコード 34
改訂履歴 38
セールス, ソリューションおよび法律情報 39

1 はじめに

Bluetooth Low Energy (BLE) は、短距離通信用に Bluetooth Special Interest Group (SIG) によって策定された超低消費電力無線規格です。BLE 物理層、プロトコルスタック、およびプロファイルアーキテクチャは、消費電力を最小限に抑えるように設計および最適化されています。BLE は Classic Bluetooth と同様に 2.4GHz の ISM 周波数帯を利用しますが、帯域幅は 125 kbps～2Mbps です。

サイプレスの PSoC 4 BLE は、BLE、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 1 つのチップに搭載したプログラマブル組込みシステム オン チップ (SoC) です。

本アプリケーションノートでは、PSoC Creator の BLE コンポーネントを使って、健康温度計プロファイルで BLE 健康温度計のアプリケーションを設計してから、CySmart セントラル エミュレーション ツールおよび CySmart モバイル アプリケーションを使用して、設計したアプリケーションを検証する方法について説明します。PSoC Creator の

BLE コンポーネントは事前に構築された標準プロファイルがあります。これにより、BLE 対応のプロジェクトでこれらのサービスを使用するのは非常に簡単です。

本文書は、ユーザーが BLE、PSoC、PSoC Creator IDE、およびサーミスタを使った温度測定の基本に精通していることを前提としています。以下のリンクを参照してください。

- AN91267 - PSoC® 4 BLE 入門
- PSoC Creator のホームページ
- AN66477 – PSoC® 3, PSoC 4, and PSoC 5LP – Temperature Measurement with a Thermistor

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 PSoC デバイスを選択し、迅速かつ効率的にデザインに統合する手助けをしています。リソースの包括的なリストについては、「KBA86521, How to Design with PSoC 3, PSoC 4, and PSoC 5LP」を参照してください。

以下は PSoC 4 BLE のリソースの要約です。

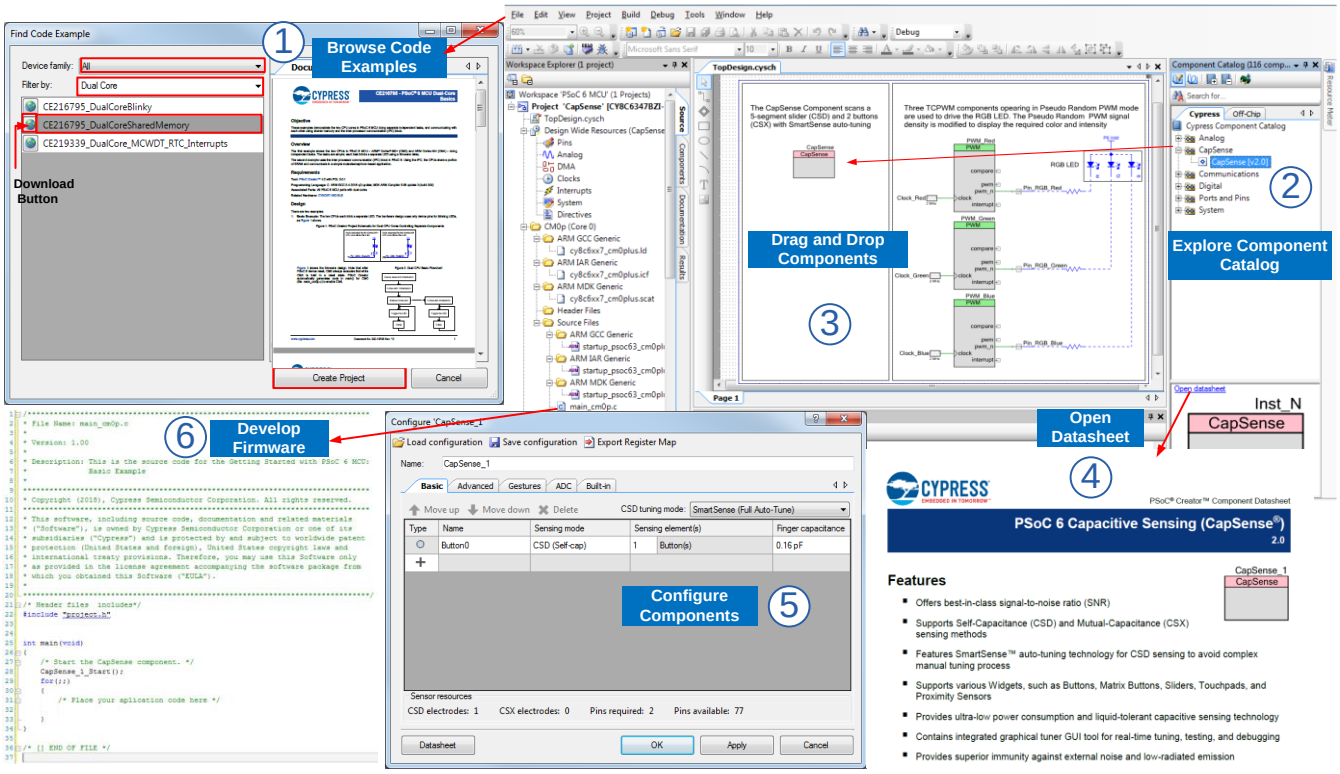
- **概要:** PSoC ポートフォリオ, PSoC ロードマップ
- **製品セレクタ:** PSoC 1, PSoC 3, PSoC 4, or PSoC 5LP。また PSoC Creator 内にデバイス選択ツールがあります。
- **データシート**: PSoC 41XX-BL および PSoC 42XX-BL デバイス ファミリ用の電気的仕様を提供し、説明します。
- **アプリケーションノートおよびサンプルコード**: 基本レベルから高度なレベルまでの幅広いトピックを提供します。PSoC Creator は追加のサンプルコードをご提供します。
- **テクニカル リファレンス マニュアル (TRM)**: 各 PSoC 4 BLE 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提供します。
- **CapSense デザイン ガイド**: PSoC 4 BLE ファミリ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 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ています。
  - Windows、iOS、Android 向けの CySmart BLE ホストエミュレーションツールは、BLE ペリフェラルアプリケーションのテストとデバッグを可能にする使いやすい GUI です。CySmart モバイルアプリのソースコードはサイプレスの Web サイトでも入手できます。

3 PSoC Creator

PSoC Creator は無料の Windows ベースの統合開発環境 (IDE) です。このキットにより、PSoC 3、PSoC 4、PSoC 5LP、および PSoC 6 ベースのシステムの同時ハードウェアとファームウェア設計が可能です。PSoC Creator により、以下のことが可能です。

1. **File** > **Code Example** メニューからサンプルコード一覧を参照
2. 100 以上のコンポーネントを含むライブラリを利用
3. 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
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構成
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 1. PSoC Creator の特長



4 カスタム サービスおよび標準サービス

Bluetooth SIG は、GATT クライアントか GATT サーバーとして構成できる一連のサービスを定義します。これらのサービスは標準サービスと呼ばれています。標準サービスの例として心拍数サービス、健康温度計サービス、血圧サービス、即時アラート サービスなどです。標準サービスの完全なリストについては Bluetooth Developer Portal を参照してください。

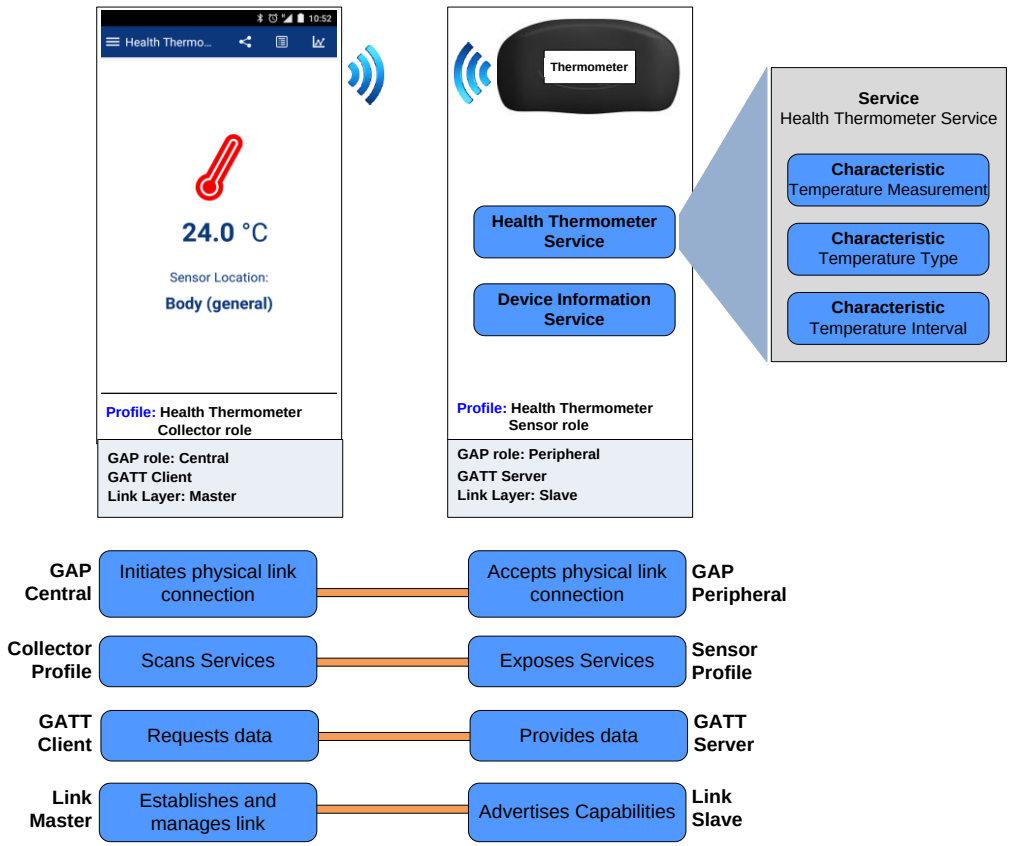
これらの標準サービスは、各種のアプリケーションに対応するように定義されています。例えば、心拍数サービスがリストバンドまたはチェスト ストラップ モニター内の心拍数センサーからのデータを報告するように構成することができます。また、特定の時間間隔で消費したエネルギー量も報告できます。

BLE 規格でユーザーはカスタム サービスという独自のサービスを作成することもできます。名前が示すように、それらは BLE 標準サービスでカバーされないサービスを定義するために使用されます。これらのサービスは、カスタムアプリケーションを持つことができる BLE デバイスを展開できるため、同様に重要です。

4.1 BLE 健康体温計

BLE 健康温度計のアプリケーション (図 2) では、温度計デバイスは、GAP ペリフェラルとして動作し、健康温度計センサー プロファイルを実装しています。一方、データを受信するモバイル デバイスは GAP セントラルとして動作し、健康温度計コレクター プロファイルを実装しています。この例では、健康温度計センサー プロファイルは、2 つの標準サービスを実装しています: 3 つの特性 (温度測定特性、温度タイプ特性、および測定間隔特性) を含む健康温度計サービスと本書で後述する 9 つの特性を含むデバイス情報サービスです。

図 2. BLE システムの設計



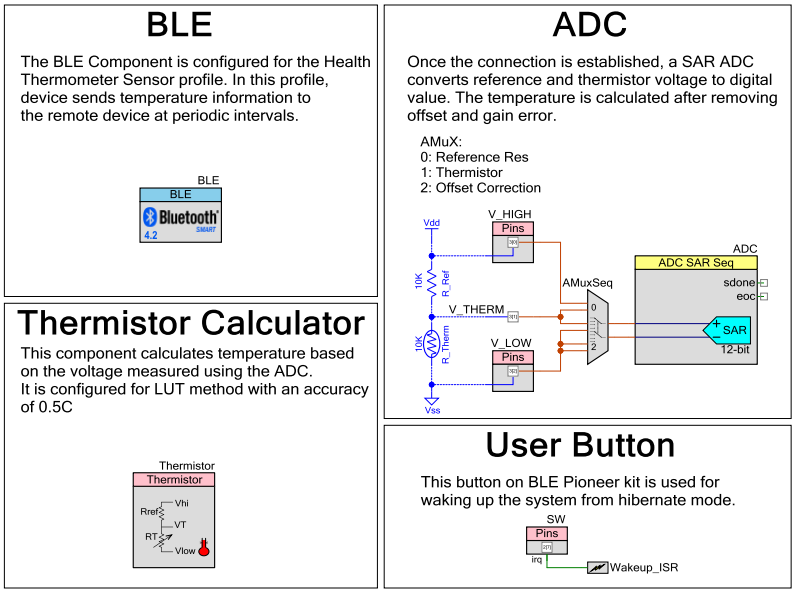
5 PSoC Creator プロジェクト: 健康温度計

このプロジェクトでは、PSoC4 BLE デバイスに統合されたものは以下通りです。

- GAP 層でペリフェラルとして、また GATT 層で GATT サーバーとして動作する BLE コンポーネント
- サーミスタの両端電圧を測定する ADC
- ADC の読み出し値を使用して温度を計算するサーミスタ計算機
- システムをハイバネート モードからウェイク アップさせるユーザー ボタン

図 3 に、健康温度計プロジェクトの PSoC Creator の回路図を示しています。

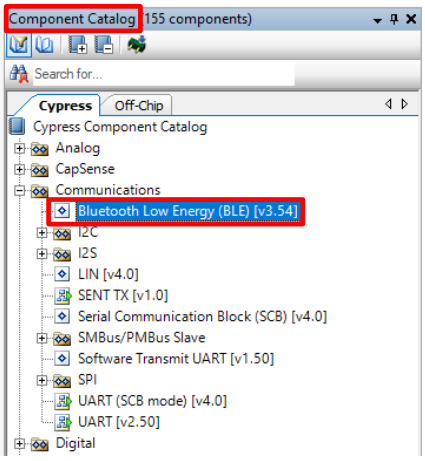
図 3. PSoC Creator の回路図



5.1 コンポーネントを設定

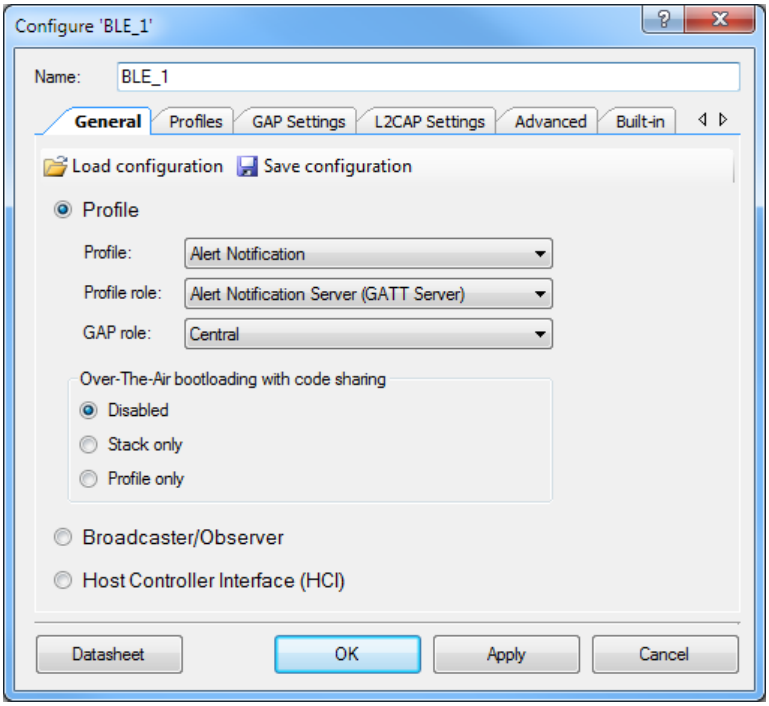
1. 新しい PSoC4100 BLE/PSoC4200 BLE 設計プロジェクトを作成します。PSoC Creator が初めての方は、PSoC Creator ホームページを参照してください。
2. BLE コンポーネント (Component Catalog > Communications) を TopDesign 回路図にドラッグ アンド ドロップします (図 4 を参照してください)。

図 4. BLE コンポーネント



3. BLE コンポーネントをダブルクリックして設定します。図 5 に示すように、設定ウィンドウが表示されます。

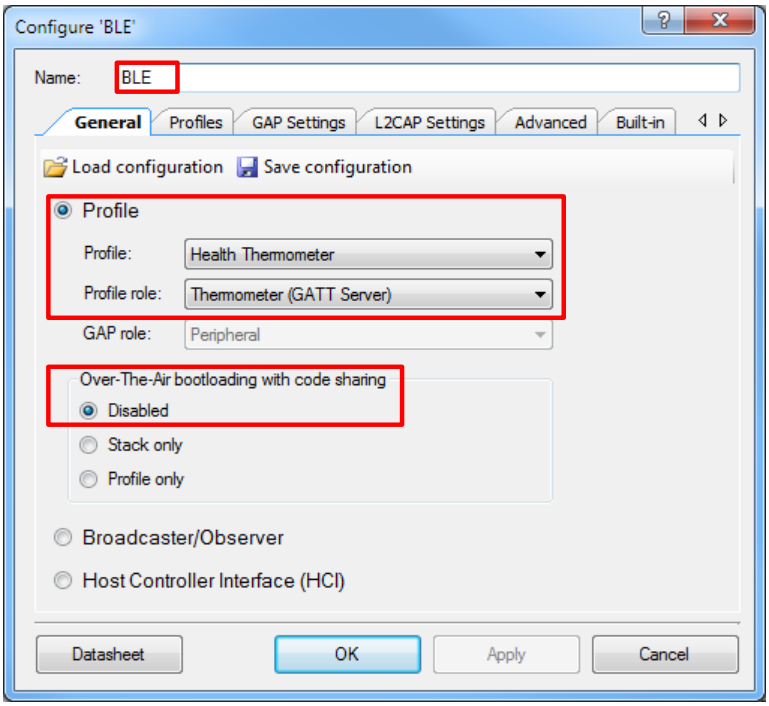
図 5. BLE コンポーネントの設定ウィンドウ



4. コンポーネント設定ウィンドウの **General** タブ (図 6 を参照) で以下の設定を行います。

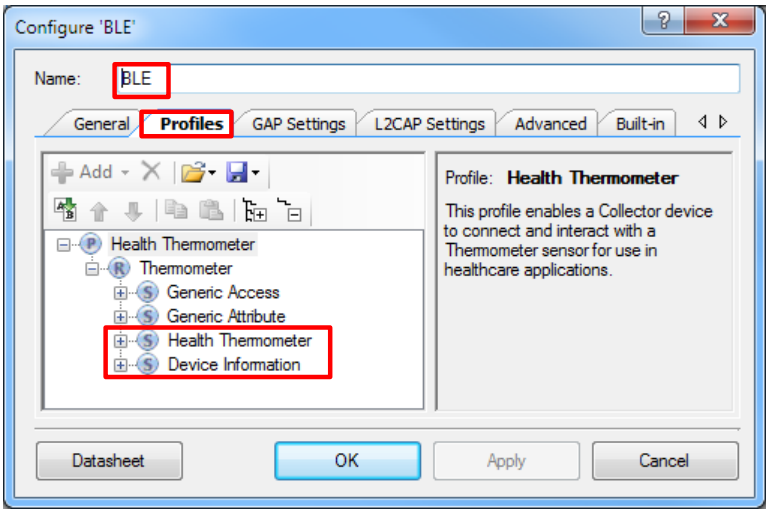
   Name: BLE (名前: BLE)
   Configuration: Profile Collection (コンフィギュレーション:プロファイル コレクション)
   Profile: Health Thermometer (プロファイル: BLE 健康温度計)
   Profile role: Thermometer (GATT Server) (プロファイル ロール: 温度計 (GATT サーバー))
   Over-The-Air bootloading with code sharing: Disabled (コード共有による無線ブートローディング: 無効)

図 6. General タブ



注: Bluetooth SIG に従って、Health Thermometer 標準プロファイルは Health Thermometer Service と Device Information Service をカプセル化します。したがって、図 7 に示すように、これらのサービスはデフォルトで追加されます。健康温度計プロファイルまたは健康温度計サービスの詳細については、Bluetooth 採用仕様を参照してください。

図 7. 健康温度計プロファイル



5. **Profiles** タブで以下の設定を行います。

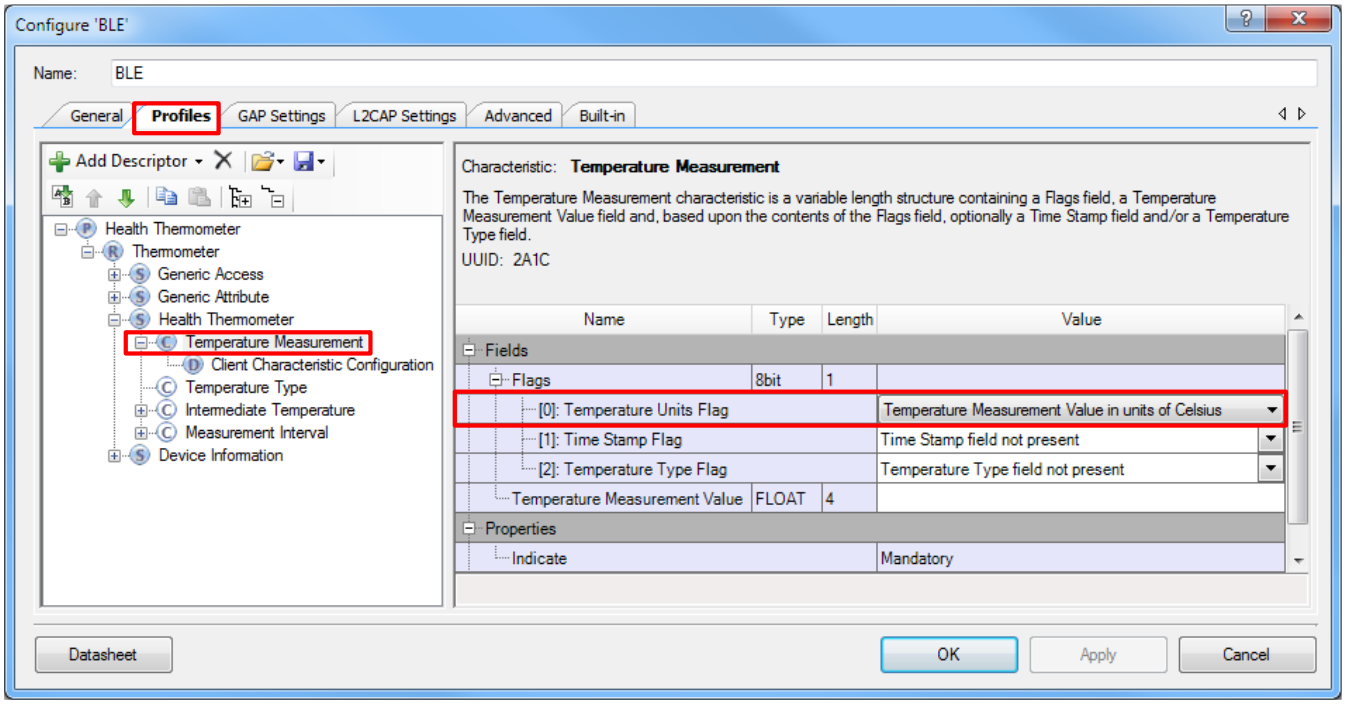
   Service: Health Thermometer (サービス: 健康温度計)

   Characteristic: Temperature Measurement (特性: 温度測定)

   Fields: Temperature Units Flag (フィールド: 温度単位フラグ)

Value: Temperature Measurement Value in units of elsius (値: 摂氏単位の温度測定値)
図 8 に温度測定特性の設定を示しています。

図 8. 温度測定特性



同様に、表 1 に従って残りのサービスと特性を更新します。「記述子」列にある“N/A”はフィールドと値が記述子でなく、特性に関連することを意味します。例えば、**Temperature Text Description** フィールドは **Health Thermometer** サービスでの **Temperature Type** の特性に属していますが、**Lower Inclusive Value** フィールドと **Upper Inclusive Value** フィールドは **Measurement Interval** 特性の **Valid Range** の記述子に属しています。

表 1. 特性の設定

サービス	特性	記述子	フィールド	値	備考
Health Thermometer (健康温度計)	Temperature Measurement (温度測定)	該当なし	Temperature Units Flag (温度単位フラグ)	Temperature Measurement Value in units of degrees elsius (摂氏単位での温度測定値)	使用可能な値は Bluetooth SIG によって定義される。BLE コンポーネントにより使用可能な値のいずれかを選択できる
	Temperature Type (温度タイプ)	該当なし	Temperature Text Description (温度テキスト説明)	Body (general) (ボディ(一般))	使用可能な値は Bluetooth SIG によって定義される。BLE コンポーネントにより使用可能な値のいずれかを選択できる
	測定間隔	該当なし	測定間隔	1	ユーザ定義 単位: 秒 範囲：1～65535
	測定間隔	有効な範囲	Lower Inclusive Value (下位包括値)	1	ユーザ定義 単位: 秒 範囲：1～65535
			Upper Inclusive Value (上位包括値)	60	ユーザ定義 単位: 秒 範囲：1～65535
Device Information (デバイス情報)	メーカー名文字列	該当なし	メーカー名	Cypress Semiconductor (サイプレスセミコンダクタ)	ユーザ定義
	モデル番号文字列	該当なし	モデル番号	1.0	ユーザ定義
	シリアル番号文字列	該当なし	連番機能	**	ユーザ定義
	ハードウェア リビジョン文字列	該当なし	ハードウェア リビジョン	CY8CKIT-042-BLE	ユーザ定義
	ファームウェア リビジョン文字列	該当なし	ファームウェアのリビジョン	**	ユーザ定義
	Software Revision String (ソフトウェア リビジョン文字列)	該当なし	Software Revision (ソフトウェア リビジョン)	PSoC Creator 4.2	ユーザ定義

残りの設定は、それぞれデフォルト値のままにしておきます。

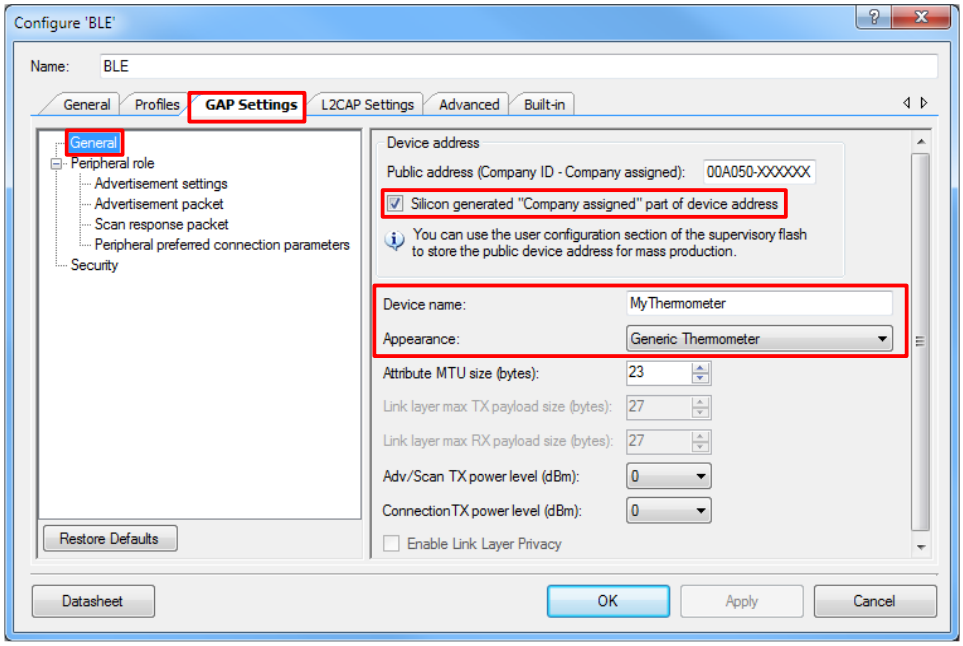
6. 表 2 に従って、**Bluetooth デバイスアドレス (BD_ADDR)** 、**デバイス名**、および[**GAP 設定**]タブの[**General** 設定]の下の **Appearance** を構成します。図 9 は、一般的な **GAP 設定**を示しています。

表 2. General 設定

名	値	備考
Public address (パブリックアドレス)	Check Silicon Generated Address (シリコンの生成したアドレスを確認)	会社 ID と会社に割り当てられた値をアドレスとして使用。これらの詳細な情報がない場合は、フィールドに所望のアドレスを追加
Device name (デバイス名)	MyThermometer	ユーザ定義
Appearance (外観)	Generic Thermometer (一般的な温度計)	使用可能な値は Bluetooth SIG によって定義される。BLE コンポーネントにより使用可能な値のいずれかを選択できる

注: パブリック アドレスは、デバイスを識別するために使用されるユニークな 48 ビットの BD_ADDR を指します。これは会社 ID (24 ビット)と会社に割り当てられた値(24 ビット)という 2 つの部分に分けられています。デフォルトでパブリック アドレスにはサイプレス セミコンダクタの社 ID がロードされています。IEEE によって割り当てられた 24 ビットのユーザーの会社 ID を使わなければなりません。

図 9. デバイス名と外観

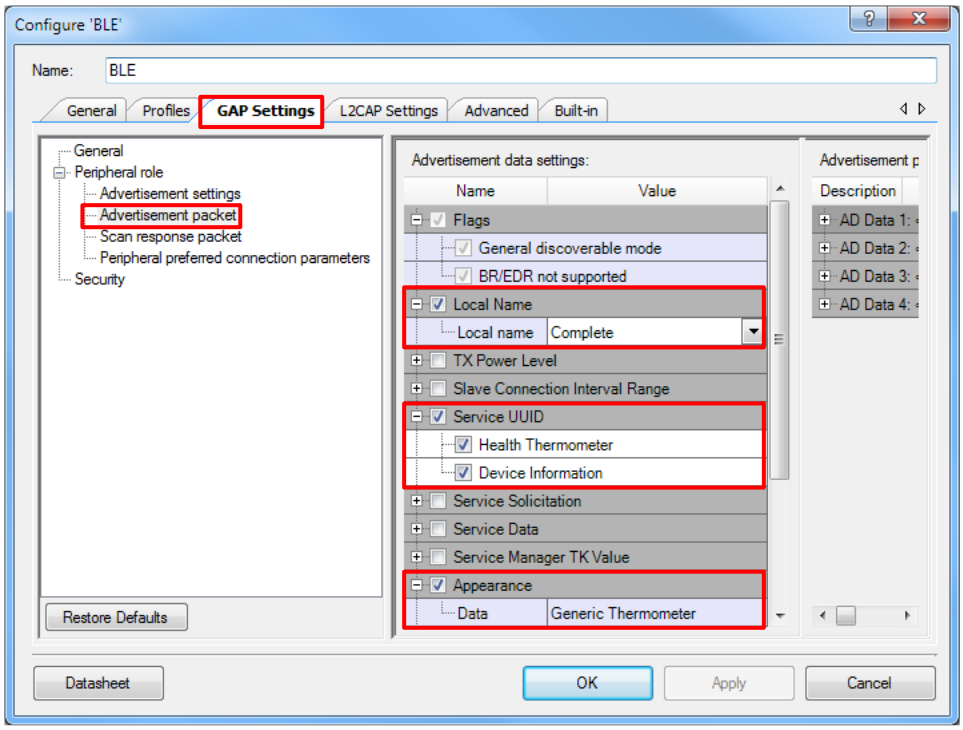


7. **Peripheral Role** の下の **Advertisement Settings** は、このプロジェクトのデフォルト値のままになります。表 3 に従って、ペリフェラルの役割の下で **Advertisement packet** 設定を構成します。図 10 は、アドバタイズパケット設定を示しています。

表 3. Advertisement Packet の設定

名	チェック ボックス	値	備考
Local Name (ローカル名)	有効	Complete (完全)	通知パケットの一部として完全な名前を送信。送信する文字数を選択することで省略名を送信することも可能
Service UUID (サービス UUID)	有効	該当なし	通知パケットの一部としてサービスの UUID (ユニバーサル一意識別子)を送信
Service UUID (サービス UUID) > Health Thermometer (健康温度計)	有効	該当なし	通知パケットの一部として健康温度計サービス UUID を送信
Service UUID (サービス UUID) > Device Information (デバイス情報)	有効	該当なし	通知パケットの一部としてデバイス情報サービス UUID 送信
外観	有効	該当なし	通知パケットの一部として外観値を送信

図 10. GAP 設定–Advertisement Packet



Scan response packet、**Peripheral preferred connection parameters**、**Security** 設定など、残りの設定をデフォルト値のままにします。また、[**L2CAP Settings**]タブの設定をデフォルト値のままにします。

8. **Apply** それから **OK** をクリックします。
9. デジタル入力ピン コンポーネントを配置して、図 11 と図 12 に示すように設定します。

図 11. 端子配置

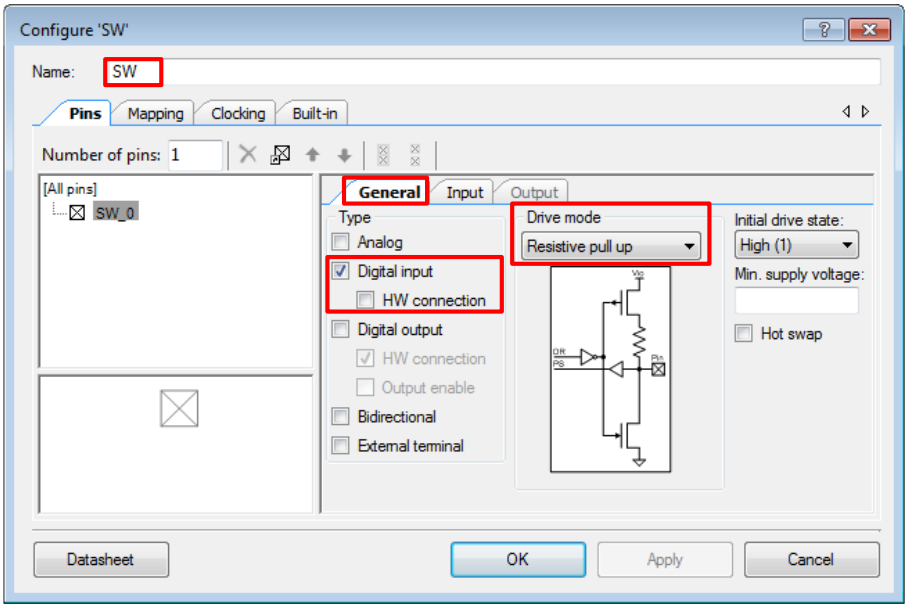
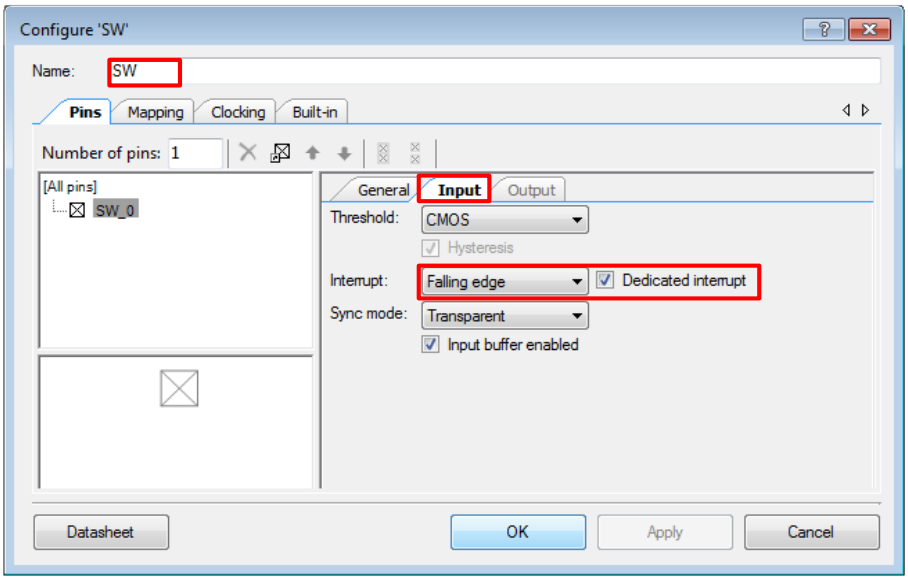


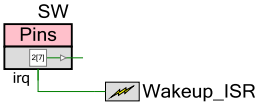
図 12. ピン割込みの設定



注：論理 HIGH という信号のデフォルト状態を維持するために、ユーザー スイッチのドライブモードは**抵抗プルアップ**として選択されます。スイッチを押すと、ピンがグランドに接続されるので、ピン上の信号が論理 HIGH から論理 LOW に駆動されます。その結果、割り込みは**立ち下がりエッジ**で設定されます。

10. 割り込みコンポーネントを配置し、SW コンポーネントの irq ピンに接続し、図 13 に示すように「Wakeup_ISR」に名前を変更します。割り込みコンポーネントは、割り込み信号を記録し、それぞれの機能をトリガーするために使用されます。

図 13. SW ピン



11. シーケンス SAR ADC コンポーネントをドラッグアンドドロップし、表 4 に従って構成します。図 14 および図 15 は、シーケンス SAR ADC コンポーネントの構成設定を示しています。

表 4. SAR ADC コンポーネントの設定

タブ	名称	値	備考
General (全般)	Name (名称)	ADC	この名前はコンポーネント用の API の接頭辞として使用される
	Clock frequency (kHz) (クロック周波数(kHz))	6000	このアプリケーションでは、高精度が不要。より速いサンプリング速度は、アクティブな時間を減らすことによって電力を節約するのに役立つ
	Vref Select (Vref 選択)	VDDA	0～V_{DDA} の入力電圧範囲
	Samples Averaged (サンプル平均)	16	16 のサンプル平均値は高周波ノイズを平均するのに役立つ
Channels (チャネル)	Sequenced channels (連続チャネル)	1	スキャンされるチャネル数
	AVG	チェックした	対応する ADC チャネルに対して平均化を有効にする

図 14. SAR ADC 設定 「General」タブ

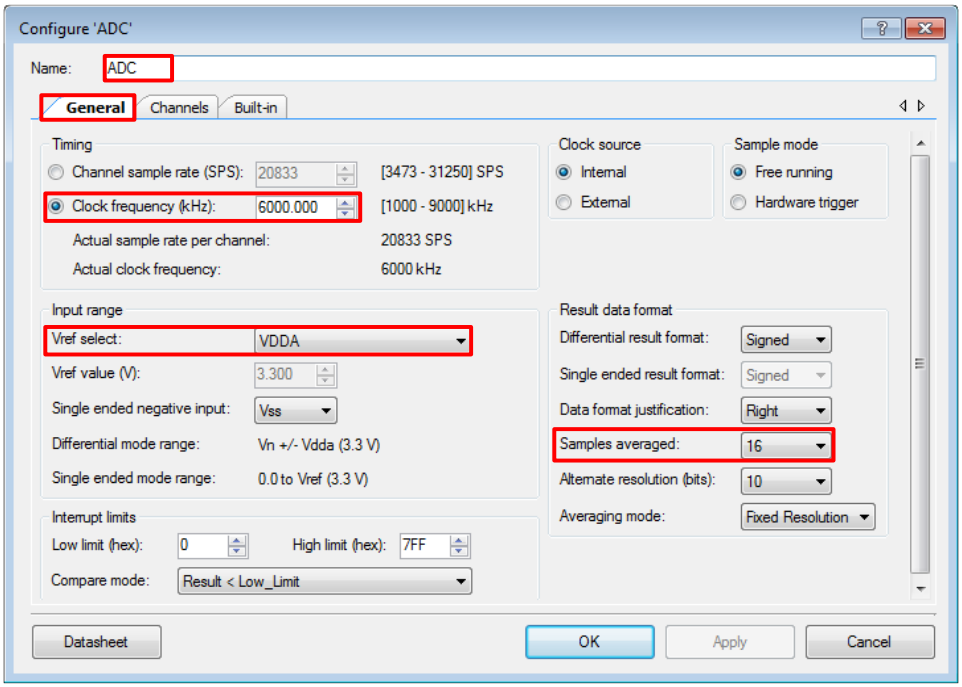
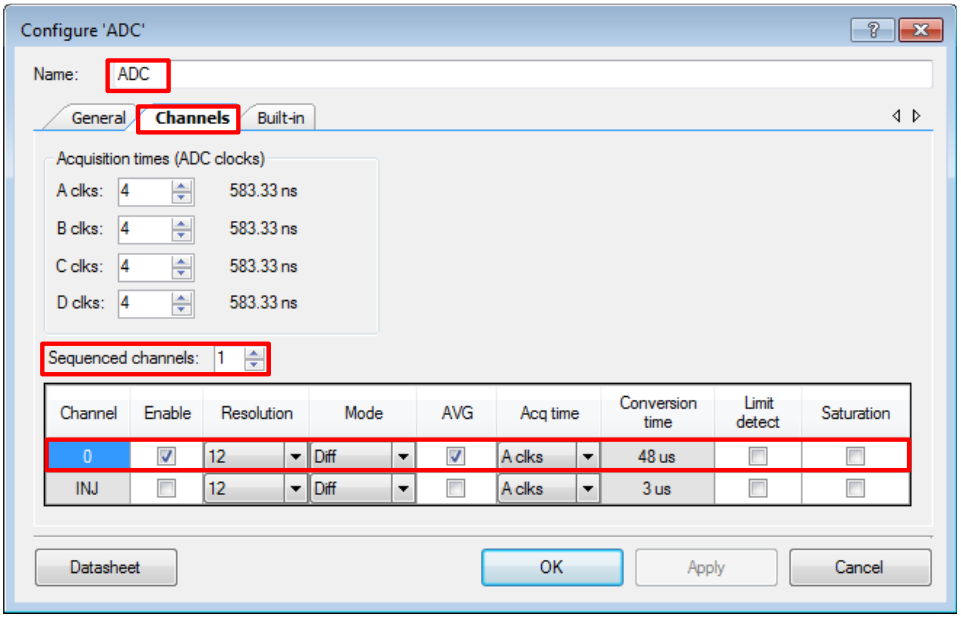


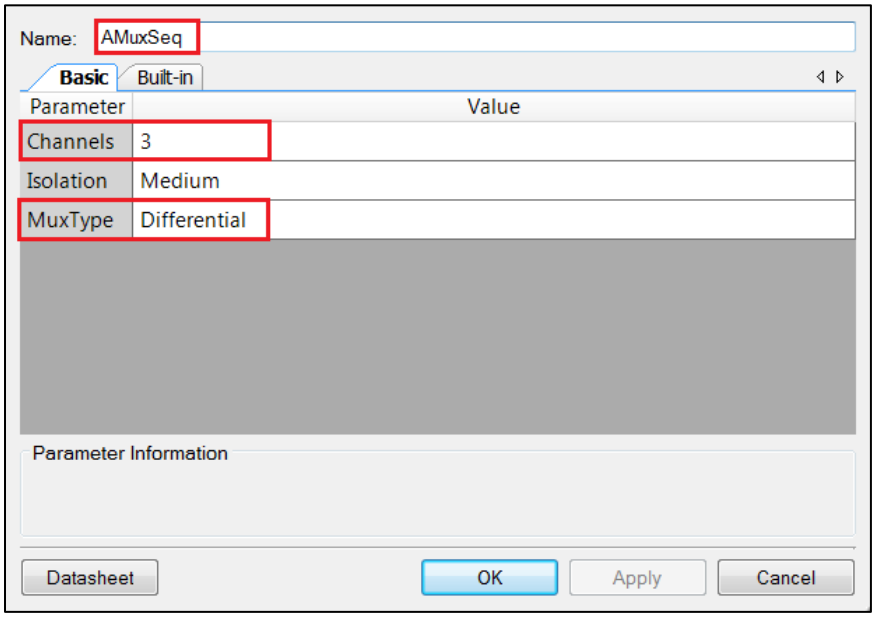
図 15. SAR ADC 設定 「Channels」タブ



12. 温度を測定するために、ステップ 11～18 に従います。

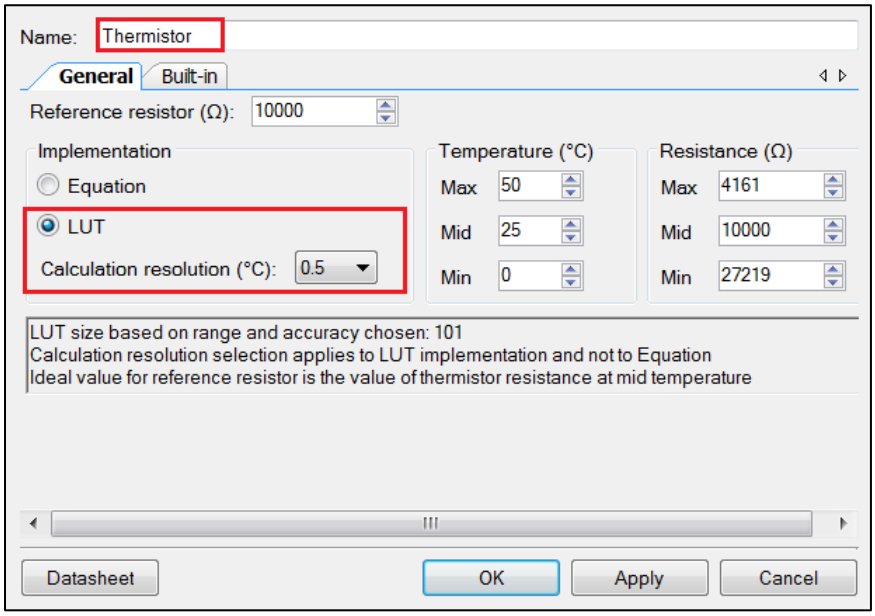
13. アナログ マルチプレクサ シーケンサー コンポーネントを配置し、図 16 に従って構成します。AMUX コンポーネントは、複数のアナログ信号を ADC 入力に多重化するために使用されます。

図 16. AMUX の設定



14. Thermistor Calculator コンポーネントを配置し、図 17 に従って構成します。これらの設定は、サーミスタを使用した温度測定用にこのコンポーネントを構成します。

図 17. サーミスタ計算機の設定



注: 正確な温度測定のために、温度 (ºC) と抵抗 (Ω) の列をサーミスタのデータシートで指定された値で更新する必要があります。

15. 2 つのアナログ ピン コンポーネントを配置して、図 18 と図 19 に示すように、設定します。

注: これらのピンはアナログ ピンとデジタル出力の両方として設定されています。デジタル出力としての設定により、ファームウェアでピンを V_{DD} または V_{SS} に駆動することができます。

図 18. アナログ ピンの設定 V_HIGH

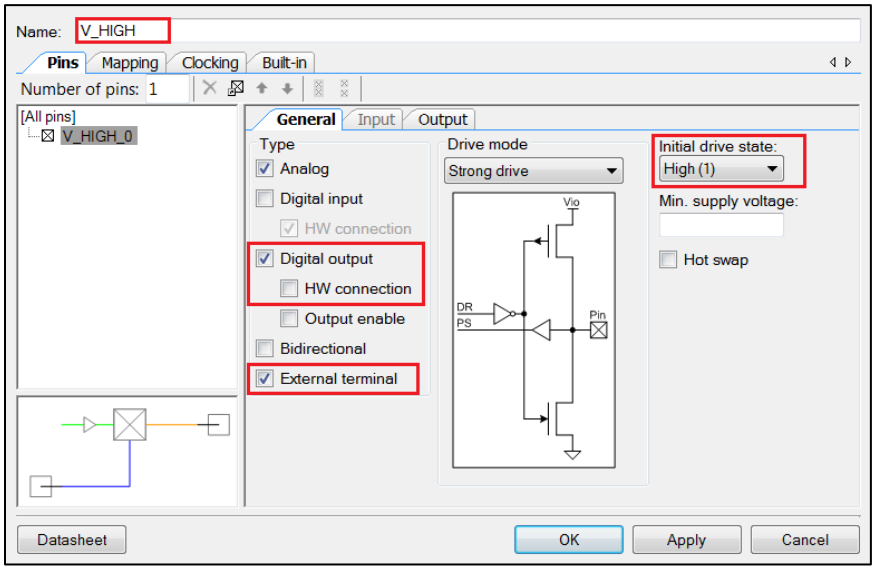
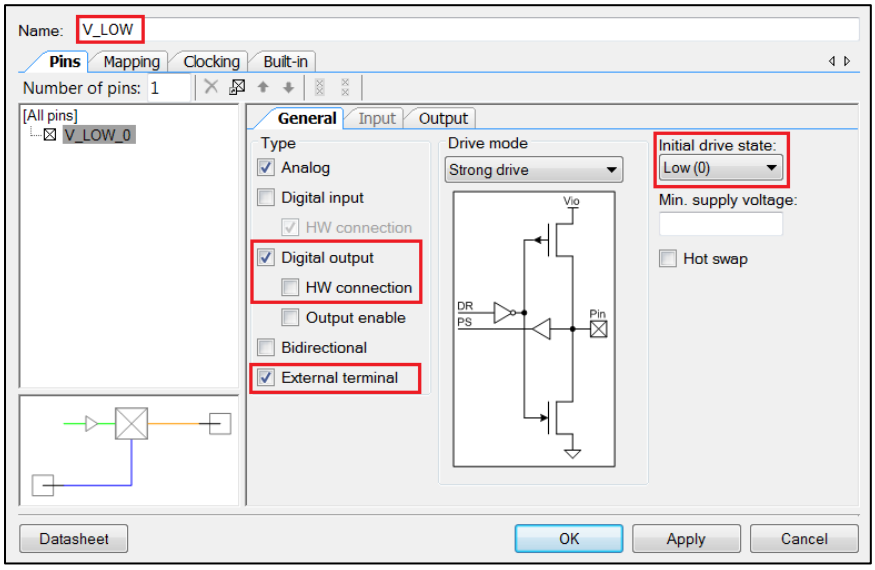


図 19. アナログ ピンの設定 V_LOW

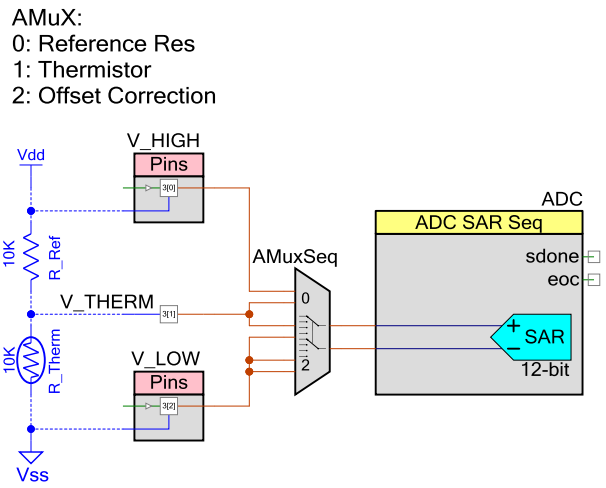


16. アナログ ピン コンポーネントを配置し、「V_THERM」という名付けます。**外部端子**を有効にします。

17. 図 20 に示す回路図のように、アナログ マルチプレクサ シーケンサ、シーケンシング SAR ADC、およびアナログ ピン コンポーネントを接続します。

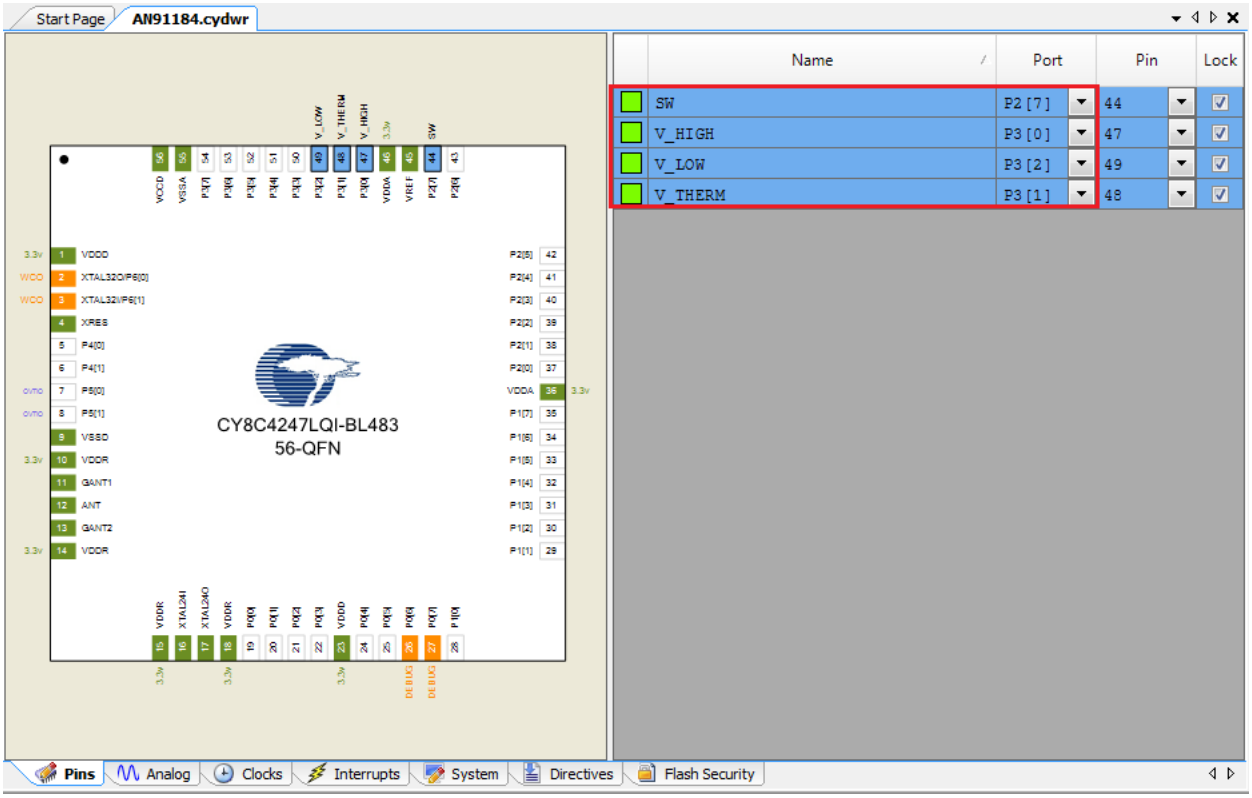
    注: すべての青色のコンポーネントが「オフチップ」カタログからのものです。これらは、ドキュメントで参照することのみを目的としていますが、回路図のハードウェア上にあるものをより詳しく説明するためにも使用できます。

図 20. 回路図 – 温度測定



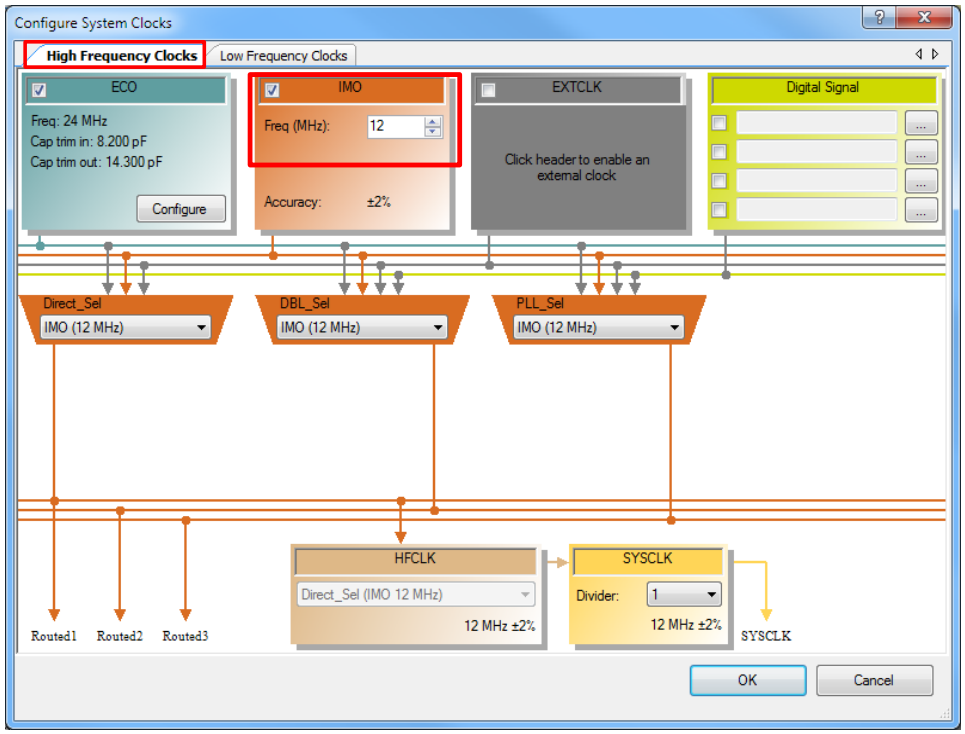
18. 設計全体リソース ウィンドウのピン タブでは、図 21 に示すようにピンを接続します。

図 21. 設計全体リソース ウィンドウのピン タブ



19. 設計全体リソース ウィンドウのクロック タブでは、図 22 に示すように、IMO 周波数を 12MHz に設定します。

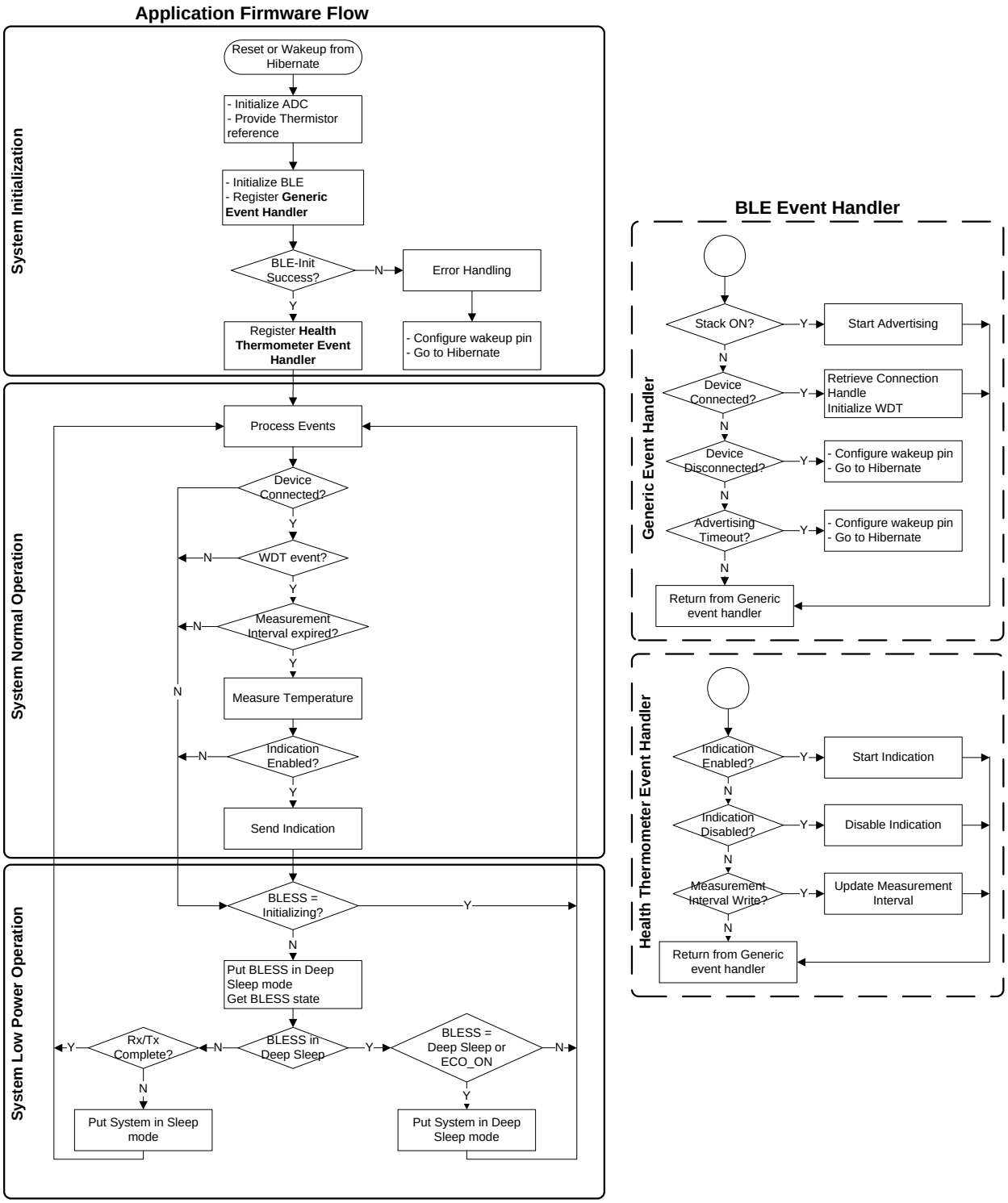
図 22. クロックの設定



5.2 ファームウェアを設定

図 23 に、健康温度計アアプリケーションのファームウェア フローを示します。

図 23. システム フローチャート



注：アプリケーションファームウェアのソースファイルは、このアプリケーションノートに含まれているプロジェクト例に含まれています。ソースファイルを独自のプロジェクトに含めるか、完成したサンプルプロジェクトをそのまま使用できます。サンプル プロジェクト用のソース ファイルは表 5 に示す 9 つがあります。

表 5. サンプル プロジェクトのソース ファイル

名	変更内容
main.c	これは主なファームウェア ファイル。このファイルに関数が 1 つだけある。 ■ `main()` – この関数はアプリケーション全体の流れを制御。この関数の一部として実行される主なタスクはシステムの初期化、およびアプリケーション制御 (BLE イベントの処理、アプリケーション フローの制御および低消費電力実装を含む)。
CommonFunctions.c/.h	アプリケーション制御に使用される一般の関数を実装。以下の関数を含んでいる。 ■ `InitializeSystem()` – システムのすべてのブロックを初期化。 ■ `PrepareForDeepSleep` – ハードウェア ブロックをディープ スリープ モードに移行することによって、システムの低消費電力動作を実現 ■ `WakupFromDeepSleep()` – ハードウェア ブロックを通常の動作に復帰。
Temperature.c/.h	ADC から測定データを読み出して、サーミスタ計算機コンポーネントで温度計測を温度を計算することで、温度測定を実現また、これにより、ユーザは温度センサーから温度を測定する代わりに、それをシミュレーションすることも可能。以下の関数を含んでいる。 ■ `MeasureSensorVoltage()` – この関数はセンサー電圧を測定するためのもの。センサー シミュレーション オプションが選択された時、この関数は使用不可。詳細については、センサー シミュレーション節を参照してください。 ■ `ProcessTemperature()` – この関数は温度値を測定またはシミュレートするためのもの。
WatchdogTimer.c/.h	これは、ウォッチドッグ タイマの機能を実装し、システム時間を追跡。以下の関数を含んでいる。 ■ `WatchdogTimer_Start()` – ウォッチドッグ タイマー (WDT0) を 1s 周期で起動し、一致時に割り込みを生成 ■ `WatchdogTimer_Isr()` – WDT 用の ISR であり、測定間隔を追跡するために使用される。これはウォッチドッグ タイマからのコールバック関数 ■ `WatchdogTimer_Stop()` – ウォッチドッグ タイマー (WDT0) を中止
BLE_HTSS.c/.h	このファイルはプロジェクトの BLE 固有の機能を処理。これは、イベント ハンドラ節で詳しく説明する BLE スタックから生成されたイベントを処理。以下の関数を含んでいる。 ■ `GenericEventHandler()` – BLE スタックが生成する一般的なイベントを処理 ■ `HtssEventHandler()` – 健康温度計サービス用に生成されたイベントを処理 ■ `ProcessBLE()` – 温度データを指示として GATT クライアントに送信 ■ `ConvertFloatTemp()` – 温度データを IEEE-754 形式から IEEE-11073 形式に変換 ■ `EnableBLE()` – BLE コンポーネントを起動し、イベント ハンドラ関数を登録

次の節では健康温度計アプリケーションの動作について説明します。完全なファームウェアが本文書に含まれていないことに注意してください。代わりに、主要な概念について詳しく説明します。完全なファームウェアについては、付属のサンプルプロジェクトを参照してください。Health Thermometer アプリケーションステートマシンは、次の 4 つの状態で構成されています。

- システムの初期化
- イベント ハンドラ
  - 一般的なイベント ハンドラ
  - 健康温度計イベント ハンドラ
- システムの通常動作

- システムの低消費電力動作

次の節でこれらのステートの詳細について説明します。

5.2.1 システムの初期化

デバイスがリセットする、またはハイバネート モードから復帰すると、このファームウェアは初期化を実行します。この初期化処理は、SAR ADC の起動、グローバル割り込みの有効化、オペアンプの起動、およびウォッチドッグ タイマーの起動を含んでいます。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 サブシステム全体の初期化が処理されます。

注: BLE コンポーネントの初期化の一部として、ユーザ コードは BLE スタックからイベントを受信するために呼び出す必要があるイベント ハンドラ関数へのポインタを渡す必要があります。図 23 に示す一般的なイベント ハンドラは BLE 初期化の一部として登録されます。Code 1 に、BLE コンポーネントを起動して一般的なイベント ハンドラを登録するためのコードを示します。

Code 1. BLE 初期化

```
apiResult = CyBle_Start(GenericEventHandler);
```

BLE コンポーネントが正常に初期化した場合、ファームウェアは健康温度計サービス用のイベントを受信するために呼び出される関数を登録し、通常動作モードに切り替えます。Code 2 に、健康温度計サービを登録するためのスニペットを示します。

Code 2. 健康温度計サービス イベント ハンドラ

```
CyBle_HtsRegisterAttrCallback(HealthThermometetEventHandler);
```

5.2.2 イベント ハンドラ

BLE コンポーネントでは、BLE スタック上で行われるあらゆる動作の結果は、イベントのリストを介してアプリケーション ファームウェアに転送されます。これらのイベントは BLE インターフェース ステータスとデータを提供します。次のようにイベントを分類できます。

- 一般的なイベント

  GAP レイヤー、GATT レイヤー、およびスタックの L2CAP レイヤーで実行される操作により、これらのイベントが生成されます。たとえば、BLE スタックが初期化されてオンになったときに CYBLE_EVT_STACK_ON イベントを受け取り、リモートデバイスとの接続が確立されたときに CYBLE_EVT_GAP_DEVICE_CONNECTED イベントを受け取り、クライアントから書き込みコマンドを受け取ったときに CYBLE_EVT_GATTS_WRITE_CMD_REQ イベントを生成します。一般的なイベントの詳細については、BLE コンポーネントの API ドキュメントを参照してください (PSoC Creator 内の BLE コンポーネントを右クリックして、**Open API Documentation** を選択)。

  BLE リンクを正常に確立し、維持するために、アプリケーション ファームウェアにはイベント ハンドラ関数がなければなりません。コード 3 は GenericEventHandler 関数の実装を示しています。BLE スタックの初期化、デバイス接続と切断、およびタイム アウトのときに生成されるイベントが処理されます。

- サービス固有イベント

  サービス固有イベントは、Bluetooth SIG が定義する標準サービスで行われる動作により発生されるイベントです。たとえば、CYBLE_EVT_HTSS_INDICATION_ENABLED イベントは、クライアントが温度構成特性の表示を有効にするためにクライアント構成特性記述子を書き込むときにサーバーによって受信されます。サービス固有のイベントの詳細については、BLE コンポーネントの API ドキュメントを参照してください。

  BLE コンポーネントはサービス固有のイベント ハンドラにこれらのイベントを転送できます。アプリケーション ファームウェアは、これらのイベントを処理するためのサービス固有イベント ハンドラー関数を含む必要があります。サービス固有のイベントハンドラーがサポートされていない場合、これらのイベントは共通イベント ハンドラー (GenericEventHandler) で処理する必要があります。Code 4 は、HtssEventHandler というサービス固有のイベントハンドラーの実装を示しています。

Code 3. 一般的なイベント ハンドラ

```
void GenericEventHandler(uint32 event, void *eventParam)
{
  switch(event)
  {
    /* This event is received when component is Started */
    case CYBLE_EVT_STACK_ON:
    {
      /* Stop watchdog to reduce power consumption during advertising */
      WatchdogTimer_Stop();
      /* Start Advertisement and enter Discoverable mode*/
      CyBle_GappStartAdvertisement(CYBLE_ADVERTISING_FAST);
      break;
    }

    /* This event is received when device is disconnected or advertising times out*/
    case CYBLE_EVT_GAP_DEVICE_DISCONNECTED:
    case CYBLE_EVT_TIMEOUT:
    {
      /* Sets the ENABLE_HIBERNATE flag to put system in Hibernate mode */
      SystemFlag |= ENABLE_HIBERNATE;
      break;
    }

    /* This event is received when connection is established */
    case CYBLE_EVT_GATT_CONNECT_IND:
    {
        /* Start watchdog timer with 1s refresh interval */
        /* Note: For this application, wakeup should be 1s because htssInterval
        *  resolution is configured as 1s */
        WatchdogTimer_Start(REFRESH_INTERVAL);
        /* Retrieve BLE connection handle */
        connectionHandle = *(CYBLE_CONN_HANDLE_T *) eventParam;
        break;
    }

    default:
    {
      /* Error handling */
      break;
    }
  }
}
```

Code 4. 健康温度計サービス イベント ハンドラ

```
void HtssEventHandler(uint32 event, void* eventParam)
{
  CYBLE_HTS_CHAR_VALUE_T *interval;
  switch(event)
  {
    /* This event is received when indication are enabled by the central */
    case CYBLE_EVT_HTSS_INDICATION_ENABLED:
    {
      /* Set the htssIndication flag */
      htssIndication = true;
      break;
    }
```

```
            /* This event is received when indication are disabled by the central */
            case CYBLE_EVT_HTSS_INDICATION_DISABLED:
            {
              /* Reset the htssIndiciation flag */
              htssIndication = false;
              break;
            }

            /* This event is received when measurement interval is updated by
            *  the central */
            case CYBLE_EVT_HTSS_CHAR_WRITE:
            {
              /* Retrive interval value */
              interval = ((CYBLE_HTS_CHAR_VALUE_T *)eventParam);
              htssInterval = interval->value->val[1];
              /* Update htssInterval with the updated value */
              htssInterval = (htssInterval << 8) | interval->value->val[0];
              break;
            }

            default:
            {
              /* Error handling */
              break;
            }
          }
        }
```

5.2.3 システムの通常動作

システムの通常動作ステートでは、ファームウェアは、周期的に CyBle_ProcessEvents()を呼び出して BLE スタック関連動作を処理し、接続が確立されたかをチェックします。

注: リンク層へのデータ送受信とアプリケーション層へのイベント生成などのあらゆる BLE スタック関連動作は CyBle_ProcessEvents()関数呼び出しの一部として実行されます。このアプリケーションでは、Code 1 はスタックを初期化しますが、スタックに関連するイベントは CyBle_ProcessEvents()関数が呼び出された時にのみ生成されます。同様に、CyBle_ProcessEvents()が呼び出された時にのみ、デバイスの接続と切断、タイムアウト宣伝、および健康温度計サービスに関連する他のイベントが生成されます。

接続が確立された場合、ファームウェアは一定の時間間隔で温度を測定します (健康温度計のサービスの測定間隔特性による設定)。温度を測定した後、指示はセントラル デバイスによって有効になっている場合、ファームウェアは指示として、BLE セントラル デバイスに温度データを送信します。

BLE アプリケーションでは、デバイスは、BLE 接続状態に応じて、通知間隔または接続間隔とも呼ばれている定期的な間隔のみでデータを送受信します。そのため、システム通常動作タスクが完了した時に、電力を節約するために、デバイスはシステムの低消費電力動作モードに入り、次の接続／通知間隔でウェイクアップします。

5.2.4 システムの低消費電力動作

システムの低消費電力ステートでは、システムは次の 3 つ可能な電力モードのいずれかで動作します。

- スリープ

  CPU がフリーですが、BLE サブシステム (BLESS) がアクティブで、データの送受信にビジーのとき、CPU はこのモードに入ります。この場合、CPU がスリープ モードに入りますが、クロックおよびレギュレータなどの残りのコアは BLE が正常に動作するためにアクティブのままです。電力を節約するために、内部主発振器 (IMO) 周波数は 3MHz に低減されます。ウェイクアップのときに、それは 12MHz に切り替えます。

- ディープ スリープ

  ファームウェアは絶えず BLESS をディープスリープ モードに移行させようとします。BLESS を正常にディープスリープ モードに移行させると、残りのシステムもディープスリープ モードに移行します。

注: BLESS をディープスリープ モードにすると、すぐにデバイスはディープスリープ モードに移行します。これが保証できない場合、ファームウェアは (ISR 処理を回避するために) 割り込みを無効にし、BLESS がディープスリープ モードか ECO_ON モードに入ったかを再度チェックします。BLESS がこれらのモードのいずれかに入った場合、デバイスは無事にディープスリープ モードに移行します。そうしない場合、デバイスは Rx/Tx イベントが完成するまで待たなければなりません。

- ハイバネート
  デバイスが切断される、または通知間隔がタイム アウトのとき、デバイスは「ハイバネート」という超低消費電力モードに入ります。このモードから復帰した後、RAM の内容はまだ保持されていますが、ファームウェアは、*main.c* の最初から実行します。

5.2.5 センサー シミュレーション

温度を測定するためのサーミスタと基準抵抗器がない場合は、温度シミュレーションモードを使用してアプリケーションをテストできます。このモードでは、温度データがシミュレートされ、測定間隔ごとに 1 0C ずつ増分されます (デフォルト値は 1 秒です)。Code 5 に示すように、これは、*Temperature.h* ファイルの MEASURE_TEMPERATURE_SENSOR 定数の値を 1 から 0 に変更することによって、アプリケーションコードで行うことができます。

Code 5. シミュレーション温度センサー

```
#define MEASURE_TEMPERATURE_SENSOR (0u)
```

5.3 ハードウェア構成

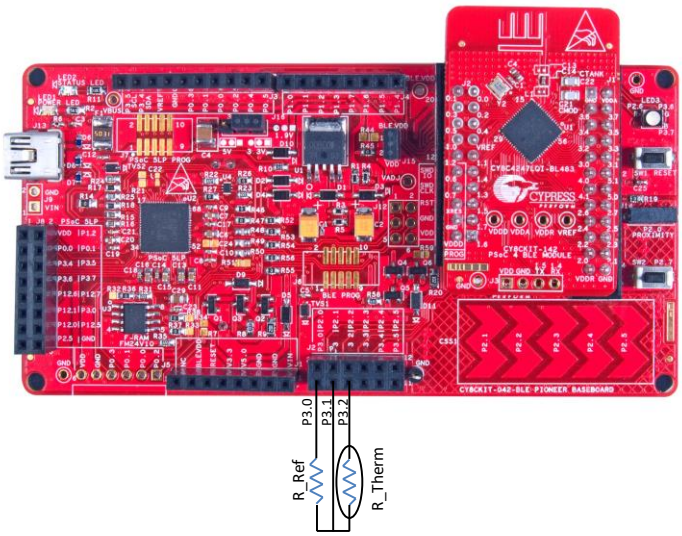
CY8CKIT-042-BLE Bluetooth Low Energy (BLE) Pioneer Kit は、サイプレスの PSoC4 BLE デバイス用の開発キットです。柔軟性に設計されており、このキットは、多くのサードパーティ製の Arduino™ シールドとのフットプリント互換性があります。このキットは、1 個の追加のヘッダーを実装して Digilent® Pmod™ペリフェラル モジュールをサポートするプロービジョンがあります。また、ボードは 1 個の CapSense®スライダー、1 個の基板搭載 1Mb F-RAM、1 個の RGB LED、1 個のプッシュ ボタン スイッチ、1 個の統合 USB プログラマ、1 個のプログラム／デバッグ ヘッダおよび USB-UART/I2C ブリッジを備えています。

1. BLE パイオニア ベースボードに CY8CKIT-142 PSoC4 BLE モジュール(赤色のモジュール)を配置します。

   このキットはサーミスタを備えていませんので、健康温度計アプリケーションをテストするために、以下のオプションの 1 つとともに BLE パイオニア キットを利用できます。

   - 外部サーミスタ – 図 24 に示すように、BLE パイオニアキットに 10kΩ の外部サーミスタと基準抵抗を接続します。

図 24. 外部サーミスタ接続

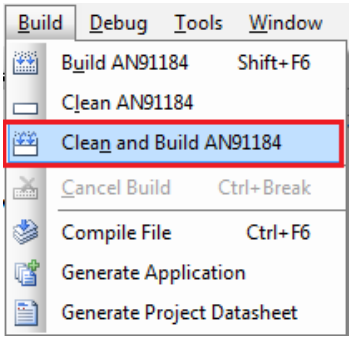


- シミュレーション センサー – ハードウェア センサーを使用する代わりに、ファームウェアは温度データをシミュレーションし、測定間隔 (デフォルト値が 1 秒) につき、温度を 1ºC インクリメントします。詳細については、センサー シミュレーション節を参照してください。

5.4 デバイスをビルドしてプログラム

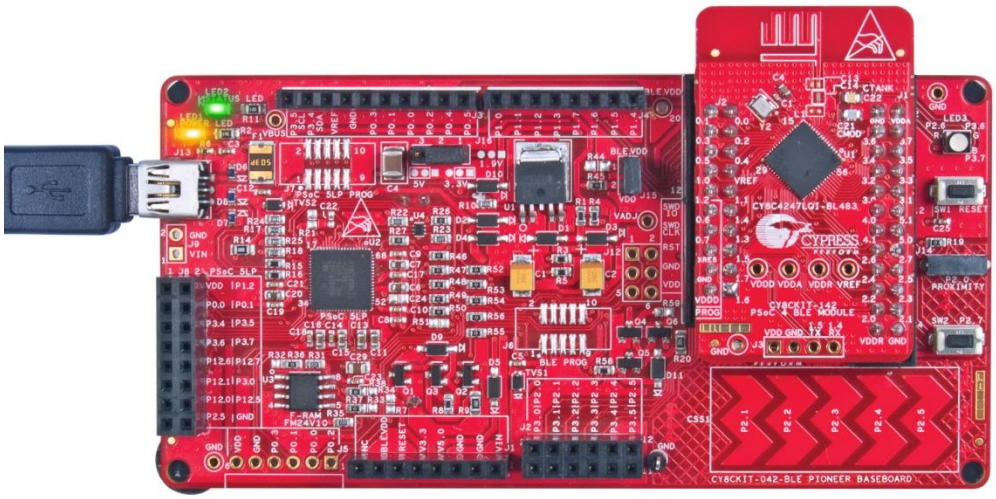
1. 図 25 に示すように、ファームウェアをビルドかつコンパイルするために、**Build** > **Build AN91184** を選択します。

図 25. プロジェクトをビルド



2. BLE パイオニア ベースボードに PSoC4 BLE モジュール (赤色のモジュール) を差し込み、その後、USB スタンダード A－ミニ B ケーブルを使用して PC にキットを接続します (図 26 を参照してください)。PC での USB エニュメレーションを完了させます。

図 26. USB ケーブルを使用して PCB に BLE パイオニア ベースボードを接続



3. 図 27 に示すように、**Debug** > **Program** を選択します。PC に接続しているのは 1 つのみのキットであれば、プログラミングが自動的に起動します。複数のキットであれば、PSoC Creator はどんなキットをプログラムするかという選択を求めます。

図 27. デバイスをプログラム

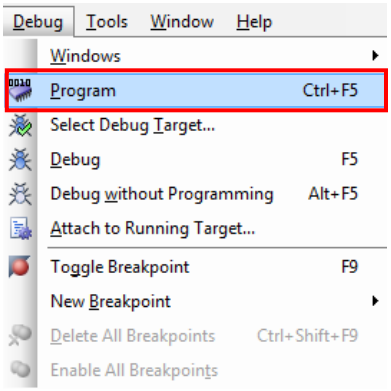
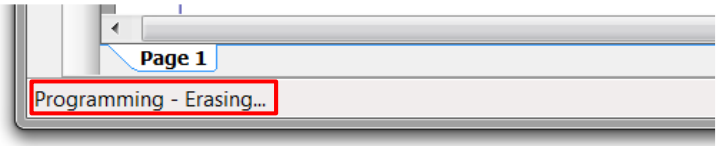


図 28 に示すように、PSoC Creator のステータス バー (画面の左下隅) でプログラミング状態を確認できます。

図 28. プログラミング状態



6 アプリケーションのテスト

CySmart セントラル エミュレーション ツールまたは CySmart モバイル アプリケーションを使用して健康温度計アプリケーションをテストできます。

6.1 CySmart セントラル エミュレーション ツール

BLE ペリフェラル デバイスの動作をテストと検証するために、CY5670 or CY5677 CySmart USB Dongle (BLE Dongle) とともに CySmart セントラル エミュレーション ツールを使用できます。

www.cypress.com/cysmart から最新の CySmart セントラル エミュレーション ツールをダウンロードしてください。

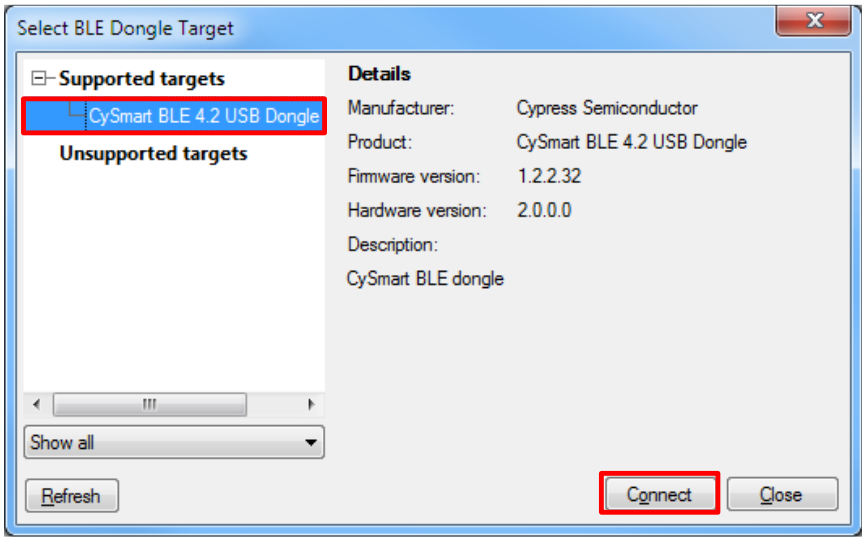
CySmart セントラル エミュレーション ツールを使用して健康温度計アプリケーションを検証するには、次の手順を行います。

1. BLE Dongle を PC に接続し、**Start** > **All Programs** > **CySmart 1.3** > **CySmart 1.3** で CySmart セントラル エミュレーション ツールを起動します。

   CySmart セントラル エミュレーション ツールは USB ドライブに接続する BLE Dongle を検出します。

2. 図 29 に示すように、BLE Dongle を接続するには **Connect** をクリックします。

図 29. BLE ドングルのターゲットを選択



3. PC に BLE Dongle が接続されている時に、図 30 に示すように、BLE デバイスを検索するために **Start Scan** をクリックします。

図 30. Start Scan (スキャン開始)

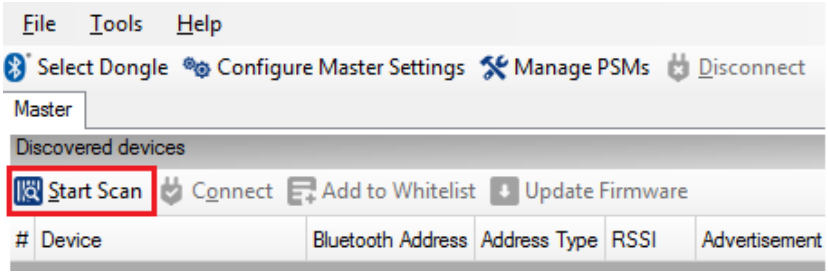
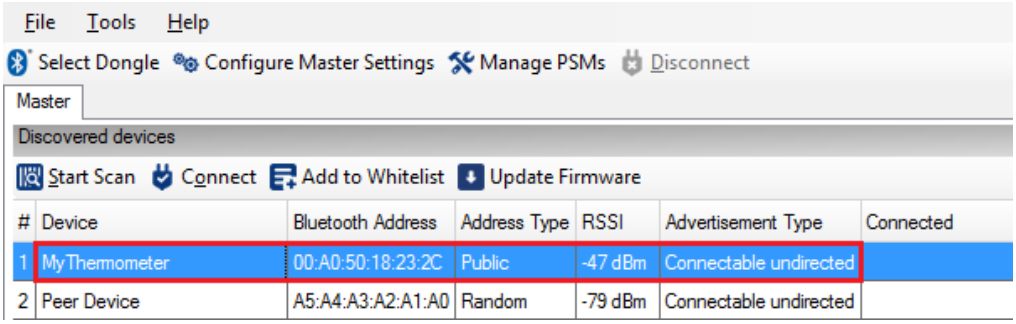


図 31 に示すように、検出された BLE デバイスは CySmart セントラル エミュレーション ツールの画面に表示されます。

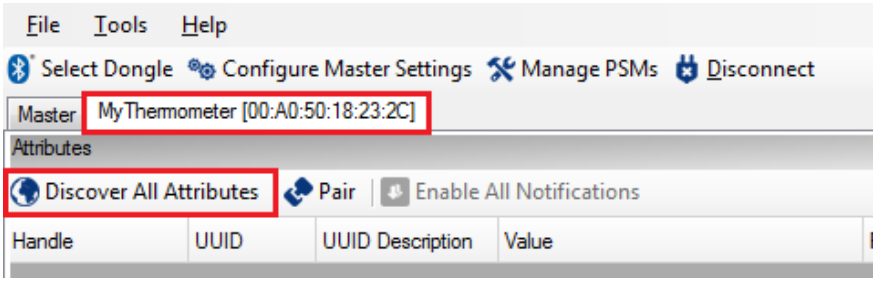
図 31. 検出されたデバイス



4. 図 31 に示すように、「MyThermometer」というデバイスを選択するために、名前をクリックしてから **Connect** をクリックします。

   BLEDongle が BLE デバイスに接続されている時、図 32 に示すように、デバイス名と BD_ADDR のある新しいタブが追加されます。

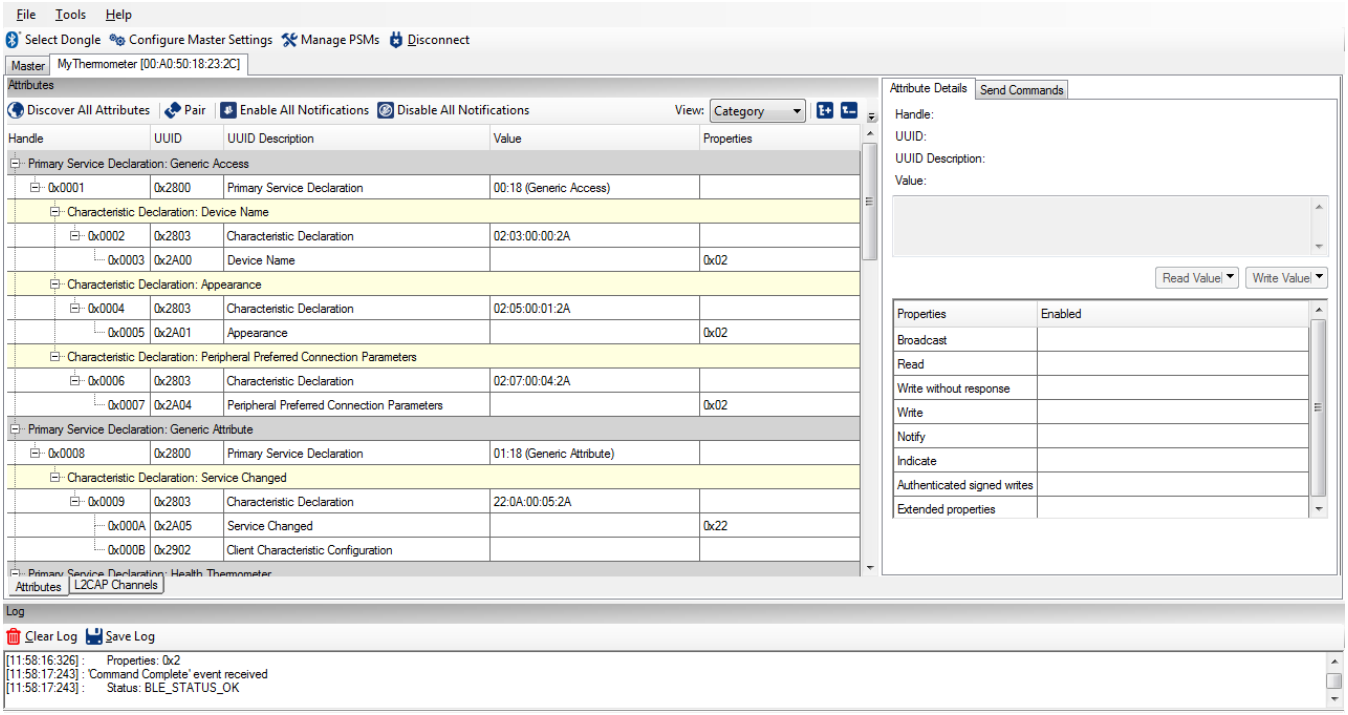
図 32. MyThermometer に接続



5. 「MyThermometer」のペリフェラルによって公開されるすべてのアトリビュートを調べるには、図 32 に示すように **Discover All Attributes** をクリックします。

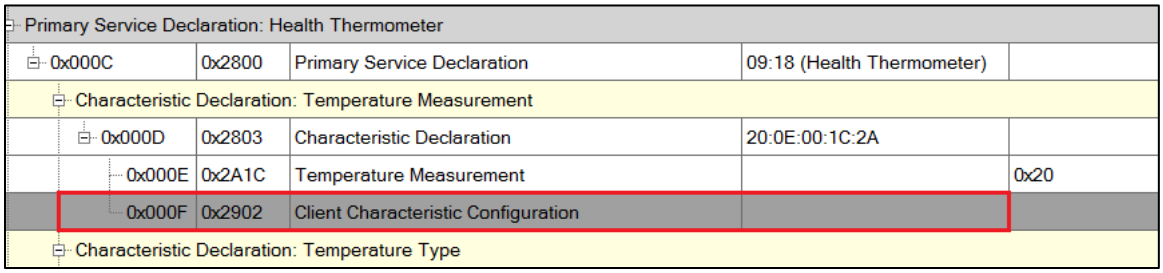
   図 33 に示すように、検出されたすべてのアトリビュートはグループ化されて表示されます。

図 33. 検出されたアトリビュート



6. 測定した温度を読み出すために、図 34 と図 35 に示すように、**Temperature Measurement** 特性の **Client Characteristic Configuration** 記述子を選択します。

図 34. 指示 (Indications) を有効にする ステップ A



7. 図 35 に示すように、指示を可能にする（測定された温度データが指示によって報告）ために、**Client Characteristic Configuration** 記述子内の **02** の値を入力します。

   Note: ビット定義の詳細については、Client Characteristic Configuration Descriptor を参照してください。

図 35. 指示 (Indications) を有効にする ステップ B

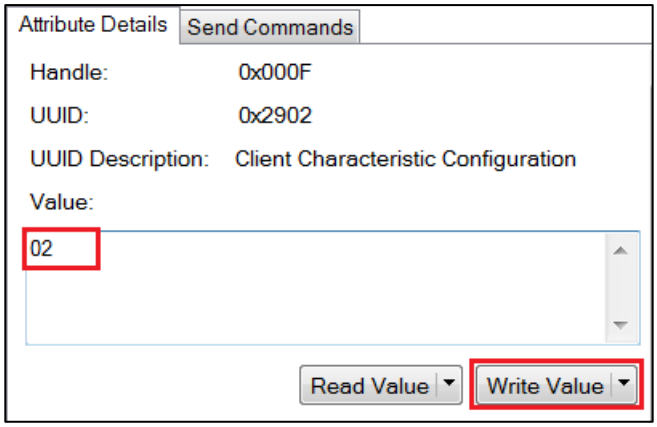


図 36 に示すように、「Temperature Measurement」アトリビュートは測定温度値で更新されます。

図 36. 測定した温度

Primary Service Declaration: Health Thermometer				
0x000C	0x2800	Primary Service Declaration	09:18 (Health Thermometer)	
Characteristic Declaration: Temperature Measurement				
0x000D	0x2803	Characteristic Declaration	20:0E:00:1C:2A	
0x000E	0x2A1C	Temperature Measurement	00:F0:00:00:FF	0x20
0x000F	0x2902	Client Characteristic Configuration	02:00	
Characteristic Declaration: Temperature Type				
0x0010	0x2803	Characteristic Declaration	02:11:00:1D:2A	
0x0011	0x2A1D	Temperature Type		0x02

読み戻される値は 5 バイトの 16 進値であり、そこで、最初のバイトが BLE コンポーネントで設定されたフラグであり、次の 3 バイト値が IEEE-11073 浮動小数点値のリトルエンディアン フォーマットで表示された温度値です。したがって、測定された温度値は 24.0ºC です (0xF0 は 240 で、小数点 1 桁で 24.0 になります)。

CySmart セントラル エミュレーション ツールの詳細については、CySmart User Guide を参照してください。

6.2 CySmart モバイル アプリケーション

サイプレスは、BLE アプリケーションを検証するためのモバイル アプリケーションを提供します。このアプリケーションはさまざまな標準およびカスタム プロファイルに対応しています。GATT データベースがビューできるようにユーザー インターフェースも提供します。

iOS デバイス用の Apple App Store および Android デバイス用の Google Play Store から CySmart アプリケーションをダウンロードできます。これらのアプリのソースコードはサイプレスの Web サイトでも入手できます。

- Apple App Store: ここをクリック。
- Google Play Store: ここをクリック。

CySmart モバイル アプリケーションを使用して、健康温度計アプリケーションを検証するには、以下の手順に従ってください。

8. 図 37 に示すように、デバイス上の CySmart アプリケーションを開きます。

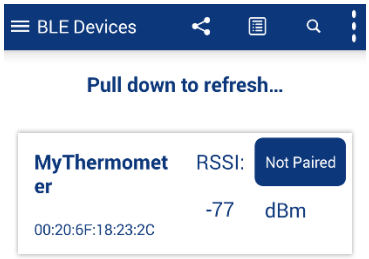
   Note: スクリーンショットは CySmart Android のアプリケーション用です。CySmart iOS アプリケーションの場合、多少の違いがあるかもしれません。

図 37. BLE コンフィギュレーション



9. Bluetooth が有効になっている場合、モバイルデバイスは BLE デバイスをスキャンし、画面にリストします。それ以外の場合は、ユーザーに Bluetooth を有効にするように求めてから、BLE デバイスを検索します。図 38 は、付近の BLE デバイスを示しています。

図 38. デバイスのリスト



10. 図 38 に示すように、「MyThermometer」というデバイスに接続するために、デバイス名をクリックします。

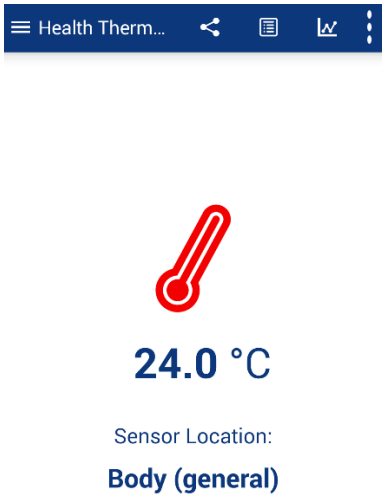
11. 接続が確立された時、図 39 に示すように、アプリケーションは自動的にすべてのアトリビュートを検出し、検出したサービスをカルーセル フォーマットで表示します。

図 39. ホーム ページ



12. 健康温度計サービスを選択します。図 40 に示すように、現時点の温度とセンサーの位置が表示されます。

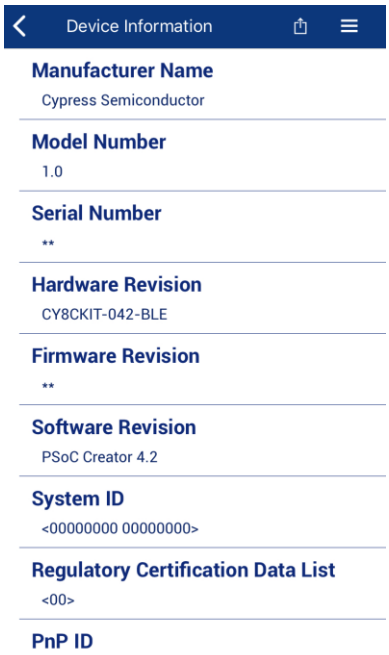
図 40. 健康温度計サービス



13. CySmart アプリケーションのホーム画面に戻るために、画面上の「Back」ボタンをクリックします。

14. 「Device Information」サービスを選択します。図 41 に示すように、現時点の温度とセンサーの位置が表示されます。

図 41. Device Information Service (デバイス情報サービス)



6.3 まとめ

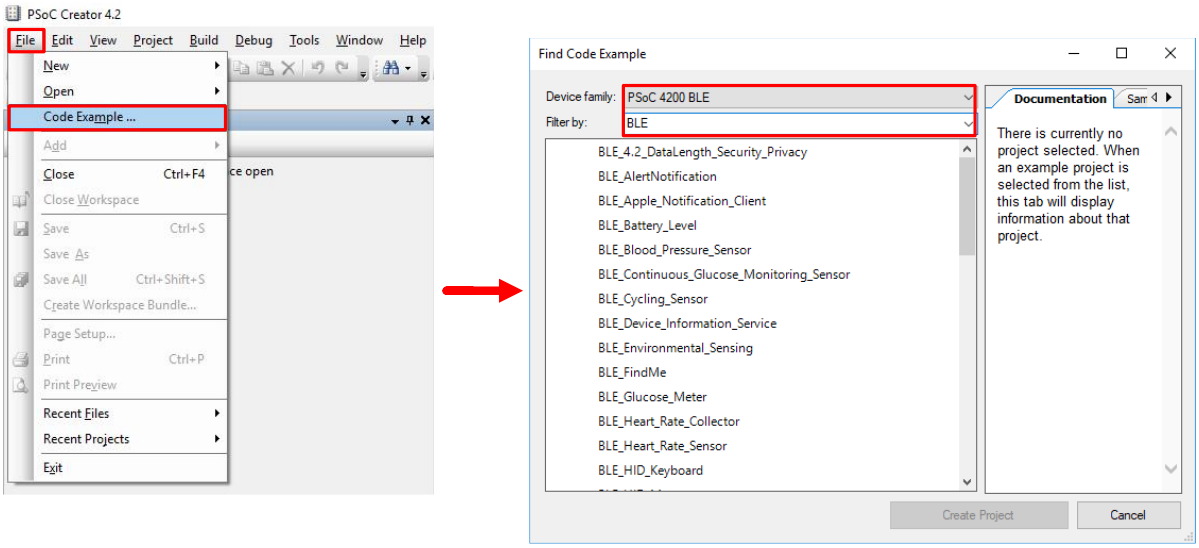
このアプリケーションノートでは、PSoC Creator BLE コンポーネントを使用して、標準の BLE プロファイルを使用して BLE Health Thermometer アプリケーションを設計する方法を検討しました。次に、CySmart Central Emulation Tool と Cypress が提供する CySmart モバイルアプリを使用して、このアプリケーションを検証しました。

7 関連文書

アプリケーションノート	
AN91267 - PSoC® 4 BLE 入門	本資料は、PSoC 4 BLE のアーキテクチャとその開発ツールを探究することをご支援し、PSoC Creator™という PSoC 4 BLE の開発ツールを使用してどのように簡単に BLE 設計を作成できるかをご説明します。
AN91162 – BLE カスタム プロファイルの作成について	カスタム BLE プロファイルを採用して PSoC 4 BLE または PRoC BLE デバイスに基づいて Bluetooth® Low Energy (BLE) アプリケーションを開発する方法について説明します。
AN92584 – Designing for Low Power and Estimating Battery Life for BLE Applications	PSoC 4 BLE デバイスを使用した低電力アプリケーションの設計について説明します。また、BLE アプリケーションの消費電流およびバッテリ寿命を計算する方法も説明し、バッテリ寿命を延ばすために消費電流を最小化するヒントとコツを提供します。
AN66477 – PSoC® 3, PSoC 4, and PSoC 5LP - Temperature Measurement with a Thermistor	PSoC®3、PSoC 4、または PSoC 5LP を使用してサーミスタで温度を測定する方法を説明します。このアプリケーションノートでは、PSoC Creator™サーミスタ計算機コンポーネントについて説明します。このコンポーネントは、数学を多用する抵抗から温度への変換を簡素化します。
AN210781 – Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity	BLE 接続を備えた PSoC 6 MCU デバイスおよび初めての PSoC Creator プロジェクトのビルド方法について説明
AN215656 – PSoC 6 MCU デュアル CPU システム設計	PSoC 6 MCU のデュアル CPU アーキテクチャおよびシンプルなデュアル CPU 設計のビルド方法について説明
AN219434 - PSoC 6 MCU ー 生成コードの IDE へのインポート	PSoC Creator によって生成されたコードを所望の IDE へエクスポートする方法を説明
PSoC Creator コンポーネント データシート	
ピン	ハードウェア リソースの物理ピンへの接続をサポート
タイマーカウンター (TCPWM)	固定機能のタイマー/カウンターの実装をサポート
クロック	ローカル クロック生成をサポート
割込み	ハードウェア信号からの割り込みの生成をサポート
デバイス資料	
PSoC® 4: PSoC 4200_BLE Family Datasheet Programmable System-on-Chip (PSoC®)	
PSoC 6 MCU: PSoC 63 with BLE Datasheet	
PSoC 6 MCU: PSoC 63 with BLE Architecture Technical Reference Manual	
PSoC 6 MCU: PSoC 62 Datasheet	
PSoC 6 MCU: PSoC 62 Architecture Technical Reference Manual	
開発キット資料	
CY8CKIT-042-BLE Bluetooth Low Energy Pioneer Kit Guide	
CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit	
CY8CKIT-062-WiFi-BT PSoC 6 WiFi-BT Pioneer Kit	
CY8CPROTO-062-4343W PSoC 6 Wi-Fi BT Prototyping Kit	
CY8CPROTO-063 BLE PSoC 6 BLE Prototyping Kit	
ツール資料	
PSoC Creator	Downloads タブでクイックスタートとユーザーガイドを確認してください

付録 A. サンプルコード

PSoC Creator の「Find Code Example」ブラウザを使用して、BLE に関連するコード例を検索およびダウンロードできます。



次の表は、上記を使用して見つけることができる BLE コード例の要約を提供します。

サンプルコード	用途	GAP ロール
BLE Find Me	このサンプル プロジェクトは、BLE コンポーネントの Find Me Profile 操作を示しています。Find Me ターゲットは、Immediate Alert Service の 1 つのインスタンスで Find Me プロファイルを利用して、クライアントがアラート用にデバイスを設定している場合にアラートを表示します。	ペリフェラル
BLE Device Information Service	このサンプルコードは、BLE コンポーネント API とアプリケーション層コールバックを構成および使用する方法を示しています。デバイス情報サービスは、BLE コンポーネントで BLE サービスの特性を構成する方法を示す例として使用されます。	ペリフェラル
BLE 4.2 Data Length Security Privacy	このサンプル プロジェクトでは、BLE 4.2 データ長拡張、認証付き LE セキュア接続 (SC) と暗号化のペアリング、およびリンク層プライバシー (LL プライバシー) を示します。	ペリフェラル
BLE Alert Notification Profile	このサンプルプロジェクトは、BLE コンポーネントのアラート通知クライアント操作を示しています。アラート通知クライアントは、アラート通知サービスの 1 つのインスタンスで BLE アラート通知プロファイルを使用して、アラート通知サーバーから「Email」、「Missed call (不在着信)」、および「SMS/MMS」アラートに関する情報を受信します。	ペリフェラル
BLE Apple Notification Client	このサンプル プロジェクトは、BLE Apple 通知クライアント アプリケーションのワークフローを示しています。アプリケーションは、GATT クライアントモードで BLE Apple 通知センター サービスを使用して、BLE Apple 通知センター サーバー (iPhone、iPod など) と通信します。	ペリフェラル
BLE Battery Level	このプロジェクトは、PSoC 4 BLE の内部 ADC を使用したバッテリー電圧の測定を実証し、BLE バッテリー アラート サービスを使用してバッテリー電圧の変化を BLE セントラル デバイスに通知します。	ペリフェラル
BLE Blood Pressure Sensor	このサンプル プロジェクトでは、BLE 血圧センサーアプリケーションのワークフローを示します。血圧センサー アプリケーションは、BLE 血圧プロファイルを使用して、血圧測定レコードをクライアントに報告します。	ペリフェラル

サンプルコード	用途	GAP ロール
BLE Continuous Glucose Monitoring Sensor	このサンプル プロジェクトは、BLE 持続血糖値測定センサー アプリケーションのワークフローを示しています。アプリケーションは BLE 持続血糖値測定プロファイルを使用して、BLE 持続血糖値測定サービスによってクライアントに CGM 測定レコードを報告し、Bond Management サービスによって一元管理します。	ペリフェラル
BLE Cycling Sensor	このサンプルは、サイクリング速度とケイデンス (Cadence) サービス (CSCS) およびサイクリング電力サービス (CPS) を示しています。サイクリング スピードとケイデンスは、サイクリング活動をシミュレートし、CSCS を使用して、シミュレートされたサイクリング スピードとケイデンス データを BLE セントラル デバイスに報告します。Cycling Power は、CPS を使用して、サイクリング電力データをシミュレートし、シミュレートされたデータを BLE セントラル デバイスにレポートします。	ペリフェラル
BLE Environmental Sensing Profile	このサンプル プロジェクトは、BLE コンポーネントの環境検知プロファイル操作を示しています。環境センサーは、環境検知およびデバイス情報サービスの 1 つのインスタンスと、環境検知プロファイルを使用して、風速測定をシミュレートします。	ペリフェラル
BLE Glucose Meter	このサンプル プロジェクトは、BLE 血糖値測定器アプリケーション ワークフローを示しています。血糖値測定器アプリケーションは、BLE 血糖値測定プロファイルを使用して、血糖値測定レコードをクライアントに報告します。また、血糖値測定器アプリケーションは、バッテリー サービスを使用して、バッテリー レベルとデバイス情報サービスを通知します。	ペリフェラル
BLE Heart Rate Collector	このサンプルプロジェクトは、BLE 心拍数測定器のワークフローを示しています。このプロジェクトは、BLE 対応の心拍センサーから心拍数データを受信し、UART を介して端末ソフトウェア上のデータを示します。	セントラル
BLE Heart Rate Sensor	このサンプル プロジェクトは、BLE 心拍センサーのワークフローを示しています。このプロジェクトは、心拍数データをシミュレートし、BLE 対応のセントラル/クライアント デバイスとの通信を実行します。	ペリフェラル
BLE HID Keyboard	このプロジェクトは、ブートおよびプロトコル モードでのキーボードの押下を示しています。このサンプルでは、セントラル デバイスからのサスペンド イベントの処理と、サスペンド時に低電力モードに入ることも示しています。	ペリフェラル
BLE HID Mouse	このプロジェクトは、ブート モードとプロトコル モードでのマウスの動きとボタンクリック HID レポートを示します。このサンプルでは、セントラル デバイスからのサスペンド イベントの処理と、サスペンド時に低電力モードに入ることも示しています。	ペリフェラル
BLE IPSP Router and Node	このサンプル プロジェクトは、BLE コンポーネントのインターネット プロトコル サポート プロファイル操作を示しています。このサンプルは、L2CAP チャネルを使用して、BLE トランスポートを介して 2 つのデバイス間に IPv6 通信インフラをセットアップする方法を示しています。このサンプルは、IPSP ルーター (GAP セントラル) と IPSP ノード (GAP ペリフェラル) の 2 つのプロジェクトで構成されています。ルーターは、生成された異なるコンテンツのパケットをループ内のノードに送信し、その後受信したデータパケットで検証します。ノードは、ルーターから受信したデータを単純にラップしてルーターに戻します。	セントラルおよびペリフェラル
BLE Navigation	このサンプル プロジェクトは、ロケーションおよびナビゲーション ポッド アプリケーションのワークフローを示しています。アプリケーションは、BLE ロケーションおよびナビゲーション プロファイルを使用して、クライアントにロケーションおよびナビゲーション情報を報告します。	ペリフェラル
BLE OTA External Memory Bootloadable and Bootloader	このサンプル プロジェクトは、BLE ブートローダー サービスを使用した OTA ファームウェアの更新を示しています。デフォルトでは、これはデバイス情報サービスを備えた BLE コンポーネントを含む通常のブートローダブル プロジェクトです。ブートローダーモードが有効になると、このサンプル プロジェクトは、ブートローダブル プロジェクトの新しいイメージの受信準備と、外部メモリへの保存準備が整います。	ペリフェラル

サンプルコード	用途	GAP ロール
BLE OTA Fixed Stack Bootloader and Bootloadable	このサンプル プロジェクトでは、カスタム リンカー スクリプトを使用して、ブートローダー プロジェクトとブートローダブル プロジェクト間でメモリブロックを共有する方法を示します。ブートローダーが API 関数を配置し、ブートローダブルが API 関数を呼び出す方法を示します。ブートローダー プロジェクトの目的は、デバイス上のブートローダブル イメージを、Bluetooth プロトコルによって OTA に送信されたイメージに置き換えることです。	ペリフェラル
BLE OTA Upgradable Stack HID Keyboard	このサンプル プロジェクトでは、アップグレード可能なアプリケーション プロジェクト (HID キーボード) およびアップグレード可能なスタック プロジェクトを BLE スタックで実装する方法を示します。このアプリケーション プロジェクトでは、スタック プロジェクトの BLE スタックを使用しています。	ペリフェラル
BLE Upgradable Stack Example Launcher	このサンプル プロジェクトでは、アップグレード可能なアプリケーション プロジェクト (HID キーボード) およびアップグレード可能なスタック プロジェクトを BLE スタックで実装する方法を示します。このアプリケーション プロジェクトでは、スタック プロジェクトの BLE スタックを使用しています。	ペリフェラル
BLE Upgradable Stack Example Stack	このサンプル プロジェクトでは、アップグレード可能なアプリケーション プロジェクト (HID キーボード) およびアップグレード可能なスタック プロジェクトを BLE スタックで実装する方法を示します。このアプリケーション プロジェクトでは、スタック プロジェクトの BLE スタックを使用しています。	ペリフェラル
BLE Phone Alert	このサンプルプロジェクトは、BLE 電話アラート ハンドラ アプリケーションのワークフローを示しています。電話アラート ハンドラ アプリケーションは、BLE 電話アラート ステータス プロファイルを使用して、サーバーのアラート状態と呼び出し音設定を監視および制御します。	ペリフェラル
BLE Proximity	このサンプルプロジェクトは、BLE コンポーネントの近接通知処理を示しています。Proximity Reporter (近接レポーター) は、接続損失サービスの 1 つのインスタンスと送信電源サービスの 1 つのインスタンスにおいて、BLE 近接プロファイルを利用して、クライアントとの接続が失われた場合にデバイスにアラートを表示します。	ペリフェラル
BLE Running Speed Cadence	このサンプル プロジェクトは、BLE コンポーネントのランニング スピードとケイデンス センサーの動作を示しています。デバイスは、ランニング/ウォーキング データ測定をシミュレートし、BLE Running Speed and Cadence Service を介して送信します。	ペリフェラル
BLE Temperature Measurement	このサンプル プロジェクトは、BLE コンポーネントの体温計プロファイルの操作を示しています。デバイスは体温計の読み取り値をシミュレートし、BLE 体温計サービスを介して送信します。また、バッテリー レベル値を測定し、BLE バッテリー サービスを介して送信します。	ペリフェラル
BLE Time Sync	このサンプル プロジェクトは、BLE コンポーネントのタイム プロファイル操作を示しています。この Time Sync のサンプルでは、Current Time Service (現在時刻サービス) の 1 つのインスタンスで BLE タイム プロファイル (タイム クライアントとして GAP ペリフェラルロール用に構成) を利用して、外部タイム サーバーからの時刻同期機能を示します。	ペリフェラル
BLE Weight Scale	このサンプル プロジェクトは、BLE コンポーネントの体重計プロファイル操作を示しています。体重計センサーは、体重計、体組成、ユーザー データ、およびデバイス情報サービスの 1 つのインスタンスを使用して、最大 4 人の登録ユーザーの体重測定をシミュレートします。	ペリフェラル
BLE Wireless Power Transmitter and Receiver	このサンプル プロジェクトでは、BLE コンポーネントのワイヤレス給電プロファイル操作を示します。このサンプルでは、ワイヤレス給電システムの受電ユニット (PRU) と送電ユニット (PTU) 間の通信を示します。PTU セントラル デバイスは、最大 8 つの PRU 周辺機器との多重接続をサポートします。PRU は、受電装置のデータをシミュレートし、ワイヤレス給電サービス (WPTS) を使用して、シミュレートされたデータを PTU に報告します。この例は、ワイヤレス送電 (GATT クライアント) とワイヤレス受電 (GATT サーバー) の 2 つのプロジェクトで構成されています。	セントラルおよびペリフェラル

サンプルコード	用途	GAP ロール
CE11181 - BLE HTTP Proxy Code Examples with PSoC 4 BLE	HTTP プロキシ サーバー プロジェクトと HTTP プロキシ クライアント プロジェクトをペアで使用して、BLE HTTP プロキシ サービス (HPS) 操作を示します。HTTP プロキシ サーバーは、HTTP プロキシ サービスの 1 つのインスタンスを使用して、BLE デバイス上の HTTP サーバーをシミュレートします。HTTP プロキシ サーバーは、HTTP プロキシ クライアントの役割を実装する他のデバイスでも動作できます。	セントラルおよびペリフェラル
CE211245 - Bluetooth Low Energy (BLE) Indoor Positioning	このサンプル プロジェクトでは、GATT 接続で設定できる BLE ブロードキャスト モードを使用して屋内ナビゲーション システムを作成する方法を示します。このプロジェクトでは、BLE Pioneer Kit を時分割多重放送局および接続可能な屋内測位サービス (IPS) サーバーとして構成します。	ペリフェラル
CE217613 - Bluetooth Low Energy (BLE) Automation IO	BLE コンポーネントの自動入出力プロファイル機能と関連 API の使用方法を示す BLE サンプルプロジェクト。このプロジェクトは、デジタル特性の 2 つのインスタンスと、アナログ特性と集約特性の 2 つのインスタンスとを備えた自動入出力サーバー (AOIS) として、BLE パイオニアキットを構成します。	ペリフェラル

改訂履歴

文書名: AN91184 – PSoC® 4 BLE - BLE アプリケーションの設計

文書番号: 002-00014

版	ECN	発行日	変更内容
**	4929877	09/25/2015	これは英語版 001-91184 Rev. *B を翻訳した日本語版 002-00014 Rev. **です。
*A	5800934	07/06/2017	更新されたロゴと著作権。
*B	6833537	03/18/2020	これは英語版 001-91184 Rev. *F を翻訳した日本語版 002-00014 Rev. *B です。

セールス, ソリューションおよび法律情報

ワールドワイドな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&バッファ	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	cypress.com/wireless

PSoC®ソリューション

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

サイプレス開発者コミュニティ

コミュニティ | サンプルコード | 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

本書で言及するその他のすべての商標または登録商標は各社の所有物です。

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

© Cypress Semiconductor Corporation, 2015-2020. 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若しくはファームウェア (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, 本段落で特に記載されているものを除き, その特許権, 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 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がない場合, Cypress は, (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, かつ組織内部でのみ, 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) 本ソフトウェアをバイナリーコード形式で外部エンドユーザーに配布すること, 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) が抵触する 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若しくはこれに伴うハードウェアに関しても, 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。いかなるコンピューティングデバイスも絶対に安全ということはない。従って, Cypress のハードウェアまたはソフトウェア製品に講じられたセキュリティ対策にもかかわらず, Cypress は, Cypress 製品への権限のないアクセスまたは使用といったセキュリティ違反から生じる一切の責任を負わない。加えて, 本書面に記載された製品には, エラッタと呼ばれる設計上の欠陥またはエラーが含まれている可能性があり, 公表された仕様とは異なる動作をする場合がある。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, いかなる製品若しく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, プログラム, 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の使用, 又は装置若しく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ようなその他の使用 (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, それの不具合が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ような装置若しく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 Cypress はそれら一切から本書により免除される。Cypress 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) から免責補償される。

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 のより完全な商標のリストは, cypress.com を参照すること。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。